Московский Государственный Университет
имени М.В. Ломоносова
Российская Академия Наук
Международная Академия Технологических Наук
Российская Академия Естественных Наук

Интеллектуальные Системы.

Теория и приложения

ТОМ **23** ВЫПУСК 3 * **2018**

МОСКВА

ISSN 2411–4448

УДК 519.95; 007:159.955
ББК 32.81

Издается с 1996 г.*

В журнале «Интеллектуальные системы. Теория и приложения» публикуются научные достижения в области теории и приложений интеллектуальных систем, новых информационных технологий и компьютерных наук.

Издание журнала осуществляется под эгидой МГУ имени М. В. Ломоносова, Научного Совета по комплексной проблеме «Кибернетика» РАН, Отделения «Математическое моделирование технологических процессов» МАТН, Секции «Информатики и кибернетики» РАЕН.

Учредитель журнала: ООО «Интеллектуальные системы».

Журнал входит в список изданий, включенных ВАК РФ в реестр публикаций материалов по кандидатским и докторским диссертациям по математике и механике.

Спонсором издания является:

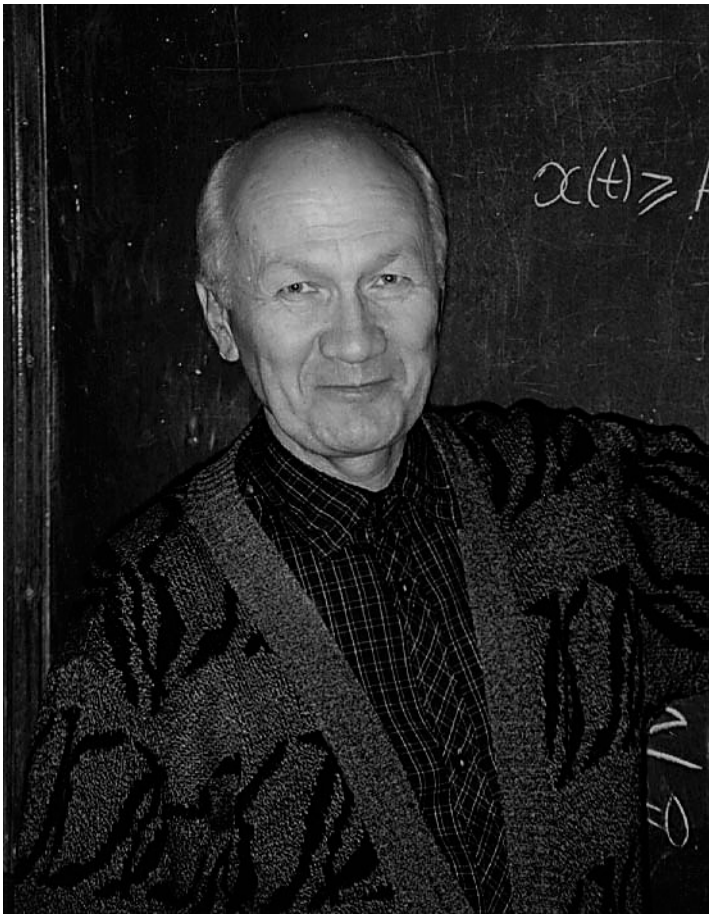
ООО «Два Облака»
Разработка корпоративных информационных систем
http://www.dvaoblaka.ru

Индекс подписки на журнал: 64559 в каталоге НТИ «Роспечать».

Адрес редакции: 119991, Москва, ГСП-1, Ленинские Горы, д. 1, механико-математический факультет, комн. 12-01.
Адрес издателя: 115230, Россия, Москва, Хлебозаводский проезд, д. 7, стр. 9, офис 9. Тел. +7 (495) 939-46-37, e-mail: mail@intsysjournal.org

*) Прежнее название журнала: «Интеллектуальные системы».

ОГЛАВЛЕНИЕ



15.12.1936-01.08.2018

Памяти Юрия Николаевича Черемных

Редакционная коллегия журнала «Интеллектуальные системы. Теория и приложения», кафедра математической теории интеллектуальных систем (МаТИС) механико-математического факультета МГУ имени М. В. Ломоносова сообщают о печальном известии – закончил свой жизненный путь наш коллега, друг и товарищ – замечательный русский ученый, математик и экономист – Юрий Николаевич Черемных. . .

Всю свою жизнь Юрий Николаевич связал с Московским университетом – завершив обучение на механико-математическом факультете МГУ, он посвятил свой математический талант исследованиям в области экономики, в частности исследованиям в области динамических моделей в экономике. Эти исследования были обобщены в 1983 г. успешной

защитой докторской диссертации «Экономико-математический анализ динамических народно-хозяйственных моделей» по специальности математические и инструментальные методы экономики.

Юрий Николаевич – один из зачинателей экономико-математических исследований в нашей стране. Ему принадлежит важная роль в становлении этого направления на экономическом факультете МГУ, в создании кафедры математических методов . Он принял участие в разработке основных курсов кафедры, в выборе правильной стратегии развития преподавания и научных исследований. Являясь профессором кафедры «Математические методы анализа экономики» экономического факультета, он поддерживал тесный контакт с кафедрой МаТИС механико-математического факультета МГУ. С сотрудниками кафедры МаТИС Юрия Николаевича связывали не только формальные отношения (с 1994 г. он являлся по совместительству профессором кафедры МаТИС), но и чисто дружеские и товарищеские отношения. Он являлся непременным участником кафедральных событий, принимал активное участие в деятельности кафедры, вел большую учебно-методическую работу со студентами и аспирантами кафедры МаТИС, выбравшими специализацию в области математического моделирования экономических процессов. Он был мощным связующим звеном в обучении математическим методам студентов-экономистов и развитии исследований экономических процессов студентами-математиками.

Широкая эрудиция , талант ученого и педагога-исследователя позволяли Юрию Николаевичу успешно работать в этих важных областях научного знания. Он является автором свыше 120 печатных работ, среди которых более двух десятков учебников и монографий, под его руководством защищены 12 кандидатских работ и несколько десятков дипломных и выпускных работ студентов экономического и механико-математического факультетов, студентов и магистрантов филиалов МГУ имени М. В. Ломоносова в ближнем зарубежье, он был автором и лектором различных курсов и спецкурсов по математике и экономико-математическим методам.

Признанием научных и педагогических заслуг Юрия Николаевича стало присвоение ему звания профессора (1986 г.), заслуженного профессора МГУ (2001 г.), избрание его действительным членом Международной Академии менеджмента (1989 г.) и действительным членом Академии технологических наук РФ (2012 г.). Он вел большую научно-организационную работу на экономическом факультет МГУ имени М. В. Ломоносова: с 1983 по 1987 гг. был заместителем декана по научной

работе, был членом Программного комитета конференции «Интеллектуальные системы и компьютерные науки», регулярно проводимой кафедрой МаТИС с 1998 года, выступал на ней с пленарными докладами по математическим методам моделирования экономики.

Юрий Николаевич был постоянным членом нашей редколлегии с момента основания журнала в 1996 г., курируя направление, связанное с применением математических методов в экономике. Он был автором ряда публикаций в нашем журнале и, кроме того, проявлял большую заботу о публикации в журнале статей научной молодежи.

Трудно и грустно писать об Юрии Николаевиче в прошедшем времени и много можно было бы еще хорошего и доброго сказать об ушедшем от нас светлом человеке – профессоре Юрии Николаевиче Черемных...

Пусть теплые и светлые воспоминания о нем останутся в наших сердцах...

Скорбим вместе с родными и близкими Юрия Николаевича, его коллегами, друзьями, со всеми, кто знал этого доброго и светлого человека.

Вечная ему память...

Гл. редактор журнала «Интеллектуальные системы. Теория и приложения» академик д.ф.-м.н., профессор Кудрявцев В.Б.
Редакционная коллегия журнала «Интеллектуальные системы. Теория и приложения»

Часть 1.
Общие проблемы теории интеллектуальных систем

Некоторые математические модели упругопластических процессов сложного нагружения

Молодцов И.Н., Бабаева Д.О.

В рамках теории упругопластических процессов А.А.Ильюшина [1], в [3] для математического моделирования процессов сложного нагружения использовано квазилинейное определяющее уравнение с тремя функционалами состояния. Калибровка определяющих функционалов там проведена с использованием экспериментальных результатов [4] (Р.А.Васин и др.) по трехмерным винтовым траекториям деформаций. Выяснилось, что отклик на винтовую траекторию деформации принимает, по исчерпанию некоторого следа запаздывания, вполне определенную форму предельного режима. Поэтому для произвольных трехмерных процессов деформации в [2] предлагалось последовательно аппроксимировать траектории деформации отрезками винтовых линий, на которых вычислять определяющие функционалы по уравнениям предельных режимов. Тогда на траекториях указанного вида реализуется соответствие геометрии траектории деформаций форме отклика. Это соответствие по А.А.Ильюшину назовем теоремой изоморфизма, уточняя для процессов высокой размерности уравнения самих винтовых сплайнов в пространстве деформаций и форм отклика. Принятый в [2] алгоритм относился исключительно к трехмерным траекториям и базировался на использовании смешанного пространственного базиса, включающего, помимо традиционных в определяющих соотношениях векторов (направляющих векторов напряжений и скоростей деформаций), еще и сам направляющий вектор деформаций. Здесь рассматриваются варианты модификации общей теории, годные для описания произвольных процессов нагружения с траекториями деформаций любой размерности. В качестве репера во всех новых теориях использованы направляющий вектор напряжений и векторы, построенные на основе векторов естественного сопровождающего репера Френе. Поскольку далее рассматриваются процессы деформации высокой размерности, то растет чис-

ло определяющих функционалов и несколько усложняются методы их идентификации.

Ключевые слова: пластичность, пластические деформации, сложное нагружение, определяющие соотношения, идентификация функционалов, теорема изоморфизма

Основная идея.

Рассматриваются процессы сложного упругопластического нагружения материалов. Для описания их свойств вводятся пятимерные эвклидовы пространства векторов–девиаторов напряжений и деформаций. Процесс деформации характеризуется в пространстве пятимерной кривой, в каждой точке которой изображается реакция или отклик материала на деформацию в виде вектора напряжений. Пусть $\overline{\sigma}$ и $\overline{\varepsilon}$ обозначают согласованные пары пятимерных векторов напряжений и деформаций, построенные на базе девиаторов соответствующих тензоров. В [2] векторы напряжений и деформаций связываются между собой определяющими уравнениями:

$$\frac{d\overline{\sigma}}{ds} = Q\frac{d\overline{\varepsilon}}{ds} + (P-Q)\left(\frac{d\overline{\varepsilon}}{ds}, \overline{n}_\sigma\right) + (N-Q)\left(\frac{d\overline{\varepsilon}}{ds}, \overline{n}'_\varepsilon\right)\overline{n}'_\varepsilon, \qquad (1)$$

$$\overline{n}'_\varepsilon \equiv \frac{\overline{n}_\varepsilon - (\overline{n}_\sigma, \overline{n}_\varepsilon)\overline{n}_\sigma}{\sqrt{\Psi}}, \Psi \equiv 1 - (\overline{n}_\sigma, \overline{n}_\varepsilon)^2,$$

в которых P, N, Q - функционалы процесса деформаций, s - длина дуги траектории деформаций. Другой формой (1) является эквивалентное представление скорости изменения напряжений в ортонормированном репере из трех векторов, связанных с направляющими векторами напряжений, деформаций и скоростей деформаций:

$$\frac{d\overline{\sigma}}{ds} = Q\left(\frac{d\overline{\varepsilon}}{ds} - \overline{n}_\sigma\left(\overline{n}_\sigma, \frac{d\overline{\varepsilon}}{ds}\right) - \overline{n}'_\varepsilon\left(\overline{n}'_\varepsilon, \frac{d\overline{\varepsilon}}{ds}\right)\right) + P\overline{n}_\sigma\left(\overline{n}_\sigma, \frac{d\overline{\varepsilon}}{ds}\right) + N\overline{n}'_\varepsilon\left(\overline{n}'_\varepsilon, \frac{d\overline{\varepsilon}}{ds}\right).$$

Рассмотрение уравнения (1) в пятимерном пространстве, где

$$\overline{n}_\sigma = \cos\theta_1\overline{n}_1 - \sin\theta_1\left(\cos\theta_2\overline{n}_2 - \sin\theta_2\left(\cos\theta_3\overline{n}_3 - \sin\theta_3\left(\cos\theta_4\overline{n}_4 - \sin\theta_4\overline{n}_5\right)\right)\right),$$

$$\overline{n}_\varepsilon = \cos\varphi_1\overline{n}_1 - \sin\varphi_1\left(\cos\varphi_2\overline{n}_2 - \sin\varphi_2\left(\cos\varphi_3\overline{n}_3 - \sin\varphi_3\left(\cos\varphi_4\overline{n}_4 - \sin\varphi_4\overline{n}_5\right)\right)\right),$$

приводит к векторному определяющему уравнению

$$\dot{\overline{n}_\sigma} = \frac{Q}{\sigma}\left(\overline{n}_1 - \cos\theta_1\overline{n}_\sigma\right) + \frac{N_1}{\sigma}\sin\theta_1\left(\overline{n}_\varepsilon - (\overline{n}_\sigma, \overline{n}_\varepsilon)\overline{n}_\sigma\right). \qquad (2)$$

Это следует из того, что эквивалентная (2) система уравнений для углов в представлении направляющего вектора напряжений в репере Френе:

$$\left\{\begin{array}{l} \dot{\theta_1} = \kappa_1 \cos\theta_2 - \frac{Q}{\sigma}\sin\theta_1 - \frac{N_1}{\sigma}\sin\theta_1\Delta_{14}, \\ \dot{\theta_2} = \kappa_2 \cos\theta_3 - \kappa_1\frac{\cos\theta_1}{\sin\theta_1}\sin\theta_2 - \frac{N_1}{\sigma}\sin\varphi_1\Delta_{24}, \\ \dot{\theta_3} = \kappa_3 \cos\theta_4 - \kappa_2\frac{\cos\theta_2}{\sin\theta_2}\sin\theta_3 - \frac{N_1}{\sigma}\sin\varphi_1\sin\varphi_2\Delta_{34}, \\ \dot{\theta_4} = \kappa_4 - \kappa_3\frac{\cos\theta_3}{\sin\theta_3}\sin\theta_4 - \frac{N_1}{\sigma}\sin\varphi_1\sin\varphi_2\sin\varphi_3\Delta_{44}, \end{array}\right. \tag{3}$$

содержит в качестве определяющих функционалов именно Q и N_1. Здесь и далее точка над величиной обозначает производную величины по длине дуги траектории деформаций, а также приняты обозначения:

$$\Delta_{14} \equiv \sin\theta_1\cos\varphi_1 - \cos\theta_1\sin\varphi_1\,(\cos\theta_2\cos\varphi_2 + \sin\theta_2\sin\varphi_2\,(\cos\theta_3\cos\varphi_3 +$$

$$+\sin\theta_3\sin\varphi_3\,(\cos\theta_4\cos\varphi_4 + \sin\theta_4\sin\varphi_4)))$$

$$\Delta_{24} \equiv \sin\theta_2\cos\varphi_2 - \cos\theta_2\sin\varphi_2\,(\cos\theta_3\cos\varphi_3 +$$

$$+\sin\theta_3\sin\varphi_3\,(\cos\theta_4\cos\varphi_4 + \sin\theta_4\sin\varphi_4))$$

$$\Delta_{34} \equiv \sin\theta_3\cos\varphi_3 - \cos\theta_3\sin\varphi_3\,(\cos\theta_4\cos\varphi_4 + \sin\theta_4\sin\varphi_4)$$

$$\Delta_{44} \equiv \sin\theta_4\cos\varphi_4 - \cos\theta_4\sin\varphi_4$$

Система уравнений (3) определяет смысл функционалов Q и N_1, как функционалов, регулирующих скорости изменения первого и второго углов. На примере уравнения (1), в [3] для трехмерных траекторий деформации проведена калибровка определяющих функционалов и предложен вариант термомеханики упругопластических процессов. Изложенная там теория вполне подходит для описания упругопластических процессов и хорошо соответствует имеющимся экспериментам. При усложнении теории требуется увеличение числа векторов основного репера в ответ на увеличение размерности траектории процесса. В самом общем случае максимальное число определяющих функционалов может достигать пяти. Для выполнения свойства объективности определяющих соотношений далее исключаются из рассмотрения реперы, прямо содержащие векторы деформации. Тогда вполне естественной выглядит конструкция ортонормированных реперов из направляющего вектора напряжений и векторов, построенных на базе векторов пятимерного репера Френе.

Новые определяющие реперы и уравнения.

1. В пятимерном пространстве выбираем базовую систему векторов $\overline{n}_\sigma, \overline{n}_1, \overline{n}_2, \overline{n}_3, \overline{n}_4$ и строим на ее основе ортонормированный репер:

$$\overline{n}_\sigma, \overline{n}'_1 \equiv \frac{\overline{n}_1 - \cos\theta_1 \overline{n}_\sigma}{\sin\theta_1},$$

$$\overline{n}'_2 \equiv \frac{\overline{n}_2 - (\overline{n}_2, \overline{n}_\sigma)\overline{n}_\sigma - (\overline{n}_2, \overline{n}'_1)\ \overline{n}'_1}{\sin\theta_2},$$

$$\overline{n}'_3 \equiv \frac{\overline{n}_3 - (\overline{n}_3, \overline{n}_\sigma)\overline{n}_\sigma - (\overline{n}_3, \overline{n}'_1)\ \overline{n}'_1 - (\overline{n}_3, \overline{n}'_2)\ \overline{n}'_2}{\sin\theta_3},$$

$$\overline{n}'_4 \equiv \frac{\overline{n}_4 - (\overline{n}_4, \overline{n}_\sigma)\overline{n}_\sigma - (\overline{n}_4, \overline{n}'_1)\ \overline{n}'_1 - (\overline{n}_4, \overline{n}'_2)\ \overline{n}'_2 - (\overline{n}_4, \overline{n}'_3)\ \overline{n}'_3}{\sin\theta_4}. \qquad (4)$$

По аналогии с (1) записываем определяющее уравнение в виде:

$$\dot{\overline{n}_\sigma} = \frac{Q}{\sigma}\sin\theta_1 \overline{n}'_1 + \frac{F}{\sigma}\sin\theta_2 \overline{n}'_2 + \frac{U}{\sigma}\sin\theta_3 \overline{n}'_3 + \frac{W}{\sigma}\sin\theta_4 \overline{n}'_4. \qquad (5)$$

с пятью определяющими функционалами Q, F, U, W и скрытым в (5) функционалом P, отвечающим за скалярные свойства материала. Эквивалентная уравнениям (5) система уравнений для углов:

$$\begin{cases} \dot{\theta_1} = \kappa_1 \cos\theta_2 - \frac{Q}{\sigma}\sin\theta_1, \\ \dot{\theta_2} = \kappa_2 \cos\theta_3 - \kappa_1 \frac{\cos\theta_1}{\sin\theta_1}\sin\theta_2 + \frac{F}{\sigma}\frac{\sin\theta_2}{\sin\theta_1}, \\ \dot{\theta_3} = \kappa_3 \cos\theta_4 - \kappa_2 \frac{\cos\theta_2}{\sin\theta_2}\sin\theta_3 - \frac{U}{\sigma}\frac{\sin\theta_3}{\sin\theta_1 \sin\theta_2}, \\ \dot{\theta_4} = \kappa_4 - \kappa_3 \frac{\cos\theta_3}{\sin\theta_3}\sin\theta_4 + \frac{W}{\sigma}\frac{\sin\theta_4}{\sin\theta_1 \sin\theta_2 \sin\theta_3}. \end{cases} \qquad (6)$$

2. Для трехмерных процессов сложного нагружения

$$\overline{n}_\sigma = \cos\theta_1 \overline{n}_1 - \sin\theta_1 \left(\cos\theta_2 \overline{n}_2 - \sin\theta_2 \overline{n}_3\right).$$

В зависимости от выбора основного репера получим разные определяющие уравнения и соответствующие им системы уравнений для углов:
2.1.

$$\{\overline{n}_\sigma, \overline{n}'_1, \overline{n}'_2\}:$$

$$\dot{\overline{n}_\sigma} = \frac{Q}{\sigma}\sin\theta_1 \overline{n}'_1 + \frac{F}{\sigma}\sin\theta_2 \overline{n}'_2, \qquad (7)$$

$$\begin{cases} \dot{\theta_1} = \kappa_1 \cos\theta_2 - \frac{Q}{\sigma}\sin\theta_1, \\ \dot{\theta_2} = \kappa_2 - \kappa_1 \frac{\cos\theta_1}{\sin\theta_1}\sin\theta_2 + \frac{F}{\sigma}\frac{\sin\theta_2}{\sin\theta_1}, \end{cases} . \qquad (7.1)$$

2.2.

$$\{\overline{n}_\sigma, \overline{n}'_1, \overline{n}'_\varepsilon\}:$$

$$\dot{\overline{n}_\sigma} = \frac{Q}{\sigma}\sin\theta_1 \overline{n}'_1 + \frac{N_1}{\sigma}\sin\theta_1 \overline{n}'_\varepsilon. \qquad (8)$$

$$\begin{cases} \dot{\theta_1} = \kappa_1 \cos\theta_2 - \frac{Q}{\sigma}\sin\theta_1 - \frac{N_1}{\sigma}\sin\theta_1\Delta, \\ \dot{\theta_2} = \kappa_2 - \kappa_1 \frac{\cos\theta_1}{\sin\theta_1}\sin\theta_2 - \frac{N_1}{\sigma}\sin\varphi_1\sin(\theta_2-\varphi_2), \end{cases} \tag{8.1}$$

$$\Delta \equiv \sin\theta_1\cos\varphi_1 - \cos\theta_1\sin\varphi_1\cos(\theta_2-\varphi_2).$$

Калибровка определяющих функционалов Вопрос калибровки определяющих функционалов на произвольных процессах деформаций решается однотипно для изображающих пространств и траекторий любой размерности. Он осуществляется в четыре этапа:

1. Этап первый: по уравнению кривой деформации определяем скалярные характеристики траектории в данной точке (кривизна и крутки);

2. Этап второй: приближаем траектории деформаций сплайнами в виде отрезков винтовых линий соответствующей размерности;

3. Этап третий: определяем формы отклика из теоремы изоморфизма и калибруем параметры отклика;

4. Этап четвертый: на данном отрезке по форме отклика из системы уравнений для углов находим определяющие функционалы и т.д.

Пример.

Трехмерное пространство. Рассмотрим трехмерный процесс деформаций с заданной траекторией.

Сплайн. В любой точке траектории деформаций с заданными на ней скалярными характеристиками уравнения:

$$\begin{gathered} \varepsilon_1 = \varepsilon_{10} + c\cos\alpha, \varepsilon_3 = c\sin\alpha, \\ \varepsilon_2 = \varepsilon_{20} + a\left(\frac{\alpha}{2\pi} + (m-1)\right), \end{gathered} \tag{8.2}$$

$$\varepsilon_{20} \equiv \frac{\varepsilon_{10}+c}{\sqrt{3}}, c \equiv \frac{\kappa_1}{\kappa_1^2+\kappa_2^2}, a \equiv \frac{2\pi\kappa_2}{\kappa_1^2+\kappa_2^2}, a_1 \equiv \frac{\kappa_2}{\sqrt{\kappa_1^2+\kappa_2^2}}$$

определяют сплайн в виде винтовой линии, проходящей через точку начала траектории с параметрами (кривизной и кручением), совпадающими с параметрами траектории деформации в начальной точке.

Параметры отклика. На основании экспериментальных данных [4] в [3] была установлена форма отклика на траекторию деформации (8.2) с четырьмя параметрами, неизвестным образом зависящими от характеристик сплайна:

$$\begin{gathered} \sigma_1 = \sigma_{10} + R\cos\beta, \sigma_3 = \sigma_{30} + R\sin\beta, \beta \equiv \alpha + \alpha_0, \\ \sigma_2 = \sqrt{\sigma(s)^2 - \sigma_1^2 - \sigma_2^2}, \sigma(s) = \sigma_0 + G's. \end{gathered} \tag{9}$$

В трехмерном случае соотношения (8.2) и (9) задают изоморфизм пространств деформаций и напряжений.

Формулы для функционалов. Имеют место следующие аналитические представления векторных характеристик процесса через параметры траектории деформаций и отклика:

$$\cos\theta_1 = \frac{\kappa_1}{\sqrt{\kappa_1^2+\kappa_2^2}}\frac{\sigma_{30}\cos\alpha - \sigma_{10}\sin\alpha + R\sin\alpha_0}{\sigma} + a_1\frac{\sigma_2}{\sigma},$$

$$\cos\theta_2 = \frac{\sigma_{10}\cos\alpha + \sigma_{30}\sin\alpha + R\cos\alpha_0}{\sigma\sin\theta_1}. \qquad (10)$$

Преобразуем уравнения (7.1) с учетом формул (10). В итоге получим систему линейных алгебраических уравнений для определяющих функционалов процесса:

$$\frac{Q}{\sigma}\sin^2\theta_1 = \frac{\kappa_2}{\sigma\sqrt{\kappa_1^2+\kappa_2^2}}\left(\dot{\sigma_2} - \frac{\sigma_2}{\sigma}\dot{\sigma}\right) + \frac{R}{\sigma}\cos\alpha_0 -$$

$$-\frac{\kappa_1}{\sqrt{\kappa_1^2+\kappa_2^2}}\frac{\sigma_{30}\cos\alpha - \sigma_{10}\sin\alpha + R\sin\alpha_0}{\sigma^2}\dot{\sigma},$$

$$\cos\theta_2\left(\frac{\dot{\sigma_2}}{\sigma} - \frac{Q}{\sigma}\cos\theta_1\right) - \frac{\sigma_{30}\cos\alpha - \sigma_{10}\sin\alpha}{\sigma\sin\theta_1}\sqrt{\kappa_1^2+\kappa_2^2} =$$

$$= \kappa_2\sin\theta_2 - \kappa_1\frac{\cos\theta_1}{\sin\theta_1} + \frac{F}{\sigma}\frac{sin^2\theta_2}{\sin\theta_1}.$$

Пятимерное пространство. В этом случае уравнение сплайна строим в виде пятимерного винта. В [1] в пятимерном пространстве решена задача построения траектории с постоянными кривизнами. Решение задачи выписано в репере, который не является ортогональным. Это решение приводим к ортонормированному реперу Френе начальной точки траектории. Результат в виде трехчленного пятимерного винта дается формулой:

$$\overline{\varepsilon} = \frac{\kappa_2\kappa_4}{k_1^2k_2^2}s\begin{pmatrix}\kappa_2\kappa_4\\0\\\kappa_1\kappa_4\\0\\\kappa_1\kappa_3\end{pmatrix} + \frac{\kappa_1}{k_1^2(k_1^2-k_2^2)}\begin{pmatrix}-\kappa_1(k_2^2-\kappa_1^2-\kappa_2^2)\frac{\sin k_1 s}{k_1}\\-(k_2^2-\kappa_1^2-\kappa_2^2)(1-\cos k_1 s)\\\kappa_2(\kappa_4^2-k_1^2)\frac{\sin k_1 s}{k_1}\\-\kappa_2\kappa_3(1-\cos k_1 s)\\\kappa_2\kappa_3\kappa_4\frac{\sin k_1 s}{k_1}\end{pmatrix} +$$

$$+\frac{\kappa_1}{k_2^2(k_1^2-k_2^2)}\begin{pmatrix}\kappa_1(k_1^2-\kappa_1^2-\kappa_2^2)\frac{\sin k_2 s}{k_2}\\(k_1^2-\kappa_1^2-\kappa_2^2)(1-\cos k_2 s)\\-\kappa_2(\kappa_4^2-k_2^2)\frac{\sin k_2 s}{k_2}\\\kappa_2\kappa_3(1-\cos k_2 s)\\-\kappa_2\kappa_3\kappa_4\frac{\sin k_2 s}{k_2}\end{pmatrix},$$

в которую входят кривизны траектории, а также две функции кривизн, см.[1]:

$$k_1^2=\frac{1}{2}\left(\kappa_1^2+\kappa_2^2+\kappa_3^2+\kappa_4^2+\right.$$

$$\left.+\sqrt{\left(\kappa_1^2+\kappa_2^2+\kappa_3^2+\kappa_4^2\right)^2-4\left(\kappa_1^2\kappa_3^2+\kappa_2^2\kappa_4^2+\kappa_1^2\kappa_4^2\right)}\right)$$

$$k_2^2=\frac{1}{2}\left(\kappa_1^2+\kappa_2^2+\kappa_3^2+\kappa_4^2-\right.$$

$$\left.-\sqrt{\left(\kappa_1^2+\kappa_2^2+\kappa_3^2+\kappa_4^2\right)^2-4\left(\kappa_1^2\kappa_3^2+\kappa_2^2\kappa_4^2+\kappa_1^2\kappa_4^2\right)}\right).$$

В результате тождественных преобразований трехчленной формулы пятимерного винта получается пятичленное представление:

$$\overline{\varepsilon}=p_1\frac{\kappa_1}{k_1^2(k_1^2-k_2^2)}\{\overline{e}_1(-1+\cos k_1 s)+\overline{e}_3\sin k_1 s\}+s\frac{\kappa_2\kappa_4}{k_1 k_2}\overline{e}_2+$$

$$+p_4\frac{\kappa_1}{k_2^2(k_1^2-k_2^2)}\{\overline{e}_4(-1+\cos k_2 s)+\overline{e}_5\sin k_2 s\} \qquad (11)$$

с известным образом зависящими от кривизн траектории функциями p_1, p_4.

Значит, в специальном ортонормированном репере $\overline{e}_1, \overline{e}_2, \overline{e}_3, \overline{e}_4, \overline{e}_5$ уравнение пятимерного винта имеет представление подобное трехмерному винту (8.2). Что касается самого репера, то, как видно, он определяется только внутренней геометрией траектории.

Указанное обстоятельство позволяет постулировать подобную (9) форму для отклика на траекторию деформаций в виде пятимерной винтовой линии:

$$\begin{cases}\sigma_1=\sigma_{10}+R_1\cos\beta, \beta\equiv k_1 s+\beta_0,\\ \sigma_3=\sigma_{30}+R_1\sin\beta,\\ \sigma_4=\sigma_{40}+R_2\cos\gamma, \gamma\equiv k_2 s+\gamma_0,\\ \sigma_5=\sigma_{50}+R_2\sin\gamma,\\ \sigma_2=\sqrt{\sigma^2-\sigma_1^2-\sigma_3^2-\sigma_4^2-\sigma_5^2}.\end{cases} \qquad (12)$$

Подчеркнем, что отличие этого представления от (9) в том, что форма отклика для трехмерных винтовых траекторий была выбрана в [3] на основе анализа результатов обработки экспериментальных данных [4], тогда как соотношение (12) представляет собой формулировку теоремы изоморфизма для пятимерных траекторий деформации. В четырехмерном или двумерном пространствах деформаций секулярные члены в уравнениях винтов (11) равны нулю и это обстоятельство соответствующим образом корректирует и формы отклика двумерных и четырехмерных процессов деформации..

Калибровка параметров отклика.

