УДК: 519.7, 621.3
MSC2010: 68M07

ОБРАТИМЫЕ ВЫЧИСЛЕНИЯ. ЧАСТЬ I

© С. И. Гуров

Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики
Ленинские горы, д. 1, стр. 52, Москва, ГСП-1, 119991, Российская Федерация
e-mail: *sgur@cs.msu.su*

© А. Е. Жуков

Московский государственный технический университет имени Н. Э. Баумана
2-я Бауманская ул., д. 5, стр. 1, Москва, 105005, Российская Федерация
e-mail: *aez_ iu8@rambler.ru*

© Д. В. Закаблуков

ООО "Алгоритмы и данные ул. Дмитрия Ульянова, д. 42, стр. 1, 117218, Москва,
Российская Федерация e-mail: *dmitriy.zakablukov@gmail.com*

© Г. В. Кормаков

Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики
Ленинские горы, д. 1, стр. 52, Москва, ГСП-1, 119991, Российская Федерация
e-mail: *egor2898@mail.ru*

Reversible Calculations. Part I.

Gurov S. I., Zhukov A. E., Zakablukov D. V., Kormakov G. V.

Abstract. In the first part of the paper, the main provisions of reversibility as a new paradigm for the development of computer technology are considered. The fundamentals of reversible logic and models of reversible calculations, reversible programming languages are stated.

Keywords: *reversible logic, reversible computation models, reversible programming languages.*

1. Введение в проблематику. Принцип Неймана–Ландауэра

Интерес к обратимым вычислениям и реализующим их схемам из обратимых элементов возник в начале 1960-х гг., когда был сформулирован казавшийся сначала парадоксальным принцип Неймана–Ландауэра: *в любой вычислительной системе, независимо от её физической реализации, при потере одного бита информации выделяется теплота в количестве не менее* $\varepsilon_0 = kT \ln 2$ Дж (k — постоянная Больцмана, T — абсолютная температура) [1].

Продолжительное время данный принцип оставался всего лишь чистой теорией, не подкреплённой результатами экспериментов, что было связано с трудностями измерения крайне малых количеств выделяемой энергии. Однако в 2012 г. в эксперименте с коллоидной частицей эффект Ландауэра удалось обнаружить на практике [2]. В 2014 г. был проведен ещё один эксперимент, показавший, что при уменьшении возможных макроскопических состояний системы в два раза выделяется минимум ε_0 тепла [3].

При комнатной температуре $\varepsilon_0 \approx 3 \cdot 10^{-21}$ Дж = 0,017 эВ — крайне малое количество. Но в пересчете на процессор суммарная рассеиваемая мощность вырастает уже до величин порядка 1 Вт[1].

Проведённый в рамках проекта International Technology Roadmap for Semiconductors ITRS-2001[2] анализ развития транзисторных технологий на период после 2001 г. показал, что при использовании 22-нм технологии интегральных микросхем (ИМС), выделение тепла составит 5–10 МВт на кв. см поверхности процессора; для сравнения: Солнце выделяет не более 6,5 кВт/см^2. Очевидно, что обеспечить необходимое охлаждение становится всё более трудным. Поэтому проблема отвода тепла уже в следующем десятилетии станет гораздо более существенной и игнорировать эффект Ландауэра, оставаясь в рамках современных технологий, уже нельзя. В результате исследователи пришли к выводу, что, без учёта тепловых шумов и требований надёжности, физический предел традиционных технологий вычислителя с плавающей точкой — 10^{22} операций в секунду [4]. Данная ситуация выглядит как конец развития вычислительной техники в рамках существующих технологий из-за «*теплового проклятия*».

[1] http://old.computerra.ru/2004/538/204845/
[2] http://www.itrs2.net

Выход видят в переходе к вычислительным устройствам, реализующим *обратимые вычисления*[3], позволяющие обойти ограничение, устанавливаемое принципом Ландауэра. При обратимых вычислениях информация не теряется, откуда следуют (теоретически) нулевые потери энергии на её обработку.

Важным является то, что обратимость необходимо поддерживать на всех уровнях вычислений: физической модели, архитектуры вычислителя, языков программирования высокого уровня и реализуемых алгоритмов: Ч. Беннетт указал, что необратимость хотя бы на одном уровне полностью разрушает положительные эффекты остальных [5]. Отсюда следует, что для создания парадигмы обратимых вычислений требуется разработать следующие новые направления:

- теорию (алгебру и логику) обратимых вычислений;
- языки и парадигмы обратимого программирования;
- методы реализации прикладных программ и алгоритмы обратимого программирования;
- обратимую схемотехнику;
- физическую реализацию обратимых элементов.

В статье дан краткий обзор основных понятий, связанных с обратимостью, рассмотрены вопросы построения схем из функциональных обратимых элементов, в том числе сбоеустойчивых и предложены основные такие элементы. Указаны направления и результаты применения схем из обратимых логических элементов в криптографии.

2. Основы обратимой схемотехники

2.1. **Понятие обратимости. Мусорные биты.** Говорят, что вычисления *логически обратимы*, если по выходным величинам можно восстановить входные. Такие вычисления реализуют на *обратимых элементах*.

Логический элемент назовём *полуобратимым*, если он осуществляет инъективное преобразование входного булевого вектора в выходной. Обычно дополнительно требуют, чтобы произвольный сигнал, направленный в обратном направлении, от выхода элемента к его входу, однозначно и безошибочно восстановил исходный входной вектор. Назовём элемент с n входами *обратимым*, если и только если он осуществляет некоторую биекцию на множестве всех 2^n возможных входов или их перестановку. Как следствие, такой элемент должен иметь n выходов. В общем случае элементы с n входами и m выходами будем называть $n \times m$-элементами.

[3] англ. *reversible*; ранее употреблялись также термины *консервативные, реверсивные* вычисления

При реализации комбинационной логики схемами из обратимых элементов, на выходе кроме *информационных битов* — значений вычисляемой функции — как правило, формируются ещё и дополнительные биты, называемые (неудачно) *мусорными* или *стоками* (англ. *garbage* и *sink* соответственно). Мусорные биты в действительности играют очень важную роль: они дают возможность по выходному вектору однозначно восстановить входной, т. е. обратить вычисления. Если их не сохранить, произойдёт рассеяние энергии, а именно этого требуется избежать.

2.2. **Обратимые комбинационные элементы.** Чтобы отличать обычные комбинационные элементы от обратимых, будем называть первые *вентилями*, а вторые *гейтами*, оставив термин *элемент* для обоих. Входы элементов будем, как правило, обозначать литерами с начала алфавита: A, B, ..., а выходы — с середины алфавита: P, Q, ... Иногда удобнее обозначения A_1, A_2, ... и P_1, P_2, ... соответственно. На схемах часто входы и выходы обозначают литерами x, y, ..., а их порядок указывают положением на элементе сверху вниз. В формулах $\oplus$ означает сумму по $mod\,2$, штрихом $'$ обозначают инвертирование, знак конъюнкции "$\cdot$" иногда опускаем.

Простейшими обратимыми элементами являются NOT и SWAP.

Инвертор (NOT) — имеет один вход A, один выход P и осуществляет безусловное инвертирование входного сигнала:

$$P = A'.$$

Обмен (SWAP) — имея два входа A, B, меняет их местами на выходах P, Q:

$$P = B, \quad Q = A.$$

Входы большинства других обратимых элементов разделяются на *управляющие*, или *адресные* и *управляемые*, или *целевые*. Значения вторых на выходе могут быть изменены в зависимости от значений первых, которые передаются на выход без изменений (их иногда называют *сквозными дорожками*). Преобразованные выходы также называют *сигнальными*.

Контролируемый инвертор (CNOT, Controlled NOT, элемент Феймана, FG) — 2×2-обратимый гейт, реализующий *условное инвертирование*. Имеет два входа A, B и два выхода P, Q. Первый вход A — управляющий, а второй вход будет инвертирован на выходе, если и только если $A = 1$:

$$P = A, \quad Q = A \oplus B.$$

Рассмотренные выше элементы *вычислительно не универсальны*: используя только их, невозможно реализовать произвольную логическую функцию. Ясно, что универсальный элемент должен обеспечить реализацию (в зависимость от значений управляющих входов) полной системы булевых функций, например, пар функций AND2 и NOT или OR2 и NOT. Кроме того, такие элементы должны обеспечивать возможность *каскадирования* — организации последовательного соединения элементов. Далее рассматриваются универсальные обратимые элементы. В наибольшим числе работ в качестве таковых используют элементы Тоффоли или Фредкина.

Вентиль Тоффоли (CCNOT, Controlled Controlled NOT, TG) — 3×3-универсальный контролируемый обратимый гейт, имеет три входа A, B, C и три выхода P, Q, R. Первые два входа A, B — управляющие: третий вход будет инвертирован на выходе, если и только если $A = B = 1$. Формулы выходов:

$$P = A, \quad Q = B, \quad R = C \oplus A \cdot B.$$

Очевидно, элемент Тоффоли действительно вычислительно универсален: при $A = B = 1$ он реализует инвертор NOT, а при $C = 0$ — AND2. На элементах Тоффоли можно осуществить операцию ветвления и передачу сигналов, обеспечивая каскадирование: при входном векторе $(A, 1, 0)$ на его выходе будет вектор $(A, 1, A)$.

Вентиль Фредкина (CSWAP, Controlled SWAP, управляемый обмен, симметрический переключатель, FRG) — 3×3-универсальный контролируемый обратимый гейт, он, как и TG, имеет три входа A, B, C и три выхода P, Q, R. Первый вход A — управляющий: если он равен 1, то остальные два входа B, C на выходе поменяются местами, в противном же случае они подаются на выход неизменными:

$$P = A, \quad Q = A' \cdot B \vee A \cdot C, \quad R = A \cdot B \vee A' \cdot C.$$

Иногда FRG изображают, как показано на рис. 1.

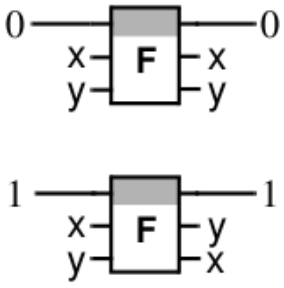

Рис. 1. Вариант изображения элемента Фредкина

Нетрудно видеть, что элемент Фредкина также вычислительно универсален: во-первых, при входе $(A, 0, 1)$ на его выходе будет (A, A, A'), что означает реализацию функций NOT и дублирования сигнала и, во-вторых, при входах $(A, B, 1)$ и $(A, B, 0)$ на его выходе будут векторы $(A, A \oplus B, B \operatorname{Im} A)$ и $(A,\ A' \cdot B,\ A \cdot B)$, т. е. реализуются пары функций (OR2, обратная импликация) и (обратная коимпликация, AND2) соответственно. Так, реализация функции XOR2 на двух гейтах Фредкина показана на рис. 2.

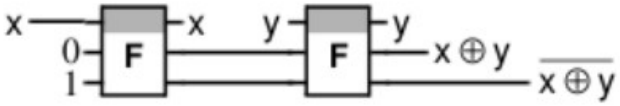

Рис. 2. Реализация функции XOR2 на гейтах Фредкина

Сам факт того, что управляемо перемешивая проводки в кабеле между входом и выходом и не используя никаких других элементов, можно получить на выходе значение любой булевой функции от входа, далеко не очевиден [6].

Рассмотренные элементы NOT, CNOT, Тоффоли и Фредкина оказываются обратными самим себе: если сигналы пройдут через два экземпляра одного элемента, они останутся неизменными. Ясно, что элемент *самообратим*, если и только если реализуемое им биективное преобразование представляется в виде совокупности транспозиций (перестановок внутри некоторых пар) элементов множества всех двоичных векторов.

Обычно схематически рассмотренные гейты изображаются как показано на рис. 3.

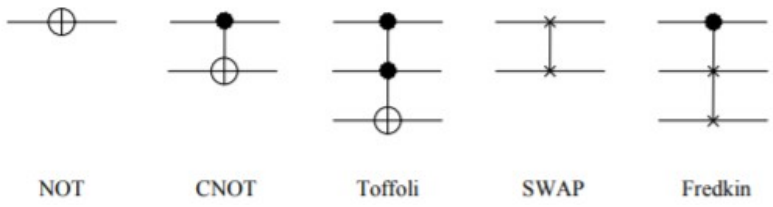

Рис. 3. Обычное схематическое изображение гейтов NOT, CNOT, TG, SWAP, FRG

Отметим ещё *обобщённый гейт Тоффоли* ($\mathrm{C}^{n-1}\mathrm{NOT}$, MCT, Multiple-Control Toffoli Gate, generalized n-bit Toffoli) — $n \times n$-гейт. Из n входов $A_1, \ldots, A_n$ первые $n-1$ — управляющие, функции n выходов $P_1, \ldots, P_n$, представляются формулами

$$P_i = A_i,\ i = \overline{1, n-1}, \quad P_n = A_1 \cdot \ldots \cdot A_{n-1} \oplus A_n.$$

Ясно, что при $n = 2$ элемент $\mathrm{C}^1\mathrm{NOT}$ есть FG, а при $n = 3$ элемент $\mathrm{C}^2\mathrm{NOT}$ есть TG.

Расширенный гейт Тоффоли (ETG) имеет $n+1$ входов $A_1, \ldots, A_n, A_{n+1}$, первые $n-1$ из которых — управляющие и $n+1$ выходов $P_1, \ldots, P_n, P_{n+1}$, из которых последние два (а не один, как у $\mathrm{C}^{n-1}\mathrm{NOT}$) сигнальные, инвертирующие управляемые сигналы A_n и A_{n+1} соответственно при 1 на всех управляющих входах:

$$P_i = A_i,\ i = \overline{1, n-1}, \quad P_n = A_1 \cdot \ldots \cdot A_{n-1} \oplus A_n \quad P_{n+1} = A_1 \cdot \ldots \cdot A_{n-1} \oplus A_{n+1}.$$

Схематическое представление элемента ETG дано на рис. 4 (обратите внимание на использование знака $\times$).

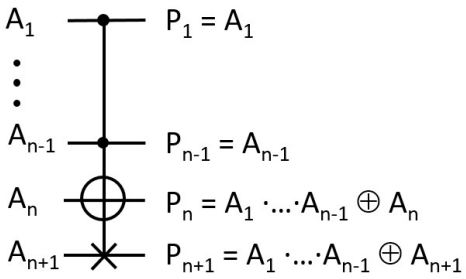

Рис. 4. Элемент ETG

Обобщённый (множественно управляемый) гейт Фредкина (обобщённый CSWAP, C^nSWAP, MCF, Multiple-Control Fredkin Gate) имеет n входов $A_1, \ldots, A_n$ из которых первые $n-2$ управляющие, и n выходов $P_1, \ldots, P_n$, последние два из которых — сигнальные, повторяющие A_n и A_{n-1}, если все управляющие входы — единичные, или не делая этого, иначе.

Вентиль Переса (PG) — обратимый 3×3-гейт, имеющий три входа A, B, C и три выхода P, Q, R:

$$P = A, \quad Q = A \oplus B, \quad R = A \cdot B \oplus C.$$

Вентиль Переса может быть реализован на гейтах TG и CNOT, соединённых последовательно, как показано на рис. 5.

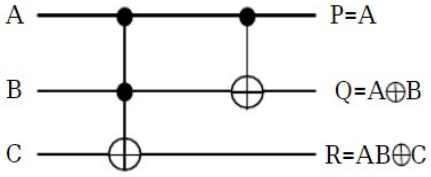

Рис. 5. Вентиль Переса, реализованный на TG и CNOT

Этот элемент интересен тем, что на его основе относительно просто может быть синтезирован полный однобитный сумматор[4] — см. рис. 6.

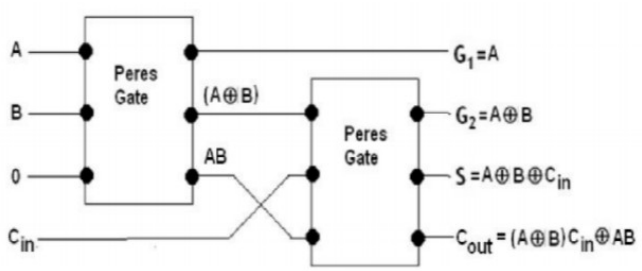

Рис. 6. Обратимый однобитный сумматор на PG: A, B — суммируемые разряды, C_{in} — перенос из предыдущего разряда, S — сумма, C_{out} — перенос в следующий разряд

Следующие два гейта удовлетворяют условию инъективного, но не биективного преобразования входа в выход. Как следствие, у них число выходов больше числа входов.

Переключатель Прайса универсальный полуобратимый гейт; имеет два входа — A, B и три выхода — P, Q, R. Логика его работы представлена формулами:

$$P = A, \quad Q = A \cdot B, \quad R = A' \cdot B.$$

Вентиль ВЗАИМОДЕЙСТВИЕ (Interaction Gate, IG) — универсальный полуобратимый гейт, имеет два входа A, B и четыре выхода P, Q, R, S, реализующих четыре функции, эквивалентные относительно преобразований Поварова-Шеннона конъюнкций входов:

$$P = A \cdot B, \quad Q = A' \cdot B, \quad R = A \cdot B', \quad S = A' \cdot B'.$$

Кроме рассмотренных, известны также гейты Кернтопфа, Марголуса, несколько гейтов Де Воса и множество других; ссылки можно найти в [7, 8].