Оценка параметров (9) или (12) проводится по уравнению:

$$\overline{\sigma}^{\cdot} = Q\overline{n}_1 + \left(\frac{d\sigma}{ds} - Q\cos\theta_1\right)\overline{n}_\sigma,$$

которое является трехчленной формулой А.А.Ильюшина, и его вполне можно считать 0-приближением всех рассматриваемых здесь уравнений (5),(7),(8), с использованием формулы

$$\cos\theta_1 = c_1k_1\frac{\sigma_{30}\cos k_1s - \sigma_{10}\sin k_1s + R_1\sin\beta_0}{\sigma} + \tag{13}$$

$$+c_2k_2\frac{\sigma_{50}\cos k_2s - \sigma_{40}\sin k_2s + R_2\sin\gamma_0}{\sigma} + a_1\frac{\sigma_2}{\sigma}.$$

В этом приближении пренебрегаем малыми величинами порядка $\sigma_{10}/\sigma, \cdots \sigma_{50}/\sigma$, (это следовало в трехмерном случае из анализа экспериментальных данных [4]) и записываем с учетом (12) первое, третье, четвертое и пятое уравнения 0-приближения. Получаем систему из 4 независимых уравнений для 4 неизвестных параметров отклика $(R_1, R_2, \beta_0, \gamma_0)$:

$$R_1 = \frac{c_1}{\lambda}\sigma\cos\beta_0, R_2 = \frac{c_2}{\lambda}\sigma\cos\gamma_0, k_1\,\mathrm{tg}\,\beta_0 = k_2\,\mathrm{tg}\,\gamma_0,$$

$$\mathrm{tg}^5\,\beta_0 + \mathrm{tg}^4\,\beta_0\frac{1}{k_1}\left(\frac{1}{\sigma}\frac{d\sigma}{ds} - \frac{a_1}{\lambda}\right) + tg^3\beta_0\left(\frac{k_2^2}{k_1^2} + 1 - \frac{c_1^2}{\lambda^2} - \frac{c_2^2}{\lambda^2}\frac{k_2^2}{k_1^2}\right) +$$

$$+\,\mathrm{tg}^2\,\beta_0\frac{1}{k_1}\left(\frac{k_2^2}{k_1^2} + 1\right)\left(\frac{1}{\sigma}\frac{d\sigma}{ds} - \frac{a_1}{\lambda}\right) + \mathrm{tg}\,\beta_0\frac{k_2^2}{k_1^2}\left(1 - \frac{c_1^2 + c_2^2}{\lambda^2}\right) + \tag{14}$$

$$+\frac{1}{k_1}\frac{k_2^2}{k_1^2}\left(\frac{1}{\sigma}\frac{d\sigma}{ds} - \frac{a_1}{\lambda}\right) = 0, \frac{1}{\lambda} \equiv \frac{Q}{\sigma}.$$

При $k_2 = 0, c_2 = 0$ отсюда следуют характеристики отклика трехмерного процесса деформаций:

$$R_1 = \frac{c_1}{\lambda}\sigma\cos\beta_0,$$

$$\mathrm{tg}^3\,\beta_0 + \mathrm{tg}^2\,\beta_0\frac{1}{k_1}\left(\frac{1}{\sigma}\frac{d\sigma}{ds} - \frac{a_1}{\lambda}\right) +$$

$$+\,\mathrm{tg}\,\beta_0\left(1 - \frac{c_1^2}{\lambda^2}\right) + \frac{1}{k_1}\left(\frac{1}{\sigma}\frac{d\sigma}{ds} - \frac{a_1}{\lambda}\right) = 0. \qquad (15)$$

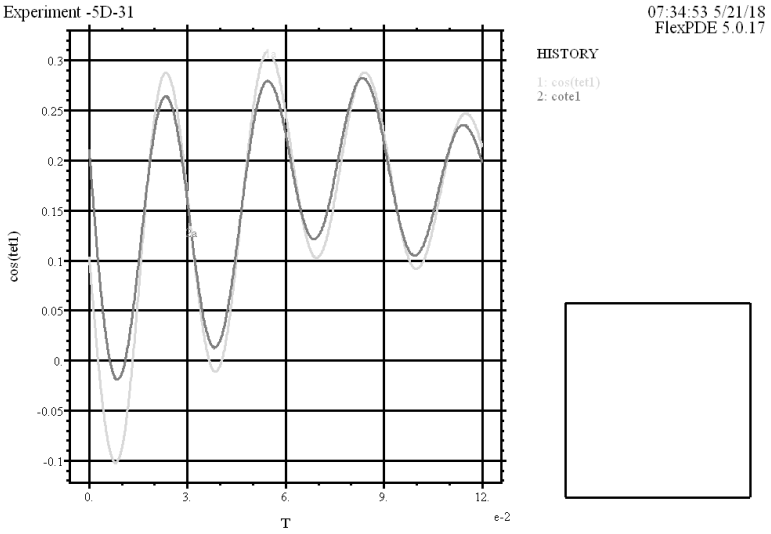
Параметры $\sigma_{10}, \sigma_{20}, \sigma_{30}, \sigma_{40}, \sigma_{50}$, в 0-приближении остаются неопределенными. Итерационная процедура для нахождения этих параметров в трехмерном случае изложена в [2] вместе с примером реализации для процессов в виде трехмерных винтовых линий. Найденные там значения этих параметров уже в 1-приближении соответствуют экспериментальным данным.

Все предложенные выше подходы были реализованы на пятимерных процессах деформаций с траекториями деформаций в виде пятимерных винтовых линий после одноосного нагружения. Мы взяли реальные эксперименты из [4] на трехмерных винтах и на их основе построили пятимерные траектории деформаций, сохранив для них те же самые номера, что были в прототипах.

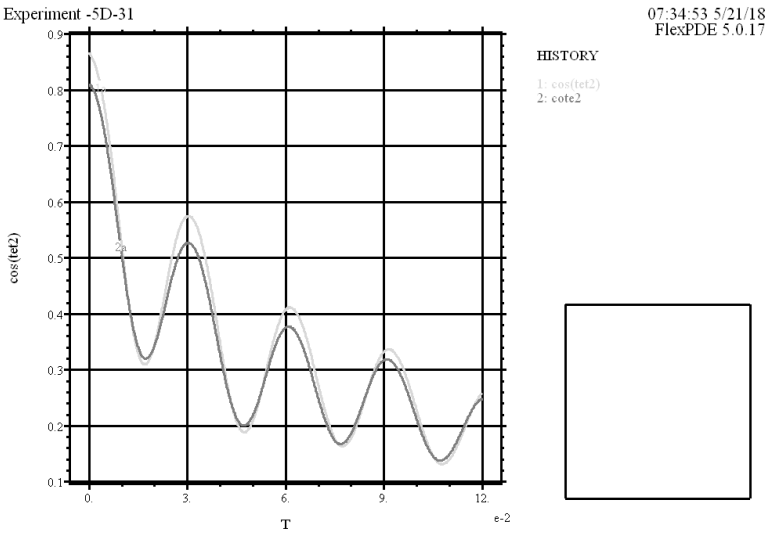
Таблица параметров рассмотренных пятимерных процессов деформаций.

Nexp	κ_1	κ_2	κ_3	κ_4	σ_{10}	σ_{30}	σ_{40}	σ_{50}	λ
25	333	666	200	200	70	30	10	10	0.005
27	100	200	50	50	50	50	70	10	0.008
29	200	200	50	50	70	50	-20	50	0.008
31	200	60	60	60	70	30	10	100	0.08
35	200	400	100	100	70	30	10	70	0.01

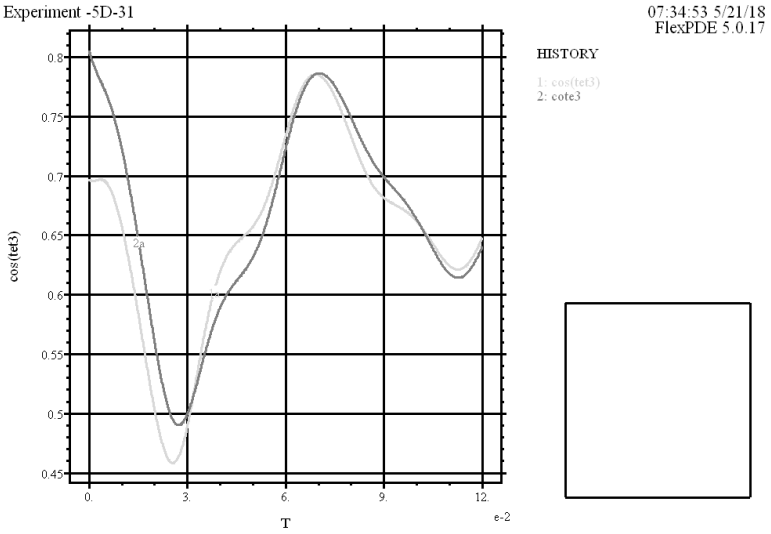
Результаты некоторых расчетов приводятся ниже.



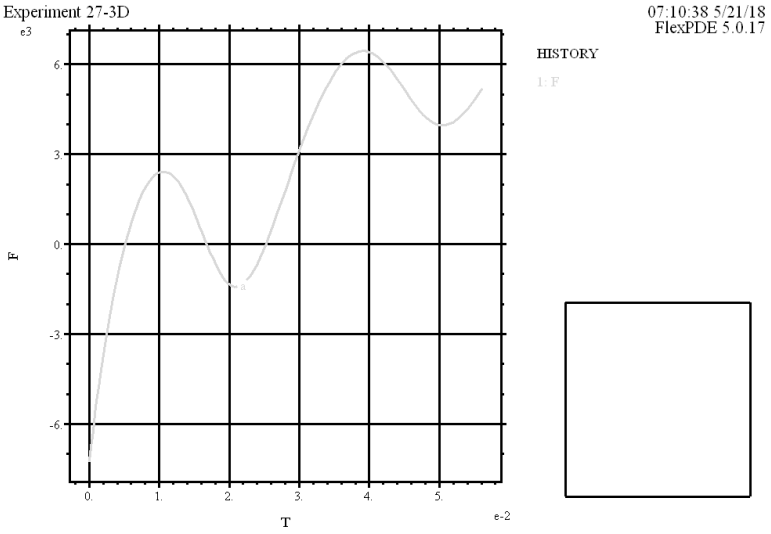
Solve 5D: Cycle=828 Time= 0.1200 dt= 1.8735e-4 P2 Nodes=9 Cells=2 RMS Err= 2.e-18



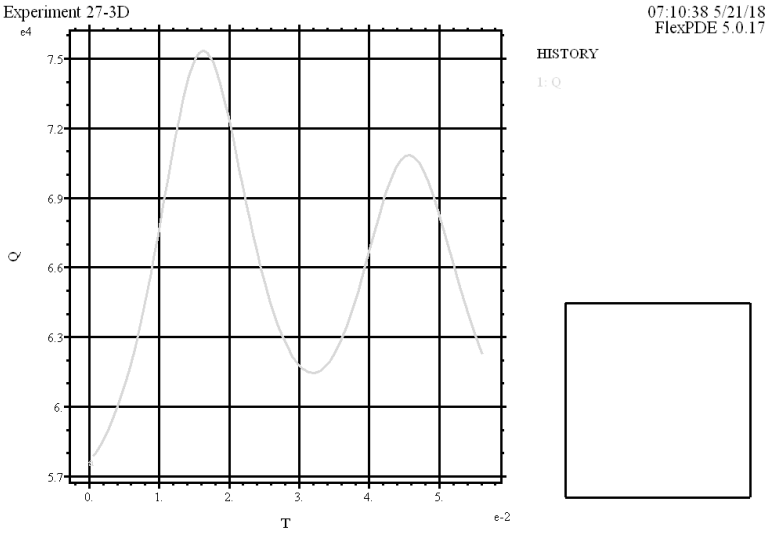
Solve 5D: Cycle=828 Time= 0.1200 dt= 1.8735e-4 P2 Nodes=9 Cells=2 RMS Err= 2.e-18



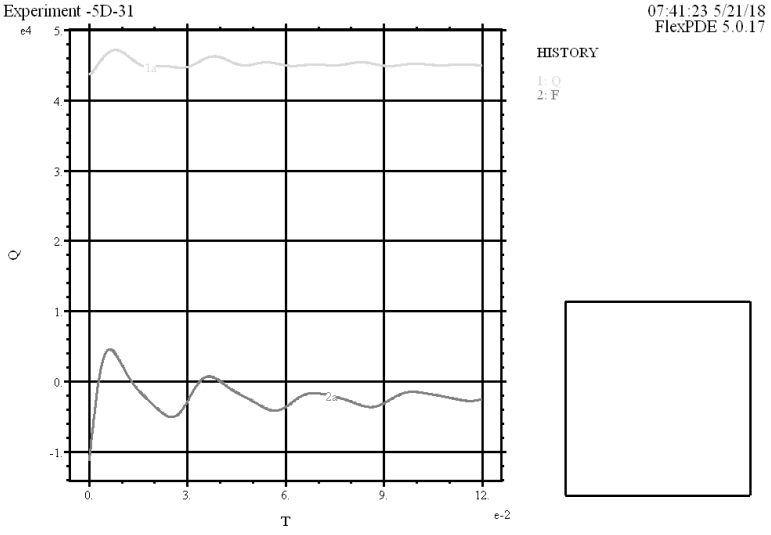
Solve 5D: Cycle=828 Time= 0.1200 dt= 1.8735e-4 P2 Nodes=9 Cells=2 RMS Err= 2.e-18



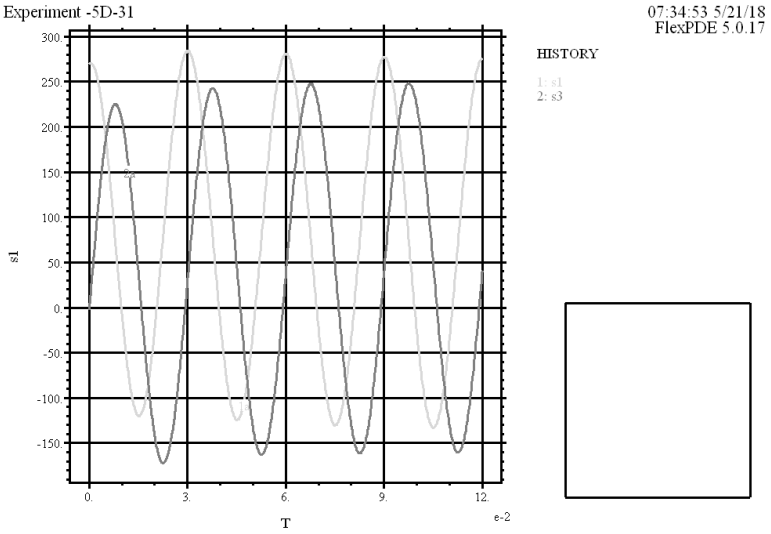
Solve1 3D: Cycle=108 Time= 0.0561 dt= 8.6779e-4 P2 Nodes=9 Cells=2 RMS Err= 1.e-12



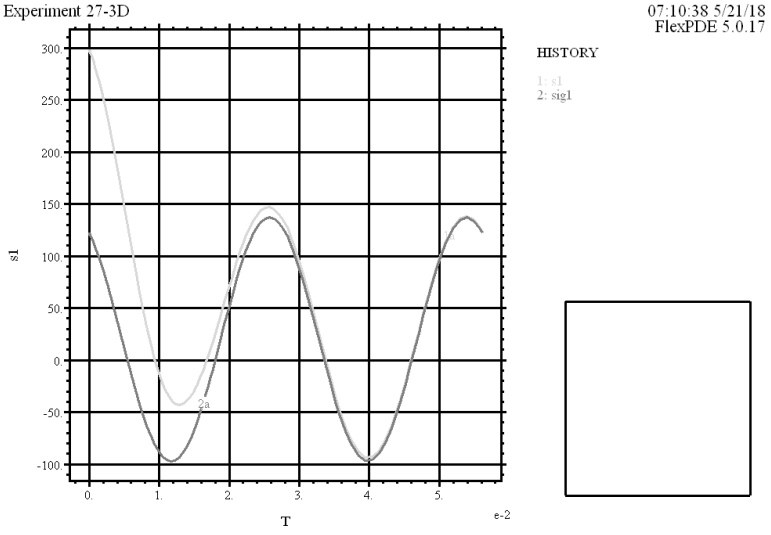
Solve1 3D: Cycle=108 Time= 0.0561 dt= 8.6779e-4 P2 Nodes=9 Cells=2 RMS Err= 1.e-12



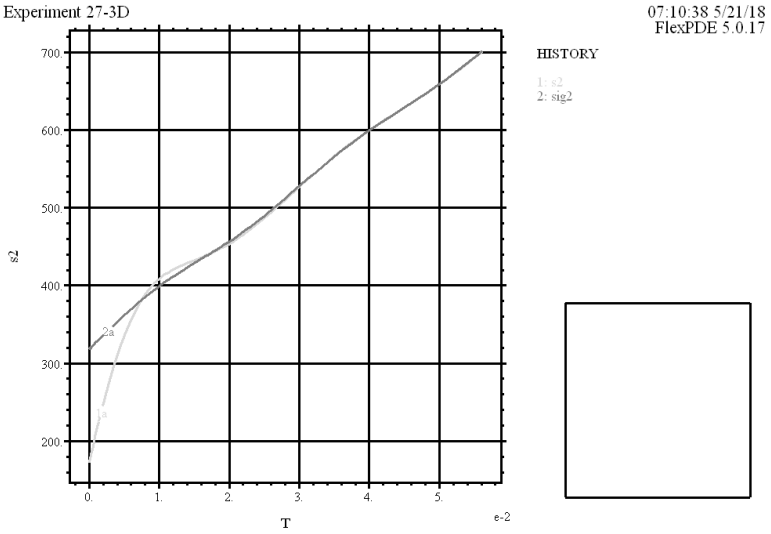
Solve 5D: Cycle=828 Time= 0.1200 dt= 1.8735e-4 P2 Nodes=9 Cells=2 RMS Err= 2.e-18



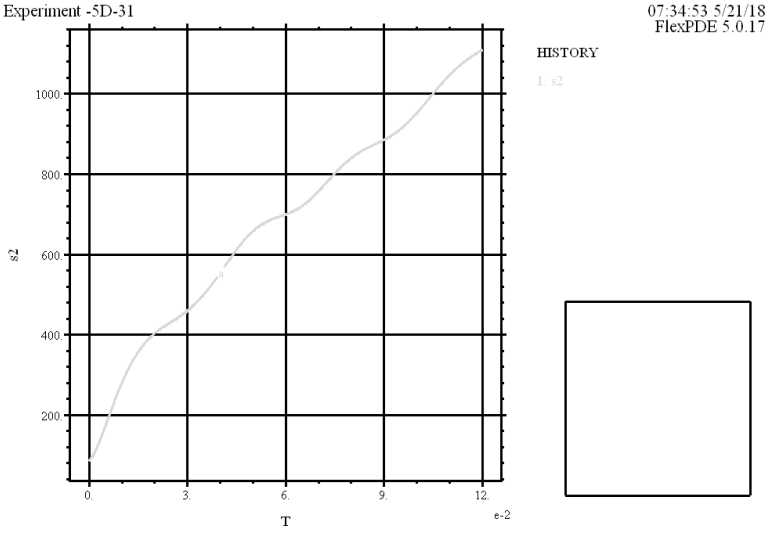
Solve 5D: Cycle=828 Time= 0.1200 dt= 1.8735e-4 P2 Nodes=9 Cells=2 RMS Err= 2.e-18



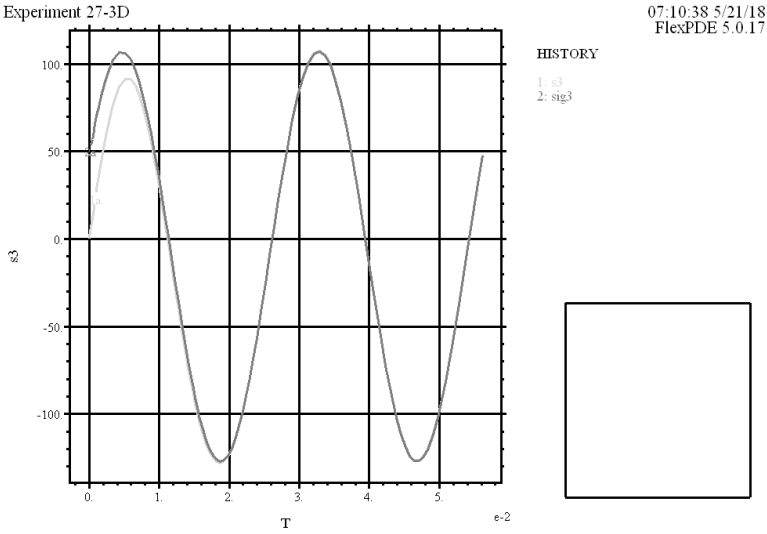
Solve1 3D: Cycle=108 Time= 0.0561 dt= 8.6779e-4 P2 Nodes=9 Cells=2 RMS Err= 1.e-12



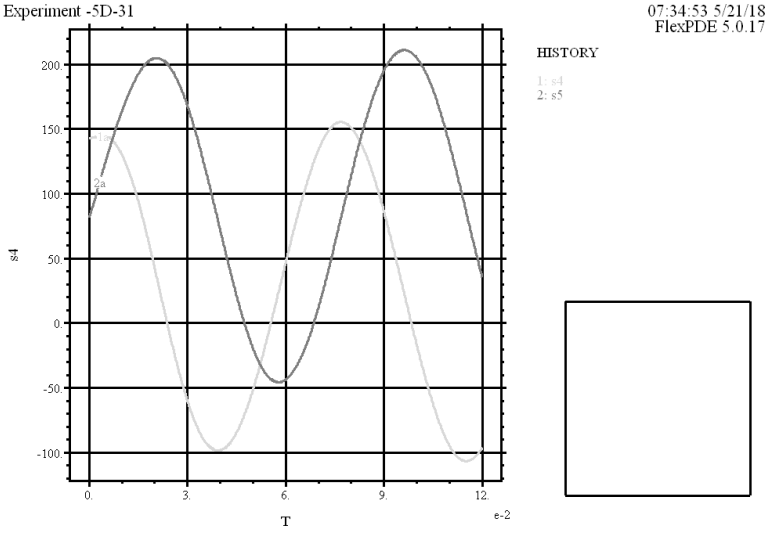
Solve1 3D: Cycle=108 Time= 0.0561 dt= 8.6779e-4 P2 Nodes=9 Cells=2 RMS Err= 1.e-12



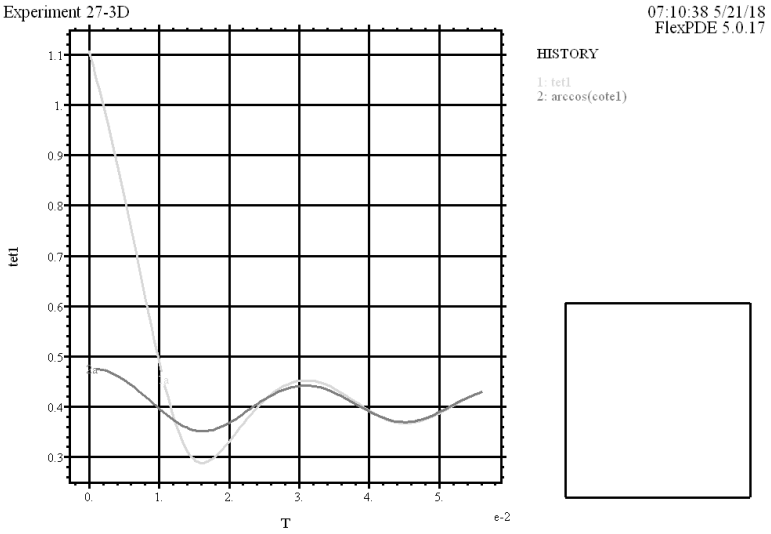
Solve 5D: Cycle=828 Time= 0.1200 dt= 1.8735e-4 P2 Nodes=9 Cells=2 RMS Err= 2.e-18



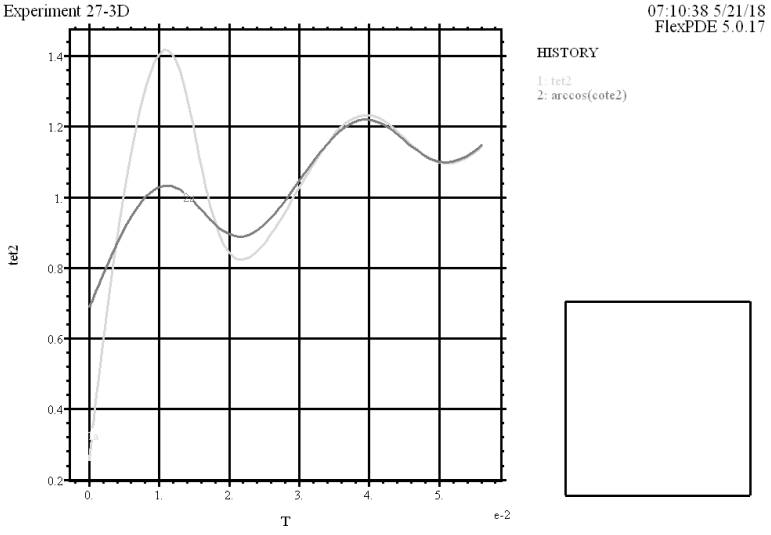
Solve1 3D: Cycle=108 Time= 0.0561 dt= 8.6779e-4 P2 Nodes=9 Cells=2 RMS Err= 1.e-12



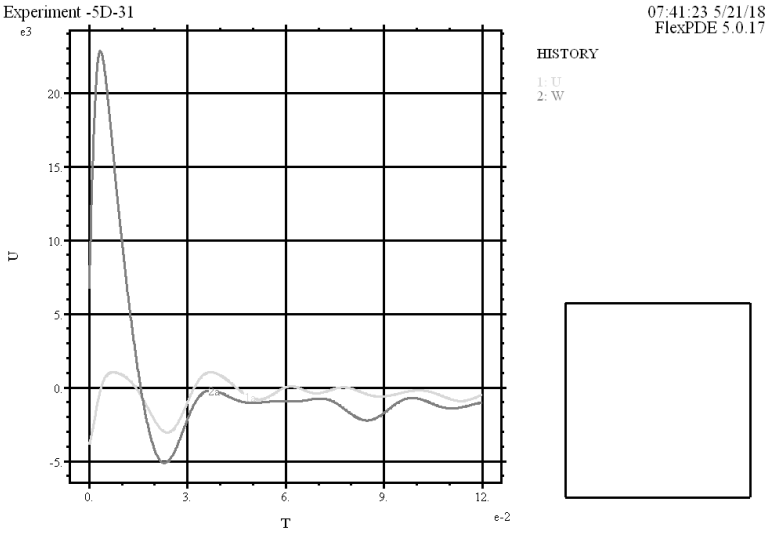
Solve 5D: Cycle=828 Time= 0.1200 dt= 1.8735e-4 P2 Nodes=9 Cells=2 RMS Err= 2.e-18



Solve1 3D: Cycle=108 Time= 0.0561 dt= 8.6779e-4 P2 Nodes=9 Cells=2 RMS Err= 1.e-12



Solve1 3D: Cycle=108 Time= 0.0561 dt= 8.6779e-4 P2 Nodes=9 Cells=2 RMS Err= 1.e-12



Solve 5D: Cycle=828 Time= 0.1200 dt= 1.8735e-4 P2 Nodes=9 Cells=2 RMS Er= 2.e-18

Список литературы

[1] Ильюшин А.А. Пластичность. Основы общей математической теории // Изд-во АН СССР, Москва,-1963,- 272 с.

[2] Молодцов И.Н., Бабаева Д.О. Некоторые вопросы верификации теории упругопластических процессов при сложном нагружении// Проблемы машиностроения и автоматизации, Москва, -2016, -№3, -с.98-105.

[3] Молодцов И.Н., Бабаева Д.О. О роли функционалов пластичности в геометрическом истолковании диссипации и описании векторных свойств материала в процессах сложного нагружения// В сборнике Упругость и неупругость. Материалы Международного научного симпозиума по проблемам механики деформируемых тел, посвященного 105-летию со дня рождения А.А.Ильюшина// Москва,-2016,-с.210-215.

[4] Вавакин А.С., Васин Р.А., Викторов В.В., Широв Р.И. Экспериментальное исследование упругопластического деформирования стали при сложном нагружении по криволинейным пространственным траекториям деформаций. Деп. в ВИНИТИ, 16.10.86, №7298-В86. 66с.

Some mathematical models of elastoplastic processes of complex loading

Molodtsov I.N., Babaeva D.O.

In the framework of the Ilyushin's theory of elastoplastic processes, in [3] for mathematical modeling of complex loading processes we use special type quasilinear equation with three state functionals. The functionals was calibrated using the experimental results [4] (R.A.Vasin, etc.) for 3D- helical trajectories of deformations. It turned out that the response on a helical trajectory of deformation takes a completely definite loading form, not exactly, but after the exhaustion of some trace of retard. On the helical trajectories of deformations the form of loading is the same: helical trajectory of deformations is becoming to helical trajectory of loading there and back. The used map preserves the geometry of space. This correspondence by Ilyushin is called as isomorphism theorem. All new theories use as the basis the directing vector of stresses and vectors constructed on the base of Frenet basis. For high-dimensional processes, the number of state functionals increases, so and the methods of their identification become more complicated. All models are completly verified. The results are given below.

Keywords: plasticity, plastic deformations, complex loading, state functional, identification of functionals, isomorphism theorem

Оценка зависимости износа и скорости записи твердотельного накопителя от размера резервной области и размера блока

Снегова Е.А., Алисейчик П.А., Алексеев Д.В.

В данной работе производится оценка зависимости избыточности записи (Write Amplification) от размера резервной области памяти (Overprovision) твердотельного накопителя (SSD) при разных значениях числа секторов в блоке N. Сборка мусора, т.е. перезапись частично заполненных блоков производится в соответствии с жадным алгоритмом (Greedy Garbage Collection).

Получена более точная формула чем в [1], поскольку отсутствует предположение $N \to \infty$. Показано, что при фиксированном размере резервной области избыточность записи является монотонно возрастающей функцией от N.

Ключевые слова: избыточность записи, резервная область памяти, сборщик мусора, твердотельный накопитель.

1. Введение

Физическое пространство памяти SSD-накопителя состоит из блоков, каждый из которых состоит из одинакового колисества страниц. Например, один блок типичного SSD-накопителя может содержать 512 страниц, при этом каждая страница может иметь объем 16 Kb. Единицей записи является страница, в то время как единицей стирания является многостраничный блок.

В силу ограничений технологии SSD, для того, чтобы записывать новые данные на страницу, страница должна быть очищена, т.е. предварительно требуется стереть весь блок, к которому она относится. Поэтому обновление данных, записанных во флеш-память происходит путем записи обновленных данных в новую область, причем область, в которую была записана старая версия подвергается процессу *инвалидации*,

т.е. страницы, содержащие старые данные помечаются как *невалидные*. Изначально все страницы считаются валидными. *Валидностью блока* называют количество содержащихся в нем валидных страниц.

При этом контроллер хранит и постоянно обновляет таблицу соответствия логических адресов физическим. Со временем количество невалидных страниц растет, а валидность блоков, соответственно, уменьшается. Контроллер периодически проводит стирание блоков, чтобы освободить место для записи. В процессе стирания производится копирование всех действительных (т.е. не подвергшихся инвалидации ранее) страниц в стираемом блоке.

Эффективность этого процесса стирания может существенно повлиять на производительность памяти, в частности, за счет дополнительного копирования может снизиться пропускная способность. Это явление называют *эффектом избыточной записи*. Избыточность записи определяется как среднее значение числа записей на SSD (в секторах), приходящееся на один сектор пользовательского запроса на запись. Избыточность записи влияет как на скорость записи, так и на износ физического носителя.

Для хранения повторно записанных страниц используется резервная область памяти — та часть физического пространства SSD, которая не доступна пользователю. Ее размер измеряется как отношение разницы размера физического пространства и логического пространства к логическому пространству (все составляющие этой формулы измеряются в секторах или байтах).

Задача оценки зависимости избыточности записи (Write Amplification, далее обозначается как W) от размера резервной области памяти (Overprovision, далее обозначается как R) твердотельного накопителя (SSD) при разных значениях числа секторов в блоке N возникла при программировании контроллера реального SSD-устройства. Предполагалось, что данный функционал должен работать с различными видами SSD-устройств, в частности, отличающимися числом секторов в блоке N. Таким образом, данная формула призвана проверить, что контроллер работает оптимальным образом, позволяя на случайном равномерно-распределенном в логическом пространстве потоке запросов получать минимально возможное значение избыточности записи.

В работе [6] была предложена вероятностная аналитическая модель для сборщика мусора со скользящим окном, был представлен метод вычисления избыточности записи. В [2] был получен результат, в котором зависимость избыточности записи от размера резервной области исходя

из предположения, что валидность блока равномерна распределена на отрезке $[v_{min}, N]$, где v_{min} — это минимальная валидность блока (блок с такой валидностью попадает в GC при жадном алгоритме "сборки мусора"), а N — число секторов в блоке. Полученная оценка избыточности записи не зависит ни от v_{min}, ни от N. В работах [4] и [8] независимо был получен аналогичный результат в предположении, что распределение валидности зависит от числа валидных страниц, что более точно соответствует действительности. Во всех этих работах используется сборщик мусора, использующий жадный алгорим. В работе [1] получена оценка $W(R)$ с использованием функции Ламберта. Эта оценка также не зависит от N, но является наиболее близкой к полученной в настоящей статье оценке $W(R, N)$, так как при при $N- > \infty$ $W(R, N)$ сходится к оценке $W(R)$ из [1].