3. Модели обратимых вычислений

3.1. **Модель бильярдных шаров.** Так называют реализующую обратимую логику абстрактную физическую модель (Billiard Ball Model, BBM), предложенную Э. Фредкиным и Т. Тоффоли [6]; из этой работы взяты рисунки 7 и 8 данного подраздела.

[4] *Полный однобитный сумматор* вычисляет 1) сумму по $mod\,2$ трёх битов — двух данных разрядов чисел и переноса из младшего разряда и 2) бит переноса в старший разряд.

Данная модель используется во многих исследований по обратимости вычислений и вычисления в ней называют *баллистическими*.

Модель представляет собой плоскость, «стол», по которой без трения (точнее, без потери энергии) в строго заданных направлениях перемещаются бильярдные шары. Вследствие обратимости времени в кинематических уравнениях, траектория движения шара полностью обратима.

Совершение логических операций моделируются изменениями траектории движения шаров при их соударении между собой. Скорости, допустимые направления и размеры шаров подобраны таким образом, что в дискретные моменты времени, соответствующие тактам, шары могут находиться только в небольшом наборе точек, образующих прямоугольную, повернутую на 45° сетку, см. рис. 7. Если шары попадают в соседние точки, между ними происходит абсолютно упругое столкновение.

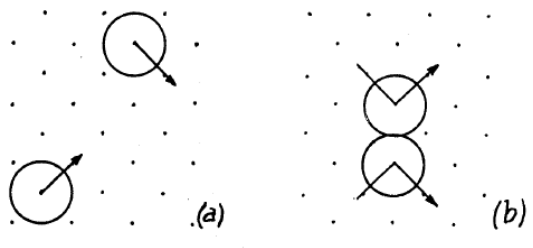

Рис. 7. (a) Шары радиуса $1/\sqrt{2}$, перемещающиеся по узлам сетки; (b) Центральное столкновение двух шаров.

Логической единицей считается наличие шара в данной точке и в данный момент времени, а нулем — его отсутствие. На плоскости могут быть установлены неподвижные стенки, или *зеркала* для отражения, сдвига, задержки и обеспечения пересечения траекторий шаров, см. рис. 8. Элемент (d) на рис. 9 называется *нетривиальным*

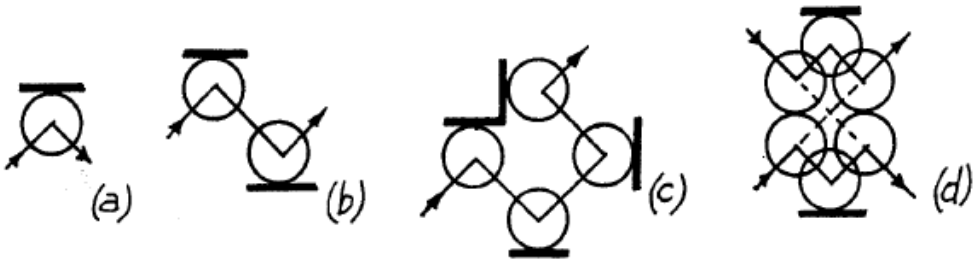

Рис. 8. Зеркала для поворота (a), сдвига (b), задержки (c) и безопасного пересечения траекторий (d) шаров.

кроссовером и позволяет шарам «как бы проходить друг через друга».

Возможность создания в рамках данной модели обратимых гейтов показана на рис. 9.

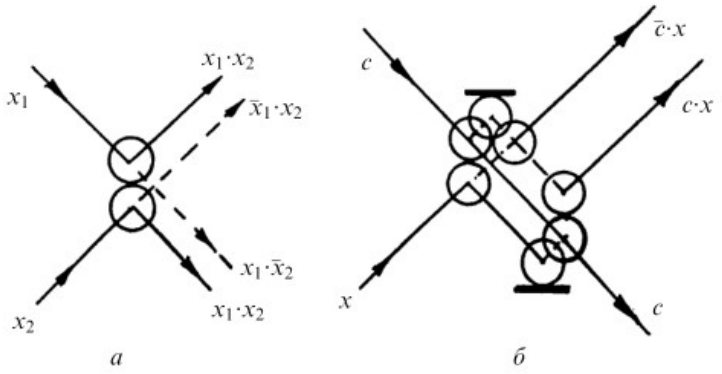

Рис. 9. Бильярдная реализация а) элемента IG и б) переключателя Прайса [9]. x_1, x_2, c, x — битовые сигналы.

Из нетривиального кроссовера, переключателей Прайса и IG можно сконструировать схему, эквивалентную гейту Фредкина.

Таким образом, возможно построение бильярдной модели компьютера. Если после того как компьютер завершил «вычисления» запустить бильярдные шары в обратном направлении, модель вернётся в исходное состояние.

Заметим, что предположение об абсолютно упругом и без потери энергии взаимодействии шаров друг с другом и с отражающими стенками является очень жёстким ограничением. Оно не влияет на теоретические рассмотрения, но существенно для физической реализации. Также ясно, что «бильярдный компьютер» чрезвычайно чувствителен к малейшим неточностям реализации: случайным отклонениям шара от правильного направления, погрешностям в установке зеркал и т. д., в результате которых ошибки «вычислений» будут накапливаться. Данные трудности существенно уменьшаются, если в качестве бильярдных шаров используется субмикроскопические частицы, например, электроны. Законы квантовой механики накладывают определённые ограничения на состояние элементарных частиц, поэтому оказывается возможным купировать малые отклонения в их движении.

Р. Меркль предложил практическую реализацию BBM [10]. Роль бильярдных шаров при этом играют электроны, пучки которых перемещаются по индуктивным контурам между конденсаторами, а стол представляет собой алмазный контейнер, обработанный определённым образом. Исследование модели проходили при температуре 1°K.

3.2. **Машины Тьюринга и обратимость.**

3.2.1. *Классическая машина Тьюринга (МТ).* Напомним, что машина Тьюринга — абстрактный компьютер, управляющее устройство которого (*головка записи-чтения*) способно находиться в одном из состояний конечного множества Q состояний. Головка читает/пишет/удаляет символы конечного алфавита $A = \{ a_0, a_1, \ldots, a_n \}$ с неограниченной в обе стороны разделённой на ячейки ленты, и передвигается вдоль неё на один шаг вправо/влево за такт работы. Выделяется особый *пустой символ*, заполняющий все ячейки ленты, кроме конечного числа тех, на которых записаны входные данные.

Управляющее устройство работает согласно последовательно выбираемым командам, которые представляют алгоритм, реализуемый данной МТ. Команда МТ

$$\langle q, a, q', a', d \rangle$$

означает: *находясь в состоянии q и считывая символ a, МТ переходит в состояние q', заменяет символ в ячейке на a' и либо сдвигается на ячейку вправо/влево ($d = \pm 1$), либо остаётся на месте ($d = 0$).* Некоторые состояния могут быть помечены как *терминальные*, и переход в любое из них означает конец работы алгоритма. Машина Тьюринга называется *детерминированной*, если каждой комбинации состояния и ленточного символа в таблице соответствует не более одной команды.

3.2.2. *Существование обратимых МТ.* И. Лесёрф и Ч. Беннетт в своих работах независимо доказали существование обратимых машин Тьюринга [5, 11]. Точнее, любую машину Тьюринга можно реализовать обратимо, т. е. любое вычисление, которое можно сделать на обычном необратимом вычислительном устройстве (компьютере), можно выполнить так, что оно будет обратимо.

Стирания символов с ленты делают обычную машину Тьюринга необратимой, поэтому Беннетт предложил добавить к ней вторую ленту, названную *лентой истории*, на которой сохраняются сведения об удаленных или изменённых данных. В конце вычислений полученный результат может быть записан на ещё одну ленту для сохранности. Затем машина Тьюринга начинает работать в обратную сторону, с помощью ленты истории она обращает одну за одной все операции, пока не вернется к исходному состоянию.

3.2.3. *Реверсивная машина Тьюринга (РМТ).* Рассмотрим другую конструкцию, обеспечивающее обратимость вычислений — реверсивную машину Тьюринга (РМТ) [12].

Главное отличие её от МТ — невозможность одновременного считывания символа с ленты и перемещения головки.

Команды РМТ

$$1)\ \langle\, q,\, a,\, q',\, a' \,\rangle, \qquad 2)\ \langle\, q,\, *,\, q',\, d \,\rangle$$

означают:

1) *находясь в состоянии q и считывая символ a, РМТ переходит в состояние q' и заменяет символ на a'*;
2) *находясь в состоянии q РМТ переходит в состояние q' и сдвигается на ячейку вправо/влево $d = +1/-1$.*

Если РМТ перешла из состояний q_1 и q_2 в одно и то же состояние q', то q_1 и q_2 оба принадлежат *записывающему типу* и они должны записать на ленту разные символы. Отсюда следует существование обратного прохода по пути вычислений.

При реализации на РМТ классических (на МТ) вычислений, проходящих за время T и требующие памяти S, потребуются уже время T_R и объём памяти S_R, оцениваемые как

$$T_R \,=\, 3 \cdot 2^{O\left(T/2^k\right)}, \quad S_R \,=\, S \cdot (1 + O(k))\,, \qquad \text{где } k \in (1, \log_2 T).$$

Мы видим, что при переходе к РМТ значительно увеличивается как число операций, так и требуемая для вычисления память.

Н. Н. Непейвода указал способ возможного суперэкспоненциального сокращения для любой системы алгебраического моделирования, в т. ч. и для РМТ, основанный на идее петербургского логика В. П. Оревкова [13, 14]. Поясним её на примере. Если в распоряжении вычислителя имеется лишь операция прибавления единицы, то вычисление экспоненты требует экспоненциального числа шагов. Запишем неявное определение функции $\varphi(x, y) = x + 2^y$ с помощью равенств

$$\varphi(x, 0) \,=\, x + 1, \qquad \varphi(x, y + 1) \,=\, \varphi(\varphi(x, y), y),$$

тогда экспонента вычисляется за линейное число шагов. Аналогично за линейное число шагов можно обеспечить вычисление суперэкспоненты и т. д.

3.3. **Обратимые клеточные автоматы.** Н. Марголус обобщая модель бильярдных вычислений, пришел к идее обратимых клеточных автоматов [15].

3.3.1. *Классические клеточные автоматы (КлА).* Клеточные автоматы — одна из старейших моделей вычислений, насчитывающая уже более 70 лет [15].

Классический КлА представляет собой упорядоченный набор ячеек памяти, образующих регулярную n-мерную решетку. Каждая ячейка памяти клеточного автомата может хранить одно значение из некоторого конечного множества (как правило — 1 бит). Время для клеточного автомата дискретно и измеряется тактами или шагами. Работу КлА определяют правила переписывания, по которому клетки на каждом шаге изменяют содержание хранимой в них информации.

Обычно эти правила одинаковы для всех ячеек. Смена состояний всех ячеек решетки происходит синхронно и одновременно на каждом такте, при этом новое значение каждой ячейки памяти является функцией от текущих значений ячеек, образующих её окрестность.

3.3.2. *Исследование обратимости.* КлА называется *обратимым*, если каждое его внутреннее состояние имеет единственный прообраз. Наиболее существенные результаты, связанные с вопросами обратимости, получены для классических КлА, заданных на бесконечных решетках. Если автомат является обратимым, обратное преобразование может быть реализовано также с помощью КлА (возможно с другой, в том числе и с большей окрестностью по сравнению с исходным автоматом) [16]. Также было показано, что для одномерных КлА задача алгоритмического распознавания обратимости является разрешимой [17]. В той же работе был построен алгоритм распознавания, имеющий экспоненциальную сложность.

Позже были построены алгоритмы для распознавания обратимости одномерных КлА, имеющие полиномиальную сложность [18–21]. Однако для клеточных автоматов на решетках размерности 2 и более измерений таких алгоритмов нет. Было установлено, что в общем случае эта задача является алгоритмически неразрешимой в том смысле, что не существует алгоритма, который для любого автомата всегда заканчивал бы свою работу в конечное время и давал бы правильный ответ [22, 23]. В работах [24, 25] исследовались границы между классами клеточных автоматов, для которых свойство обратимости является алгоритмически разрешимым, и теми, для которых оно алгоритмически неразрешимо, и получен критерий разрешимости свойства обратимости для классов клеточных автоматов фиксированной размерности и с фиксированным числом состояний ячейки.

Построение обратимых КлА в случае размерностей, больших 1, наталкивается на значительные трудности. Известно лишь несколько типов обратимых двумерных КлА, основными из которых являются блочные клеточные автоматы [15, 26] и клеточные автоматы второго порядка [27–29]. Автоматы этих типов отличаются от классических клеточных автоматов, однако доказано, что они могут быть эмулированы классическими КлА (возможно, с значительно большим размером окрестности и числом состояний ячейки).

3.3.3. *Пример обратимого двумерного блочного КлА.* Автомат работает следующим образом [15]. Плоскость разбивается на квадраты размером 2×2 непрерывными и пунктирными линиями, причём разбивка пунктирными линиями сдвинута относительно разбивки непрерывными линиями на один квадрат по диагонали. В цитируемой работе в стартовой конфигурации берется квадрат размером 512×512 клеток, раскрашенных случайным образом в черный и белый цвета. Затем в каждый нечётный момент времени правило используется для перекраски клеток разбиения, заданного непрерывными линиями, а в каждый чётный момент времени то же самое правило применяется к клеткам разбиения, заданного пунктиром.

3.3.4. *Клеточные автоматы конечного размера.* В большинстве приложений решетка КлА имеет конечные размеры. Это порождает т. н. «проблему краевых клеток»: необходимо определить, как задавать значения функции для ячеек, у которых отсутствует часть соседей. Чаще всего в соответствии со свойством однородности для разрешения этой проблемы противоположные края решетки клеточных автоматов отождествляются, образуя многомерный тор (в одномерном случае — кольцо). Известны и другие варианты решения проблемы краевых клеток, например, введение «нулевой границы» (null-boundary), когда значения для отсутствующих соседей полагаются равными 0.

Важной разновидностью КлА конечного размера являются *обобщенные клеточные автоматы (ОКлА)*. Такие автоматы впервые описаны в работах С. Кауффмана под именем «булевой сети» (Boolean network) и предназначались для моделирования генетических процессов в биологии [30]. Затем ОКлА был предложен в работе [31], где был назван «неоднородным клеточным автоматом»[5]. ОКлА математически можно описать следующим образом.

Пусть задан ориентированный граф $G = (V, E)$, где $V = \{v_1, \ldots, v_N\}$ — множество вершин графа, E — множество дуг, а δ_i — полустепень захода для вершины v_i и при этом входящие в данную вершину дуги пронумерованы числами $1, \ldots, \delta_i$.

[5] в более поздних работах этот термин использовался в другом смысле

Будем считать, что с каждой вершиной v_i связана ячейка памяти, содержащая булеву переменную m_i, и булева функция $f_i(x_1, \ldots, x_{\delta_i})$ — локальная функция связи i-й вершины.

Обобщенный клеточный автомат это автономный автомат, его внутренним состоянием в момент времени t называется заполнение массива ячеек $(m_1(t), m_2(t), \ldots, m_N(t))$. Функция переходов задаёт отображение множества состояний в себя и определяет следующее состояние автомата. При этом заполнение ячеек памяти обобщенного КлА описывается уравнением:

$$m_i(t) \;=\; f_i\left(m_{n(i,1)}(t-1), m_{n(i,2)}(t-1), \ldots, m_{n(i,\delta_i)}(t-1)\right),$$

где $m_i(t)$ — состояние i-й ячейки памяти в момент времени t, а $n(i,j)$ — номер вершины, из которой исходит дуга, входящая в вершину i и имеющая номер j.

Однородный обобщенный клеточный автомат — это ОКлА, граф которого является регулярным по входу (т. е. число дуг, заходящих в вершину одинаково для всех вершин) и при этом локальная функция связи для всех ячеек одинакова. В противном случае обобщенный клеточный автомат будет называться *неоднородным*.

Вопросы обратимости для КлА конечного размера в принципе — всегда разрешимы, и основная задача состоит в нахождении приемлемых критериев для проверки обратимости, алгоритмов для реализации обратного преобразования и оценки сложности этих алгоритмов. В настоящее время эти вопросы весьма мало исследованы, результатов очень немного, а те, какие есть, — не внушают большого оптимизма на быстрое продвижение и скорые успехи в этом направлении. Так в работе [32] проводились попытки исследовать свойство обратимости двумерных КлА на множестве конфигураций, помещающихся в некоторый квадрат. Было установлено, что задача распознавания обратимости в этой постановке является $co-NP$-полной. В свою очередь, для ОКлА, как показано в [33], задача восстановления предыдущего состояния (а значит и начального заполнения) обобщенного клеточного автомата является NP-трудной, а в случае КлА с локальной функцией связи от 2 переменных, задача о существовании предыдущего состояния принадлежит классу P.