2. Постановка задачи и результат

Далее опишем типичный упрощенный алгоритм работы флеш-контроллера. Память флеш контроллера состоит из трех основных компонент.

1) кэш объемом не менее одного блока;

2) таблица Block(lsa), в которой каждому логическому сектору (lsa) сопоставлен физический номер блока, в который он записан, или сопоставлено число -1, если данный сектор не используется;

3) таблица валидности блоков Validity(block), в котроой каждому блоку сопоставлена его валидность, то есть число таких секторов (lsa), у которых Block(lsa) = block.

Таблицы контроллера Block(lsa) и Validity(block) по сути составляют *систему трансляции адресов* (Flash Translation Layer, FTL).

Типичный алгоритм контроллера выглядит следующим образом. От управляющего устройства поступает запросна запись последовательного участка логических секторов. Постепенно эти сектора попадают в кэш. При этом «старые» данные записанные в данном логическом секторе становятся невалидными, поэтому значение Validity(Block(lsa)) уменьшается на единицу, а затем Block(lsa) присваивается значение -1.

Если в кэше оказалась хотя бы одна полная страница, то происходит запись данных из кэша на SSD. Для этого либо берется текущий открытый блок и записывается следующая подряд идущая страница, либо

сначала ищется пустой блок, объявляется текущим открытым блоком, а затем в него записывается первая страница.

Обозначим текущий открытый блок как *cur_block*. Тогда для каждого сектора lsa, поступившего из кэша на SSD, таблицы FTL менеяются следующим образом: $Block(lsa) = cur\_block$, $Validity(cur\_block) += 1$.

Поиск пустого блока — задача части контроллера под названием сборщик мусора (Garbage Collector). В настоящей статье рассматривается "жадный алгоритм"GC, при котором GC берет блок с наименьшей валидностью и все валидные данные помещает к кэш, а сам блок стирает, освобождая для новых записей.

Очевидно, что такой алгоритм работы контроллера возможет только в ситуации, когда объем памяти SSD (физическое пространство) превышает совокупный объем логического пространства.

Ниже приводится оценка избыточности записи W от объема резервной памяти R и числа секторов в блоке N. Верны следующие оценки:

1) Пусть $R_k = \frac{N}{N-k} * [\frac{1}{k+1} + ... + \frac{1}{N}] - 1$ $(k = 0, ...N-1)$, тогда

$$W(R_k) = \frac{N}{N-k}.$$

2) При $R \in [R_{k+1}, R_k]$ выполнено равенство

$$\frac{R - R_{k+1}}{R_k - R_{k+1}} = \frac{W - W_{k+1}}{W_k - W_{k+1}},$$

то есть W линейно зависит от R.

3) Если $R > \frac{1}{2} + .. + \frac{1}{N}$, то $W = 1$.

При разных значениях N зависимость $W(R)$ выглядит следующим образом (см. рис. 1-3):

Замечание: при при $N \to \infty$ приведенная выше оценка $W(R, N)$ сходится к оценке $W(R)$ из [1].

3. Оценка зависимости $W(R, N)$ в узловых точках

Обозначим логическое пространство за единицу, а физическое — за $1+R$. Без ограничения общности, будем считать, что размер сектора совпадает с размером страницы. Данное предположение упрощает выкладки, но никак не влияет на полученный результат. Кроме того, будем считать, что в одну единицу времени приходит одно из следующих двух событий:

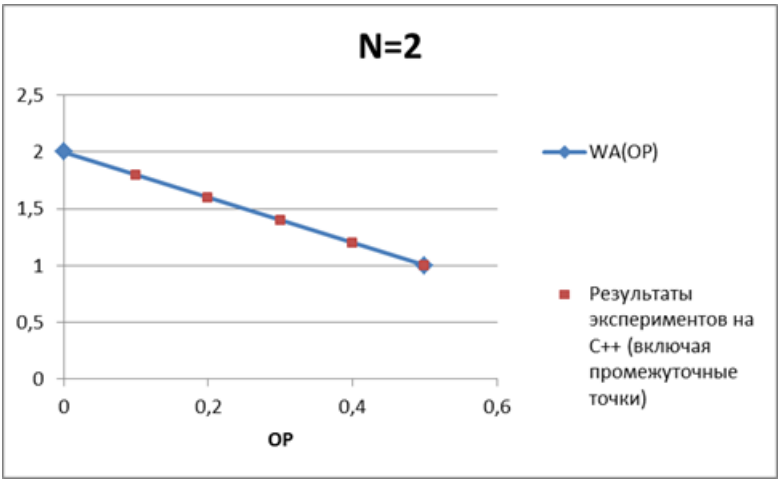


Рис. 1.

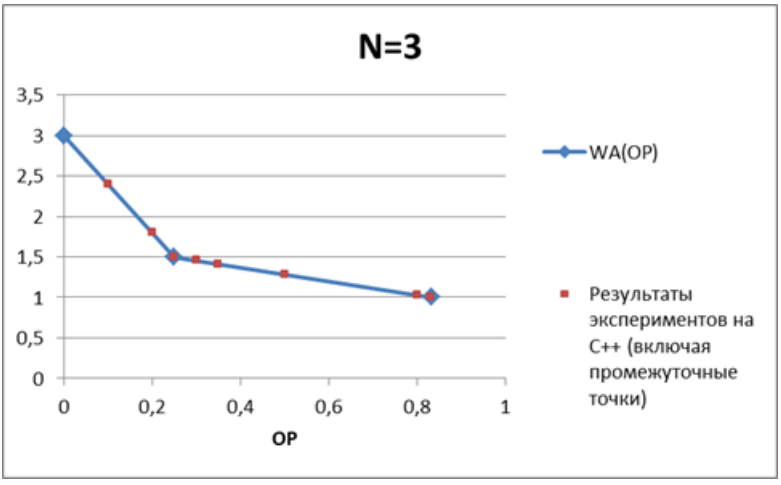


Рис. 2.

1) Выполнение одного запроса на запись.

   Поступает один запрос (то есть соответствующие данные попадают в кэш) и происходит инвалидация "старых" данных (изменение таблиц Validity и Block). Кроме того, если в кэше набралась одна страница секторов, то она записывается на SSD (в физическое пространство) в тот же самый момент времени.

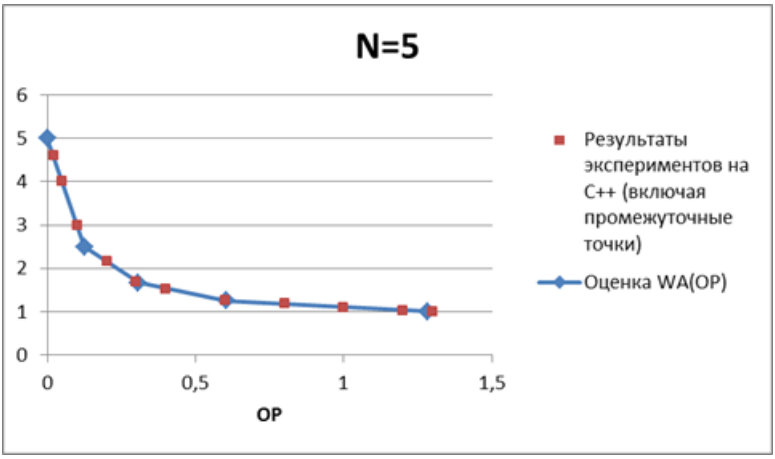


Рис. 3.

2) Работа GC — "превращение блока с минимальной валидностью в блок с валидностью N".

   Если в физическом пространстве не осталось больше свободного места, то есть записанная на предыдущем шаге страница была последней в открытом блоке, то GC ищет блок с минимальной валидностью, все валидные данные переписывает в кэш, блок очищает и объявляет его новым открытым блоком. В случае если минимальная валидность не нулевая, то в кэше оказывается минимум один сектор. Тогда все данные из кэша переписываются в физическое пространство, частично заполняя новый открытый блок.

Пусть k — минимальная валидность, C — скорость, с которой GC "превращает блок с минимальной валидностью в блок с валидностью N X_n — это доля блоков с валидностью n в общем числе блоков, составляющих физическое пространство, то есть верно, что

$$X_{k+1} + X_{k+2} + ... + X_N = \frac{1+R}{N}. \quad (1)$$

Используем предположение о равномерности распределения пользовательских запросов в логическом пространстве. Вероятность увеличения числа блоков с валидностью $n = k+1, .., N-1$ равна произведению числа блоков с валидностью $n+1$ - X_{n+1}, и размеру блоков с валидностью $n+1$, то есть самому числу $n+1$. Аналогично, вероятность уменьшения числа блоков размера n равна произведению числа блоков с валидностью

n - X_n и размеру блока n. Таким образом,

$$X'_n = X_{n+1}(n+1) - X_n \cdot n.$$

Уравнения, описывающие работу системы:

$$X'_k = X_{k+1}(k+1) - C$$

$$X'_{k+1} = X_{k+2}(k+2) - X_{k+1} \cdot (k+1) \tag{2}$$

$$.....$$

$$X'_N = C - X_N \cdot N.$$

В равновесной ситуации верно, что

$$X'_k = X'_{k+1} = ... = X'_N = 0. \tag{3}$$

Кроме того, $X_k = 0$, так как если $X_k > 0$, то с ненулевой вероятностью $k-1$ будет минимальной валидностью, что противоречит нашему изначальному предположению. Из (1), (2) и (3) следуют условия на R, при котором минимальная валидность блока равняется k:

$$R = \frac{N}{N-k} * [\frac{1}{k+1} + ... + \frac{1}{N}] - 1, \tag{4}$$

W в этом случае определяется следуюбщим образом:

$$W = \frac{N}{N-k}.$$

4. Оценка завистмости $W(R, N)$ в промежуточных точках

Осталось показать, что в промежуточных точках зависимость $W(R, N)$ линейно по R при фиксированном N.

Перепишем уравнения, описывающие работу системы для случая, когда минимальная валидность может равняться k или $k+1$.

$$\frac{\triangle X_k}{\triangle t} = X_{k+1}(k+1) - C * I_{X_k \neq \emptyset}$$

$$\frac{\triangle X_{k+1}}{\triangle t} = X_{k+2}(k+2) - X_{k+1} * (k+1) - C * I_{X_k = \emptyset}$$

$$\frac{\triangle X_{k+2}}{\triangle t} = X_{k+3}(k+3) - X_{k+2} * (k+2)$$

....

$$\frac{\triangle X_N}{\triangle t} = C - X_N * N$$

Пусть в α — доля случаев, когда минимальная валидность блока k, а в $1-\alpha$ случаях - минимальная валидность $k+1$, $\alpha \in [0,1]$.

Тогда

$$C = \frac{1}{N-k-(1-\alpha)},$$

$$W = \frac{N}{N-k-(1-\alpha)}.$$

$$X_{k+1} + X_{k+2} + ... + X_N = \frac{1+R}{N} \Rightarrow$$

$$\frac{1}{N-k-(1-\alpha)} * [\frac{\alpha}{k+1} + \frac{1}{k+2} ... + \frac{1}{N}] = \frac{1+R}{N}, \tag{5}$$

откуда следует, что

$$\alpha = \frac{[\frac{1}{k+2} + ... + \frac{1}{N}] * N - (N-k-1) * (1+R)}{1+R-\frac{N}{k+1}},$$

$$W = \frac{N * [1+R-\frac{N}{k+1}]}{(N-k-1) * [-\frac{N}{k+1}] + [\frac{1}{k+2} + ... + \frac{1}{N}] * N},$$

то есть уравнение линейно по R.

5. Заключение

Полученные оценки были проверены путем симуляции работы твердотельного накопителя. Показано, что в случае конечного числа секторов они являются более точными, чем оценка из [2].

В дальнейшем планируется обобщение полученного результата на случай неравномерного распределения частоты обновления данных. Т.е. когда данные делятся на "горячие" (т.е. часто обновляемые) и "холодные" (редко обновляемые).

Список литературы

[1] Stoica R., Ailamaki A. Improving flash write performance by using update frequency //Proceedings of the VLDB Endowment. – 2013. – Т. 6. – №. 9. – С. 733-744.

[2] Agarwal R., Marrow M. A closed-form expression for write amplification in NAND flash //GLOBECOM WorkshRs (GC Wkshps), 2010 IEEE. – IEEE, 2010. – С. 1846-1850.

[3] Sah C. T. Fundamentals of solid state electronics. – World Scientific Publishing Company, 1991.

[4] Desnoyers P., Analytic modeling of SSD write performance // SYSTOR '12 Proceedings of the 5th Annual International Systems and Storage Conference, Article No. 12

[5] Desnoyers P., Analytic Models of SSD Write Performance // ACM Transactions on Storage, 2012.

[6] Hu X.-H., Eleftheriou E. , Haas R., Iliadis I., Pletka R. Write Amplification Analysis in Flash-Based Solid State Drives // SYSTOR, 2009.

[7] Yang Y., Zhu J., Analytical modeling of garbage collection algorithms in hotness-aware flash-based solid state drives // Mass Storage Systems and Technologies (MSST), 2014, 30th Symposium on.

[8] Xiang L., Kurkoski B.M., An improved analytic expression for write amplification in NAND flash // Computing, Networking and Communications (ICNC), 2012 International Conference on

[9] Wang W.-N., A simplified Model of Write Amplification for Solid State Drives Adopting Page level Address Translation Mechanism // ICEEAC, pp.2156-2160, 2010.

The estimation of dependency SSD wear level and write speed on overprovision and block size
Snegova E.A., Aliseychik P.A., Alexeev D.V.

This paper is devoted to estimation of SSD write amplification dependency on R — overprovision and N — number of sectors in a block. Garbage collector uses greedy algorithm.

More precise formula than in[1] is obtained because it does not depend on $N \to \infty$ assumption. Proved that for fixed overprovisioning the write amplification is monotonic increasing function of N.

Keywords: Write amplivication, overprovision, SSD, garbage collector.

Часть 2.
Специальные вопросы теории интеллектуальных систем

Некорректность интуиционистской логики относительно L-реализуемости.

Коновалов А. Ю.

Для каждого счетного расширения L языка арифметики определяется абсолютная L-реализуемость предикатных формул. Доказывается, что интуиционистская логика не является корректной относительно этих семантик.

Ключевые слова: конструктивная семантика, реализуемость, абсолютная реализуемость, формальная арифметика, интуиционистская логика.

Будем считать, что язык формальной арифметики LA содержит обозначения для всех примитивно рекурсивных функций, а также константы для обозначения всех натуральных чисел. Расширение LA' языка LA получается добавлением к LA предикатных символов P_i^n и функциональных символов f_i^n для всех $i \geq 0,\ n \geq 1$. Валентность символов P_i^n и f_i^n полагается равной n. Формулы языка LA' строятся обычным образом из атомов и логических констант $\top, \bot$ при помощи логических связок $\wedge$, $\vee$, $\to$ и кванторов $\exists$, $\forall$. Выражение $\neg A$ условимся рассматривать как сокращение для формулы $A \to \bot$. Будем считать, что фиксирована геделева нумерация языка LA'. Формулу языка LA' с геделевым номером z обозначаем через Φ_z. Через $\ulcorner \Phi \urcorner$ обозначаем геделев номер формулы Φ.

Фиксируем расширение L языка LA и интерпретацию $\mathcal{N}_L$ языка L такие, что L — подъязык языка LA', а интерпретация $\mathcal{N}_L$ является продолжением стандартной интерпретацией языка LA. *Униформизацией* формулы $\Phi(x_1, \ldots, x_n, y)$ языка L, не содержащей параметров, отличных от $x_1, \ldots, x_n, y$, будем называть формулу

$$\Phi(x_1, \ldots, x_n, y) \wedge (\forall y_1 < y)\, \neg \Phi(x_1, \ldots, x_n, y_1),$$

которую обозначим $\Phi^U(x_1, \ldots, x_n, y)$. Каждая такая формула задает частичную функцию $f : \mathbb{N}^n \to \mathbb{N}$, где $f(k_1, \ldots, k_n) = k$, если и только

если $\mathcal{N}_L \models \Phi^U(k_1, \ldots, k_n, k)$, т. е. формула $\Phi^U(k_1, \ldots, k_n, k)$ истинна в интерпретации $\mathcal{N}_L$. Через I_n^L обозначаем множество геделев номеров формул языка L, не содержащих параметров отличных от $x_1, \ldots, x_n, y$. Если $z \in I_n^L$, то посредством $\varphi_z^{L,n}$ обозначим n-местную частичную функцию, задаваемую формулой Φ_z^U. В выражениях вида $\varphi_z^{L,n}$ обычно будем опускать второй верхний индекс там, где он может быть восстановлен из контекста.

Будем говорить, что частичная функция $\psi(x_1, \ldots, x_n)$ определима в языке L формулой $A(x_1, \ldots, x_n, y)$ этого языка, если имеет место $(k_1, \ldots, k_n) = n \iff \mathcal{N}_L \models A(k_1, \ldots, k_n, n)$ для всех натуральных чисел $k_1, \ldots, k_n, n$. Отметим, что n-местная функция ψ определима в языке L тогда и только тогда, когда найдется натуральное число $z \in I_n^L$, для которого выполняется соотношение $\psi(x_1, \ldots, x_n) \simeq \varphi_z^{L,n}(x_1, \ldots, x_n)$.

Представляет интерес рассмотрение варианта конструктивной логики, основанного на использовании определимых в языке L функций как конструктивного способа получения одних реализаций из других. Понятие L-реализуемости для языка LA можно определить по аналогии с рекурсивной реализуемостью Клини [1, §82]. Однако нетрудно убедиться, что возникающая при этом семантика совпадает со стандартной классической семантикой языка LA. Поэтому более уместным представляется рассмотрение L-реализуемости сразу в контексте абсолютной реализуемости предикатных формул [2].

Предикатные формулы строятся обычным образом из атомов $P(v_1, \ldots, v_n)$, где P есть n-местная предикатная переменная, а $v_1, \ldots, v_n$ — предметные переменные, при помощи логических констант $\top$, $\bot$, связок $\wedge$, $\vee$, $\to$ и кванторов $\forall$, $\exists$.

Пусть фиксированы примитивно-рекурсивные двухместная функция c, которая взаимно однозначно нумерует все пары натуральных чисел, и одноместные обратные функции p_1 и p_2, так что выполняются соотношения $\mathsf{p}_1(\mathsf{c}(x, y)) = x$ и $\mathsf{p}_2(\mathsf{c}(x, y)) = y$. В выражениях вида $\mathsf{p}_1(t)$, $\mathsf{p}_2(t)$ обычно будем опускать скобки.

Следуя [2], n-местным обобщенным предикатом будем называть всякую функцию типа $\mathbb{N}^n \to 2^{\mathbb{N}}$. Пусть A — предикатная формула, f — отображение, которое каждой предикатной переменной из A ставит в соответствие обобщенный предикат соответствующей валентности. В этом случае отображение f будем называть оценкой формулы A. Временно введем в язык логики предикатов константы для обозначения всех натуральных чисел. Формулы с этими константами будем называть предикатными формулами расширенного языка.

Для каждого натурального числа e, произвольной замкнутой предикатной формулы расширенного языка A и оценки f определим отношение $e\,\mathbf{r}_f^L\,A$ (число e реализует A при оценке f):

1) неверно $e\,\mathbf{r}_f^L\,\bot$;

2) верно $e\,\mathbf{r}_f^L\,\top$;

3) $e\,\mathbf{r}_f^L\,P(a_1,\ldots,a_n) \rightleftharpoons e \in f(P)(a_1,\ldots,a_n)$, если P есть n-местная предикатная переменная;

4) $e\,\mathbf{r}_f^L\,(\Phi\wedge\Psi) \rightleftharpoons \mathsf{p}_1 e\,\mathbf{r}_f^L\,\Phi$ и $\mathsf{p}_2 e\,\mathbf{r}_f^L\,\Psi$;

5) $e\,\mathbf{r}_f^L\,(\Phi\vee\Psi) \rightleftharpoons (\mathsf{p}_1 e = 0$ и $\mathsf{p}_2 e\,\mathbf{r}_f^L\,\Phi)$ или $(\mathsf{p}_1 e = 1$ и $\mathsf{p}_2 e\,\mathbf{r}_f^L\,\Psi)$;

6) $e\,\mathbf{r}_f^L\,\exists x\,\Phi(x) \rightleftharpoons \mathsf{p}_2 e\,\mathbf{r}_f^L\,\Phi(\mathsf{p}_1 e)$;

7) $e\,\mathbf{r}_f^L\,\forall x_1,\ldots,x_n\,(\Phi(x_1,\ldots,x_n)\to\Psi(x_1,\ldots,x_n)) \rightleftharpoons e \in I_{n+1}^L$ и для всех[1] $s,\ a_1,\ldots,a_n \in \mathbb{N}$ верно

$$s\,\mathbf{r}_f^L\,\Phi(a_1,\ldots,a_n) \Longrightarrow !\varphi_e^L(a_1,\ldots,a_n,s) \text{ и } \varphi_e^L(a_1,\ldots,a_n,s)\,\mathbf{r}_f^L\,\Psi(a_1,\ldots,a_n),$$

если $n \geq 0$;

8) $e\,\mathbf{r}_f^L\forall x_1,\ldots,x_n\,\Phi \rightleftharpoons e\mathbf{r}_f^L\forall x_1,\ldots,x_n\,(\top\to\Phi)$, если $n>0$, формула Φ не начинается с квантора $\forall$, и логическая связка $\to$ не является главной в Φ.

Будем говорить, что замкнутая предикатная формула A является *абсолютно L-реализуемой*, если для любой оценки f формулы A найдется такое натуральное число e, что $e\,\mathbf{r}_f^L\,A$. По аналогии с определением примитивно рекурсивно реализуемой секвенции из работы С. Салехи [3] распространим на секвенции понятие абсолютной L-реализуемости.

$$e\,\mathbf{r}_f^L\,A(\overline{x}) \Rightarrow B(\overline{x}) \rightleftharpoons e\,\mathbf{r}_f^L\,\forall\overline{x}\,(A(\overline{x})\to B(\overline{x})),$$

где $\overline{x} = x_1,\ldots,x_n$. Будем говорить, что секвенция $A \Rightarrow B$ является *абсолютно L-реализуемой*, если для любой оценки f предикатных формул A и B найдется такое натуральное число e, что $e\,\mathbf{r}_f^L\,A \Rightarrow B$.

Верна следующая теорема.

Теорема 1. *Секвенция $\top\to P(x)\Rightarrow P(x)$ не абсолютно L-реализуема.*

Из теоремы 1 следует, что формула $\forall x\,((\top\to P(x))\to P(x))$ не является абсолютно L-реализуемой. Таким образом, справедлива следующая теорема.

Теорема 2. *Интуиционистское исчисление предикатов не является корректным относительно семантики абсолютной L-реализуемости.*

[1] Однако, если в списке $x_1,\ldots,x_n$ на некоторых позициях i и j стоят одинаковые переменные x_i и x_j, то мы не допускаем рассмотрение тех списков $a_1,\ldots,a_n$, в которых $a_i \neq a_j$.

Список литературы

[1] Клини С. К. Введение в метаматематику. М.: ИЛ, 1957.

[2] Плиско В. Е. Абслолютная реализуемость предикатных формул // Изв. АН СССР. Сер. матем. 1983. **47**, №2. 315–334.

[3] Salehi S. Primitive recursive realizability and basic arithmetic // Bull. Symbol. Logic. 2001. **7**, N 1. 147—148.

The intuitionistic logic is not sound with L-realizability.
Konovalov A. Yu.

An absolute L-realizability of predicate formulas is introduced for all countable extensions L of the language of arithmetic. It is proved that the intuitionistic logic is not sound with this semantics.

Keywords: constructive semantics, realizability, absolute realizability, formal arithmetic, intuitionistic logic.

О семантическом анализе юридических текстов

Перпер Е.М., Гасанов Э.Э., Кудрявцев В.Б.

Рассматривается задача создания программы, осуществляющей семантический анализ юридических текстов на русском языке. Получив на вход текст, программа должна выдать схему вычисления значений всех сущностей, описанных в тексте. В работе приведены приемы, позволяющие автоматически строить такую схему при наличии морфологической информации о всех словах текста.

Ключевые слова: семантический анализ, синтаксический анализ, юридический текст, логическая формула, прием.

1. Введение

Обычно проверить для текста, правильно ли выделен его смысл, сложно, но в некоторых частных случаях сделать это проще. Одним из таких частных случаев является построение по тексту какого-либо закона, посвященного бухгалтерскому учету, схем вычисления значений упомянутых в законе объектов. Можно считать, что смысл закона выделен правильно, если и только если вычисленные по этим схемам значения объектов совпадают со значениями, вычисленными в соответствии с текстом закона.

Программе, создающей такие схемы, легко найти практическое применение, например, она могла бы использоваться в ERP-системах (Enterprise Resource Planning — Управление ресурсами предприятия), позволяющих бухгалтерам автоматически заполнять формы отчетности. При изменениях в законодательстве, касающихся изменения правил заполнения форм отчетности, производителям ERP-систем ([1, 2] и т.д.) приходится вручную менять программу. Этого не потребуется, если программа будет создаваться автоматически по тексту нормативно-правового акта.

Задачу построения данных схем предлагается решать в несколько этапов.

Первым этапом является создание по каждому предложению рассматриваемого текста его синтаксического дерева (дерева зависимостей) — дерева связей между словами в предложении. Определение дерева зависимостей см., например, на сайте Национального корпуса русского языка [3].

Большая часть парсеров — программ, осуществляющих синтаксический анализ — основывается либо на машинном обучении на синтаксически размеченных корпусах [4], либо на использовании строго определенных правил, позволяющих находить связи между словами [5].

Для программ первого типа необходима большая база синтаксически размеченных текстов (то есть, предложений, для которых синтаксический граф уже построен). Для русского языка можно выделить две базы синтаксически размеченных текстов — это Национальный корпус русского языка (доступ к этому корпусу ограничен) и проект Universal Dependencies [6]. Проект содержит корпуса для более чем 60 языков; цель проекта — разработка единой модели синтаксической разметки для разных языков [7]. Корпус русского языка проекта Universal Dependencies содержит более 66000 предложений, и его можно свободно использовать для обучения парсеров.

Что касается программ, работающих на основе правил, то они не требуют базы данных синтаксически размеченных текстов. Тем не менее, создание набора правил, который позволял бы проводить синтаксический анализ произвольных предложений — сложная задача. Одной из моделей языка, используемых программами, осуществляющими обработку естественного языка на основе правил, является модель «Смысл $\Leftrightarrow$ Текст» (Мельчук [8]). Вводя эту модель, Мельчук постулирует, что естественный язык — это преобразователь из текста в смысл и обратно. Мельчук строит формальную модель языка, состоящую из нескольких уровней (в т.ч., семантического и синтаксического), и вводит наборы правил, с помощью которых должен совершаться переход обрабатываемого представления текста на другой уровень. В идеале эти наборы правил должны обеспечивать переход от текста к его смыслу и обратно. Строго описанные правила оказалось удобно использовать в различных программах, осуществляющих обработку естественного языка (не обязательно русского), например, для автоматической генерации текста по определенным правилам [9].

Вероятно, наиболее развитой системой, осуществляющей обработку русского языка, в настоящее время является Abbyy Compreno. Она проводит синтаксический и семантический анализ текста [10], а также ис-

пользуется в программах, извлекающих информацию из неструктурированного текста, и программах, проводящих классификацию документов. Доступ к этой системе ограничен.

Для автора работы основным недостатком парсеров, имеющихся в свободном доступе, оказалась крайняя трудоемкость исправления совершаемых ими ошибок. По этой причине автор был вынужден отказаться от их использования.

В данной работе парсер строится на основе правил. Тот факт, что в работе рассматриваются не произвольные тексты, а тексты нормативно-правовых актов, касающихся бухгалтерского учета, значительно упрощает создание необходимого набора правил.

После синтаксического анализа текста осуществляется семантический анализ — выделение в тексте семантических (смысловых) связей между его фрагментами. В настоящей работе семантический анализ состоит в том, что по каждому синтаксическому графу строится формула логики предикатов, которая определяет условия, накладываемые на рассматриваемые в тексте объекты.

Наконец, по всем формулам вместе создается модель закона, представляющая собой совокупность схем вычисления значений объектов, упомянутых в законе. Каждая такая схема является своеобразным аналогом алгебраического дерева вычислений [11]. Построение формул логики предикатов и создание модели закона осуществляются, как и синтаксический анализ, на основе строго определенных правил (также называемых в настоящей работе приемами).

Общая идея алгоритма построения модели закона впервые была опубликована в [12], а идея используемого в работе метода синтаксического анализа — в работе [13]. Часть приемов, приведенных в этих работах, используется и в настоящей работе.

2. Основные понятия и формулировка результатов

Введем понятие модели юридического документа в соответствии с тем, как это было сделано в [12].

Рассмотрим ориентированный граф, каждой вершине которого сопоставлена некоторая процедура одного из следующих видов:

1)получение значения из конкретного поля в памяти. Считается, что это значение передается по всем ребрам, выходящим из такой вершины;

2)запись значения в конкретное поле в памяти. В такую вершину должно вести единственное ребро. Считается, что записываемое значение поступает в данную вершину по этому ребру;

3)вычисление некоторой арифметической функции одного или двух аргументов, логической функции одного или двух аргументов либо унарного или бинарного отношения. В такую вершину должно входить столько ребер, сколько аргументов у функции или отношения. По каждому из этих ребер в вершину поступает значение соответствующего аргумента функции (отношения). Значение функции (отношения) передается по всем ребрам, выходящим из этой вершины.

4)выбор одного значения из нескольких. В эту вершину для некоторого натурального числа $m \geq 2$ должно вести $m + 1$ ребро. Эти ребра должны быть пронумерованы числами от 0 до m. По ребру с номером m в вершину поступает число $i \in Z, 0 \leq i \leq m - 1$. По каждому ребру, выходящему из вершины, передается значение, поступившее в вершину по ребру с номером i.

Будем относить вершину к i-му виду вершин, $i \in \{1, 2, 3, 4\}$, если ей сопоставлена процедура i-го вида.

Из вершин графа 2-го вида выделим одну.

Этому графу сопоставляется процедура его вычисления: для выделенной вершины вычисляется значение, поступающее в нее по единственному входящему ребру. Это значение вычисляется с помощью процедуры, сопоставленной вершине, из которой выходит соответствующее ребро.

Для любой вершины 1-го, 3-го, 4-го вида вычисление значения, передаваемого по ребрам, выходящим из этой вершины, происходит следующим образом.

Для вершины 1-го вида это значение берется из соответствующего поля в памяти.

Для вершины 3-го вида сначала вычисляется значение, поступающее в вершину по одному из входящих в нее ребер. Если этого недостаточно для вычисления значения функции (отношения), то вычисляется значение, поступающее в вершину по другому ребру. В любом случае, дальше вычисляется значение функции (отношения) на полученных значениях аргументов, это значение и передается по выходящим из вершины ребрам.

Для вершины 4-го вида сначала вычисляется число i, поступающее в вершину по входящему в него ребру с максимальным номером, затем

вычисляется значение, поступающее в вершину по ребру номер i. Это значение и передается по выходящему из вершины ребру.

Если результат процедуры вычисления графа при любых исходных значениях полей совпадает с результатом заполнения соответствующего поля в форме отчетности в соответствии с текстом закона, назовем этот граф моделью вычисления поля. Граф, содержащий в себе в качестве подграфа модель вычисления любого поля из формы отчетности, назовем моделью юридического документа.

Первым этапом построения модели юридического документа по его тексту является синтаксический анализ.

Синтаксическое отношение — это отношение между парой слов предложения, причем одно из слов является главным в этом отношении, а другое — зависимым. Каждое синтаксическое отношение в зависимости от частей речи участвующих в отношении слов, их морфологических характеристик и т.д., может быть отнесено к определенному классу синтаксических отношений.

На вход программе, осуществляющей синтаксический анализ, поступает последовательность *лексем*. Лексема, в свою очередь, представляет собой последовательность символов. Это может быть слово, число, знак препинания. Текст предложения разбивается на последовательность лексем в процессе *лексического анализа*.

Помимо последовательности лексем, на вход синтаксическому анализу поступает последовательность *токенов*. Токен создается для каждого слова предложения в процессе морфологического анализа и представляет собой тройку, в которую входят: само слово; *лемма* — каноническая форма слова (например, для существительного это будет то же слово, но в именительном падеже и единственном числе); набор *морфологических характеристик* (для существительного это род, падеж, число и т.д., для глагола это вид, время и т.д).