3.4. **Другие модели обратимых вычислений.** Известны и другие модели обратимых вычислений, укажем некоторые из них.

Броуновская машина (Ч. Беннетт). Идея — случайные блуждания, корректируемые с помощью незначительных количеств управляющей энергии [34].

Адиабатическое переключение (Р. Меркль). Адиабатическим называют процесс, протекающий без подвода и отвода теплоты. Идея — все переключения на гейтах

делаются при одинаковом напряжении и при этом энергия не тратится. При изменении напряжения между переключениями энергии тратится тем меньше, чем более плавно происходит изменение. В идеале будут почти нулевые выбросы тепла; см. [12]. Минусом является низкая скорость работы.

Модель стержней и пазов (Ч. Беннетт и Р. Ландауэр). Элемент управления — стержень с выступом, имеющий одну степень свободы. Взаимодействие моделируется блокировкой стержня при попадании выступов в одну область. Стержни размещены внутри матрицы с фиксированными каналами и снабжены эластичными фиксаторами, ограничивающими движение по оси и обеспечивающими дополнительное сдерживание стержня, который претерпел столкновение. Стержни не двигаются, пока их не толкнут, и только с толчком можно выяснить, заблокирован ли стержень. В каждый момент времени стержень можно толкнуть, освободить или оставить как есть [35].

4. Математика обратимой логики

4.1. **Модели схем из функциональных обратимых элементов.** Схема из функциональных элементов классически определяется как ориентированный граф без циклов с помеченными рёбрами и вершинами. Правильно сформированная обратимая схема — ациклическая комбинационная логическая схема, в которой все элементы обратимы и соединены друг с другом последовательно без ветвлений. В самой простой математической модели обратимых схем все элементы имеют одинаковое количество входов и выходов n. В ориентированном графе, описывающем такую обратимую схему, все вершины нумеруются от 1 до l и имеют ровно n занумерованных входов и выходов. При этом i-й выход m-й вершины, $m < l$, соединяется только с i-м входом $(m + 1)$-й вершины. Входами обратимой схемы являются входы первой вершины, а выходами — выходы l-й вершины. Величина l называется *сложностью схемы*.

Для описания модели схем из обратимых элементов, имеющих не n входов, а меньше, вводится понятие несущественных входов наравне с адресными и целевыми. Значения на несущественных входах, как следует из названия, не влияют на значение на целевом выходе и передаются на соответствующие выходы без изменений. Для перехода от этой модели к предыдущей потребуется в качестве функциональных элементов рассматривать все возможные схемы сложности 1, построенные на n проводниках.

Модель обратимых схем можно свести к модели функциональных схем с *ограниченной памятью*. В случае обратимой схемы с n входами каждой линии ставится в

соответствие один из n регистров памяти (ячеек памяти), хранящих результат вычислений на каждом шаге работы схемы. Входы и выходы элементов, подключенные к линиям схемы, считывают и записывают значения в соответствующий линии регистр памяти.

Перед началом работы в регистры памяти записываются начальные значения, в конце работы из них считывается результат. Для реализации биективного отображения на множестве двоичных векторов длины n необходимо как минимум n регистров памяти. Если требуемое отображение невозможно реализовать при помощи заданного семейства обратимых элементов на n регистрах памяти, но можно реализовать на $(n + q)$ регистрах памяти, то говорят, что схема реализует отображение с q дополнительными входами. В большинстве случаев перед началом работы в регистры памяти, соответствующие дополнительным входам, записывают нулевые значения, однако могут быть записаны и единичные значения.

Обратимый элемент по определению задаёт биективное отображение на множестве $\mathbb{Z}_2^n$ — множестве двоичных векторов длины n. Любое такое преобразование можно описать подстановкой на данном множестве. Следовательно, последовательное соединение обратимых элементов задаёт подстановку, равную произведению соответствующих подстановок. Отсюда следует очевидная связь между обратимыми схемами с n входами и подстановками из симметрической группы S_{2^n}. Так если взять элементы, обратные элементам обратимой схемы Σ, и соединить их в обратном порядке, то мы получим схему Σ_r, которая задает биективное отображение, обратное к отображению, задаваемому схемой Σ. Будем называть такую схему *зеркальной*. Если обратимая схема состоит только из самообратимых элементов, то обратное отображение реализует схема, состоящая из тех же самых элементов, но соединенных в обратном порядке.

4.2. **Функциональная полнота семейств обратимых элементов.** Семейство логических элементов, из которых строиться схема, называют *библиотекой*. Если в ней присутствуют элементы с переменным числом входов, выходов, изменяемых свойств, то такая библиотека называется параметрической. При этом конкретный элемент выбирается в зависимости от значения некоторого параметра. Применение параметрических библиотек значительно облегчает процесс создание логической схемы. Схему, построенную из элементов библиотеки $\mathscr{B}$, называют $\mathscr{B}$-схемой. Подстановка называется $\mathscr{B}$-конструируемой если она может быть реализована $\mathscr{B}$-схемой.

Множество всех подстановок, реализуемых $\mathscr{B}$-схемами с n входами и n выходами, является группой, обозначаемой $\langle\mathscr{B}\rangle_n$. Эта группа, в свою очередь, является подгруппой симметрической группы S_{2^n} (может быть совпадающей со всей S_{2^n}).

Подстановки, соответствующие элементам библиотеки $\mathscr{B}$, являются образующими элементами группы $\langle\mathscr{B}\rangle_n$. В связи с этим задачи построения схем из обратимых элементов, реализующих элементы группы подстановок, и получения оценок для их сложности сводятся к задачам нахождения длин элементов группы подстановок в заданной системе образующих, длины самой группы и мощностей её слоев. При этом естественно возникает вопрос функциональной полноты заданного семейства обратимых элементов: какие биективные отображения (какой класс отображений) могут быть реализованы и при каком количестве дополнительных входов.

Библиотека, образованную элементами NOT, CNOT, TG называют NCT-библиотекой [36]. Будем называть *N-конструируемой* (*C-конструируемой, T-конструируемой*) подстановку, построенную исключительно с помощью элементов NOT (соответственно CNOT, TG). В работе [37] установлены следующие результаты.

- Группа S_{2^n} содержит 2^n N-конструируемых подстановок, $\prod\limits_{i=0}^{n-1}\left(2^n-2^i\right)$ C-конструируемых подстановок, $\dfrac{1}{2}\,(2^n-n-1)!$ T-конструируемых подстановок. Все три указанных множества являются подгруппами в S_{2^n}.
- В схеме с $n>3$ входами каждый из элементов NCT-библиотеки задаёт чётную подстановку, как следствие и сама обратимая схема также реализует чётную подстановку.
- Для любой заданной подстановки на множестве двоичных векторов длины $n\leqslant 3$ существует реализующая её NCT-схема с n входами.
- Для любой заданной чётной подстановки на множестве двоичных векторов длины $n\geqslant 4$ существует реализующая её NCT-схема с n входами.
- Для любой заданной нечётной подстановки на множестве двоичных векторов длины $n\geqslant 4$ не существует реализующей её NCT-схемы с n входами, однако существует реализующая ее NCT-схема с $(n+1)$ входами (один дополнительный).

Зафиксируем набор библиотек $\mathscr{B}_1,\ldots,\mathscr{B}_k$ и построим $(\mathscr{B}_1|\ldots|\mathscr{B}_k)$-*схему*, присоединяя к выходу $\mathscr{B}_i$-схемы входы $\mathscr{B}_{i+1}$-схемы, $i=1,\ldots,k-1$. Входом построенной $(\mathscr{B}_1|\ldots|\mathscr{B}_k)$-схемы есть вход $\mathscr{B}_1$-схемы, а выходом — выход $\mathscr{B}_k$-схемы.

Каждая подстановка из S_{2^n} при $n=1,2,3$ и каждая четная подстановка при $n\geqslant 4$ может быть реализована некоторой (T|C|T|N)-схемой и, следовательно, является NCT-конструируемой [37].

Аналогично, для других семейств обратимых элементов можно установить их функциональную полноту. Пусть $\mathscr{B}$ — библиотека из обратимых логических элементов. Тогда $\mathscr{B}$ универсальна, если для любого n и любой подстановки $\pi \in S_{2^n}$ существует такое q, что некоторая $\mathscr{B}$-схема вычисляет π, используя q линий дополнительной (вспомогательной) памяти. Доказано, что для любой универсальной библиотеки $\mathscr{B}$ и достаточно большого n подстановки из знакопеременной группы A_{2^n} являются $\mathscr{B}$-конструируемыми, а подстановки из S_{2^n} реализуются с помощью не более чем одной дополнительной линии (вспомогательной памяти) [37].

Приведем примеры универсальных библиотек, имеющих прикладное значение для криптографии. Библиотека Тоффоли–подобных обратимых элементов $\sigma : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ для которых координатные функции $\sigma_i(x_1, \ldots, x_n)$, $i = 1, \ldots, n$, задаются системой

$$\begin{aligned} y_i &= x_i \oplus f\,(x_{i_1}, \ldots, x_{i_k}), \\ y_j &= x_j \ \text{ при } j \neq i, \end{aligned}$$

где $\oplus$ — операция сложения по $mod\,2$, а функция $f : \mathbb{Z}_2^k \to \mathbb{Z}_2$. В работе [38] доказывается, что при $n > 1$, $k = n - 1$ множество указанных элементов порождает группу S_{2^n}, а при $n \geqslant 4$ и фиксированном $2 \leqslant k \leqslant n - 2$ — группу A_{2^n}.

Другая интересная библиотека состоит из подстановок $\sigma : \mathbb{Z}_2^{2m} \to \mathbb{Z}_2^{2m}$ вида $\sigma(a, b) = (b,\ f(a, b))$, где $a, b \in \mathbb{Z}_2^m$, $f : \mathbb{Z}_2^m \times \mathbb{Z}_2^m \to \mathbb{Z}_2^m$. Для случая $f(a, b) = a \oplus \varphi(b)$, φ — произвольное отображение $\mathbb{Z}_2^m \to \mathbb{Z}_2^m$, $\oplus$ — операция покоординатного сложения по $mod\,2$. Доказано, что множество указанных подстановок порождает группу $A_{2^{2m}}$ [39].

4.3. **Схемы с дополнительными входами.** Считаем, что схема из обратимых элементов с n значимыми и q дополнительными входами реализует функцию $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, если подавая на значимые входы значение аргумента функции $\mathbf{x}$, подавая 0 на дополнительные входы, на m значимых выходах мы получаем значение $\mathbf{y}$ (значения, полученные на незначимых выходах, игнорируются). Если в конце работы схемы значение на незначимом выходе не равно входной константе, линия называется *мусорной*, а полученное значение — *вычислительным мусором*. Достаточно часто мусорными считаются все незначимые выходы. В общем, число основных (значимых) входов плюс число дополнительных входов должно равняться числу основных (значимых) выходов плюс число незначимых выходов, включая мусорные, см. рис. 10.

Рис. 10. Входы и выходы обратимой схемы

Чтобы реализовать необратимое отображение с помощью обратимых логических элементов необходимо сделать его обратимым, добавив дополнительные входы на которые подаются константы 0. Для реализации необратимой функции, в которой совпадающие выходные наборы встречаются до M раз, требуется как минимум $\lceil \log_2 M \rceil$ дополнительных линий [40]. В частности, для любого заданного отображения на множестве двоичных векторов длины n существует реализующая её обратимая схема, состоящая из элементов данного семейства и имеющая не более $2n$ входов (n дополнительных).

Поскольку произвольная схема с $n+q$ входами, построенная из обратимых элементов реализует некоторую подстановку на множестве двоичных векторов длины $n+q$, реализация этой схемой функции $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ может быть математически описана при помощи расширяющего отображения $\varphi_{n,n+q} : \mathbb{Z}_2^n \to \mathbb{Z}_2^{n+q}$ вида

$$\varphi_{n,n+q}\left(\langle x_1, \ldots, x_n\rangle\right) = \langle x_1, \ldots, x_n, 0, \ldots, 0\rangle$$

и редуцирующего отображения $\psi^{\pi}_{n+q,n} : \mathbb{Z}_2^{n+q} \to \mathbb{Z}_2^n$ вида

$$\psi^{\pi}_{n+q,m}\left(\langle x_1, \ldots, x_{n+q}\rangle\right) = \langle x_{\pi(1)}, \ldots, x_{\pi(m)}\rangle,$$

где π — некоторая подстановка на множестве $\mathbb{Z}_{n+q}$.

Тогда обратимая схема с $(n+q) \geqslant m$ входами, задающая подстановку $g : \mathbb{Z}_2^{n+q} \to \mathbb{Z}_2^{n+q}$, реализует отображение $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ с использованием $q \geqslant 0$ дополнительных входов (дополнительной памяти), если существует такая подстановка $\pi \in \mathbb{Z}_{n+q}$, что

$$\psi^{\pi}_{n+q,m}\left(g\left(\varphi_{n,n+q}(\mathbf{x})\right)\right) = f(\mathbf{x}),$$

где $\mathbf{x} \in \mathbb{Z}_2^n$, $f(\mathbf{x}) \in \mathbb{Z}_2^m$.

В основополагающей работе Ч. Беннетта была предложена общая конструкция для обратимого вычисления произвольной (обратимой или необратимой) функции [5]. Конструкция Беннетта схематично представлена на рис. 11. Здесь $\mathbf{f}$ — схема

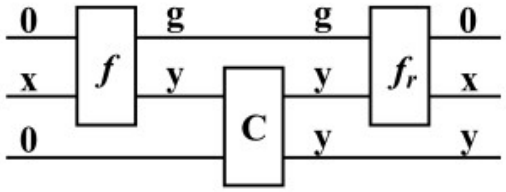

Рис. 11. Конструкция Беннетта

из обратимых элементов, реализующая функцию $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, $\mathbf{x} \in \mathbb{Z}_2^n$ — вход функции, $\mathbf{y} \in \mathbb{Z}_2^m$ — ее выход, $\mathbf{g}$ — вычислительный мусор, $\mathbf{f}_r$ — зеркальная схема. В конструкции используется подсхема копирования C, которая может быть построена из элементов CNOT (рис. 12).

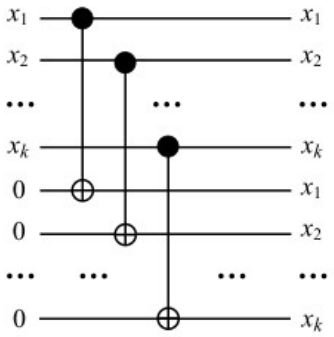

Рис. 12. Схема копирования

В развитие этой идеи Т. Тоффоли [36] была предложена общая конструкция NCT-схемы для вычисления произвольной (обратимой или необратимой) функции (см. ниже п. 5.1).

С помощью любой библиотеки, содержащей универсальный элемент, можно реализовать обратимой схемой любое заданное отображение на множестве двоичных векторов длины n, однако количество необходимых для этого дополнительных входов схемы может кардинально отличаться для различных библиотек: оно может быть либо константным, либо зависящим от количества элементов в схеме. От количества дополнительных входов обратимой схемы зависит и класс отображений на множестве двоичных векторов, реализуемых при помощи заданного семейства обратимых элементов, даже если оно содержит универсальный элемент. К примеру, гейт Фредкина является универсальным, однако при помощи обратимой схемы без дополнительных входов, состоящей только из элементов данного типа, невозможно реализовать отображение, изменяющее вес Хэмминга двоичного вектора (количество единичных

координат в нем). Помимо этого от количества дополнительных входов обратимой схемы существенно зависит и её сложность (будет рассмотрено в части II статьи).

5. Обратимые языки программирования

5.1. **Обратимые надстройки над функциями.** Для обеспечения обратимости при вычислениях используют функции, инъективные по первому аргументу, которые получают из обычных функций, применяя процедуру обратимой надстройки.

Определение. Бинарную функцию $A \times B \xrightarrow{\odot} C$ называют *инъективной по первому аргументу*, если

$$a_1 \odot b = a_2 \odot b \;\Rightarrow\; a_1 = a_2.$$

Пример: $n + 3 = m + 3 \;\Rightarrow\; n = m$; антипример: $n \times 0 = m \times 0 \;\not\Rightarrow\; n = m$.

Теорема. *Пусть $\odot$ — бинарная функция, инъективная по первому аргументу. Тогда существует функция $C \times B \xrightarrow{\oslash} A$ такая, что*

$$(a \odot b) \oslash b \;=\; a.$$

Неформально говорят, что операция $\oslash$ «обратна» к операции $\odot$. Примеры пар $(\odot, \oslash)$: для чисел — $(+, -)$, при работе с ненулевыми числами — $(\times, \div)$, для булевых векторов — $(\oplus, \oplus)$.