Выходом синтаксического анализа является *синтаксическое дерево*, также называемое *деревом зависимостей*. Это ориентированное дерево. Каждой его вершине сопоставлен токен. Дуга в синтаксическом дереве ведет из вершины A в вершину B тогда и только тогда, когда сопоставленные этим вершинам слова связаны в русском языке синтаксическим отношением, причем главным в этом отношении является слово, соответствующее вершине A. Дуге при этом сопоставлено название отношения.

В данной работе, как и в [13], для синтаксического анализа предлагается использовать (возможно, в несколько упрощенном виде) подход, применяемый А.С. Подколзиным для автоматического решения различ-

ных математических задач [14]. В применении к синтаксическому анализу этот подход состоит в следующем. В программе имеется список *правил*, которые позволяют находить синтаксические связи между словами. Для каждого токена и каждого правила проверяется, применимо ли это правило к данному токену; если да, то создается продиктованное этим правилом синтаксическое отношение.

Каждое правило состоит из трех частей. Первая часть проверяет, подходит ли слово для этого правила: обладает ли оно нужным набором морфологических характеристик. В большинстве случаев значение леммы не проверяется, однако есть и правила, которые работают с конкретными леммами. В тех случаях, когда целью правила является построения синтаксического отношения, в котором рассматриваемое слово было бы зависимым, проверяется также, что слово еще не является зависимым ни в каком построенном синтаксическом отношении. Объясняется эта проверка просто: каждое слово может входить в какое угодно число синтаксических отношений в качестве главного, но лишь в одно — в качестве зависимого.

Вторая часть заключается в поиске слова, которое может входить в синтаксическое отношение с рассматриваемым словом. В некоторых правилах ищется не одно слово, а несколько, притом таких, что каждое из них могло бы образовывать синтаксические отношения либо с другим найденным словом, либо с рассматриваемым словом.

Наконец, третья часть строит синтаксические отношения между рассматриваемым словом и найденными словами. В том случае, если слово прошло проверку в первой части правила, и для него были найдены подходящие слова во второй части правила, будем говорить, что правило *применимо* к слову. Таким образом, если правило применимо к слову, то в результате применения правила к этому слову строится одно или несколько новых синтаксических отношений.

Надо заметить, что описанный в работе алгоритм включает в себя элементы семантического анализа — это видно по названию некоторых отношений, по использованию списков существительных, формируемых исходя из смыслового значения этих существительных, и по нескольким другим признакам. Дело в том, что определенную работу по семантическому анализу удобно выполнять одновременно с синтаксическим анализом.

Для большей части создаваемых алгоритмом синтаксических отношений обозначения классов, к которым относятся эти отношения, взяты из [5], но для некоторых отношений используются специальные названия:

это позволяет облегчить работу с синтаксически разобранным предложением на следующих этапах.

Второй этап построения модели закона состоит в упрощении синтаксического графа каждого предложения. Цель этого упрощения — получить граф, в котором каждой вершине соответствует одна сущность (либо одно или несколько служебных слов).

На третьем этапе построения модели юридического документа происходит отождествление сущностей из различных предложений. В результате одну и ту же сущность во всех построенных на следующем этапе формулах будет выражать одна и та же переменная.

На четвертом этапе по каждому предложению исходного текста и соответствующему синтаксическому графу строится логическая формула. В результате связи между сущностями, о которых идет речь в предложении, оказываются выраженными с помощью математических операций.

На пятом этапе по совокупности всех формул для каждого поля из формы отчетности строится модель вычисления этого поля. Каждая такая модель фактически является программой, вычисляющих значение поля в соответствии с необходимыми и достаточными для этого данными. Совокупность всех таких моделей — модель закона — и будет той программой, которую нужно создать.

Для каждого этапа решения задачи приведены приемы, с помощью которых этот этап осуществляется. Приведенных приемов достаточно, чтобы построить фрагмент модели положения ПБУ 6/01 [15] по пунктам этого закона, касающимся годовой суммы амортизационных отчислений по объектам основных средств. Дальнейшие исследования будут касаться накопления большого числа приемов, что позволит строить модели различных нормативно-правовых актов.

3. Построение связного синтаксического графа предложения

Прежде чем перейти к описанию правил, заметим, что на слова, которые могли бы образовывать синтаксические отношения друг с другом, накладываются определенные ограничения. Если некоторая часть предложения заключена в круглые скобки, то никакое слово из этой части не может быть главным в синтаксическом отношении, в котором зависимое слово находится вне этих скобок. Применению основного алгоритма построения синтаксического дерева предшествует определение глубины

вложенности каждого слова предложения в круглые скобки (в дальнейшем будем коротко именовать ее глубиной). Эта глубина вычисляется как разность числа открывающих и числа закрывающих круглых скобок перед словом.

Подробнее остановимся на том, как осуществляется вторая часть правила. Происходит последовательный перебор всех слов предложения, осуществляемый либо в сторону начала предложения, либо в сторону его конца. Перебор начинается со слова, следующего после рассматриваемого слова в том направлении, в котором осуществляется перебор. Если рассматриваемое слово должно быть главным в синтаксическом отношении и имеет большую глубину, чем слово, до которого дошел перебор, то перебор завершается неудачей. Если рассматриваемое слово должно быть зависимым в синтаксическом отношении и имеет меньшую глубину, чем слово, до которого дошел перебор, то перебор также завершается неудачей. Если перебор дошел до начала либо конца предложения, а в некоторых случаях — до начала либо конца *клаузы* (простого предложения в составе сложного), в которой содержалось рассматриваемое слово, и нужное слово не найдено, перебор снова заканчивается неудачей. Если же нужное слово найдено, перебор успешно завершается, и найденное слово используется в следующей части правила для построения синтаксического отношения. В тех случаях, когда ищется несколько слов, может быть осуществлено несколько переборов.

Будем называть слово *допустимым*, если его глубина не является препятствием для включения этого слова в синтаксическое отношение.

Перед основным алгоритмом построения синтаксических отношений, совершается несколько подготовительных действий.

1) Для каждого тире в предложении создается токен, который в качестве леммы содержит само тире, а его набор морфологических характеристик состоит из одного элемента — указания на то, что тире следует рассматривать как глагол. Действительно, в большинстве случаев тире заменяет глагол «есть» в какой-либо форме.

2) Предложение разбивается на сегменты по некоторым знакам препинания — запятым, точкам с запятой, двоеточиям, притом рассматриваются только те из них, что имеют нулевую глубину. Во многих случаях сегмент совпадает с клаузой, но не всегда, так как запятые не только разделяют простые предложения в составе сложного, но и выполняют другие роли (например, отделяют друг от друга однородные члены предложения).

3) В предложении выделяются группы идущих подряд токенов, которые с точки зрения синтаксического анализа рассматриваются как единое целое. Делается это по той причине, что применение общих правил построения синтаксических отношений к словам из этой группы может привести к ошибкам, поэтому необходимо выделить эти группы до основного цикла построения синтаксических отношений.

   Группа токенов, о которой идет речь, может представлять собой

   а) предлог: «исходя из», «в течение», «в отношении», «в соответствии с», «в связи с» и др.;

   б) союз: «а также», «прежде чем» и др.;

   в) устойчивое словосочетание: «иметь место» и др.;

   г) графическое сокращение: «и т.д.», «и т.п.» и др.

   Алгоритм использует отдельный словарь, содержащий рассматриваемые группы токенов. Каждая группа токенов из рассматриваемого предложения ищется в словаре, и в случае успеха поиска каждая пара идущих подряд слов из этой группы соединяется отношением «НЕДЕЛИМ», где главным объявляется то из двух слов, которое находится ближе к началу предложения. Кроме того, если группа представляет собой предлог или союз, то считается, что каждое слово из группы является соответственно предлогом или союзом.

4) При синтаксическом анализе юридических текстов удобно также выделить группы токенов, обозначающие статьи, пункты, подпункты нормативных актов. Пусть в предложении непосредственно после слова «статья», «пункт», «подпункт» находятся один или несколько токенов, каждый из которых соответствует некоторой букве или числительному. Тогда для каждого такого токена создается отношение «ПУНКТ», в котором этот токен является зависимым, а токен, соответствующий слову «статья», «пункт» или «подпункт» — главным.

В основном алгоритме синтаксического анализа, помимо попыток применения к слову правил построения синтаксических отношений, для этого слова производятся некоторые вычисления, которые могут быть в дальнейшем использованы при рассмотрении не только этого слова, но и

последующих слов. Например, таким образом находится и запоминается последний встретившийся в рассмотренной части предложения глагол, а также последний встретившийся глагол перед последним встретившимся двоеточием.

Приступим, наконец, к описанию правил, из которых состоит основной алгоритм синтаксического анализа. Заметим, что некоторые части речи в процессе применения правил приравниваются к другим частям речи. В большинстве случаев это явно оговаривается. Исключение — местоимения: местоимения-существительные во всех правилах приравниваются к существительным, а местоимения-прилагательные — к прилагательным, если находятся непосредственно перед прилагательным или существительным, и к существительным в противном случае. В правилах это явно не оговаривается, но, например, когда речь в правиле идет о существительном, предполагается, что это может быть также и местоимение-существительное, и местоимение-прилагательное, если оно приравнено к существительному.

Правила 1 и 2 применяются к тире. С их помощью осуществляется попытка соединить тире с глаголом или причастием, которые были заменены этим тире. Эти правила (как и некоторые другие) относятся к числу сложных, так как с их помощью создаются связи между различными сегментами предложения.

1) Правило, создающее для тире синтаксическое отношение «ВАРИАНТ» между этим тире и последним глаголом, находящимся перед последним двоеточием перед тире. Если такой глагол существует, то он был найден ранее. В создаваемом отношении этот глагол является главным элементом, а тире — зависимым.

2) Правило, которое для тире проверяет, является ли союз первым элементом сегмента, в котором это тире находится. Если является, то происходит перебор сегментов от предыдущего к первому в предложении, пока первым элементом сегмента является тот же союз и не достигнуто начало предложения. Если у сегмента, на котором перебор прекратился, вторым словом является тот же союз, а первым — глагол или причастие, то между этим глаголом (причастием) и тире создается синтаксическое отношение «ВАРИАНТ», в котором глагол (причастие) — главный элемент, а тире — зависимый.

3) Правило, которое для слова «не» ищет ближайшее находящееся после него допустимое слово и создает синтаксическое отношение «ОТР» («отрицание»), в котором найденное слово является главным элементом, а «не» — зависимым.

4) Правило, которое для существительного ищет ближайшее находящееся после него допустимое слово, не являющееся прилагательным в родительном падеже. Если такое слово найдено, и это — существительное в родительном падеже, то создается синтаксическое отношение «ГЕНИТ-ИГ» («генитивная именная группа») с рассматриваемым словом в качестве главного элемента и найденным существительным в родительном падеже в качестве зависимого элемента.

5) Правило, ищущее для предлога «на» ближайшее к нему среди находящихся перед ним слов сегмента допустимое слово из списка существительных, обозначающих величину или значение («стоимость», «цена», «размер» и т.д.). Если между этим существительным и рассматриваемым предлогом «на» не расположены глагол или тире, создается синтаксическое отношение «ОПР» («определительное») с найденным существительным в качестве главного слова и предлогом «на» в качестве зависимого слова. Это правило, как и следующие два, касаются случаев присоединения предлога к существительному. Случай присоединения предлога к глаголу рассматривается после этих правил и действует «по умолчанию».

6) Правило, ищущее для предлога «от» ближайшее к нему среди находящихся перед ним слов сегмента допустимое слово из списка существительных, обозначающих доход («доход», «выручка» и т.д.). Если между этим существительным и рассматриваемым предлогом не расположены глагол или тире, создается синтаксическое отношение «ОПР» с найденным существительным в качестве главного слова и предлогом «от» в качестве зависимого слова.

7) Правило, ищущее для предлога «по» ближайшее к нему среди находящихся перед ним слов сегмента допустимое слово из списка существительных («положение», «списание» и т.д.). Если между этим существительным и рассматриваемым предлогом «по» не расположены глагол или тире, создается синтаксическое отношение «ОПР» с найденным существительным в качестве главного слова и предлогом «по» в качестве зависимого слова.

8) Правило, ищущее для предлога «о» («об») ближайшее к нему среди находящихся перед ним слов сегмента допустимое слово из списка существительных («законодательство», «информация», «решение» и т.д.). В случае успеха поиска создается синтаксическое отношение «ОПР» с найденным существительным в качестве главного слова и предлогом «о» («об») в качестве зависимого слова.

9) Правило, проверяющее для существительного в именительном падеже, есть ли глагол в сегменте, где это существительное находится. Пусть глагола там нет, и непосредственно перед последним двоеточием, встретившимся до этого сегмента, находится то же самое существительное (возможно, в другом падеже). Тогда создается отношение «А_ИМЕННО», где главным словом является найденное перед двоеточием существительное, а зависимым — рассматриваемое слово.

10) Правило, рассматривающее союз «то есть» или его графическое сокращение «т.е.». Если непосредственно перед этим союзом находится существительное, то осуществляется поиск допустимого существительного после этого союза, и в случае удачи создаются два отношения: «А_ИМЕННО», где главным словом является существительное, находящееся перед союзом, а зависимым — сам союз, и «СОЮЗ», где главным словом является этот союз, а зависимым — существительное, находящееся после него.

11) Правило, ищущее для количественного числительного ближайшее к нему среди находящихся после него допустимых существительных в родительном падеже и подходящем числе. Если такое существительное нашлось, и между ним и рассматриваемым количественным числительным нет ничего, кроме союзов и других количественных числительных, то строится синтаксическое отношение «КОЛИЧ» («количественное»), где найденное слово является главным, а числительное — зависимым словом. Если последнее допустимое слово предложения, находящееся перед числительным, обозначает количественное отношение («больше», «ниже», «равное» и т.д.), то создается также синтаксическое отношение «ДОП» («дополнение»). В этом отношении главным объявляется слово, обозначающее количественное отношение, а зависимым — ранее найденное существительное.

12) Правило, ищущее для порядкового числительного ближайшее к нему среди находящихся после него и согласованных с ним допустимых существительных. Если такое существительное нашлось, и между ним и рассматриваемым количественным числительным нет ничего, кроме союзов и других количественных числительных, то строится синтаксическое отношение «НОМЕР», где найденное слово является главным, а числительное — зависимым словом.

13) Правило, проверяющее, является ли последнее допустимое слово предложения, идущее в предложении до числительного, словом, которое может обозначать количественное отношение («больше», «ниже», «равное» и т.д.). Если это так, то строится синтаксическое отношение «КОЛИЧ», где найденное слово является главным, а числительное — зависимым словом.

14) Правило, которое для предлога, а также для слов «исходя» (если следующее за ним слово – «из») и «как», ищет допустимый глагол, чтобы связать их синтаксическим отношением «ГЛ_ДОП» («глагол – дополнение»), в котором глагол будет главным словом, а рассматриваемое правилом слово — зависимым. Глагол ищется в том же сегменте, где находится рассматриваемое слово, сначала в направлении от этого слова к концу сегмента, а в случае неудачи — в направлении от этого слова к началу сегмента. В данном правиле к глаголу приравниваются отглагольные существительные и причастия. Если нужного слова найти не удалось, а рассматриваемое правилом слово — «по» или «при», ищется в направлении от него к концу предложения ближайший допустимый глагол, не находящийся в клаузе, начинающейся со слова «который». В случае, когда нужное слово все еще не найдено, в качестве него берется последний глагол, находящийся перед последним (до рассматриваемого правилом слова) двоеточием, если такой существует.

15) Правило, ищущее для существительного, не находящегося в именительном падеже, ближайшее к нему среди находящихся перед ним слов сегмента допустимое слово, являющееся глаголом, существительным, страдательным причастием (не согласованным с существительным), кратким причастием, предлогом либо союзом (но не союзом «или», «и», «а»). Если такого слова нет и сегмент отделен от предыдущего точкой с запятой, то в качестве искомого слова рассматривается последний глагол в первом сегменте пред-

ложения, а при наличии предлога непосредственно после этого глагола — этот предлог. Создается синтаксическое отношение «ДОП» («дополнение»), в котором главным является найденное слово, а зависимым — рассматриваемое существительное.

16) Правило, ищущее для не находящегося в именительном падеже существительного, чей сегмент отделен от предыдущего запятой, первое в предыдущем сегменте (считая от его начала) существительное в том же падеже. Если такое существительное найдено, и оно является зависимым в каком-либо синтаксическом отношении, то добавляется такое же отношение, в котором зависимым словом является рассматриваемое правилом существительное.

17) Правило, ищущее для существительного в именительном падеже допустимый глагол, чтобы связать их синтаксическим отношением «ПОДЛ» («подлежащее»), в котором глагол будет главным словом, а существительное — зависимым. Глагол ищется так же, как и в предыдущем правиле, за тем исключением, что отглагольное существительное в данном случае к глаголу не приравнивается.

18) Правило, которое ищет для наречия допустимый глагол, чтобы связать их синтаксическим отношением «ГЛ_ОБСТ» («глагол – обстоятельство»), где глагол будет главным словом, а наречие — зависимым. Глагол ищется так же, как и в правиле 14.

19) Правило, ищущее для слова «если» допустимый глагол, чтобы связать их синтаксическим отношением «ЕСЛИ», в котором глагол будет главным словом, а существительное — зависимым. Глагол ищется так же, как и в правиле 14.

20) Правило, которое для слова «если», являющегося зависимым в каком-либо синтаксическом отношении, ищет ближайший к нему допустимый глагол, находящийся в предложении до сегмента, в котором находится рассматриваемое слово. Если такое слово найдено, то создается отношение «УСЛ» («условие»), где найденный глагол будет главным словом, а зависимым будет слово, являющееся главным в отношении, где «если» является зависимым.

21) Правило, которое для полного причастия, находящегося непосредственно после запятой, ищет ближайшее к нему в сторону начала предложения существительное, имеющее тот же падеж и число,

причем если это число — единственное, то и тот же род. Если такое существительное нашлось, создается синтаксическое отношение «ПРИЧ_СУЩ» («причастие – существительное»), в котором главным словом является существительное, а зависимым — причастие.

22) Правило, которое для прилагательного или полного причастия ищет ближайшее к нему в сторону конца предложения существительное, имеющее тот же падеж и число, причем если это число — единственное, то и тот же род. Если такое существительное нашлось, создается синтаксическое отношение «ПРИЛ_СУЩ» («прилагательное – существительное») (если рассматриваемое слово — прилагательное) или «ПРИЧ_СУЩ» (если рассматриваемое слово — причастие), где главным словом является существительное, а зависимым — рассматриваемое слово.

23) Правило, обрабатывающее союзы «и», «или», «а», не находящиеся в начале сегмента. Сначала ищется слово w_1 — допустимое слово, ближайшее к рассматриваемому союзу по направлению к началу предложения. Кроме того, ищется слово w_2 — допустимое слово, ближайшее к рассматриваемому союзу по направлению к концу предложения. Если w_2 — полное причастие, краткое причастие, глагол либо отглагольное существительное, то от рассматриваемого союза в сторону начала предложения ищется слово w_3 — ближайшее допустимое слово, удовлетворяющее следующим условиям: если w_2 — полное причастие, то w_3 должно быть полным причастием либо прилагательным, имеющим тот же падеж, что w_2; если w_2 — глагол либо краткое причастие, то найденное слово также должно быть глаголом либо кратким причастием и иметь то же число, что и w_2; если w_2 — отглагольное существительное, то w_3 должно быть существительным в том же падеже.

    Если w_2 не является полным причастием, кратким причастием, глаголом либо отглагольным существительным, ищется ближайшее к союзу по направлению к концу предложения допустимое слово w_4, удовлетворяющее следующим условиям: если w_1 — прилагательное либо полное причастие, то найденное слово также должно быть прилагательным либо полным причастием и иметь тот же падеж, что и w_1; если w_1 — существительное, то найденное слово должно быть существительным и иметь тот же падеж, что и w_1; если w_1 — глагол либо краткое причастие, то найденное слово также должно быть глаголом либо кратким причастием и иметь то же

число, что и w_1. Заметим, что w_1 может совпадать с w_3, а w_2 — с w_4.

Если подходящее слово найдено, то создается два синтаксических отношения «ОДНОР_ИГ» («однородные именные группы»), которые содержит рассматриваемый союз в качестве главного слова и найденные слова w_1 и w_4 (либо w_2 и w_3, если w_2 — полное причастие) в качестве зависимых. Если слово w_1 (w_3) до применения данного правила участвовало в каком-либо синтаксическом отношении в качестве зависимого, то оно заменяется в этом отношении на рассматриваемый союз.

24) Правило, обрабатывающее союзы «и», «или», «а», находящиеся в начале сегмента. Правило в основном аналогично предыдущему, отличие заключается в том, что в качестве w_2 выбирается не любое допустимое слово, ближайшее к рассматриваемому союзу по направлению к концу предложения, а только глагол или краткое причастие. Затем так же, как это делается в предыдущем правиле, для w_2 ищется слово w_3, и с участием этих слов и рассматриваемого союза создаются синтаксические отношения «ОДНОР_ИГ».

25) Правило, проверяющее, является ли последнее допустимое слово предложения, идущее в предложении до количественного числительного в именительном падеже, существительным. Если да, то создается отношение «НОМЕР», в котором найденное существительное — главное слово, а числительное — зависимое.

26) Правило, проверяющее, является ли существительным последнее отличное от частицы «не» допустимое слово предложения, идущее в предложении до слова, обозначающего количественное отношение. Если это так, то строится синтаксическое отношение «ОПР», где найденное существительное — главный элемент, а слово, обозначающее количественное отношение — зависимый.

27) Правило, обрабатывающее слово, чья лемма — «который». Сначала в предыдущем сегменте ищется ближайшее к рассматриваемому слову допустимое существительное, имеющее то же число, что и рассматриваемое слово. Затем ищется ближайшее к рассматриваемому слову по направлению к концу предложения допустимое слово, являющееся глаголом, кратким причастием или тире. Если нужные слова найдены, создается синтаксическое отноше-

ние «ПРИДАТ_ОПР» («придаточное определительное»), в котором главным словом является найденное существительное, а зависимым — найденный глагол, краткое причастие или тире.

Описанные выше правила расположены в том же порядке, в каком их рассматривает алгоритм. Если несколько правил независимо друг от друга пытаются создать разные отношения, в которых одно и то же слово было бы зависимым, сработает только то из правил, которое было рассмотрено раньше. Таким образом, при изменении порядка применения правил результат может измениться.

Помимо правил, касающихся построения синтаксических отношений, имеются три правила, изменяющих уже созданные синтаксические отношения. Все эти правила так или иначе касаются построения множественного актанта — отношения между однородными членами предложения. Перечислим их.

28) Правило, которое для каждого союза «и», «или», «а» просматривает все отношения «ОДНОР_ИГ», где этот союз является главным словом. Если среди зависимых слов есть слова $w_1, w_2, ..., w_k, k \in N$ из списка существительных, обозначающих величину или значение, и ровно одно слово не из этого списка, то отношения, в которых рассматриваемый союз — главный, разрываются, а в том отношении, где этот союз является зависимым, он заменяется на единственное найденное слово не из списка существительных, обозначающих величину или значение. Далее происходит перебор вершин ориентированного дерева от вершины, которой соответствует рассматриваемый союз, по направлению к корню дерева. Пусть вершине, получаемых при этом переборе, соответствует слово w из списка существительных, обозначающих величину или значение. Тогда перебор прекращается; найденное слово в том отношении, где оно является зависимым, заменяется на рассматриваемый союз; создаются отношения «ОДНОР_ИГ», в которых главным словом является рассматриваемый союз, а зависимыми — слова $w, w_1, w_2, \ldots, w_k$.

Данное правило введено по следующей причине: оказывается полезным считать, что существительные, обозначающие значение, могут быть однородными только с существительными, обозначающими значение. Это правило применяется сразу после применения правила 23 или 24.

Следующие три правила применяются тогда, когда для предложения уже построено дерево зависимостей. Эти правила применяются к каждой подходящей вершине дерева.

29) Правило, создающее множественный актант из слов, являющихся зависимыми от одного и того же слова в отношении «ОДНОР_ИГ». Множественный актант удобно представлять как вершину, из которой исходит ребро с меткой «МНА» к каждой вершине, соответствующей слову из множественного актанта. Основная работа по созданию такого множественного актанта уже проделана при применении правил 23 и 24. Теперь же достаточно указать, что каждая вершина, из которой исходит хотя бы одно ребро с меткой «ОДНОР_ИГ» (в действительности, как видно из построения отношений «ОДНОР_ИГ», не бывает вершин, из которых выходило бы ровно одно такое ребро), соответствует множественному актанту, а метку ребра поменять на « МНА ».

30) Правило, создающее множественный актант из слов, являющихся зависимыми от одного и того же слова в отношении «ВАРИАНТ» (т.е. из нескольких тире, заменяющих одно и то же слово) и из самого этого слова. Достаточно указать, что каждая вершина, из которой исходит хотя бы одно ребро с меткой «ВАРИАНТ», соответствует множественному актанту, а метку ребра поменять на « МНА ».

31) Правило, создающее множественный актант из всех слов $w_1, w_2, \ldots, w_k$, являющихся зависимыми от одного и того же слова w в отношении с одним и тем же названием v (но не «ОДНОР_ИГ», так как этот случай рассматривается отдельно, и не «ГЛ_ДОП», так как такие слова не являются однородными). В дерево зависимостей добавляется новая вершина w_0, куда из вершины, соответствующей слову w, опускается ребро с меткой v. Все ребра, ведущие в вершины, соответствующие словам $w_1, w_2, \ldots, w_k$, удаляются, и в каждую из этих вершин из w_0 проводится ребро с меткой « МНА ». Наконец, вершина w_0 помечается как соответствующая множественному актанту.

Приведем пример предложения, для которого описанный алгоритм построил дерево зависимостей. Рассмотрим следующее предложение: «При способе уменьшаемого остатка годовая сумма амортизационных

отчислений определяется исходя из остаточной стоимости объекта основных средств на начало отчетного года и нормы амортизации, исчисленной исходя из срока полезного использования этого объекта и коэффициента не выше 3, установленного организацией». Это предложение — несколько измененная часть пункта 19 ПБУ 6/01 [15].

Будем считать, что на вход программы, осуществляющей синтаксический анализ, поступила последовательность лексем, на которые было разбито предложение, а также следующие токены (они получены благодаря морфологическому разбору, произведенному программой, созданной на проекте АОТ [5]):

1) При ПРИ ПРЕДЛ,
2) способе СПОСОБ С,но,мр,пр,ед,
3) уменьшаемого УМЕНЬШАТЬ ПРИЧАСТИЕ,стр,пе,нс,но,од,нст,мр,рд,ед,
4) остатка ОСТАТОК С,но,мр,рд,ед,
5) годовая ГОДОВОЙ П,но,од,жр,им,ед,
6) сумма СУММА С,но,жр,им,ед,
7) амортизационных АМОРТИЗАЦИОННЫЙ П,но,од,рд,мн,
8) отчислений ОТЧИСЛЕНИЕ С,но,ср,рд,мн,
9) определяется ОПРЕДЕЛЯТЬСЯ Г,дст,нп,нс,Зл,нст,ед,
10) исходя ИСХОДИТЬ нп,нс,
11) из ИЗ
12) остаточной ОСТАТОЧНЫЙ П,кач,но,од,жр,рд,ед,
13) стоимости СТОИМОСТЬ С,но,жр,рд,ед,
14) объекта ОБЪЕКТ С,но,мр,рд,ед,
15) основных ОСНОВНОЙ П,но,од,рд,мн,
16) средств СРЕДСТВО С,но,ср,рд,мн,
17) на НА ПРЕДЛ,

18) начало НАЧАЛО С,но,ср,вн,ед,

19) отчетного ОТЧЕТНЫЙ П,кач,но,од,мр,рд,ед,

20) года ГОД С,но,мр,рд,ед,

21) и И СОЮЗ,

22) нормы НОРМА С,но,жр,вн,рд,им,ед,мн,

23) амортизации АМОРТИЗАЦИЯ С,но,жр,рд,ед,

24) , ,

25) исчисленной ИСЧИСЛИТЬ ПРИЧАСТИЕ,стр,пе,св,но,од,прш,жр,пр,тв, дт,рд,ед,

26) исходя ИСХОДИТЬ нп,нс,

27) из ИЗ

28) срока СРОК С,но,мр,рд,ед,

29) полезного ПОЛЕЗНЫЙ П,кач,но,од,ср,рд,ед,

30) использования ИСПОЛЬЗОВАНИЕ С,но,ср,рд,ед,

31) этого ЭТОТ МС-П,но,од,мр,рд,ед,

32) объекта ОБЪЕКТ С,но,мр,рд,ед,

33) и И СОЮЗ,

34) коэффициента КОЭФФИЦИЕНТ С,но,мр,рд,ед,

35) не НЕ ЧАСТ,

36) выше ВЫШЕ Н,

37) 3 3 ЧИСЛ-П,но,од,жр,тв,ед,

38) , ,

39) установленного УСТАНОВИТЬ ПРИЧАСТИЕ,стр,пе,св,но,од,прш,ср,мр, вн,рд,ед,

40) организацией ОРГАНИЗАЦИЯ С,но,жр,тв,ед,

41) . .

В результате применения описанного алгоритма без использования правил 29 и 31 создаются следующие синтаксические отношения (у отношения сначала указывается главное слово, а потом зависимое):

1) НЕДЕЛИМ(исходя, из) по правилу 3а предварительной обработки (таких отношений создается два, так как «исходя из» встречается в предложении дважды);
2) ГЛ_ДОП(определяется, при) по правилу 14;
3) ГЕНИТ_ИГ(способе, остатка) по правилу 4;
4) ДОП(при, способе) по правилу 15;
5) ПРИЧ_СУЩ(остатка, уменьшаемого) по правилу 22;
6) ПРИЛ_СУЩ(сумма, годовая) по правилу 22;
7) ГЕНИТ_ИГ(сумма, отчислений) по правилу 4;
8) ПОДЛ(определяется, сумма) по правилу 17;
9) ПРИЛ_СУЩ(отчислений, амортизационных) по правилу 22;
10) ГЛ_ДОП(определяется, исходя) по правилу 14;
11) ПРИЛ_СУЩ(стоимости, остаточной) по правилу 22;
12) ГЕНИТ_ИГ(стоимости, объекта) по правилу 4;
13) ДОП(из, стоимости) по правилу 15 (однако затем по правилу 23 это отношение заменяется на отношение
14) ДОП(из, и));
15) ГЕНИТ_ИГ(объекта, средств) по правилу 4;
16) ПРИЛ_СУЩ(средств, основных) по правилу 22;
17) ОПР(стоимости, на) по правилу 5;
18) ГЕНИТ_ИГ(начало, года) по правилу 4;
19) ДОП(на, начало) по правилу 15;

20) ПРИЛ_СУЩ(года, отчетного) по правилу 22;

21) ОДНОР_ИГ(и, стоимости) по правилу 23;

22) ОДНОР_ИГ(и, нормы) по правилу 23;

23) ГЕНИТ_ИГ(нормы, амортизации) по правилу 4;

24) ПРИЧ_СУЩ(амортизации, исчисленной) по правилу 21;

25) ГЛ_ДОП(исчисленной, исходя) по правилу 14;

26) ГЕНИТ_ИГ(срока, использования) по правилу 4;

27) ДОП(из, срока) по правилу 15 (однако затем по правилу 23 это отношение заменяется на отношение

28) ДОП(из, и));

29) ПРИЛ_СУЩ(использования, полезного) по правилу 22;

30) ГЕНИТ_ИГ(использования, объекта) по правилу 4;

31) ПРИЛ_СУЩ(объекта, этого) по правилу 22;

32) ОДНОР_ИГ(и, срока) по правилу 23;

33) ОДНОР_ИГ(и, коэффициента) по правилу 23;

34) ОТР(выше, не) по правилу 3;

35) ОПР(коэффициента, выше) по правилу 26;

36) КОЛИЧ(выше, 3) по правилу 13;

37) ПРИЧ_СУЩ(коэффициента, установленного) по правилу 21;

38) ДОП(установленного, организацией) по правилу 15.

В случае применения правил 29 и 31 к вершинам построенного дерева зависимостей, метки ребер, соответствующих отношениям 21, 22, 32 и 33, поменяются с «ОДНОР_ИГ» на «МНА». Кроме того, вершины, из которых эти ребра выходят, будут помечены как соответствующие множественному актанту. Как можно заметить, правило 29 будет применено дважды, а правило 31 — ни разу.