Обратимая надстройка над унарной функцией $f(y)$ есть бинарная функция $g(x, y) \;=\; (\, x \odot f(y),\, y\,)$, такая, что операция $\odot$ обратима по первому аргументу.

$$\begin{array}{lcl} x & \longrightarrow & x \odot f(y) \\ y & \longrightarrow & y \end{array} \qquad\qquad \begin{array}{lcl} (x \odot f(y)) \oslash f(y) = x & \longleftarrow & x \odot f(y) \\ y & \longleftarrow & y \end{array}$$

Рис. 13. Схема обратимого вычисления функции $f(y)$

Пример: если $g(x, y) \;=\; (x \,+\, f(y),\, y)$, то $g^{-1}(x, y) \;=\; (x \,-\, f(y),\, y)$. Пусть $f(y) = \sin\, y$, x — произвольное, тогда

$$\begin{aligned} g(x, y) \;&= (x + \sin\, y,\, y) \\ g^{-1}(x, y) \;&= (x - \sin\, y,\, y) \\ g^{-1}(x + \sin\, y,\, y) \;&= (x + \sin\, y - \sin\, y,\, y) \;=\; (x,\, y) \end{aligned}$$

Обратимые надстройки существуют у всех функций. Обратная детерминированность программы позволяет проводить вычисления программы в обратном порядке («инверсность»).

Двунаправленные программы называют *линзами* (lens) [41].

5.2. **Язык Janus.** Язык Janus — первый обратимый язык программирования, интерпретатор был построен Т. Йокоямой и Р. Глюком [42]. Преобразователь и интерпретатор Janus свободно доступны[6]. Janus также реализован на языке Prolog.

В языке Janus все функции заменены обратимыми надстройками, в циклах и условиях возвращается информация о пройденном пути («история» Ч. Беннетта), параметры процедур передаются только по ссылке, глобальных переменных нет, при инициализации процедур все переменные и элементы массива обнуляются, а стеки опустошаются.

Janus — императивный язык программирования: исполнение программы состоит в последовательном выполнении команд. Программа *prog* на языке Janus состоит из основной процедуры p_{main}, за которой следует последовательность d_proc^* определений процедур p:

$$prog ::= p_{main}\, d\_proc^*$$

Основная процедура p_{main} не имеет параметров и состоит из указания типов переменных и оператора. Определение процедуры имеет вид

`procedure` <имя>((<тип скаляр/массив><имя>)*)<тело процедуры>.

Имеются следующие типы переменных данных: скаляр (32-разрядное неотрицательное целое), одномерный массив целых $x[c]$ и стек целых. Логические значения суть 'целое ненулевое' = *true*, 'целое нулевое' = *false*. Массивы индексируются целыми числами, начиная с нуля.

Управляющие структуры языка Janus традиционны. Это операторы (*statement*) присваивания (`+=`, `-=`, `^`), обмена двух значений (`<=>`), условный (*if* e_1 *then* s_1 *else* s_2 *fi* e_2) и цикла (*from* e_1 *do* s_1 *until* e_2).

Логика их работы проиллюстрирована на рис. 14 и 15 [41].

В Janus'е также имеются операторы работы со стеком — поместить данное значение в стек извлечь его из стека (*push, pop*), прямого (*call*) и обратного (*uncall*) вызовов процедуры.

Пример: процедура вычисления чисел Фибоначчи [43]. Вычисление n-го числа Фибоначчи в Janus программируется с запоминанием n-го и $(n-1)$-го чисел, что делает неинъективную функцию вычисления чисел инъективной. Программа `fib` состоит только из обратимых операций (`+=`, `-=`) и обратимого условного оператора с критериями входа и выхода.

```
procedure fib(int x1, int x2, int n)
  if n=0 then x1 += 1
              x2 += 1
```

[6] http://topps.diku.dk/pirc/janus-playground/

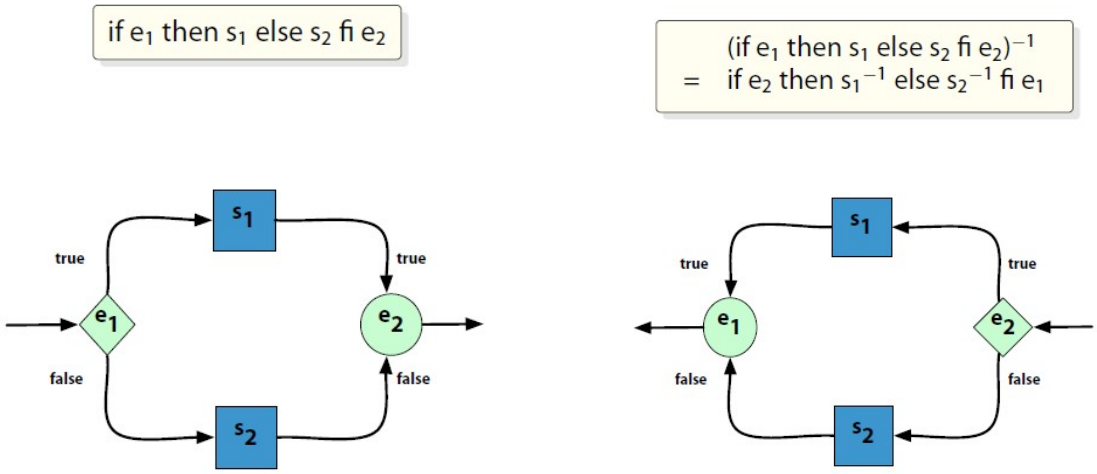

Рис. 14. Прямое а) и обратное б) вычисление условия.

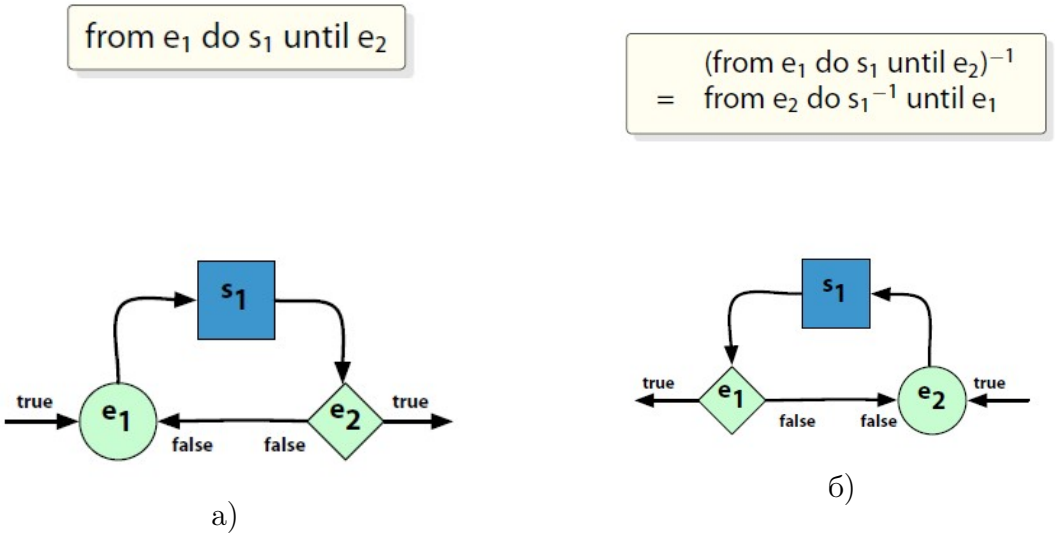

Рис. 15. Прямое а) и обратное б) вычисление цикла.

```
        else n -= 1
             call fib(x1,x2,n)
             x1 += x2
             x1 <=> x2    // swap
fi x1=x2
```

При прямом вызове процедуры, положив $n = 4$, получим $x1 = 5$, $x2 = 8$, т. е. 5-е и 6-е числа Фибоначчи и $n = 0$:

```
procedure fib_fwd(int x1,int x2,int n)
    n += 4
    call fib(x1,x2,n) // forward execution
```

При обратном вызове процедуры, положив $x1 = 5$, $x2 = 8$, получим $x1 = x2 = 0$ и $n = 4$ — исходные значения парметров:

```
procedure fib_bwd(int x1,int x2,int n)
    x1 += 5
    x2 += 8
    uncall fib(x1,x2,n) // backward execution
```

Известны и другие обратимые языки: логический язык CRL, PsiLisp (Ψ-LISP, Baker (1992)), R (Frank (1999)), Inv (Mu, Hu, and Takeichi (2004)); все ссылки можно найти в [12, 44].

При функциональном программировании каждая процедура должна проектироваться одновременно с процедурой отката.

Обратимое программирование тесно связано с алгебраическим (функциональным). Некоторые результаты по алгебрам программ в связи с обратимыми вычислениями можно найти в работах Н. Н. Непейводы [14].

Список литературы

1. Landauer, R. (1961) Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development.* 5. p. 183–191.

2. Berut, A. et al. (2012) Experimental verification of Landauer's principle linking information and thermodynamics. *Nature.* 483. p. 187–189.

3. Jun, Y. & Gavrilov, M., & Bechhoefer, J. (2014) High-Precision Test of Landauer's Principle in a Feedback Trap. *Phys. Rev. Lett..* arXiv:1408.5089 (113). p. 71-82.

4. DeBenedictis, E. (2004) Will Moore's Law be Sufficient?. *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing.* 1. p. 45–57.

5. Bennett C. H. (1973) Logical reversibility of computation. *IBM J. Res. Develop.* (17 (6)). p. 525–532.

6. Fredkin, E. & Toffoli, T. (1982) Conservative Logic. *International Journal of Theoretical Physics.* (21(3/4)). p. 632–644.

7. Бобков, С. Г. Высокопроизводительные вычислительные системы. — М.: НИИСИ РАН, 2014. — 296 с.

   Bobkov, S. G. (2014) *High performance computing systems.* Moscow: NIISI RAN.

8. Saeedi, M. & Markov, I. L. Synthesis and Optimization of Reversible Circuits — A Survey // arxiv.org [Электронный ресурс]. — Режим доступа: https://arxiv.org/pdf/1110.2574, свободный — (01.09.2019)

9. Маймистов, А. И. Обратимые логические элементы — новая область применения оптических солитонов // Квантовая электроника. — М: ФГБУН Физический институт им. П.Н. Лебедева РАН, 1995. — Т. 22 (10). — С. 1044–1048.

   Maimistov, A. I. (1995) Reversible logic elements — a new field of application of optical solitons. *Quantum electronics.* 22 (10). p. 1044–1048.

10. Merkle, R. C. (1993) Reversible Electronic Logic Using Switches. *Nanotechnology.* 4. p. 21–40.

11. Lecerf, Y. (1963) Machines de turing réversibles. *C. R. Acad. Sci. Paris.* 257. p. 2597–2600.

12. Непейвода, Н. Н. & Непейвода, А. Н. Реверсивные вычисления // www.botik.ru [Электронный ресурс]. — Режим доступа: http://www.botik.ru/cprc/bichkova/seminarICPU/materials/19032012/mat19032012.pdf, свободный — (01.09.2019)

13. Непейвода, Н. Н. Прикладная логика / Непейвода, Н. Н.. — Новосибирск: Изд-во Новосибирского университета, 2000. — 521 с.

14. Непейвода, Н. Н. Алгебры как альтернатива численному параллелизму // 2012.nscf.ru [Электронный ресурс]. — Режим доступа: http://2012.nscf.ru/Tesis/Nepeivoda.pdf.

15. Тоффоли, Т., Марголус, Н. Машины клеточных автоматов / Т. Марголус, Тоффоли Н. — М.: Мир, 1991. — 280 с.

    Toffoli, T. & Margolus, N. (1987) *Cellular automata machines: A new environment for modeling.* Cambridge, Mass.: MIT Press.

16. Richardson, D. (1972) Tesselations with local transformations. *Journal of Computer and System Sciences.* 6. p. 373–388.

17. Amoroso, S. & Patt, Y. N. (1972) Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *J. Comput. System Sci.*. 6 (5). p. 448–464.

18. Sutner, K. (1991) De Bruijn graphs and linear cellular automata. *Complex Systems*. 5 (1). p. 19–30.

19. Sutner, K. Linear Cellular Automata and de Bruijn Automata // Cellular Automata. Mathematics and Its Applications / Delorme M., Mazoyer J. (eds). — Springer, Dordrecht, 1998. — 460. — С. 303–319.

20. Culik, K. (1987) On invertible cellular automata. *Complex Systems*. 1 (6). p. 1035–1044.

21. Hillman, D. (1991) The structure of reversible one-dimensional cellular automata. *Physica D: Nonlinear Phenomena*. 52 (2–3). p. 277–292.

22. Kari, J. (1990) Reversibility of 2D cellular automata is undecidable. *Physica D*. 45. p. 379–385.

23. Kari, J. (1994) Reversibility and surjectivity problems of cellular automata. *Journal of Computer and System Science*. 48 (1). p. 149–182.

24. Кучеренко, И. В. О разрешимости обратимости клеточных автоматов // Интеллектуальные системы. — М.: МГУ имени М. В. Ломоносова, Механико-математический факультет, кафедра Математической теории интеллектуальных систем, 2004. — Т. 8, № 1–4. — С. 465–482.

25. Кучеренко, И. В. Обратимые клеточные автоматы // Дисс... канд. физ.-мат. наук. — М., 2012. — 147 с.

    Kucherenko, I. V. (2012) *Reversible Cellular Automata. Thesis ... PhD (Phys&Math)*. Moscow.

26. Schiff, J. L. (2007) *Cellular automata. A Discrete View of the World*. A John Wiley & Sons Inc., Publication. University of Auckland.

27. Margolus, N. (1984) Physics-like models of computation. *Physica D: Nonlinear Phenomena*. 10. p. 81–95.

28. Vichniac, G. (1984) Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena*. 10. p. 96–115.

29. Wolfram, S. (1984) Cellular Automata as Models of Complexity. *Nature*. 311. p. 419–424.

30. Kauffman, S. A. (1969) Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.* 22. p. 437–467.

31. Сухинин, Б. М. Разработка генераторов псевдослучайных двоичных последовательностей на основе клеточных автоматов // НАУКА И ОБРАЗОВАНИЕ: Научное издание МГТУ им. Н. Э. Баумана. — М.: Национальный Электронно–Информационный Консорциум, 2010. — № 9. — С. 8.

Sukhinin, B. M. (2010) Development of pseudorandom binary sequence generators based on cellular automata. *SCIENCE AND EDUCATION: Scientific publication of MSTU named after N. E. Bauman.* № 9. p. 8.

32. Durand, B. (1994) Inversion of 2D cellular automata: some complexity results. *Theoretical Computer Science.* 134 (2). p. 387–401.

33. Ключарёв, П. Г. *NP*-трудность задачи о восстановлении предыдущего состояния обобщенного клеточного автомата // НАУКА И ОБРАЗОВАНИЕ: Научное издание МГТУ им. Н. Э. Баумана. — М.: Национальный Электронно–Информационный Консорциум, 2012. — № 1. — С. 11.

Klucharev, P. G. (2012) *NP*-difficulty of the problem of restoring the previous state of a generalized cellular automaton. *SCIENCE AND EDUCATION: Scientific publication of MSTU named after N. E. Bauman.* 1. p. 11.

34. Bennett C. H. (1989) Time/space trade-offs for reversible computation. *SIAM J. Comput.* 18 (4). p. 766–776.

35. Bennett, C. H. & Landauer, R. (1985) The Fundamental Physical Limits of Computation. *Scientific American.* July. p. 48–56.

36. Toffoli T. Revercible Computing // Automata, Languages and Programming / de Bakker (ed.). — Springer-Verlag, 1980. — С. 632–644.

37. Shende, V. V. & Prasad, A. K. & Markov, I. L. & Hayes, J. P. (2003) Synthesis of Reversible Logic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.* 22 (6). p. 710–722.

38. Coppersmith, D. & Grossman, E. (1975) Generators for certain altemating groups with applications to cryptography. *SIAM J. Appl. Math.* 29 (4). p. 624–627.

39. Even, S. & Goldreich, O. (1983) DES-like functions can generate the alternating group. *IEEE Trans. Inform. Theory.* IT-29 (6). p. 863–865.

40. Маслов, Д. Роль обратимости в компьютерных технологиях будущего // [Электронный ресурс]. — Режим доступа:
http://www.kinnet.ru/cterra/538/33163.html, свободный — (01.09.2019).

Maslov, D. (2004) *The role of reversibility in computer technology of the future.* [Online] Available from:
http://www.kinnet.ru/cterra/538/33163.html. [Accessed: 01.09.2019, free].

41. Foster, J. N. (2010) *Bidirectional programming languages Technical Report MS-CIS-10-08.* Department of Computer & Information Science University of Pennsylvania — March 13.

42. Yokoyama, T. & Gluck, K. (2007) A reversible programming language and its invertible self-interpreter. *Proceedings of the 2007 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation (PEPM '07), ACM, New York, NY, USA.* 1. p. 144–153.

43. Yokoyama, T. (2010) Reversible Computation and Reversible Programming Languages. *Electronic Notes in Theoretical Computer Science.* 253 (6). p. 71–81.

44. Kalyan, S. & Perumalla, P. (2014) *Introduction to Reversible Computing.* CRC Press.