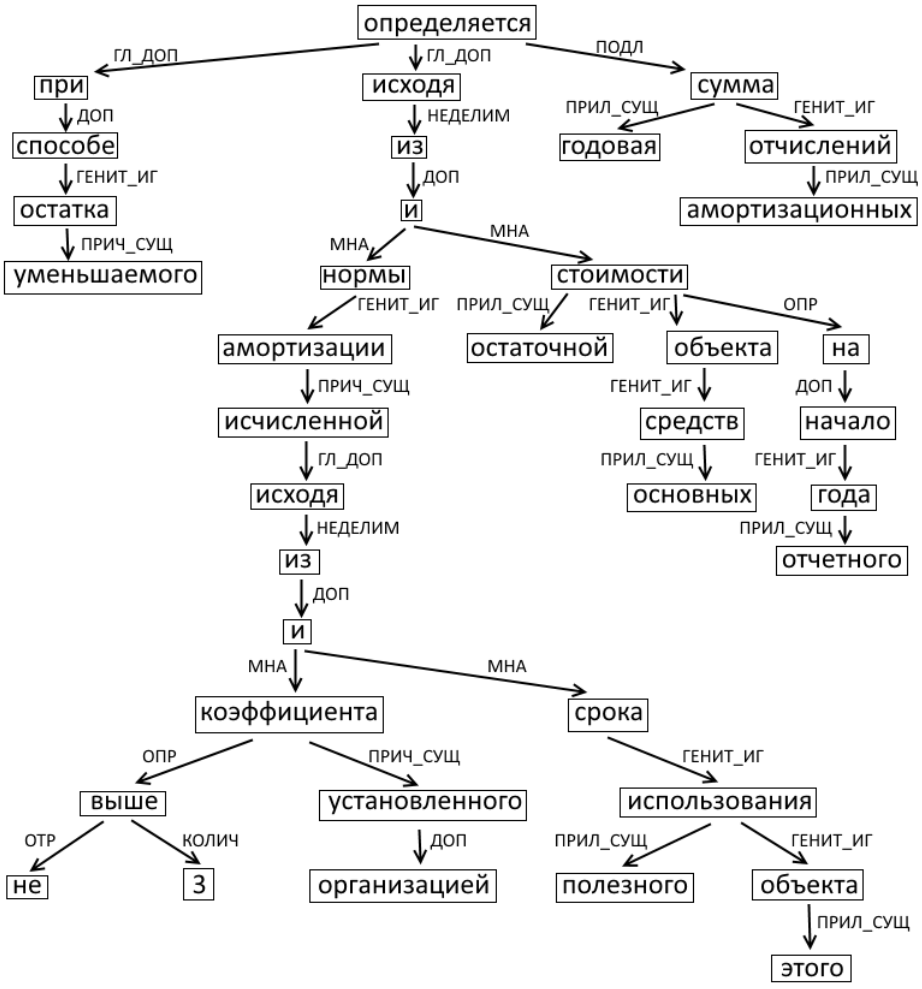


Рис. 1. Синтаксический граф

4. Упрощение синтаксического графа предложения

После синтаксического разбора каждая вершина в графе, за исключением вершин, добавленных в граф по правилу 31, будет соответствовать

некоторому токену. Однако некоторой сущности в предложении может соответствовать не одно слово, а сочетание слов.

Рассмотрим произвольное поддерево синтаксического графа, состоящее из двух или более вершин. Пусть его вершины — $\alpha_1, \alpha_2, \ldots, \alpha_n$, причем α_1 — его корень. Тогда для вершин $\alpha_1, \alpha_2, \ldots, \alpha_n$ определена операция *объединения вершин*, состоящая из следующих шагов:

1) в граф добавляется вершина α;

2) ребро, входящее в вершину α_1, перенаправляется в вершину α;

3) для каждого ребра e из каждой вершины $\alpha_i, i \leq n$, кроме ребер, ведущих в какую-либо вершину $\alpha_j, j \leq n$, проводится ребро из вершины α, ведущее в ту же вершину, что и e, и обладающее той же меткой;

4) вершине α сопоставляется набор всех токенов, сопоставленных вершинам $\alpha_1, \alpha_2, \ldots, \alpha_n$.

5) вершины $\alpha_1, \alpha_2, \ldots, \alpha_n$ и все входящие в них и выходящие из них ребра удаляются.

Граф будем называть *упрощенным синтаксическим графом*, если он получен из синтаксического графа предложения путем нескольких (в т.ч., 0) применений операции объединения вершин. Мы будем строить упрощенный граф таким образом, чтобы в нем каждой вершине соответствовала либо одна сущность, либо одно или несколько служебных слов (предлогов, союзов и т.д).

На объединяемые вершины могут быть наложены определенные ограничения:

а) вершину α может быть запрещено объединять с любой вершиной, в которую из α ведет ребро, если вершине α сопоставлены слова (сочетания слов) «за исключением», «кроме», «числитель», «знаменатель», «соотношение», «а также», «исходя», «один из», «в течение», «по», «при»; если вершине α сопоставлено количественное отношение («больше», «свыше» и т.д.);

б) вершину α может быть запрещено объединять с вершиной, из которой в α ведет ребро, если вершина α соответствует множественному актанту; если ребру, ведущему в вершину α, сопоставлено одно из следующих синтаксических отношений: «ПРИДАТ_ОПР», «В_СЛУЧАЕ», «В_ОТНОШЕНИИ», «ОТР», «УСЛ», «ЕСЛИ», «КОЛИЧ»; если хотя бы одному ребру, выходящему из вершины α, сопоставлено синтаксическое отношение «ПРИДАТ_ОПР».

Цель данных ограничений - избежать объединения вершин, которым приписаны слова, обозначающие логические функции и отношения. В некоторых случаях эти ограничения могут игнорироваться.

Упрощение графа производится с помощью перечисленных ниже приемов. Во всех случаях указано, игнорирует ли прием ограничения а и б.

1) Пусть «способ» — одно из слов, соответствующих некоторой вершине α, и пусть из α в некоторую вершину β ведет ребро, соответствующее синтаксическому отношению «ГЕНИТ_ИГ». Тогда объединяются вершины α, β и все вершины, в которые можно перейти из β. Ограничения а и б игнорируются.

2)Пусть из некоторой вершины α в некоторую вершину β ведет ребро, соответствующее синтаксическому отношению «НЕДЕЛИМ». Тогда объединяются вершины α и β. Ограничения а и б игнорируются.

3) Пусть некоторой вершине α сопоставлен один токен, являющийся предлогом (но не «по», «при», «кроме»), и из α выходит лишь одно ребро. Тогда α объединяется с вершиной, в которую из α ведет ребро. Ограничение а учитывается, ограничение б - игнорируется.

4) Путь несколько идущих подряд слов предложения составляют некоторую сущность из хранящегося в отдельном файле списка сущностей (например, «объект основных средств», «физическое лицо» и т.д). Тогда объединяются все вершины, соответствующие этим словам. Ограничения а и б игнорируются.

5) Пусть из вершины α в вершину β ведет ребро. Тогда α и β объединяются. Ограничения а и б учитываются.

Приемы используются в следующем порядке: сначала происходит попытка применить прием 1 там, где это возможно; затем то же делается с приемом 2. Далее в цикле по всем вершинам происходит попытка применения приема 3, после чего в цикле по всем словам предложения происходит попытка использовать прием 4. Наконец, везде, где возможно, используется прием 5.

Нетрудно заметить, что прием 5 фактически является основным, тогда как почти все остальные приемы нужны лишь для случаев, когда необходимо проигнорировать хотя бы одно из ограничений а и б.

В результате применения указанных приемов к синтаксическому графу, изображенному на рисунке 1, мы получим упрощенный синтаксический граф, изображенный на рисунке 2.

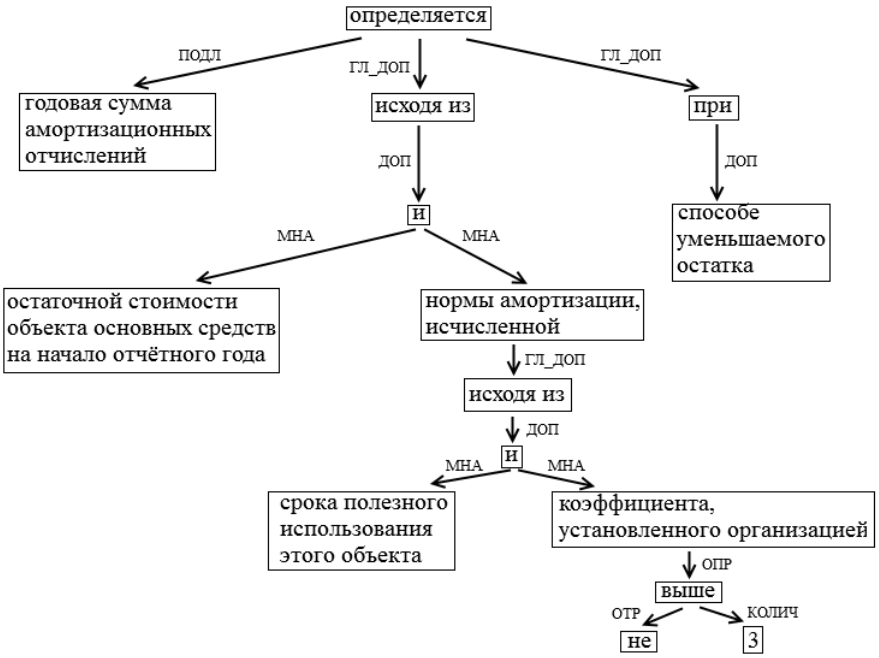


Рис. 2. Упрощенный синтаксический граф

5. Отождествление сущностей

Пусть по каждому предложению текста закона уже построена формула. Переменные этих формул соответствуют определенным текстовым фрагментам.

Перед построением формулы каждой сущности необходимо сопоставить переменную. Для того, чтобы одной и той же сущности в разных предложениях была сопоставлена одна и та же переменная, проводится процедура отождествления сущностей. Часто сущность полностью определяется текстовым фрагментом (предполагается, что если слова в двух фрагментах текста отличаются только своей формой, например, находятся в разных падежах, то эти два фрагмента определяют одну и ту же сущность). В некоторых случаях этого, однако, не происходит. Для представления множества всех сущностей удобно использовать ориентированное дерево, каждой вершине которого (кроме корня) сопоставлена некоторая сущность и некоторый текст. Если в этом дереве из вершины,

которой сопоставлена сущность A, ведет ребро в вершину, которой сопоставлена сущность B, то это означает, что B — атрибут (т.е. свойство или действие) A.

Перед построением дерева сущностей каждое личное местоимение («он», «она», «его» и др.) заменяется на слово, на которое это местоимение указывает. Для этого ищется ближайшее к личному местоимению в сторону начала предложения слово, согласованное с этим местоимением в роде и числе.

Дерево сущностей строится по упрощенному синтаксическому графу предложения. Этот граф представляет собой ориентированное дерево. Алгоритм вначале рассматривает его корень, затем — все вершины, которые из корня ведет ребро, и т.д. Вначале дерево сущностей состоит из двух вершин: корня α_0, которому не сопоставлена никакая сущность, и вершины α_1, в которую из корня ведет ребро и которой сопоставлена сущность «объект основных средств».

Обозначим рассматриваемую вершину упрощенного синтаксического графа через α. Если вершине α соответствует только одно или несколько служебных слов, то этой вершине не соответствует никакая сущность, и рассмотрение этой вершины сразу же прекращается. Каждой из остальных вершин упрощенного синтаксического графа будет соответствовать одна вершина в дереве сущностей.

Рассмотрим в упрощенном синтаксическом графе цепь, ведущую из корня в вершину α. Обозначим ближайшую к α (но отличную от нее) вершину этой цепи, которой соответствует некоторая вершина в строящемся дереве сущностей, через β (если таковая вершина найдется). Вершину цепи, в которую из β ведет ребро, обозначим через δ (эта вершина может совпадать с α). Пусть вершине β в строящемся дереве сущностей соответствует вершина β'.

Рассмотрение вершины α заключается в попытке использования следующих приемов.

1) Если из α в некоторую вершину γ ведет ребро, соответствующее синтаксическому отношению «ПОДЛ», то для целей построения дерева сущностей считается, что в упрощенном синтаксическом графе есть ребро из γ в α, но нет ребра из α в γ, и что ребро, ведущее (в действительности) в α (если оно есть), ведет в γ. Таким образом, мы считаем, что сказуемое является атрибутом подлежащего, а не наоборот.

2) Пусть в тексте, соответствующем вершине α, есть слово «объект», причем из соответствующей этому слову вершины первоначального (т.е. неупрощенного) синтаксического графа не выходит ребро, которому сопоставлено синтаксическое отношение «ГЕНИТ_ИГ»; либо пусть в тексте, соответствующем α, есть слова «объект основных средств». Тогда вершине α соответствует вершина дерева сущностей, в которую ведет ребро из α_1, и которой сопоставлена сущность, соответствующая α, и сопоставленный α текст без слова «объект» (во втором случае — без слов «объект основных средств»).

3) Если вершина β отсутствует, то вершине α соответствует вершина дерева сущностей, в которую ведет ребро из корня, и которой сопоставлены текст и сущность, соответствующие вершине α.

4) Пусть ребру из β в δ сопоставлено синтаксическое отношение «ДОП», «ГЛ_ДОП» либо «КОЛИЧ», а вершине β сопоставлены слова «в случае», «кроме», «за исключением», «в отношении», «исходя из», «при», «по», «для», «пониматься», «являться», либо слова, обозначающие количественное отношение («больше», «равный» и т.д). Тогда вершине α соответствует вершина дерева сущностей, в которую ведет ребро из корня, и которой сопоставляются текст и сущность, соответствующие вершине α.

5) Пусть ребру из β в δ сопоставлено синтаксическое отношение «УСЛ», а из α не выходит ребро, соответствующее синтаксическому отношению «ПОДЛ». Тогда вершине α соответствует вершина дерева сущностей, в которую ведет ребро из корня, и которой сопоставляются текст и сущность, соответствующие вершине α.

6) Вершине α в строящемся дереве сущностей будет соответствовать вершина, в которую ведет ребро из β', и которой сопоставлены те же текст и сущность, что и вершине α.

Порядок применения приемов следующий: сначала используется прием 1; далее происходит попытка применения приемов 2–6 (в порядке возрастания номера приема). Если попытка использования одного из приемов 2–6 успешна, то рассмотрение вершины α сразу же прекращается. Нетрудно заметить, что ситуация, когда все приемы 2–6 неприменимы, невозможна, так как ровно один из приемов 3 и 6 всегда применим.

В процессе построения дерева сущностей каждой сущности сопоставляется переменная.

После того, как дерево сущностей построено, происходит небольшое изменение текста некоторых вершин этого дерева: из текста удаляется слово «исчисленный», если в упрощенном синтаксическом графе из соответствующей этому тексту вершины ведет ребро в вершину со словами «исходя из». Кроме того, из текста удаляются указательные местоимения («этот», «такой» и т.д.).

На рисунке 3 изображено дерево сущностей, построенное по упрощенному синтаксическому графу, изображенному на рисунке 2.

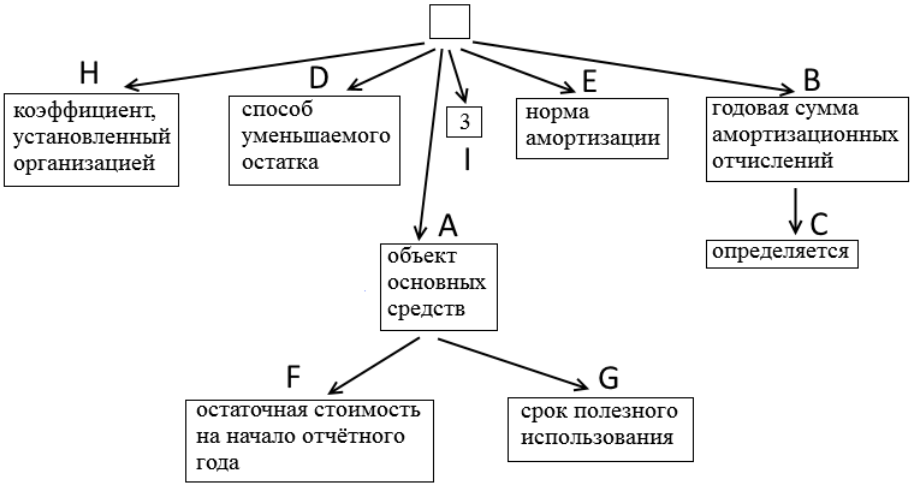


Рис. 3. Дерево сущностей

6. Построение логической формулы

Для каждого предложения по его упрощенному синтаксическому графу (с использованием дерева сущностей) строится логическая формула. При этом используются следующие приемы.

1) Если в предложении присутствует одно из следующих слов или сочетаний слов: «по», «в течение», «в случаях», «при», «в отношении», «если», то управляемая этим словом или сочетанием слов

часть предложения определяет область применения статьи закона либо некоторое условие. Пусть в соответствующую этому слову вершину γ ведет ребро из некоторой вершины α. Вершину, из которой в α ведет ребро (если таковая найдется) обозначим как β. Вершину, в которую из γ ведет ребро (если таковая найдется; таких вершин не может быть более одной) обозначим как δ. Хотя бы одна из вершин β и δ будет существовать. Возможны четыре случая:

а) вершине β сопоставлено тире. Это означает, что в предложении указано сразу несколько различных условий применения статьи закона, причем при каждом условии применяется только соответствующая ему часть статьи. Фактически в этом случае статью можно разделить на несколько независимых частей, что и осуществляется при построении формулы.

   В рассматриваемом случае вершина β существует и соответствует множественному актанту. Пусть $\alpha_1, \ldots, \alpha_r$ — вершины, в которые из β ведет ребро с меткой $<<\ >>$ (α — одна из этих вершин). Тогда итоговая формула будет иметь вид $A_1 \& \ldots A_r$, где A_i — формула, построенная по исходному графу, у которого все вершины $\alpha_j, j \neq i$ удалены, а вершины β и α_i объединены.

   Все последующие случаи рассматриваются в предположении, что случай а не имеет места.

б) вершина δ существует. Пусть A — формула, соответствующая графу, состоящему из всех вершин, достижимых из δ (будем считать, что каждая вершина достижима из себя), и соединяющих их ребер, B — формула, соответствующая графу, состоящему из всех остальных вершин (кроме γ) и соединяющих их ребер. Тогда итоговая формула будет иметь вид $(A \to B)$;

в) вершина δ не существует, а β не соответствует множественному актанту. Пусть A — формула, соответствующая графу, состоящему из всех вершин, достижимых из α, и соединяющих их ребер, B — формула, соответствующая графу, состоящему из всех остальных вершин (кроме γ) и соединяющих их ребер. Тогда итоговая формула будет иметь вид $(A \to B)$;

г) вершина δ не существует, а β соответствует множественному актанту. Это означает, что слова «по», «в течение», «в случа-

ях», «при», «в отношении», «если» относятся ко всему множественному актанту. Пусть B — формула, соответствующая графу, состоящему из всех вершин, достижимых из β, и соединяющих их ребер, C — формула, соответствующая графу, состоящему из всех остальных вершин и соединяющих их ребер. Тогда итоговая формула будет иметь вид $(B \to C)$.

2) Если в предложении некоторая его часть подчинена действию с помощью слова «кроме», то это означает, что действие выполняется тогда и только тогда, когда не выполняется условие, выраженное частью предложения, управляемой словом «кроме». Пусть это слово сопоставлено вершине α. Вершину, из которой в α ведет ребро, обозначим β. Пусть A — формула, соответствующая подграфу, состоящему из всех вершин, достижимых из β, B — формула, соответствующая графу, состоящему из всех остальных вершин (кроме α) и соединяющих их ребер. Итоговая формула будет иметь вид $(\neg A \sim B)$.

3) Слово «не» означает логическое отрицание. Пусть это слово сопоставлено вершине α, в которую входит ребро, выходящее из вершины β. Если по вершине β построено логическое выражение A, то после применения приема оно превращается в $(\neg A)$.

4) Союз «либо», «или» при перечислении однородных членов предложения,если речь не идет о перечислении аргументов некоторой функции, означает логическое «или» либо функцию, истинную, когда истинно значение ровно одного ее аргумента. Пусть вершина α — это множественный актант, которому соответствует несколько однородных членов предложения, а сами эти однородные члены предложения сопоставлены вершинам $\alpha_1, \ldots, \alpha_r$, в каждую из которых ведет ребро из α. Пусть также в вершину α не ведет ребро из вершины, которой сопоставлены слова «исходя из», и хотя бы одной из вершин $\alpha, \alpha_1, \ldots, \alpha_r$, либо вершине, в которую из $\alpha, \alpha_1, \ldots, \alpha_r$ ведет ребро, сопоставлено слово «или» («либо»). Возможны 2 случая:

   а) среди рассматриваемых вершин $\alpha_j, j \in \{1, \ldots, r\}$, найдется вершина α_i такая, что из нее выходит ребро, соответствующее синтаксическому отношению «УСЛ». Пусть это ребро ведет в вершину γ. Обозначим через A формулу, построенную по графу, состоящему из всех вершин, достижимых из γ, и

соединяющих их ребер; через B — формулу, построенную из всех остальных вершин, достижимых из α_i, и соединяющих их ребер. Если осталась лишь одна вершина $\alpha_j, i \neq j$, то обозначим через формулу, построенную по графу, состоящему из всех вершин, достижимых из α_j, и соединяющих их ребер. Если же таких вершин α_j, что $j \neq i$, больше одной, то обозначим через формулу, полученную применением 4.1 или 4.2 ко всем рассматриваемым вершинам $\alpha_s, s \in \{1, \ldots, r\}$, кроме α_i, которую мы исключим из рассмотрения. Тогда результатом применения приема будет формула $(A\&B \vee \neg A\&C)$;

б) среди рассматриваемых вершин $\alpha_j, j \in \{1, \ldots, r\}$, нет ни одной вершины α_i такой, что из нее выходит ребро, соответствующее синтаксическому отношению «УСЛ». Тогда, если обозначить через $A_j, j \in \{1, \ldots, r\}$, формулу, построенную по графу, состоящему из всех вершин, достижимых из α_j, то в результате применения приема будет создана формула, являющаяся дизъюнкцией всех формул A_j, соответствующих рассматриваемым вершинам $\alpha_j, j \in \{1, \ldots, r\}$;

5) Союз «и» или отсутствие союзов при перечислении однородных членов предложения, если речь не идет о перечислении аргументов некоторой функции, означает логическое «и» либо «или». Пусть вершина α — это множественный актант, которому соответствует несколько однородных членов предложения, а сами эти однородные члены предложения сопоставлены вершинам $\alpha_1, \ldots, \alpha_r$, в каждое из которых ведет ребро из α. Пусть также в вершину α не ведет ребро из вершины, которой сопоставлены слова «исходя из», и никакой из вершин $\alpha, \alpha_1, \ldots, \alpha_r$ и вершин, в которую из $\alpha, \alpha_1, \ldots, \alpha_r$ ведет ребро, не сопоставлены слова «или», «либо», «а также». Если однородные члены предложения — глаголы, в результате применения приема будет создана формула $(A_1\&A_2\& \ldots \&A_r)$. В противном случае результатом применения приема будет формула $(A_1 \vee A_2 \vee \ldots \vee A_r)$.

6) Слова «исходя из» означают некоторую функцию. Пусть эти слова сопоставлены вершине α, а β' — вершина, из которой в α идет ребро. Если вершине β' соответствует глагол, обозначим через β вершину, в которую из β' ведет ребро с меткой «ПОДЛ»; иначе обозначим через β саму вершину β'. Обозначим через γ вершину, в

которую ведет ребро из α (найдется лишь одна такая вершина γ). Пусть B — переменная, соответствующая вершине β. Возможно несколько случаев.

а) Вершине γ не сопоставлен множественный актант. Пусть C — переменная, обозначающая объект, определенный текстовым фрагментом, соответствующим вершине γ. Тогда в результате применения приема графу, состоящему из всех вершин, достижимых из β', и соединяющих их ребер будет поставлена в соответствие формула $((B = f_\beta(C))\&G)$, где f_β — некоторая функция, в тексте закона не определенная явно, G — формула, полученная путем применения приема 8 к вершине γ, либо приемов 6 или 9 к вершине, в которую из γ ведет ребро; если указанные приемы неприменимы, то G — тождественная истина.

$G_i, i \in 1, 2$ — формула, полученная путем применения приема 8 к вершине α'_i, либо приемов 6 или 9 к вершине, в которую из α'_i ведет ребро; если условия применения ни одного из этих приемов не соблюдены, то G_i — тождественная истина.

б) Вершине γ сопоставлен множественный актант. Пусть соответствующие ему однородные члены предложения сопоставлены вершинам $\gamma_1, \ldots, \gamma_r$, в каждое из которых ведет ребро из γ. Пусть никакой из вершин $\gamma, \gamma_1, \ldots, \gamma_r$ и вершин, в которую из $\gamma, \gamma_1, \ldots, \gamma_r$ ведет ребро, не сопоставлены слова «или», «либо», «а также». Пусть $A_j, j \in \{1, \ldots, r\}$ — переменная, соответствующая вершине γ_j, либо поставленная в соответствие вершине γ_j вспомогательная переменная, если γ_j соответствует множественный актант. Тогда в результате применения приема графу, состоящему из всех вершин, достижимых из β', и соединяющих их ребер будет поставлена в соответствие формула $((B = f_\beta(A_1, A_2, \ldots, A_r))\&G_1\& \ldots \&G_r)$, где f_β - некоторая функция, в тексте закона не определенная явно, а G_j — формула, полученная путем применения приема 8 к вершине γ_j, либо приемов 6 или 9 к вершине, в которую из γ_j ведет ребро (если найдется такая вершина, удовлетворяющая условиям применения хотя бы одного из этих приемов), и тождественная истина иначе.

в) Вершине γ сопоставлен множественный актант. Пусть соответствующие ему однородные члены предложения сопоставлены

вершинам $\gamma_1, \ldots, \gamma_r$, в каждое из которых ведет ребро из γ, и хотя бы одной из вершин $\gamma, \gamma_1, \ldots, \gamma_r$, либо вершине, в которую из $\gamma, \gamma_1, \ldots, \gamma_r$ ведет ребро, сопоставлено слово «или» («либо»). Тогда в результате применения приема графу, состоящему из всех вершин, достижимых из β', и соединяющих их ребер будет поставлена в соответствие формула $D\&G$, где D — формула, строящаяся с помощью приема из пункта 7, примененного к вершине γ. G — формула, полученная путем применения приема 6 или приема 9 к вершине γ, если она удовлетворяет условиям применения хотя бы одного из этих приемов, и тождественная истина иначе.

7) Пусть вершине γ сопоставлен множественный актант, и из вершины α, которой сопоставлены слова «исходя из», ведет ребро либо в γ, либо в вершину, из которой в γ можно попасть, переходя по ребрам с меткой «МНА» (в соответствии с их направлением). Пусть вершины β' и β ищутся для вершины α так же, как в пункте 6. Пусть также соответствующие γ однородные члены предложения сопоставлены вершинам $\gamma_1, \ldots, \gamma_r$, в каждое из которых ведет ребро из γ, и хотя бы одной из вершин $\gamma, \gamma_1, \ldots, \gamma_r$, либо вершине, в которую из $\gamma, \gamma_1, \ldots, \gamma_r$ ведет ребро, сопоставлено слово «или» («либо»). Пусть $D_j, j \in \{1, \ldots, r\}$ — это $f_\beta(A_j)$, если вершине γ_j не сопоставлен множественный актант; в противном случае это соответствующая множественному актанту формула, строящаяся с помощью приема из пункта 7, примененного к вершине γ_j. Пусть также G_j — формула, полученная путем применения приема 8 к вершине γ_j, либо приемов 6 или 9 к вершине, в которую из γ_j ведет ребро (если найдется такая вершина, удовлетворяющая условиям применения хотя бы одного из этих приемов), и тождественная истина иначе. Тогда будет создана формула $D_1\&G_1 \vee \ldots \vee D_r\&G_r$.

8) Пусть среди слов, соответствующих вершине α, есть слово «соотношение». Это означает, что данной вершине соответствует функция, представляющая собой отношение двух величин. Из вершины α в этом случае ведет ребро в некоторую вершину α', соответствующую множественному актанту, а из вершины α' выходит два ребра с меткой «МНА». Пусть α_1 и α_2 — вершины, в которые входят эти ребра; эти вершины соответствуют числителю и знаменателю отношения. Объект, находящийся в числителе или знаменателе отношения, соответствует либо самой вершине α_i, либо вершине, в

которую из α_i ведет ребро; обозначим вершину, которой соответствует этот объект, через α'_i. В случае, когда α_i и α'_i не совпадают, из вершины α_i должно вести ребро в вершину, которой соответствует слово «числитель» либо слово «знаменатель», что позволяет определить, в числителе или знаменателе отношения находится объект. В том случае, когда α_i и α'_i совпадают, считается, что в числителе оказывается объект, чей текстовый фрагмент находится в предложении раньше.

В результате применения приема графу, состоящему из всех вершин, достижимых из α, и соединяющих их ребер будет поставлена в соответствие формула $((A = C_1/C_2)\&G_1\&G_2)$, где A — переменная, сопоставленная вершине α; C_1 и C_2 — переменные, соответствующие объектам, находящимся в числителе и знаменателе отношения; $G_i, i \in 1, 2$ — формула, полученная путем применения приема 8 к вершине α'_i, либо приемов 6 или 9 к вершине, в которую из α'_i ведет ребро; если условия применения ни одного из этих приемов не соблюдены, то G_i — тождественная истина.

9) Для слов, обозначающих количественные отношения («больше», «меньше», «равный» и т.д.), в формуле будет присутствовать соответствующий предикат. Пусть α — вершина, которой сопоставлено одно из этих слов, γ — вершина, из которой в α ведет ребро. Пусть выходящее из α ребро, которому соответствует синтаксическое отношение «КОЛИЧ», ведет в вершину β. Пусть A – переменная, соответствующая вершине β, C — переменная, соответствующая вершине γ. Тогда графу, состоящему из вершин α, β, γ и соединяющих их ребер, будет соответствовать формула $pr_\alpha(C, A)$, где $pr_\alpha(C, A)$ — соответствующий вершине α предикат.

Применение приема означает проверку для вершины всех условий, указанных в тексте приема, и построение некоторой формулы, если эти условия выполнены. Предполагается, что один и тот же прием не может быть применен к одной и той же вершине (обозначенной во всех приемах как α) более одного раза, за исключением случаев, когда прием обращается рекурсивно к себе.

Алгоритм построения формулы по графу, состоящему из всех вершин, достижимых из данной, и соединяющих их ребер, состоит в следующем. Проверяется, является ли данная вершина вершиной α из приемов 2–6, 8, 9. Если да, то применяется все приемы, которые можно применить

(в порядке возрастания номеров), притом используемые приемы могут вызвать этот же алгоритм для некоторых вершин, к которым из данной идет ребро. Если ни один из приемов 2–6, 8, 9 нельзя применить, то данной вершине сопоставляется переменная, обозначающая объект, определенный текстовым фрагментом, соответствующим этой вершине. Производится конъюнкция этой переменной и всех формул, построенных с помощью этого же алгоритма по вершинам, в которые из данной вершины идет ребро.

Итоговый алгоритм построения формулы в соответствии с описанными приемами следующий.

Если условия приема 1 выполнены для какой-либо вершины (не должно быть более одной такой вершины), то в результате применения этого приема исходный граф разбивается на 2 подграфа (кроме случая, когда выполнены условия приема 1а: в этом случае рассматриваются несколько графов, полученных из исходного, и каждый такой граф разбивается на 2 подграфа). В каждом из этих подграфов выбирается корень, и к ним поочередно применяется описанный выше алгоритм построения формулы по графу, состоящему из всех вершин, достижимых из данной.

Если условия приема 1 не выполнены ни для какой вершины, то в графе выбирается корень, и к нему применяется описанный выше алгоритм построения формулы по графу, состоящему из всех вершин, достижимых из данной.

На рисунке 4 изображены упрощенный синтаксический граф вместе с сопоставленными его вершинам переменными, а также формула, построенная по упрощенному синтаксическому графу.

7. Построение модели закона по логическим формулам

Когда по каждому предложению закона построена логическая формула, можно перейти непосредственно к построению модели. Как и на всех предыдущих этапах, это можно сделать с помощью применения определенных приемов к имеющимся формулам.

1) Пусть формула имеет вид $A \to B$, где в B имеется выражение вида $C = D$, но нет выражения вида $E = C$. Найдем в уже построенном фрагменте графа модели закона вершину второго вида, в которой

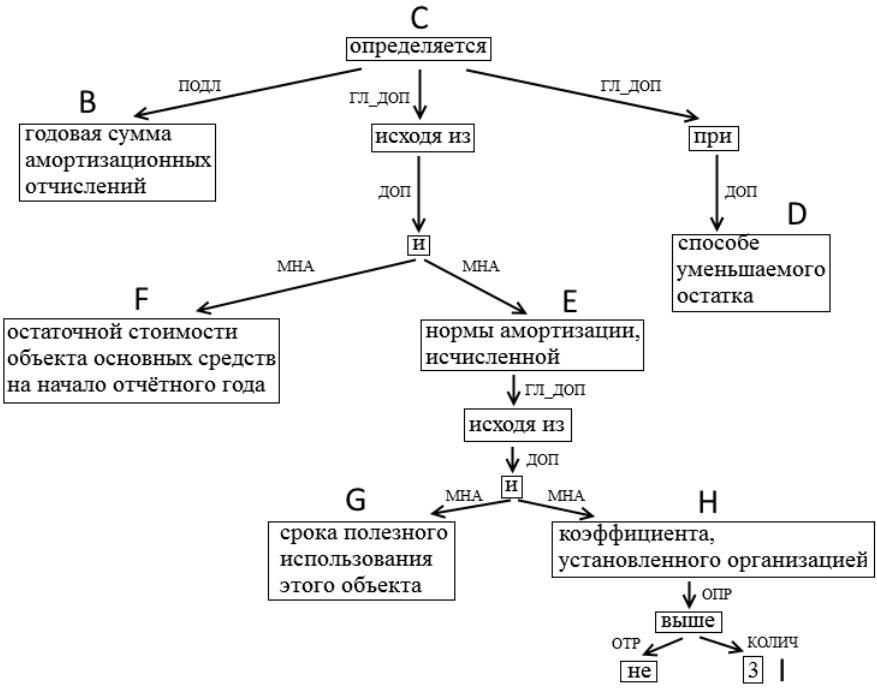


Рис. 4. Формула, построенная по упрощенному синтаксическому графу (f, g — еще не определенные функции)

происходит запись значения в поле памяти, соответствующее C из рассматриваемой формулы. Возможны 2 варианта.

а) Такая вершина есть. Обозначим ее α. Рассмотрим вершину, из которой в α ведет ребро. Обозначим ее β. Далее пройдем из β по цепочке ребер с номером 0 против их направления, пока эта цепочка не закончится. Обозначим вершину, в которую мы попадем, как γ. Далее выполним действия, указанные в пункте 1в.

б) Такой вершины нет. Тогда выберем какую-либо не рассмотренную ранее вершину и будем считать ее вершиной второго

вида, соответствующей C из рассматриваемой формулы. Обозначим эту вершину α. Выберем еще одну не рассмотренную ранее вершину, обозначим ее γ. Выпустим из нее ребро в вершину α. Далее выполним действия, указанные в пункте 1в.

в) Будем считать γ вершиной четвертого вида. Рассмотрим какие-либо три новые вершины, выпустим из них в γ по одному ребру, пронумеруем эти ребра числами 0, 1 и 2. Тогда вершине, из которой в γ ведет ребро номер 2, будет соответствовать вычисление логического выражения A. К вершине, из которой в γ ведет ребро номер 1, следует применить прием 3 относительно переменной C. Вершина, из которой в γ ведет ребро номер 0, будет, возможно, рассмотрена в результате применения 1а к одной из оставшихся формул. Если этого не произойдет, значит, процедура вычисления модели никогда не затронет эту вершину.

2) Пусть в формуле имеется выражение вида $A = B$, нет выражения вида $C = A$, и прием 1 к формуле неприменим. Тогда выберем какую-либо не рассмотренную ранее вершину и будем считать ее вершиной второго вида, соответствующей A из рассматриваемой формулы. Обозначим эту вершину α. Выберем еще одну не рассмотренную ранее вершину, обозначим ее γ. Выпустим из нее ребро в вершину α. Вершине γ будет соответствовать вычисление функции B (при помощи приема 3).

3) По фрагменту формулы, соответствующему вычислению значения переменной B, строится один из следующих фрагментов модели (прием 3б используется, если 3а неприменим):

а) Пусть в формуле есть подформула вида $A\&(B = C) \vee \&D$, причем в D входит выражение вида $B = E$. Выберем из всех таких подформул данной формулы (для фиксированного B) ту, что не содержится в другой такой подформуле. Пусть α — вершина, соответствующая вычислению значения B. Будем считать, что это — вершина четвертого вида. Выберем какие-либо три новые вершины, выпустим из них в α по одному ребру, пронумеруем эти ребра числами 0, 1 и 2. Тогда вершине, из которой в α ведет ребро номер 2, будет соответствовать вычисление логического выражения A. Вершине, из которой

в α ведет ребро номер 1, будет соответствовать вычисление функции C (смотрите прием 3б). Если D имеет вид $(B = E)$, вершине, из которой в α ведет ребро номер 0, будет соответствовать вычисление функции E (также с помощью приема 3б); в противном случае следует использовать прием 3а, рассматривая эту вершину в качестве α, а D в качестве выбранной подформулы.

б) Если в формуле есть подформула $B = A$, где A — некоторая переменная, и нет выражений вида $A = C$, где C — некоторая переменная или функция, то будем считать, что рассматриваемая вершина — первого вида, и в ней происходит получение значения A из соответствующего поля памяти. В противном случае речь идет о вычислении некоторой арифметической функции от нескольких аргументов $A_1, \ldots, A_k, k \in N$. Если эта функция в тексте закона не задана явно, то можно, например, потребовать у пользователя ее задания. Если функция уже задана, то ей можно сопоставить несколько вершин третьего типа, моделирующих вычисление этой функции (так, чтобы последний этап вычисления функции проходил в вершине α). Ребра, передающие значения аргументов $A_1, \ldots, A_k$, будут входить в некоторые из этих вершин. К каждой вершине, из которой эти ребра выходят, применяется прием 3 относительно соответствующей переменной.

4) Пусть переменная A — аргумент некоторой функции, и на эту переменную наложено некоторое условие (в виде бинарного или унарного отношения, которому она должна удовлетворять). Тогда выполнение этого условия проверяется в той вершине, в которой вычисляется значение A. В случае невыполнения условия программа требует у пользователя изменить значение переменных, с помощью которых вычисляется A (как правило, это собственно переменная A).

5) Пусть формула имеет вид $A \to B_1 \& \ldots \& B_k, \in N$, где $B_1, \ldots, B_k$ — переменные. Тогда такая формула преобразуется в конъюнкцию формул $A \to B_1, \ldots, A \to B_k$, к которым можно применять прием 6.

6) Пусть формула имеет вид $A \to B$, где B — переменная, A — подформула, не содержащая никаких переменных, кроме булевых.

Пусть B' — переменная B (точнее, соответствующая ей сущность), определенная с помощью всех остальных формул ($B' = 0$, если B не определяется из оставшихся формул). Тогда B определяется как $A \vee B'$.

7) Пусть формула имеет вид $A \to (B \sim C)$ либо $A \to (C \sim B)$, где B — переменная, A, C — подформулы, не содержащие никаких переменных, кроме булевых. Если C — тоже переменная, то для того, чтобы отличить B от C, можно использовать синтаксический граф: в нем соответствующие этим переменным текстовые фрагменты должны быть связаны каким-либо синтаксическим отношением. За B принимается переменная, чей текстовый фрагмент является главным в этом синтаксическом отношении. Пусть B' — переменная B (точнее, соответствующая ей сущность), определенная с помощью всех остальных формул ($B' = 0$, если B не определяется из оставшихся формул). Тогда B определяется как $A\&C \vee B'$.

8) Если функция, определяющая некоторую булеву переменную, не содержит никаких переменных, кроме булевых, то она моделируется с помощью несколько вершин третьего типа. В некоторые из этих вершин будут входить ребра, соответствующие переменным.

Алгоритм построения модели закона с помощью этих приемов следующий. Сначала для каждой формулы, не содержащей никаких переменных, кроме булевых, применяется прием 5; затем ко всем таким формулам применяются приемы 6 и 7; наконец, к каждой такой формуле применяется прием 8. Ко всем остальным формулам применяется прием 1 или 2, а затем — прием 4.

На рисунке 5 изображен фрагмент модели закона, соответствующий предложению «При способе уменьшаемого остатка годовая сумма амортизационных отчислений определяется исходя из остаточной стоимости объекта основных средств на начало отчетного года и нормы амортизации, исчисленной исходя из срока полезного использования этого объекта и коэффициента не выше 3, установленного организацией». Этот фрагмент построен по формуле, приведенной на рисунке 4, с помощью дерева сущностей, откуда были получены соответствующие переменным формулы сущности.

Предполагается, что выяснить, какая именно функция имеется в виду в законе, если эта функция в тексте закона не задана явно, можно, предоставив этот выбор пользователю программы, либо получив эти данные

из каких-либо других нормативно-правовых актов. Последнее решение сложнее реализуется и не всегда может дать результат; в программе в настоящее время реализовано первое решение.

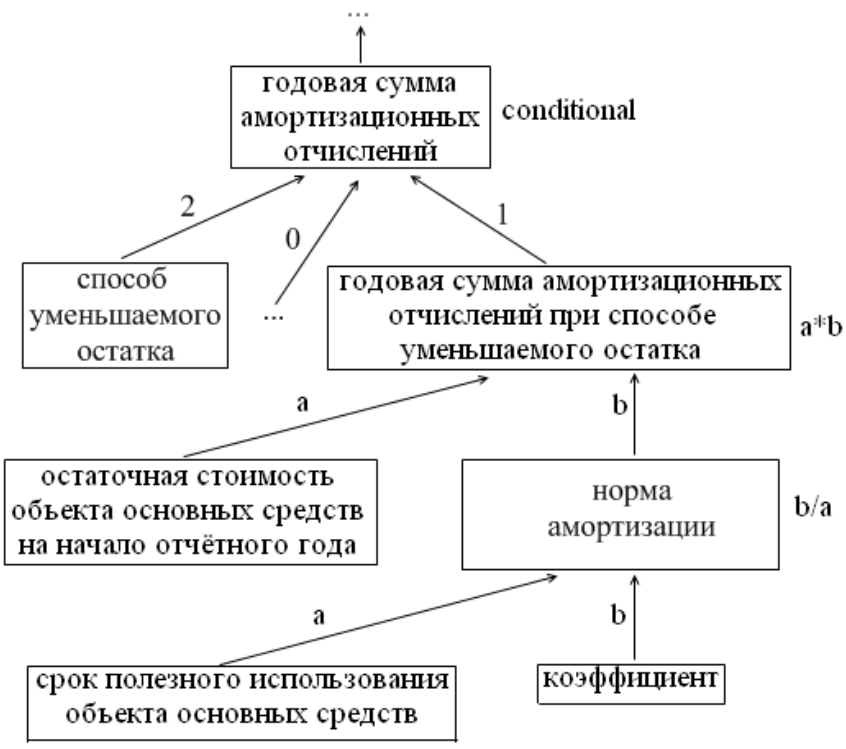


Рис. 5. Фрагмент модели закона.

8. Заключение

В работе описан метод, позволяющий по тексту нормативно-правового акта, касающегося бухгалтерского учета, автоматически строить схемы вычисления значений объектов, упомянутых в данном нормативно-правовом акте. В частности, описаны приемы синтаксического анализа; упрощения синтаксического графа; отождествления сущностей; представления упрощенного синтаксического графа в виде формулы; построения по всем формулам схем вычисления значений объектов. С помощью

данного метода созданы схемы вычисления значений объектов, описанных в положении по бухгалтерскому учету ПБУ 6/01.

Описанные в работе методы обработки текста, написанного на естественном языке, можно использовать при решении различных задач математической лингвистики.

Список литературы

[1] ERP и цифровое ядро. [https://www.sap.com/cis/products/erp.html]

[2] Microsoft Dynamics 365. [https://dynamics.microsoft.com]

[3] Национальный корпус русского языка. Синтаксически размеченный корпус русского языка: информация для пользователей. [http://www.ruscorpora.ru/instruction-syntax.html]

[4] Chen D., Manning C.D. A Fast and Accurate Dependency Parser using Neural Networks // Proceedings of EMNLP 2014. — 2014. — P. 740–750

[5] Сокирко А. Семантические словари в автоматической обработке текста (по материалам системы ДИАЛИНГ). [http://www.aot.ru/docs/sokirko]

[6] Проект Universal Dependencies. [http://universaldependencies.org]

[7] de Marneffe M., Dozat T., Silveira N., Haverinen K., Ginter F., Nivre J., Manning C.D. Universal Stanford Dependencies: A cross-linguistic typology // In Proc. of the 9th Conference on Language Resources and Evaluation (LREC). — 2014. — P. 4585–4592.

[8] Мельчук И. А. Опыт теории лингвистических моделей «Смысл ⇔ Текст». — М.: Школа «Языки русской культуры», 1999. — 368 с.

[9] Iordanskaja L., Kittredge R., Polguere A. Lexical Selection and Paraphrase in a Meaning-Text Generation Model // Natural Language Generation in Artificial Intelligence and Computational Linguistics. SECS. Boston, Springer, 1991. — **119**. — P. 293–312.

[10] Anisimovich K.V., Druzhkin K.Ju., Minlos F.R., Petrova M.A., Selegey V.P., Zuev K.A. Syntactic and semantic parser based on ABBYY Compreno linguistic technologies // Международная конференция по компьютерной лингвистике «Диалог-2012». — 2012. — **2**. — P. 91–103.

[11] Ben-Or M. Lower Bounds For Algebraic Computation Trees // Proc. 15th ACM Annu. Symp. Theory Comput. — 1983. — P. 80—86.

[12] Кудрявцев В.Б., Гасанов Э.Э., Перпер Е.М. Автоматическая генерация компьютерной программы, моделирующей нормативно-правовой акт // Интеллектуальные системы. Теория и приложения (ранее: Интеллектуальные системы по 2014, вып. 2, ISSN 2075-9460). — 2014. — **18**, №2. — С. 133–156.

[13] Перпер Е.М. О синтаксическом анализе юридических текстов // Интеллектуальные системы. Теория и приложения (ранее: Интеллектуальные системы по 2014, вып. 2, ISSN 2075-9460). — 2016. — **20**, №2. — С. 31–49.

[14] Подколзин А. С. Система автоматического решения задач по элементарной алгебре // Дискретная математика. —– 1994. — **6**, №4. — С. 35-–57

[15] Приказ Министерства финансов Российской Федерации от 30.03.2001 N 26н «Об утверждении Положения по бухгалтерскому учету «Учет основных средств» ПБУ 6/01» [http://base.consultant.ru/cons/cgi/online.cgi?req=doc;base=LAW; n=111056]

On the semantic analysis of juridical documents

Perper E.M., Gasanov E.E., Kudryavtsev V.B.

The paper considers the task of designing a program which would perform semantic analysis of a juridical document in Russian language. For each entity described in a text, the program must construct a scheme which computes the value of this entity. The paper contains rules which allow to construct such schemes if morphological information about all words of the text is provided.

Keywords: semantic analysis, syntactic analysis, juridical document, logical formula, rule.

Библиотеки с поддержкой длинной арифметики на GPU

Собянин П.И.

Рассматриваются вычислительные библиотеки, использующие архитектуру CUDA для работы с целыми числами произвольной длины. Сравниваются возможности и производительность находящихся в открытом доступе таких библиотек.

Ключевые слова: длинная арифметика, GPU, CUDA.

1. Введение

Часто в задачах криптографии с открытым ключом требуется использование типов данных, не являющихся базовыми, в т.ч. чисел произвольной длины:

- существуют алгоритмы с открытым ключом, которые основаны на возведении в степень в группе или кольце. Например, в алгоритме RSA[1] – для шифрования передаваемого сообщения, в алгоритме DSA[1] – для генерации открытого ключа, в схеме Эль-Гамаля[1] и протоколе Диффи-Хеллмана[1] – для генерации секретного ключа шифрования.

- для криптографии важно исследовать решение задач факторизации[1] и дискретного логарифмирования[1] (например, для тех же алгоритмов RSA и DSA);

В работе исследуется возможность реализации "длинной" арифметики на GPU путем привлечения существующих решений.

Задача данной статьи – исследовать возможности и производительность библиотек, использующих архитектуру CUDA[2]. Насколько можно судить по имеющимся публикациям, сравнение имеющихся инструментов еще не проводилось.

Здесь мы сравним две таких библиотеки – **GARPREC**[3][4] и **CUMP**[5].

Дальнейшая часть статьи построена следующим образом: в разделе 2 рассматриваются приемы реализации библиотек и распараллеливания алгоритмов; раздел 3 посвящен описанию сравнительного эксперимента, его результатам и выводам; в разделе 4 подведены итоги текущей работы и освещены дальнейшие планы в предметной области.

2. Обзор реализаций

В настоящий момент разработано несколько библиотек, реализующих длинную арифметику на GPU: **GPUMP** [6] и уже упоминавшиеся **GARPREC** и **CUMP**. Вкратце опишем каждую.

GPU Multiple-Precision library (GPUMP). В 2010-м году Кайонгом Жао (Kaiyong Zhao) и Сяовенем Чжу (Xiaowen Chu) была представлена библиотека GPUMP, поддерживающая "длинную" арифметику для CUDA. Она способна выполнять арифметические операции с целыми числами произвольной фиксированной точности на архитектуре CUDA. Эта библиотека поддерживает операции сложения и вычитания (в т.ч. и по заданному модулю), умножения, деления, умножения и нахождения остатка от деления методами Монтгомери[7]. Все операции используют последовательные алгоритмы вычислений. Реализовано параллельное покоординатное выполнение операций для векторов операндов.

Авторы сравнивают производительность библиотек GPUMP и GNU MP, что поддерживает "длинные" вычисления на CPU. Проводятся тесты производительности для каждой операции над большим количеством пар чисел разного размера. В каждой паре размер чисел одинаков – 512, 1024 либо 2048 бит. Производительность измеряется в количестве операций в секунду. Авторы показывают, что производительность библиотеки GPUMP в 7-9 раз выше в зависимости от операции и размеров чисел. Использованное для расчетов оборудование: CPU – Intel Core i7 с рабочей частотой 2.8 ГГц, видеокарта – XFX GTX280 с GPU NVIDIA GT200 (240 вычислительных ядер, работающих на частоте 1.24 ГГц).

GARPREC. Эта библиотека основана на библиотеке **ARPREC** [8], что поддерживает арифметические операции с "длинными" числами на CPU. GARPREC также поддерживает основные арифметические опера-

ции, такие как сложение, умножение, взятие частного, корня, экспоненты, а так же логарифма, синуса и косинуса. Как и у первой библиотеки, все операции используют последовательные алгоритмы вычислений. Вычислительная сложность алгоритма сложения линейная по размеру операндов, а умножения – $O(n \log n)$.

Тесты производительности аналогичны тем, что применяются к библиотеке GPUMP – бинарные и унарные операции применялись к большому множеству чисел разных размеров; вычисления производились как на GPU, так и CPU. Прирост производительности расчетов на GPU по сравнению с CPU многократный: для сложения – в 12 раз, для умножения – в 8-9 раз в зависимости от размера чисел, для деления – в 10 раз.

CUMP. Эта библиотека была разработана Такатоши Накаямой (Takatoshi Nakayama) в 2012 году. Она основана на библиотеке **GMP** [9]. CUMP поддерживает операции сложения, вычитания и умножения. Вычислительная сложность алгоритма сложения – $O(n)$, сложность операции умножения зависит от алгоритма, который автоматически выбирается самой библиотекой в зависимости от размеров операндов, и может составлять как $O(n^2)$ (для алгоритма умножения "столбиком"), так и $O(n \log n)$ (для алгоритма быстрого преобразования Фурье[10]). Число n здесь – размер операндов в битах.

Во всех библиотеках каждая арифметическая операция выполняется на одной "нити" (вычислительном ядре) GPU, т.е. соответствующие процедуры не распараллелены. Реализовано параллельное покоординатное выполнение операций для векторов.

Необходимо отметить, что только две из описанных библиотек находятся в открытом доступе – это GARPREC и CUMP. В силу этого обстоятельства тестирование библиотеки GPUMP не проводилось.

3. Эксперимент

Описание. Мы тестировали две операции: сложение и умножение пары чисел. Такой выбор обусловлен тем, что именно эти две операции поддержаны одновременно в каждой из библиотек. В качестве входных данных эксперимента генерировались два случайных вектора. Для каждой операции проводилось два теста: в первом варьировался размер вектора при константном размере чисел (он был равен 1000 десятичных зна-

ков), во втором – размер чисел при константном размере вектора (100000 компонент).

Мы тестировали библиотеки на видеокарте Tesla C2070, на которой установлен графический процессор Tesla T20 с 448 вычислительными ядрами, работающих на частоте 1.15 ГГц.

Результаты. На рис. 1-2 по оси X отложена длина операндов, на рис. 3-4 – длина векторов. По оси Y на всех рисунках отложено время выполнения операции в секундах. На всех изображениях пунктирный график отражает результаты работы библиотеки GARPREC, сплошной – библиотеки CUMP.

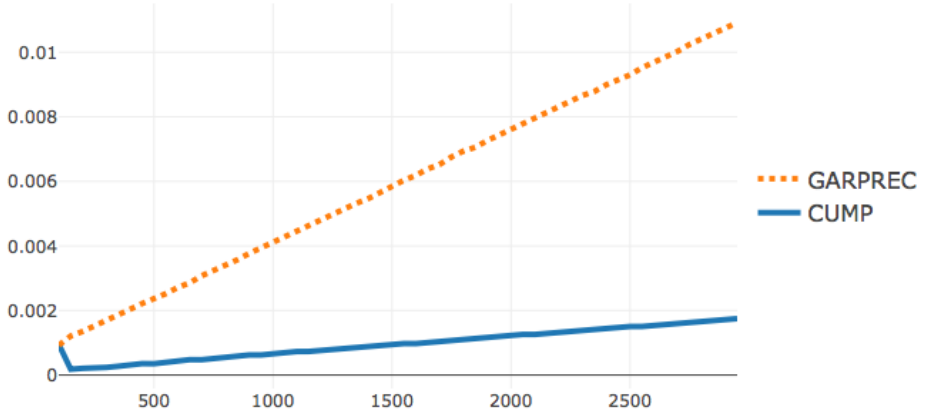


Рис. 1. Скорость сложения в зависимости от размера чисел.

Результаты тестов показывают, что порядок роста функций совпадает с теоретическим: в трех случаях из четырех рост линейный, а в оставшемся – порядка $O(n \log n)$, как и ожидалось.

Также видно, библиотека CUMP является более производительной, чем GARPREC: сложение в указанных условиях производится в 5 раз быстрее, умножение – в 20 раз при одинаковой сложности используемых алгоритмов. Причины этого отмечены в [11].

Но нельзя не отметить очень ограниченную функциональность CUMP: всего поддержаны лишь три простейшие арифметические операции, и это вряд ли делает ее пригодной для применениях в каких-то серьезных областях. Надеяться на развитие функционала этой библио-

теки также не приходится – последнее обновление её исходного кода датировано 2012-м годом[5].

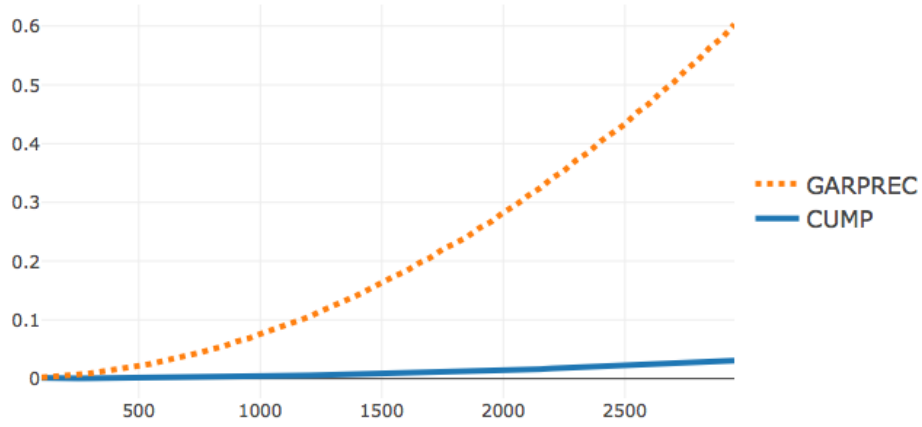


Рис. 2. Скорость умножения в зависимости от размера чисел.

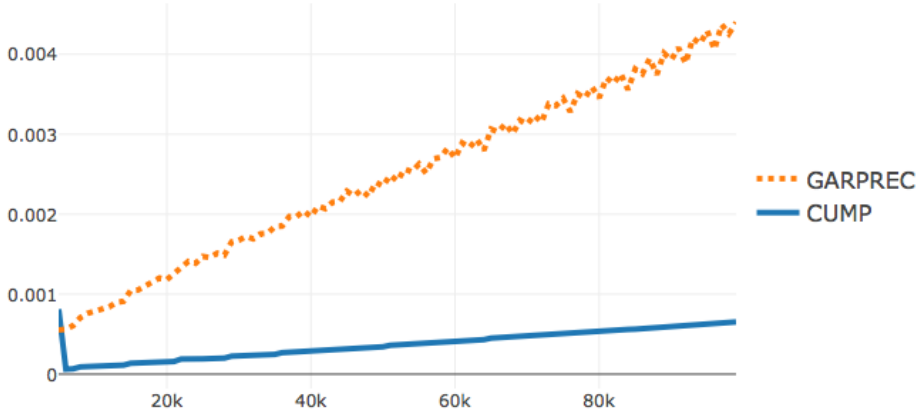


Рис. 3. Скорость сложения в зависимости от размера векторов.

4. Заключение

Операции с целыми числами произвольной длины являются важной составляющей криптографии с открытым ключом. В статье мы описали и сравнили возможности и производительность существующих библиотек,

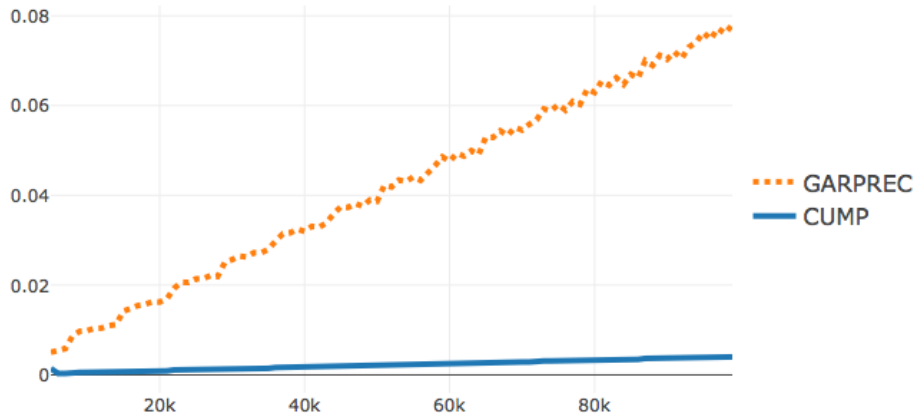


Рис. 4. Скорость умножения в зависимости от размера векторов.

поддерживающих работу с таким типом данных. В дальнейшем планируется реализовать собственную библиотеку с более широким функционалом.

Автор выражает искреннюю признательность А. В. Галатенко за постановку задачи и обсуждение результатов работы.

Список литературы

[1] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. Handbook of Applied Cryptography. *CRC Press, Fifth Printing, August 2001.*

[2] NVIDIA CUDA. http://developer.nvidia.com/object/cuda.html.

[3] GARPREC: https://code.google.com/archive/p/gpuprec/.

[4] Mian Lu, Bingsheng He, Qiong Luo. Supporting Extended Precision on Graphics Processors.

[5] CUMP Library. https://github.com/skystar0227/CUMP.

[6] Kaiyong Zhao, Xiaowen Chu. GPUMP: a Multiple-Precision Integer Library for GPUs.

[7] Peter Montgomery. Modular Multiplication Without Trial Division. *Mathematics of Computation, vol. 44 no. 170, pp. 519–521, April 1985.*

[8] ARPREC. http://crd-legacy.lbl.gov/ dhbailey/mpdist/.

[9] GMP Library. https://gmplib.org/.

[10] Brigham, E. Oran. The Fast Fourier Transform. *New York, USA: Prentice-Hall, 2002.*

[11] https://mail.haskell.org/pipermail/glasgow-haskell-users/2006-September/010963.html

GPU Multiple-Precision Arithmetic Libraries

Sobyanin P.I.

CUDA libraries supporting multiple-precision integer arithmetic are considered. Features and performance of such open-source libraries are compared.

Keywords: multiple-precision arithmetic, GPU, CUDA.

Часть 3.
Математические модели

Получение верхней оценки на хроматическое число графов заданной толщины и обхвата

Адилов С.Ш.

Данная работа посвящена изучению свойств графов с заданными параметрами толщины и обхвата. Приведена верхняя оценка на хроматическое число графов, зависящая от толщины k и обхавата g, где $k \geq 1$ и $g \geq 3$. В частности, для бипланарных графов с обхватом не менее 10 доказана 5-раскрашиваемость.

Ключевые слова: хроматическое число, обхват, толщина, планарный граф, бипланарный граф.

Введение

Вопрос наличия связей между обхватом и хроматическим числом произвольных графов был изучен венгерским математиком Палом Эрдёшем, который, используя вероятностный метод, доказал, что для любых положительных l и r существует граф с обхватом $g \geq l$ и хроматическим числом $\chi \geq r$. [1] Другими словами, существуют графы со сколь угодно большими обхватом и хроматическим числом. Однако, как известно, для планарных графов (графов с толщиной 1) имеет место аналитически доказанная теорема Хивуда, согласно которой такие графы допускают правильную раскраску в 5 цветов.[2] Если же графы при этом не содержат треугольников (толщина равна 1 и обхват не менее 4), то по теореме Грётча (Грётша) верна их 3-раскрашиваемость.[3] Для произвольных бипланарных графов, то есть графов с толщиной 2, доказано, что их хроматическое число не превосходит 12.[4] Как выяснилось в статье [5], эта оценка улучшаема до 8 для бипланарных графов, не содержащих треугольников. Таким образом, правильная раскраска может быть связана с такими параметрами, как толщина и обхват. Данная работа показывает общую оценку хроматического числа графов с толщиной $k \geq 1$ и обхватом $g \geq 3$.

Основные определения

В статье рассматриваются обыкновенные графы, то есть конечные неориентированные графы без петель и кратных ребер.

Определение. *Графом* G называется пара (V, E), где V – конечное непустое множество и E – множество неупорядоченных пар различных элементов из V. Элементы V называются *вершинами* графа, элементы E – *ребрами*. (n, m)-*графом* будем называть граф с n вершинами и m ребрами.

Определение. *Плоским графом* называется изображение графа на плоскости, причем вершинам графа сопоставлены точки на плоскости, а ребрам – кусочно гладке линии, соединяющие соответствующие вершины так, что никакие два ребра не имеют общих точек, кроме концевых вершин.

Определение. *Гранью* называется область, ограниченная ребрами в плоском графе и не содержащая внутри себя вершин и ребер графа. Одна из граней не ограничена и называется *внешней* гранью, а остальные – *внутренними* гранями. *Границей грани* будем считать множество вершин и ребер, принадлежащих этой грани.

Определение. Граф называется *планарным*, если он представим в виде плоского графа.

Определение. Пусть $G = (V, E)$ – некоторый граф, r – натуральное число. *Правильной вершинной* r*-раскраской* графа G называется произвольная функция вида $f : V \to \{1, \ldots, r\}$ такая, что $f(u) \neq f(v)$ для любых смежных вершин u и v. *Хроматическим числом* $\chi(G)$ графа G называется наименьшее r, для которого G имеет правильную вершинную r-раскраску.

Определение. *Толщина* графа G есть наименьшее число планарных подграфов, объединение которых дает G.

Определение. Граф $G = (V, E)$ называется *бипланарным* (*двупланарным*), если его толщина равна 2, то есть существуют подграфы $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ графа G такие, что G_1, G_2 - планарные и $G = (V_1 \cup V_2, E_1 \cup E_2)$.

Определение. *Обхватом* графа называется минимальная из длин его циклов.

Вспомогательные утверждения

Лемма 1. *Пусть G – планарный (n, m)-граф с обхватом g. Тогда справедливо неравенство:*

$$m \leq \frac{g}{g-2}(n-2).$$

Доказательство. Пусть f – число граней плоского изображения графа G. Каждое ребро графа G, не являющееся мостом (ребром, не содержащимся ни в одном цикле), принадлежит двум граням. Всякая грань ограничена по крайней мере g ребрами. Отсюда получим оценку снизу удвоенного числа ребер графа G, то есть

$$gf \leq 2m.$$

Так как G – планарный граф, справедлива формула Эйлера:

$$f = m - n + 2.$$

Получим

$$g(m-n+2) \leq 2m \iff m \leq \frac{g}{g-2}(n-2).$$

□

Лемма 2. *Для любого (n, m)-графа с толщиной k и обхватом g справедливо неравенство*

$$m \leq \frac{kg}{g-2}(n-2).$$

Доказательство. Имеем $G := G_1 \cup G_2 \cup \ldots \cup G_k$, где G_j – планарные (n_j, m_j)-графы, $j = \overline{1,k}$. Согласно лемме 1

$$m_j \leq \frac{g}{g-2}(n_j - 2) \leq \frac{g}{g-2}(n-2).$$

Получим

$$m \leq m_1 + m_2 + \ldots + m_k \leq \frac{kg}{g-2}(n-2).$$

□

Лемма 3. *В любом графе с толщиной k и обхватом не менее g существует вершина степени не более $d = \left\lceil \frac{2kg}{g-2} \right\rceil - 1$.*

Доказательство. Предположим обратное: степень любой вершины такого графа не меньше $d+1$. Тогда имеем 2 неравенства:
так как по предположению каждой вершине инцидентны не менее $d+1$ ребер,

$$2m \geq (d+1)n,$$

и согласно лемме 2

$$m \leq \frac{kg}{g-2}(n-2).$$

Объединив неравенства, получим

$$(d+1)n \leq 2 \cdot \frac{kg}{g-2}(n-2) \Longleftrightarrow \left\lceil \frac{2kg}{g-2} \right\rceil n \leq \frac{2kg}{g-2}(n-2) \leq \left\lceil \frac{2kg}{g-2} \right\rceil (n-2).$$

Полученное неравенство $n \leq n-2$ не имеет решений. Противоречие. □

Основные результаты

Теорема 1. *Любой граф с толщиной k и обхватом не менее g допускает правильную раскраску не более чем в $\chi = \left\lceil \frac{2kg}{g-2} \right\rceil$ цветов.*

Доказательство. Применим индукцию по количеству вершин n. Очевидно, что такой граф с $n \leq g$ вершинами является либо лесом, либо циклом длины g. Лес раскрашивается в 2 цвета, цикл − в 3. А поскольку при любых $k \geq 1$ и $g \geq 3$

$$\chi = \left\lceil \frac{2kg}{g-2} \right\rceil = \left\lceil \frac{2k(g-2)+4k}{g-2} \right\rceil = \left\lceil 2k + \frac{4k}{g-2} \right\rceil \geq 3,$$

теорема верна.

Пусть теорема верна для графов с $n > g$ вершинами. Согласно лемме 3 найдется вершина степени не более $\chi - 1$. Удалим эту вершину и все инцидентные ей ребра. По предположению индукции полученный граф допускает правильную раскраску не более чем в χ цветов. Добавив удаленную вершину и все инцидентные ей ребра обратно, получим, что ее тоже можно раскрасить в худшем случае в один из χ цветов, поскольку все смежные $\chi - 1$ вершины могут иметь $\chi - 1$ цветов. Теорема доказана. □

Следствие 1. *Следующая таблица показывает связь обхвата и хроматического числа бипланарных графов:*

Обхват	3	4	5	6, 7, 8, 9	≥ 10
Верхняя оценка хроматического числа	12	8	7	6	5

Заключение

Полученные результаты обобщают оценки хроматического числа графов произвольной толщины и обхвата. Однако получена лишь оценка сверху и в ряде случаев она оказывается завышенной, так как существуют более сильные утверждения. Например, при $k = 1$ и $g = 4$ получим планарный граф без треугольников. Согласно теореме 1 такие графы 4-раскрашиваемы, в то время как по теореме Грёча хроматическое число планарных графов без треугольников равно 3, как уже было отмечено во введении. Таким образом, становится актуальным вопрос улучшаемости оценки хроматического числа графов с фиксированными толщиной и обхватом. Для случая $k = 2$ и $g = 4$ в статье [5] показана нижняя оценка в 5 цветов.

Список литературы

[1] Paul Erdős Graph theory and probability // Canadian Journal of Mathematics. — 1959. — Т. 11. — С. 34—38.

[2] Харари Фрэнк Теория графов
Изд. 2-е. — М.: Едиториал УРСС, 2003. — 296 с.

[3] H. Grötzsch Zur Theorie der diskreten Gebilde, VII: Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel // Wiss. Z. Martin-Luther-U., Halle-Wittenberg, Math.-Nat. Reihe. — 1959. — Т. 8. — С. 109–120.

[4] L.W. Beineke Biplanar Graphs:: A Survey
https://www.sciencedirect.com/science/article/pii/S0898122197002149

[5] Ищенко Р.А. Хроматическое число бипланарных графов без треугольников // Интеллектуальные системы. - 2014 — Т.18, вып.4 - с.223-226

[6] В.А. Емеличев, О.И. Мельников, В.И. Сарванов, Р.И. Тышкевич Лекции по теории графов
Р.И. − М.: Наука. Гл. ред. физ.-мат. лит., 1990. − 384 с.

An upper bound for the chromatic number of graphs with given thickness and girth

S. Sh. Adilov

The thickness of a graph G is the minimum number of planar graphs whose union is G. In this paper, we consider an upper bound for the chromatic number of graphs depending on thickness k and girth g, where $k \geq 1$ and $g \geq 3$. In particular, for biplanar graphs with girth at least 10 we obtain 5-colorability.

Keywords: chromatic number, girth, thickness, planar graph, biplanar graph.

О мере множества законов движения точки, реализуемых клеточными автоматами

Калачев Г.В., Титова Е.Е.

В работе рассматривается модель движения точки на экране, представляющем из себя бесконечный вправо одномерный клеточный автомат. Выделено подмножество множества состояний экрана, которое называется изображением точки на экране. Закон движения определяется бесконечной последовательностью из нулей и единиц, которые в каждый момент времени задают остановку или движение точки соответственно. Описан алгоритм реализации на экране широкого класса законов движения и исследована мера Бернулли множества реализуемых законов движения. Показано, что почти все законы движения являются реализуемыми. Также доказано, что относительно тихоновской топологии множество реализуемых законов движения относится к первой категории Бэра, то есть мало.

Ключевые слова: клеточный автомат, универсальный экран, движение точки, закон движения, мера Бернулли, тихоновская топология, категории Бэра.

1. Введение

Клеточный автомат — это математическая модель в дискретном пространстве и дискретном времени. Состояние ячейки пространства в каждый момент времени определяется в зависимости от состояний соседних ячеек и самой ячейки в предыдущий момент. Правила изменения состояний могут отличаться в разных ячейках. Частным случаем клеточного автомата является однородная структура — клеточный автомат, в котором правила изменения состояний одинаковы для всех ячеек. Однородными структурами и клеточными автоматами моделируются многие реальные физические и химические процессы [7].

Впервые идея клеточных автоматов отмечена в работах венгерско-американского математика Джона фон Неймана в 1940-х годах [8]. Нейман изучал проблему самовоспроизводящихся систем, и описал универсальный конструктор, строящий самовоспроизводящийся автомат [3].

Вплоть до конца 60-х идея клеточных автоматов была забыта.

В 1960х годах была выдвинута Эдвардом Муром и доказана Муром и Джоном Майхиллом Теорема сада Эдема о конфигурациях, не имеющих родителя.

В 1970 г. в октябрьском выпуске журнала Scientific American в рубрике Мартина Гарднера «Математические игры» было впервые опубликовано описание игры «Жизнь», автором которой был математик Джон Конвей. Впоследствии было показано, что игра «Жизнь» может эмулировать универсальную машину Тьюринга.

В 1983 британский ученый Стивен Вольфрам опубликовал первую из серии статей, исследующих элементарный клеточный автомат, то есть, автомат имеющий только два состояния. Вольфрам выдвигает предположение, что элементарный автомат, названный Правило 110 может быть универсальным. В 1990 году этот факт доказан ассистентом Вольфрама Мэтью Куком.

В 2002 году Вольфрам опубликовал текст «Новый тип науки» (A New Kind of Science), где широко аргументировал, что достижения в области клеточных автоматов не являются изолированными, и имеют большое значение для всех областей науки.

На механико-математическом факультете МГУ имени М. В. Ломоносова начиная с 1960х годов исследованием свойств однородных структур занимались профессор Валерий Борисович Кудрявцев [4], профессор Александр Сергеевич Подколзин [10][11], Анатолий Александрович Болотов [1]. Результатом их работы стала, в том числе, монография [6].

Данная работа является продолжением некоторых результатов диссертации Титовой Елены Евгеньевны «Конструирование изображений клеточными автоматами» [12].

В настоящей работе исследуются одномерные клеточные автоматы, которые реализуют различные конфигурации, движущиеся на бесконечном экране в зависимости от последовательности, подаваемой на управляющий вход. Бесконечный экран является бесконечной вправо последовательностью ячеек, в каждой из которых находится экземпляр конечного автомата, входами которого являются состояния соседних автоматов; на левый вход первой ячейки подаётся управляющая последовательность. Задаётся подмножество состояний автомата, соответствую-

щих точке в ячейке экрана. Соответственно, каждая ячейка экрана либо пуста, либо в ней стоит точка в зависимости от текущего состояния автомата в ячейке. В работе рассматривается случай, когда на экране, начиная с некоторого момента появляется одна точка, которая перемещается вправо в соответствии с заданным законом, который называется законом движения. Закон движения задаётся бесконечной последовательностью из нулей и единиц, которая начинается с единицы, и в которой i-й элемент определяет, перемещается точка в момент времени на одну позицию вправо или остаётся на месте.

Задача заключается в описании множества всех законов движения, реализуемых на некотором экране с некоторой управляющей последовательностью. Изначально эта задача была поставлена и частично исследована в диссертации Титовой Е.Е [12]. В частности, было найдено множество реализуемых законов движения, а также приведён пример нереализуемого закона движения, однако часть законов движения осталась не охваченной. Целью данной работы было оценить, какую долю составляют реализуемые законы движения.

В работе введены классы $\mathcal{F}_a$ законов движения с ограниченной средней скоростью и показано, что все законы движения из этих классов реализуемы на некотором экране. На множестве всех законов движения введена бернуллиева мера и показано, что мера множества законов движения с ограниченной скоростью равна 1. Таким образом, относительно бернуллиевой меры почти все законы движения реализуемы. Также в работе рассматривается вопрос о том, какую долю занимают реализуемые законы движения среди законов движения с неограниченной скоростью.

В силу невозможности введения какой-либо естественной меры на всём множестве законов движения с неограниченной скоростью, рассматриваются достаточно большие подмножества этого множества, представляемые в виде конкатенации множеств слов фиксированной длины. На таких подмножествах естественным образом вводится равномерная мера. В работе показано, что для определённая таким образом мера реализуемых законов движения, попадающих в данное подмножество, равна 0.

Кроме меры множества реализуемых законов движения, рассматривается топологическая категория этого множества [9]. На множестве законов движения можно ввести топологию, порождённую множествами с фиксированным префиксом. В работе доказано, что относительно такой топологии множество реализуемых законов движения относится к первой категории Бэра, то есть является малым с топологической точки зрения.

Таким образом, в работе показано, что почти все законы движения реализуемы, но с топологической точки зрения они образуют малое множество.

2. Определения и формулировка результатов

Согласно [5], *конечный инициальный автомат* — это шестёрка $\mathcal{A} = (A, Q, B, \varphi, \psi, q_0)$, где A — *входной алфавит*, Q — *множество состояний*, которое является конечным подмножеством некоторого фиксированного счётного множества, B — *выходной алфавит*, $\varphi : Q \times A \to Q$ — *функция переходов*, $\psi : Q \times A \to B$ — *функция выходов*, q_0 — *начальное состояние*.

Элементарным автоматом будем называть конечный инициальный автомат $\mathcal{A}$ с двумя входами и одним выходом, такой что:

$A = Q^2$, $B = Q$ для некоторого множества Q, $|Q| \geq 2$, $\psi = \varphi = \varphi(q, l, r)$, причём $\varphi(q_0, q_0, q_0) = q_0$.

Экраном бесконечной длины называется бесконечная последовательность из одинаковых элементарных автоматов $\mathcal{A}$, причём левый вход первого автомата называется *свободным*, к входам остальным входам элементарных автоматов присоединены выходы автоматов, стоящих в двух соседних с ним клетках, то есть у автомата имеется *левый* вход l, *правый* вход r и q — текущее состояние автомата. Выходом автомата $\mathcal{A}$ в заданный момент времени является его состояние в этот момент времени.

Бесконечный экран с элементарным автоматом $\mathcal{A}$ будем обозначать $S(\mathcal{A})$. Если элементарный автомат имеет q состояний, будем говорить, что экран $S(\mathcal{A})$ также имеет q состояний.

В множестве Q состояний элементарного автомата $\mathcal{A}$ выделим непустое подмножество L, не содержащее начальное состояние, и элементы этого множества будем называть *метками*.

В случае, когда на экране имеется ровно одна метка, будем называть её *точкой*.

Законом движения для бесконечного экрана будем называть сверхслово $F = F(1)F(2)...F(n)... \in \{0, 1\}^\infty$, начинающееся с единицы. Вве-

дём обозначения

$l_F(i) = \sum_{t=1}^{i} F(i)$ — число единиц в префиксе длины i сверхслова F,

$s_F(i) = \sum_{t=1}^{i} (1 - F(i))$ — число нулей в префиксе длины i сверхслова F.

Будем говорить, что *на экране* $S(\mathcal{A})$ *реализуется движение точки по закону* F, если существует момент времени $t_0 \in \mathbb{N}$ такой, что выполняются следующие условия:

1) до момента t_0 (включительно) на экране нет меток; момент t_0 будем называть *моментом начала движения* или *началом движения*;

2) для всех $i \in \mathbb{N}$ в момент времени $t_0 + i$ на экране присутствует ровно одна метка в позиции $l_F(i)$.

Таким образом, изменение позиции метки на экране в i-й момент после начала движения ($i \geq 1$) соответствует i-й букве в сверхслове F, а именно, если $F(i) = 0$, то в i-й момент метка остается в той же ячейке, где была в $(i-1)$-й момент, если $F(i) = 1$, то в i-й момент метка сдвинется на одну ячейку вправо.

Будем говорить, что закон движения F *можно реализовать на экране* $S(\mathcal{A})$, если существует такая управляющая последовательность $u \in Q^{\infty}$, при подаче которой на свободный вход экрана $S(\mathcal{A})$ на нём реализуется закон движения F.

Экран $S(\mathcal{A})$ будем называть *универсальным для множества законов движения* $\mathcal{F}$, если для любой закон движения F из $\mathcal{F}$ можно реализовать на экране $S(\mathcal{A})$.Множество всех таких экранов будем обозначать через $\mathcal{U}(\mathcal{F})$.

Множество всех законов движения, реализуемых на некотором экране, обозначим через Impl.

Множество всех законов движения, реализуемых на экране с не более, чем q состояниями, обозначим через $\mathrm{Impl}(q)$.

Через $\mathrm{Impl}(q, t_0)$ обозначим множество законов движения, реализуемых экраном с q состояниями таким образом, что метка появляется на экране не позже, чем в момент времени t_0.

Из определений следует, что

$$\mathrm{Impl} = \bigcup_{q=2}^{\infty} \bigcup_{t_0=1}^{\infty} \mathrm{Impl}(q, t_0). \tag{1}$$

Если $S(\mathcal{A})$ — экран, то через $Q(S)$ обозначим число состояний элементарного автомата $\mathcal{A}$.

Если $\mathcal{F} \subseteq \{0,1\}^\infty$, то обозначим $Q(\mathcal{F}) = \min_{S \in \mathcal{U}(\mathcal{F})} Q(S)$. В случае $\mathcal{U}(\mathcal{F}) = \varnothing$ формально полагаем $Q(\mathcal{F}) = \infty$.

Для закона движения $F = F(1)F(2)...F(n)...$ определим среднюю скорость движения в первые t моментов времени

$$v_F(t) = \frac{1}{t} \sum_{i=1}^{t} F(i) = \frac{l_F(t)}{t}.$$

Обозначим через $\mathcal{F}_a$ множество законов движения F, для которых

$$\limsup_{t \to \infty} v_F(t) \leq a.$$

Теорема 1. *Если $a < 1$, то $\mathcal{F}_a \subset$ Impl и $Q(\mathcal{F}_a) \leq 24 \cdot 2^{1/(1-a)}$.*

Другими словами, все законы движения из множества $\mathcal{F}_a$ при $a < 1$ можно реализовать на экране с $24 \cdot 2^{1/(1-a)}$ состояниями.

Через $\mathcal{F}_{<1}$ обозначим множество законов движения F, для которых

$$\limsup_{t \to \infty} v_F(t) < 1.$$

Заметим, что $\mathcal{F}_{<1} = \bigcup_{0 \leq a < 1} \mathcal{F}_a$. Применяя теорему 1 для всех $a < 1$, получим

Следствие 1. *$\mathcal{F}_{<1} \subset$ Impl, то есть все законы движения из $\mathcal{F}_{<1}$ могут быть реализованы на некотором экране.*

Теперь оценим, насколько мало или велико множество реализуемых законов движения с двух разных точек зрения.

Топологическая характеризация множества Impl. Через U_α обозначим множество всех последовательностей, начинающихся с α, то есть

$$U_\alpha = \{\alpha\beta \mid \beta \in \{0,1\}^\infty\}.$$

На множестве $\{0,1\}^\infty$ стандартным образом определяется топология (называемая тихоновской), база которой есть семейство $\{U_\alpha \mid \alpha \in \{0,1\}^*\}$.

С топологической точки зрения множество является малым, если оно относится к первой категории Бэра, то есть представляется в виде счётного объединения нигде не плотных множеств. По теореме Бэра само

множество $\{0,1\}^\infty$ относится ко второй категории Бэра, как полное метрическое пространство[1].

Теорема 2. *Множество* $\mathrm{Impl}(q,t_0)$ *нигде не плотно относительно введённой топологии.*

На языке сверхслов это означает, что для каждого $\alpha \in \{0,1\}^*$ найдётся такое $\beta \in \{0,1\}^*$, что ни одно сверхслово вида $\alpha\beta\gamma$ (где $\gamma \in \{0,1\}^\infty$), не лежит в множестве $\mathrm{Impl}(q,t_0)$.

Учитывая (1), получим следствие.

Следствие 2. *Множество* Impl *относится к первой категории Бэра.*

Метрическая характеризация множества Impl. На множестве $\{0,1\}^\infty$ можно определить вероятностную меру. Наиболее естественным образом это можно сделать, если представить случайный закон движения, как последовательность независимых бернуллиевских случайных величин $\Xi = (\xi_1, \xi_2, ..., \xi_n, ...)$, причём ξ_i принимает значения 0 и 1 с вероятностью 1/2. По усиленному закону больших чисел имеем

$$v_\Xi(t) = \frac{\xi_1 + ... + \xi_t}{t} \to \frac{1}{2} \quad \text{п.н.},$$

то есть, с вероятностью 1 закон движения Ξ лежит в $F_{1/2}$, а значит по теореме 1 его можно реализовать на экране с 96 состояниями.

Таким образом, с вероятностной точки зрения почти все законы движения реализуемы на одном экране с 96 состояниями. Учитывая следствие 2, видим, что множество Impl хотя и мало с топологической точки зрения, является множеством полной меры.

Множество $\mathrm{Impl} \setminus \mathcal{F}_{<1}$**.** Интересным является вопрос про реализуемость законов движения, не попадающих в множество $\mathcal{F}_{<1}$. Обозначим множество таких законов движения через $\mathcal{F}_{=1}$ и сформулируем теорему, показывающую, что в некотором вероятностном смысле в множестве $\mathcal{F}_{=1}$ мало реализуемых законов движения. Множество $\mathcal{F}_{=1}$ состоит из всех законов движения F, для которых

$$\limsup_{t\to\infty} v_F(t) = 1. \tag{2}$$

[1]Фактически, рассматриваемая топология задаётся 2-адической метрикой, если рассмотреть $\{0,1\}^\infty$, как множество 2-адических чисел. В этой метрике U_α — это шар радиуса $2^{-|\alpha|}$ с центром в произвольной точке множества U_α.

Существенную сложность оценки доли реализуемых законов движения составляет формализация понятия «доля» относительно множества $\mathcal{F}_{=1}$. Проблема состоит в том, что вероятность множества $\mathcal{F}_{=1}$ равна 0, и условная вероятность $\mathbf{P}(\mathrm{Impl}|\mathcal{F}_{=1})$ не определена. Более того, событие $(\Xi \in \mathcal{F}_{=1})$ является хвостовым, то есть не зависит от любого префикса последовательности Ξ. Это означает, что при естественном способе определения вероятности следовало бы считать, что все префиксы встретятся с одинаковой вероятностью. Однако, такое «естественное» предположение не позволяет определить счётно-аддитивную вероятностную меру на $\mathcal{F}_{=1}$. Поэтому, чтобы, тем не менее, как-то оценить долю реализуемых законов движения в $\mathcal{F}_{=1}$, оставаясь при этом в рамках предположения о некоторой равномерности распределения вероятностей, представим множество $\mathcal{F}_{=1}$ в виде объединения подмножеств, на каждом из которых можно определить вероятностную меру, чтобы она была в определённом смысле равномерной.

Для начала переформулируем условие (2) в следующем виде: существует возрастающая последовательность целых чисел $\{n_i\}_{i=0}^{\infty}$ и последовательность положительных чисел $\{\varepsilon_i\}_{i=1}^{\infty}$ такая, что

$$\varepsilon_i \searrow 0 \text{ при } i \to \infty \quad \text{и} \quad \frac{1}{n_i - n_{i-1}} \sum_{j=n_{i-1}+1}^{n_i} F(j) \geq 1 - \varepsilon_i \text{ при } i \in \mathbb{N}. \quad (3)$$

Для фиксированных последовательностей $n = \{n_i\}$ и $\varepsilon = \{\varepsilon_i\}$ введём класс $\mathcal{F}_{n,\varepsilon}$ таких законов движения F, для которых выполнено (3). Фактически, последовательности n и ε задают подпоследовательность и ограничение снизу скорости сходимости v_F к 1. Заметим, что в условии (3) можно взять $\varepsilon_i = \varepsilon_i^0 := 1/i$ и подобрать n_i так, чтобы $n_0 = 0$ и $n_i > 2n_{i-1}$ при $i \geq 1$ (будем писать $n \in N^0$). Таким образом, $\mathcal{F}_{n,\varepsilon} \subseteq \mathcal{F}_{n^0,\varepsilon^0}$ для некоторой последовательности $n^0 \in N^0$.

Для последовательности $n = \{n_i\}_{i=1}^{\infty}$ определим множества

$$B_i^n = \left\{ f \in \{0,1\}^{n_i - n_{i-1}} \;\middle|\; \frac{1}{n_i - n_{i-1}} \sum_{j=1}^{n_i - n_{i-1}} f_j \geq 1 - \varepsilon_i^0 \right\}.$$

Тогда $\mathcal{F}_{n,\varepsilon^0} = \prod_{i=1}^{\infty} B_i^n$ (под произведением понимается конкатенация множеств). Заметим, что поскольку все слова из B_i^n имеют одинаковую длину, то любое слово $F \in \mathcal{F}_{n,\varepsilon^0}$ единственным образом представляется в виде конкатенации слов $f^i \in B_i^n$, $i \in \mathbb{N}$. На каждом из множеств B_i можно ввести равномерную вероятностную меру, положив $\mathbf{P}(f) = |B_i^n|^{-1}$

для всех $f \in B_i^n$. Случайный закон движения из $\mathcal{F}_{n,\varepsilon^0}$ определим, как конкатенацию независимых случайных слов $f^i \in B_i^n$. Тогда на множестве $\mathcal{F}_{n,\varepsilon^0}$ индуцируется вероятностная мера, которую обозначим через $\mathbf{P}_n$.

Теорема 3. *Для любой последовательности $n \in N^0$ выполнено*

$$\mathbf{P}_n(\mathrm{Impl} \cap \mathcal{F}_{n,\varepsilon^0}) = 0.$$

Условие $n \in N^0$ означает, что класс $\mathcal{F}_{n,\varepsilon^0}$ достаточно широк, и включает в себя не только быстро сходящиеся последовательности. Таким образом, теорема утверждает, что в достаточно широких подклассах $\mathcal{F}_{=1}$ доля реализуемых законов движения равна 0.

Здесь следует отметить, что если убрать ограничение $n \in N^0$, то можно подобрать такую последовательность n, что класс $\mathcal{F}_{n,\varepsilon^0}$ будет очень узким, и включать в себя только законы движения вида $a_1 ... a_k 1^\infty$, которые, очевидно, все являются реализуемыми.

3. Доказательства

Отрезком предобработки экрана в момент t будем называть отрезок подряд идущих ячеек на экране, удовлетворяющий следующим условиям:

1) началом отрезка является ячейка, следующая после самой правой метки на экране;

2) состояние самой правой ячейки отрезка отлично от 0;

3) состояние всех ячеек экрана, следующих за последней (самой правой) ячейкой отрезка предобработки равно 0;

4) среди состояний ячеек отрезка нет меток.

3.1. Построение универсального экрана для класса $\mathcal{F}_a$

Поскольку $a < 1$, то существует такое $p \in \mathbb{N}$, что $a < p/(p+1)$. А именно, можно взять $p = \lfloor 1/(1-a) \rfloor$.

Идея алгоритма. Алгоритм строится в несколько этапов. На нескольких первых этапах реализуется вспомогательная разметка экрана, а на последнем этапе при помощи этой разметки реализуется непосредственно само движение. А именно, имеются следующие этапы:

1) Расстановка точек остановки на экране. Для этого в каждый момент времени со скоростью 1 запускается сигнал B_t, кодирующий в себе 2^p возможных расстановок признаков остановки на отрезке длины p. Как только сигнал B достигает не размеченного участка экрана (это происходит в позиции pt), в следующие p моментов времени происходит «расшифровка» сигнала B_t, и в некоторых из ячеек $pt+1$, $pt+2$, ..., $pt+p$ проставляются признаки остановки $S0$, а в остальных — признаки продолжения движения $S1$.

2) Запуск «подталкивающих» сигналов P со скоростью 1 в нужные моменты времени.

3) Реализация требуемого закона движения. Для этого с начала экрана запускается сигнал D со скоростью 1. Как только справа сигнал D оказывается в клетке с признаком остановки $S0$, он останавливается и ждёт, пока слева придёт «подталкивающий» сигнал P. Как только такой сигнал пришёл, D продолжает двигаться вправо до следующего признака остановки.

При правильном подборе всех описанных сигналов, таким способом можно реализовать любой закон движения из $\mathcal{F}_a$. Условие $a < 1 - 1/p$ гарантирует, что вся расстановка признаков остановки S будет готова к моменту её использования сигналом D.

Доказательство теоремы 1. Зададим элементарный автомат $\mathcal{A} = (Q, \varphi, q_0)$. В качестве множества состояний возьмём множество $Q = Q_B \times Q_S \times Q_P \times Q_D$, где

$$\begin{aligned} Q_B &= \{0\} \cup \left\{B_\alpha \mid \alpha \in \{0,1\} \cup \{0,1\}^2 \cup \ldots \cup \{0,1\}^p\right\}, \\ Q_S &= \{0, S0, S1\}, \\ Q_P &= \{0, P\}, \\ Q_D &= \{0, D\}. \end{aligned}$$

Начальным состоянием является состояние $q_0 = (0,0,0,0)$. Определим функцию перехода $\varphi = (\varphi_B, \varphi_S, \varphi_P, \varphi_D)$ покомпонентно.

1) Функция φ_B:

$$\begin{aligned} \varphi_B((B_{x\alpha}, 0, *, *), *, *) &= B_\alpha, && \text{если } x \in \{0,1\}, |\alpha| \geq 1, \\ \varphi_B((B_x, 0, *, *), *, *) &= 0, && x \in \{0,1\}, \\ \varphi_B((B_\alpha, Sx, *, *), *, *) &= B_\alpha, && x \in \{0,1\}, \alpha \in \{0,1\}^*, \\ \varphi_B(*, *, *) &= 0 && \text{в остальных случаях.} \end{aligned}$$

2) Функция φ_S:

$$\begin{aligned}
\varphi_S(*,(B_{x\alpha},0,*,*),*) &= Sx, && x \in \{0,1\},\\
\varphi_S(*,(0,0,*,*),*) &= 0,\\
\varphi_S(*,(*,Sx,*,*),*) &= Sx, && x \in \{0,1\}.
\end{aligned}$$

3) Функция φ_P:

$$\begin{aligned}
\varphi_P((*,*,x,0),*,*) &= x, && x \in Q_P,\\
\varphi_P((*,*,x,D),*,*) &= 0, && x \in Q_P,
\end{aligned}$$

4) Функция φ_D:

$$\begin{aligned}
\varphi_D((*,*,P,D),(*,*,*,0),*) &= D,\\
\varphi_D((*,S1,0,D),(*,*,*,0),*) &= D,\\
\varphi_D((*,*,*,0),(*,S0,0,D),*) &= D,\\
\varphi_D(*,*,*) &= 0 && \text{в остальных случаях.}
\end{aligned}$$

В качестве множества меток возьмём подмножество $Q_B \times Q_S \times Q_P \times \{D\}$.

Для доказательства теоремы 1 покажем, что для любого закона движения $F \in \mathcal{F}_a$ можно подобрать такую управляющую последовательность, что на экране $S(\mathcal{A})$ будет реализовываться закон движения F. Обозначим эту последовательность через $\{\mathcal{C}_F(w)\}_{w=1}^{\infty}$,

$$\mathcal{C}_F(w) = (c_F^B(w), c_F^S(w), c_F^P(w), c_F^D(w)).$$

Построим входную последовательстьноть $\left\{c_F^B(w)\right\}_{w=1}^{\infty}$. Пусть I_0 — множество всех таких $i \in \mathbb{N}$, для которых выполнено $F(i) = 0, F(i-1) = 1$. То есть I_0 — это множество моментов времени, в которые движущаяся точка на экране должна останавливаться. Тогда по определнию функции переходов на разметке экрана в эти моменты должно стоять значение $S0$, а в остальные моменты — $S1$. Заметим, что $l_F(i) = i - s_F(i)$. Тогда последовательность $M = \{m_j\}_{j=1}^{\infty}$, которая должна появиться на разметке экрана, определяется по правилам:

$$\begin{aligned}
m_j &= S0 && \text{при } j = l_F(i) \text{ для некоторого } i \in I_0;\\
m_j &= S1 && \text{в остальных случаях.}
\end{aligned}$$

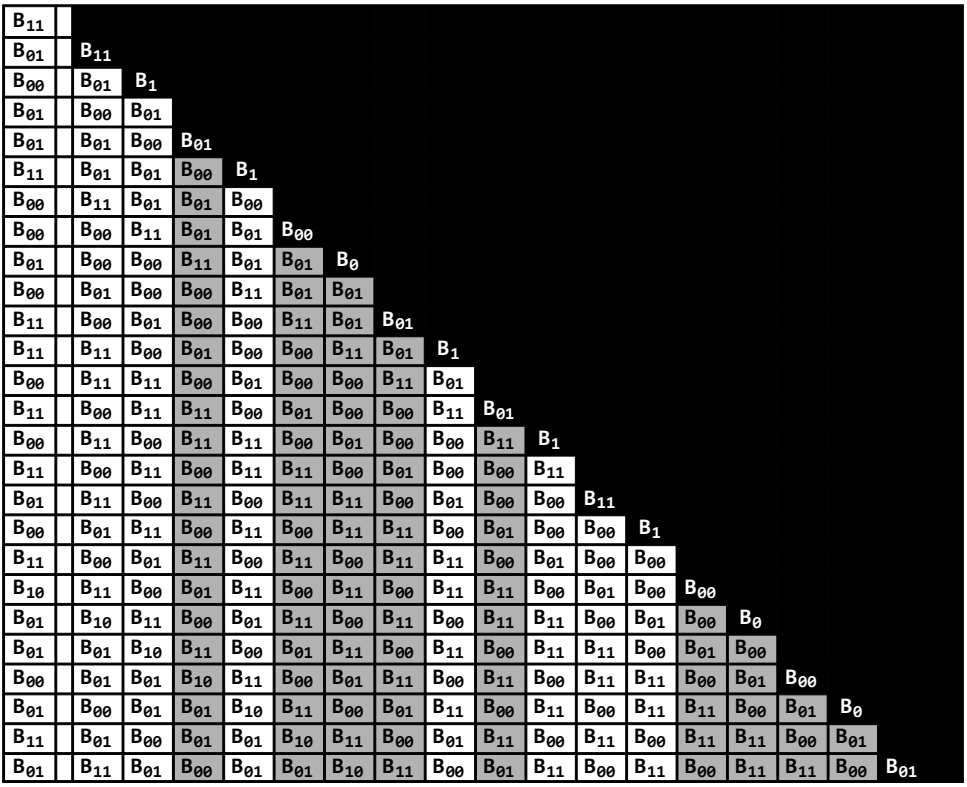


Рис. 1. Пример разметки экрана (цветом изображена компонента Q_S, внутри написана компонента Q_B).

Из элементов последовательности M составим слова α_n длины p:

$$\alpha_n = m_{(n-1)p+1} m_{(n-1)p+2} ... m_{np} \qquad \text{для любого } n \in \mathbb{N}.$$

Из определения функции переходов φ_B видно, что после появления на не размеченной части экрана сигнала B_α, разметка распаковывается по одному символу за один такт. Далее, с задержкой в один такт сигнал $B_{\alpha+1}$ достигает не размеченной части экрана. Таким образом, на экране образуется разметка $m_1, m_2, ...$, и разметка распаковывается со скоростью $\frac{p}{p+1}$.

Определим $c_F^B(w) = B_{\alpha_w}$ для любого $w \in \mathbb{N}$.

Построим теперь $\left\{c_F^P(w)\right\}$. Пусть I_1 — последовательность всех таких $i \in \mathbb{N}$, для которых выполнено $F(i) = 1, F(i-1) = 0$. Следовательно, в момент i после начала движения метка на экране должна сдвинуться

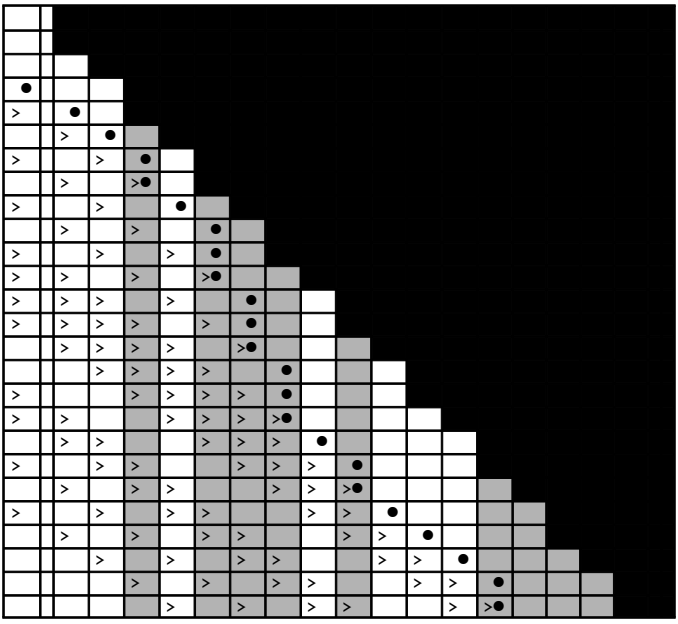


Рис. 2. Реализация закона движения на размеченном экране (• означает метку на экране, а > обозначает, что компонента Q_P равна P).

вправо. По определению функции переходов состояний это означает, что в момент $i-1$ в ячейке с меткой третья координата равна P. Координата точки на экране в момент $i-1$ равна $i-1-s_F(i-1)$.

Тогда в j-й момент после начала движения движущий сигнал определяется следующей последовательностью $\{p_j\}_{j=0}^{\infty}$:

$$\begin{aligned} p_j &= P && \text{при } j = s_F(i-1) \text{ для некоторого } i \in I_1; \\ p_j &= 0 && \text{в остальных случаях.} \end{aligned}$$

Определение p_j корректно, поскольку минимальное $i \in I_1$ не может быть меньше 2. Также, по определению функции φ_P видно, что сигнал P распространяется со скоростью 1. По построению p_j ясно, что к моменту $i-1$ сигнал достигнет ячейки с координатой $i-1-s_F(i-1)$.

Проверим, что для любого закона движения $F \in \mathcal{F}_a$ существует момент $t_0 \in \mathbb{N}$ для запуска сигнала D, что этот сигнал будет всегда двигаться по размеченному сигналами $S0$ и $S1$ отрезку экрана. Заметим, что в момент времени $t+t_0$ длина длина размеченного отрезка экрана

равна

$$\left\lceil (t_0+t)\frac{p}{p+1} \right\rceil \geq (t_0+t)\frac{p}{p+1}.$$

Поэтому достаточно найти такое t_0, что для любого $t \in \mathbb{N}$ для координаты сигнала D на экране выполняется неравенство $l_F(t) < (t_0+t)\frac{p}{p+1}$.

По условию теоремы имеем $\limsup_{t\to\infty} v_F(t) \leq a$, значит для произвольного $\varepsilon \in (0, p/(p+1)-a)$ существует такое $t_* \in \mathbb{N}$, что для любого $t > t_*, t \in \mathbb{N}$ выполнено

$$v_F(t) \leq a+\varepsilon < \frac{p}{p+1},$$

$$\frac{l_F(t)}{t} < \frac{p}{p+1},$$

$$l_F(t) < t\frac{p}{p+1}.$$

Если взять $t_0 = t_* + 1$, то для всех $t \in \mathbb{N}$ выполнено

$$l_F(t) < (t_0+t)\frac{p}{p+1}.$$

То есть, мы сможем запустить сигнал D таким образом, чтобы он двигался всегда по размеченной части экрана.

Определим теперь каждую компоненту входной последовательности $\mathcal{C}_F(w) = (c_F^B(w), c_F^S(w), c_F^P(w), c_F^D(w))$:

$$\begin{aligned} c_F^B &= B_{\alpha_1}B_{\alpha_2}B_{\alpha_3}..., \\ c_F^S &= (S1)^\infty, \\ c_F^P &= 0^{t_0}p_0p_1..., \\ c_F^D &= 0^{t_0-1}D0^\infty. \end{aligned}$$

Покажем, что на построенном экране с помощью указанной входной последовательности строится движение точки по закону F. Для этого достаточно показать по индукции, что для произвольного $i \in \mathbb{N}$ в момент времени t_0+i позиция точки на экране будет равна $l_F(i)$.

База индукции ($i = 1$). В момент времени t_0 окрестность самой левой ячейки экрана имеет вид $((*, S1, 0, D), (*, *, 0, 0), (*, *, 0, 0))$, значит по определению компоненты функции перехода φ_D, в момент времени t_0+1 компонента D станет равна D.

Шаг индукции. Предположим, что в момент времени (t_0+i-1) позиция точки на экране равна $l_F(i-1)$ и покажем, что в следующий момент времени точка окажется в позиции $l_F(i)$. Рассмотрим 2 случая.

1) Пусть $F(i)=0$. Тогда существует такое k, что $F(i-k-1)=1, F(i-k)=0$. По определению последовательности M для $j=l_F(i-k)$ выполнено $m_j=S0$. Заметим, что

   $$j=l_F(i-k)=i-k-s_F(i-k)=i-k-(s_F(i)-k)=i-s_F(i)=l_F(i).$$

   Значит, в ячейке стоит стоп-сигнал, и в следующий момент точка не поменяет своего положения, то есть останется в позиции $l_F(i-1)=l_F(i-1)+F(i)=l_F(i)$.

2) Пусть $F(i)=1$. Рассмотрим два подслучая.

   а) Если $F(i-1)=1$, то в ячейке с точкой (в позиции $l_F(i-1)$) стоит $S0$, если найдется $j\in I_0$, для которого $l_F(j)=l_F(i-1)$. Так как $F(i-1)=1$, то для всех $j<i-1$ выполнено $l_F(j)<l_F(i-1)$. Очевидно, $(i-1)\notin I_0$, а для $j>i-1$ выполнено $l_F(j)>l_F(i-1)$. Значит, такого $j\in I_0$, для которого $l_F(j)=l_F(i-1)$ не нашлось, следовательно, $m_{i-1}=S1$.

   б) Если $F(i-1)=0$, то по определению $i\in I_1$, то есть $p_j=P$ для $j=s_F(i-1)$, и в ячейке с точкой в третьей координате будет стоять P, и точка сдвинется вправо.

   Значит в обоих подслучаях точка сдвинется вправо и окажется в позиции $l_F(i-1)+1=l_F(i-1)+F(i)=l_F(i)$.

Шаг индукции доказан, значит на экране воспроизведется движение точки по закону F.

Количество состояний элементарного автомата $\mathcal{A}$ равно

$$|Q|=|Q_B|\cdot|Q_S|\cdot|Q_P|\cdot|Q_D|=12\sum_{i=0}^{p}2^i<24\cdot 2^p.$$

Подставляя $p=\lfloor 1/(1-a)\rfloor$, получим требуемую в теореме оценку. Теорема доказана. □

3.2. Оценка доли законов движения в множестве $\mathcal{F}_{=1}$

Если $A \subseteq \{0,1\}^\infty$, то через $A\big\rfloor_k$ обозначим множество всех префиксов слов из множества A длины k, то есть

$$A\big\rfloor_k = \left\{\alpha \in \{0,1\}^k \mid \exists \beta \in \{0,1\}^\infty : \alpha\beta \in A\right\}.$$

Через $D_{k,z}$ обозначим множество слов $w \in \{0,1\}^k$, в которых не более z нулей.

Лемма 1. *Для любых $k, z \in \mathbb{N}$, а также любых $t_0, q \in \mathbb{N}$ выполнено*

$$\log_q \left|\mathrm{Impl}(q,t_0)\big\rfloor_k \cap D_{k,z}\right| \le q^3 + t_0 + z.$$

Доказательство. Зафиксируем некоторую последовательность

$$w \in \mathrm{Impl}(q,t_0)\big\rfloor_k \cap D_{k,z}.$$

Тогда существует закон движения $F \in \mathrm{Impl}(q,t_0)$ такой, что $w = F\big\rfloor_k$. Поскольку $F \in \mathrm{Impl}(q,t_0)$ существует элементарный автомат $\mathcal{A}$ и управляющая последовательность $u \in E_q^\infty$ такая, что на экране реализуется движение точки по закону F, причём в момент времени t_0 на экране уже есть метка. Пусть t – последний момент, когда на экране не было метки. Тогда в момент времени $t' > t$ позиция метки будет $t' - t - s_F(t'-t) =: pos(t')$. Поскольку радиус окрестности 1, то состояние ячейки $pos(t')$ в момент t' зависит лишь от префикса управляющей последовательности u до момента $t' - pos(t') = t + s_F(t'-t) \le t_0 + s_F(t'-t)$.

Поскольку $F\big\rfloor_k \in D_{k,z}$, то $s_F(k) \le z$. Таким образом, движение точки до момента $t+k$ управляется первыми $t + s_F(k) \le t_0 + z$ символами последовательности u. С другой стороны, это движение однозначно задаёт последовательность $F\big\rfloor_k = w$. Таким образом w однозначно задаётся автоматом $\mathcal{A}$ с q состояниями и $t_0 + z$ символами из E_q.

1) Элементарный автомат однозначно задаётся своей функцией перехода $\varphi : Q \times Q \times Q \to Q$. Значит существует не более q^{q^3} элементарных автоматов с q состояниями.

2) Всего имеется q^{t_0+z} различных последовательностей из E_q длины $t_0 + z$.

Перемножив число элементарных автоматов и число различных управляющих последовательностей, получим требуемую оценку

$$\left|\mathrm{Impl}(q,t_0)\big\rfloor_k \cap D_{k,z}\right| \le q^{q^3} q^{t_0+z}.$$

Взяв логарифм по основанию q, получим требуемую оценку. Лемма доказана. $\square$

Доказательство теоремы 2. Чтобы показать, что множество Impl нигде не плотно, рассмотрим произвольное открытое множество $U \subseteq \{0,1\}^\infty$. В нём содержится открытое подмножество U_α для некоторого $\alpha \in \{0,1\}^*$. Через z обозначим число нулей в слове α.

Рассмотрим множества $G_k = \alpha D_{2^k,k}$. тогда $G_k \subseteq D_{2^k+|\alpha|,k+z}$, и по лемме 1 имеем

$$\log_q \left| \mathrm{Impl}(q,t_0)\big\rfloor_{2^k+|\alpha|} \cap G_k \right| \leq q^3 + t_0 + k + z.$$

Оценим теперь мощность множества G_k.

$$|G_k| = |D_{2^k,k}| = \sum_{j=0}^{k} C_{2^k}^j \geq C_{2^k}^k = \frac{2^k}{k} \cdot \frac{2^k-1}{k-1} \cdot \ldots \cdot \frac{2^k-(k-1)}{k-(k-1)} \geq \left(\frac{2^k}{k}\right)^k.$$

Учитывая, что параметры q, t_0 и z фиксированы, легко видеть, что при достаточно большом k выполнено

$$\log_q |G_k| \geq k^2 \log_q 2 - k \log_q k > q^3 + t_0 + k + z \geq \log_q \left| \mathrm{Impl}(q,t_0)\big\rfloor_{2^k+|\alpha|} \cap G_k \right|.$$

Это означает, что $G_k \setminus \mathrm{Impl}(q,t_0)\big\rfloor_{2^k+|\alpha|} \neq \varnothing$, то есть существует такое $\gamma \in G_k$, что $\gamma \notin \mathrm{Impl}(q,t_0)\big\rfloor_{2^k+|\alpha|}$, а это в свою очередь означает, что $U_\gamma \cap \mathrm{Impl}(q,t_0) = \varnothing$. Осталось заметить, что γ, как элемент G_k, имеет префикс α, значит $U_\gamma \subseteq U_\alpha \subseteq U$.

Таким образом, для множества U мы нашли его открытое подмножество U_γ, в котором нет ни одного элемента $\mathrm{Impl}(q,t_0)$. Учитывая, что U — произвольное открытое множество, получаем, что $\mathrm{Impl}(q,t_0)$ нигде не плотно. $\square$

Лемма 2. *Для любой последовательности $n \in N^0$ и любых $t_0, q \in \mathbb{N}$ при достаточно больших $k \in \mathbb{N}$ выполнено*

$$\log_q \left| \mathrm{Impl}(q,t_0)\big\rfloor_{n_k} \cap B_1^n \ldots B_k^n \right| \leq q^3 + t_0 + 3\frac{n_k}{k}.$$

Доказательство. Учитывая лемму 1, достаточно показать, что $B_1^n \ldots B_k^n \subseteq D_{k,\lfloor 3n_k/k \rfloor}$. Рассмотрим слово $w \in B_1^n \ldots B_k^n$. Тогда $w = w_1 \ldots w_k$, $w_i \in B_i^n$.

По определению множества B_i^n в последовательности w_i доля нулей не более $1/i$. Учитывая условие $n_i > 2n_{i-1}$, получим

$$s_w(n_k) \le \sum_{i=1}^{k} \frac{n_i - n_{i-1}}{i} \le \frac{n_k}{k} \sum_{i=1}^{k} \frac{n_i}{n_k} \cdot \frac{k}{i} < \quad \langle \text{заменим } i \mapsto k-j \rangle$$
$$< \frac{n_k}{k} \sum_{j=0}^{k-1} \left(\frac{2}{3}\right)^j \left(\frac{3}{4}\right)^j \frac{k}{(k-j)}.$$

Несложно убедиться, что взяв k достаточно большим, $\left(\frac{3}{4}\right)^j \frac{k}{(k-j)} \le 1$ для всех $j = 0, ..., k-1$, а значит

$$s(n_k) \le \frac{n_k}{k} \sum_{j=0}^{k-1} \left(\frac{2}{3}\right)^j \le 3\frac{n_k}{k}.$$

Применяя лемму 1, получим требуемую оценку. □

Лемма 3. *Для любой последовательности $n \in N^0$ и любых $t_0, q \in \mathbb{N}$ выполнено*

$$\mathbf{P}_n(\mathrm{Impl}(q, t_0) \cap \mathcal{F}_{n,\varepsilon^0}) = 0.$$

Доказательство. Для краткости введём обозначение $A := \mathrm{Impl}(q, t_0)$. При любом k выполнено

$$A \cap \mathcal{F}_{n,\varepsilon^0} \subseteq \left(A\big\rfloor_{n_k} \cap B_1^n ... B_k^n\right) \cdot \prod_{i=k+1}^{\infty} B_i^n.$$

Поскольку мера $\mathbf{P}_n$ определена на множестве $\mathcal{F}_{n,\varepsilon^0}$, как на произведении множеств B_i^n, то для всех $k \in \mathbb{N}$ выполнено

$$\mathbf{P}_n(A \cap \mathcal{F}_{n,\varepsilon^0}) \le \frac{|A\big\rfloor_{n_k} \cap B_1^n ... B_k^n|}{|B_1^n ... B_k^n|}. \tag{4}$$

Заметим, что множество B_k^n включает в себя все последовательности длины $n_k - n_{k-1}$, в которых ровно $z_k := \lfloor (n_k - n_{k-1})/k \rfloor$ нулей. Таким образом,

$$|B_1^n ... B_k^n| \ge |B_i^n| \ge C_{n_k - n_{k-1}}^{z_k} = \prod_{i=0}^{z_k - 1} \frac{n_k - n_{k-1} - i}{z_k - i} \ge \left(\frac{n_k - n_{k-1}}{z_k}\right)^{z_k} \ge k^{z_k}. \tag{5}$$

Поскольку $n \in N^0$, имеем $n_k - n_{k-1} > n_k/2$ и $n_k \geq 2^k$, отсюда при $k \geq 4$ выполнено

$$z_k \geq \left(1 - \frac{2k}{2^k}\right) \frac{n_k - n_{k-1}}{k} \geq \frac{n_k}{4k}.$$

Подставляя оценку для z_k в (5) и применяя логарифм к обеим частям получим, что при достаточно большом k выполнена оценка

$$\log_q |B_1^n ... B_k^n| \geq \frac{n_k}{4k} \log_q k.$$

Оценим логарифм $\mathbf{P}_n(A \cap \mathcal{F}_{n,\varepsilon^0})$, используя (4) и лемму 2:

$$\log_q(\text{П.Ч. (4)}) \leq q^3 + t_0 + 3\frac{n_k}{k} - \frac{n_k}{4k} \log_q k =$$
$$= \underbrace{q^3 + t_0}_{\text{const}} + \underbrace{\frac{n_k}{k}}_{\geq 2^k/k} \left(3 - \frac{1}{4} \log_q k\right) \to -\infty \quad \text{при} \quad k \to \infty.$$

Поскольку в (4) параметр k можно выбрать произвольно, получим $\mathbf{P}_n(A \cap \mathcal{F}_{n,\varepsilon^0}) = 0$. Лемма доказана. □

Доказательство теоремы 3. Из (1), счётной аддитивности вероятностной меры $\mathbf{P}_n$ и леммы 3 получим

$$\mathbf{P}_n(\text{Impl} \cap \mathcal{F}_{n,\varepsilon^0}) \leq \sum_{t_0=1}^{\infty} \sum_{q=2}^{\infty} \mathbf{P}_n(\text{Impl}(q, t_0) \cap \mathcal{F}_{n,\varepsilon^0}) = 0.$$

Теорема доказана. □

Список литературы

[1] Болотов А. А. О задачах сводимости и выразимости для однородных структур со входами и выходами // ДАН СССР — 1980. — Т. 254. №1 — С. 14–16.

[2] Верещагин Н. К., Успенский В. А., Шень А. Колмогоровская сложность и алгоритмическая случайность. — М.: МЦНМО, 2013.

[3] Глушков В. М., Амосов Н. М., Артеменко И. А. Энциклопедия кибернетики. Том 1. — Киев: УСЭ, 1974.

[4] Кудрявцев В. Б. О мощности множеств предполных множеств некоторых функциональных систем, связанных с автоматами // Проблемы кибернетики. — 1965. — Вып. 13 — С. 45–74.

[5] Кудрявцев В. Б., Алешин С. В., Подколзин А. С. Введение в теорию автоматов. — М.: Наука, 1985.

[6] Кудрявцев В. Б., Подколзин А. С., Болотов А. А. Основы теории однородных структур. — М.: Наука, 1990.

[7] Мур Э. Ф. Математические модели самовоспроизведения. В кн.: Математические проблемы в биологии: пер. с англ. – М.: Мир, 1966. — С. 36–62.

[8] Фон Нейман Дж. Теория самовоспроизводящихся автоматов. — М.: Мир, 1971.

[9] Окстоби Дж. Мера и категория. — М.: Мир, 1974.

[10] Подколзин А. С. О сложности моделирования в однородных структурах // Проблемы кибернетики. — Вып. 30 — 1975. — С. 199–255.

[11] Подколзин А. С. Об универсальных однородных структурах // Проблемы кибернетики. — Вып. 34. — 1978. — С. 109–131.

[12] Титова Е. Е. Конструирование изображений клеточными автоматами // Интеллектуальные системы. — 2014. — Т. 18, № 1. — С. 105–121.

[13] Хинчин А. Я. Основные законы теории вероятностей. — М.: ГТТИ, 1932.

[14] Чашкин А. В. Дискретная математика. — М.: Издательский центр «Академия», 2012.

On the size of the set of motion rules realizable by cellular automata

Kalachev G.V., Titova E.E.

A model of motion of a dot on the screen is considered. The screen is a semi-infinite one-dimensional cellular automaton. A particular subset of states of the screen is called the image of the dot. A rule of motion

is defined by an infinite sequence of zeros and ones, corresponding respectively to the "stop"and "go"commands. For a broad class of motion rules, an algorithm for implementing the given rule is described. A Bernoulli probability measure of realizable motion rules is explored. It is shown that almost all motion rules are realizable. Also it is shown that the set of realizable motion rules is meager with respect to the product topology.

Keywords: cellular automaton, universal screen, motion of the dot, rule of motion, Bernoulli measure, product topology, Baire category.

Критерий совпадения V-реализуемости формул расширения L языка арифметики с классической семантикой языка L.

Коновалов А. Ю.

Пусть L — некоторое расширение языка арифметики, V — некоторый класс числовых функций. Определяется понятие V-реализуемости для формул языка L таким образом, что индексы функций из класса V используются для интерпетации импликации и квантора всеобщности. Доказывается, что семантика V-реализуемости для языка L совпадает с классической семантикой этого языка тогда и только тогда, когда класс V содержит все функции, определимые в языке L.

Ключевые слова: конструктивная семантика, реализуемость, обобщенная реализуемость, формальная арифметика.

Пусть V — некоторое множество частичных функций натурального аргумента. Элементы множества V назовем V-функциями. Будем считать, что для каждого натурального числа n имеется нумерация всех n-местных V-функций. А именно, определено множество индексов $\mathsf{I}_n^V \subseteq \mathbb{N}$ вместе с отображением, которое каждому натуральному числу $z \in \mathsf{I}_n^V$ ставит в соответствие n-местную V-функцию $\varphi_z^{V,n}$, и при этом всякая n-местная V-функция есть $\varphi_z^{V,n}$ для некоторого $z \in \mathsf{I}_n^V$. Будем считать, что множество V вместе с вышеописанной нумерацией обладает следующими свойствами:

C1) V содержит все частично-рекурсивные функции;

C2) если ψ есть n-местная V-функция, s — перестановка на множестве $\{1, \ldots, n\}$, то функция ψ', определенная условным равенством $\psi'(x_1, \ldots, x_n) \simeq \psi(x_{s(1)}, \ldots, x_{s(n)})$, является V-функцией;

C3) если ψ есть n-местная V-функция, то функция ψ', определенная условным равенством

$$\psi'(x_1, \ldots, x_n, x_{n+1}) \simeq \psi(x_1, \ldots, x_n),$$

является V-функцией;

C4) композиция V-функций есть V-функция;

C5) если ψ_1, ψ_2 суть $(n+1)$-местные V-функции, то функция ψ', определенная условным равенством

$$\psi'(x_1,\ldots,x_n) \simeq \mu x\,[\psi_1(x_1,\ldots,x_n,x) = \psi_2(x_1,\ldots,x_n,x)],$$

является V-функцией (μ — оператор минимизации);

C6) если ψ_1, ψ_2 суть n-местные V-функции, χ — всюду определенная n-местная V-функция, то функция ψ', определенная условным равенством

$$\psi'(x_1,\ldots,x_n) \simeq \begin{cases} {}_1(x_1,\ldots,x_n), & \text{если } \chi(x_1,\ldots,x_n)=1; \\ {}_2(x_1,\ldots,x_n), & \text{иначе,} \end{cases}$$

является V-функцией;

C7) для каждой $(n+m)$-местной V-функции ψ найдется всюду определенная m-местная V-функция ψ', что справедливо условное равенство

$$\varphi^{V,\,n}_{\psi'(x_{n+1},\ldots,x_{n+m})}(x_1,\ldots,x_n) \simeq \psi(x_1,\ldots,x_n,x_{n+1},\ldots,x_{n+m}).$$

Множество всех частично-рекурсивных функций обладает свойствами C1–C7. Другим примером функций, обладающих свойствами C1–C7, могут служить все арифметические функции или все гиперарифметические функции с подходящей нумерацией (см. [1], [2]).

Будем считать, что язык формальной арифметики LA содержит обозначения для всех примитивно рекурсивных функций, константы для обозначения всех натуральных чисел, а также логические константы $\top$ (истина) и $\bot$ (ложь), которые считаются атомарными формулами (атомами). Расширение LA' языка LA получается добавлением к LA предикатных символов P_i^n и функциональных символов f_i^n для всех $i \geq 0,\ n \geq 1$. Валентность символов P_i^n и f_i^n полагается равной n. Формулы языка LA' строятся из атомов при помощи логических связок $\wedge$, $\vee$, $\to$ и кванторов $\exists$, $\forall$, причем квантор $\forall$ используется следующим образом: если A и B — формулы, $\overline{x} = x_1,\ldots,x_n$ — список переменных, то выражение $\forall\overline{x}\,(A \to B)$ считается формулой. Такое определение формулы навеяно идеями из базисной арифметики (см. [3]). Выражение $\neg A$ условимся рассматривать как сокращение для формулы $A \to \bot$. Выражение $A(x_1,\ldots,x_n)$ означает, что все свободные переменные формулы A находятся в списке $x_1,\ldots,x_n$. Будем считать, что фиксированы расширение L языка LA и интерпретация $\mathcal{N}_L$ языка L такие, что L — подъязык

языка LA', и интерпретация $\mathcal{N}_L$ является продолжением стандартной интерпретацией языка LA. Заметим, что при этом $\mathcal{N}_L \models \top$ и $\mathcal{N}_L \not\models \bot$.

Представляет интерес рассмотрение варианта конструктивной логики, основанного на использовании V-функций как конструктивного способа получения одних реализаций из других. Понятие V-реализуемости для языка L определим по аналогии с рекурсивной реализуемостью Клини [4, §82].

Пусть фиксированы примитивно-рекурсивные двухместная функция c, которая взаимно однозначно нумерует все пары натуральных чисел, и одноместные обратные функции p_1 и p_2, так что выполняются соотношения $\mathsf{p}_1(\mathsf{c}(x,y)) = x$ и $\mathsf{p}_2(\mathsf{c}(x,y)) = y$. В выражениях вида $\mathsf{p}_1(t)$, $\mathsf{p}_2(t)$ обычно будем опускать скобки.

Для каждого натурального числа e и произвольной замкнутой формулы Φ языка L определим отношение $e\,\mathbf{r}^V\,\Phi$ (e V-реализует Φ) индукцией по построению формулы Φ:

1) $e\,\mathbf{r}^V\,\Phi \rightleftharpoons \mathcal{N}_L \models \Phi$, если Φ — атом языка L;

2) $e\,\mathbf{r}^V\,(\Phi \wedge \Psi) \rightleftharpoons \mathsf{p}_1 e\,\mathbf{r}^V\,\Phi$ и $\mathsf{p}_2 e\,\mathbf{r}^V\,\Psi$;

3) $e\,\mathbf{r}^V\,(\Phi \vee \Psi) \rightleftharpoons (\mathsf{p}_1 e = 0$ и $\mathsf{p}_2 e\,\mathbf{r}^V\,\Phi)$ или $(\mathsf{p}_1 e = 1$ и $\mathsf{p}_2 e\,\mathbf{r}^V\,\Psi)$;

4) $e\,\mathbf{r}^V\,\exists x\,\Phi(x) \rightleftharpoons \mathsf{p}_2 e\,\mathbf{r}^V\,\Phi(\mathsf{p}_1 e)$;

5) $e\,\mathbf{r}^V\,\forall x_1,\ldots,x_n\,(\Phi(x_1,\ldots,x_n) \to \Psi(x_1,\ldots,x_n)) \rightleftharpoons e \in \mathsf{I}^V_{n+1}$ и для всех[1] натуральных чисел s, $a_1,\ldots,a_n$, если верно $s\,\mathbf{r}^V\,\Phi(a_1,\ldots,a_n)$, то определено $\varphi_e^{V,n+1}(a_1,\ldots,a_n,s)$ и $\varphi_e^{V,n+1}(a_1,\ldots,a_n,s)\,\mathbf{r}^V\,\Psi(a_1,\ldots,a_n)$.

Замкнутую формулу Φ языка L назовем *V-реализуемой* (обозначение: $\mathbf{r}^V\,\Phi$), если найдется такое натуральное число e, что $e\,\mathbf{r}^V\,\Phi$.

Семантики расширений языка арифметики, основанные на понятии V-реализуемости для некоторых конкретных классов V, рассматривались в работах [1] и [2]. Там исследовались соотношения таких семантик с базисной и интуиционисткой логикой. Сейчас мы установим критерий совпадения семантик, основанных на понятии V-реализуемости, с классической логикой.

Будем говорить, что n-местная частичная функция ψ определима в языке L формулой $\Phi(x_1,\ldots,x_n,y)$ этого языка, если имеет место

$$(k_1,\ldots,k_n) = k \Longleftrightarrow \mathcal{N}_L \models \Phi(k_1,\ldots,k_n,k)$$

для всех натуральных чисел $k_1,\ldots,k_n,k$. Множество всех функций, определимых в языке L, обозначим $\mathsf{F}(L)$.

[1] Однако, если в списке $x_1,\ldots,x_n$ на некоторых позициях i и j стоят одинаковые переменные x_i и x_j, то мы не допускаем рассмотрение тех списков $a_1,\ldots,a_n$, в которых $a_i \neq a_j$.

Верны следующие теоремы.

Теорема 1. *Пусть* $V \supseteq \mathsf{F}(L)$. *Тогда имеет место*

$$\mathbf{r}^V \Phi \Longleftrightarrow \mathcal{N}_L \models \Phi$$

для всех замкнутых формул Φ *языка* L.

Теорема 2. *Пусть* $V \not\supseteq \mathsf{F}(L)$. *Тогда найдется формула* $\Phi(z)$ *языка* L, *для которой формула* $\forall z\,(\Phi(z) \vee \neg\Phi(z))$ *не является* V*-реализуемой.*

Список литературы

[1] Коновалов А. Ю., Плиско В. Е. О гиперарифметической реализуемости // Мат. зам. 2015. **98**, №5. 725—746.

[2] Коновалов А. Ю. Арифметическая реализуемость и базисная логика // Вестн. Моск. ун-та. Матем. Механ. 2016. №1. 52–56.

[3] Provably total functions of basic arithmetic // Math. Log. Quart. 2003. **49**. N 3. 316—322.

[4] Клини С. К. Введение в метаматематику. М.: ИЛ, 1957.

The V-realizability for L-formulas coincides with the classical semantics iff V contains all L-definable functions.

Konovalov A. Yu.

Let L be an extension of the language of arithmetic, V a class of number-theoretical functions. A notion of the V-realizability for L-formulas is defined in such a way that indexes of functions in V are used for interpreting the implication and the universal quantifier. It is proved that the semantics for L based on the V-realizability coincides with the classical semantics iff V contains all L-definable functions.

Keywords: constructive semantics, realizability, generalized realizability, formal arithmetic.

Синтез легкотестируемых схем при однотипных константных неисправностях на входах и выходах элементов

Попков К.А.

Доказаны следующие утверждения: для любого натурального k и любой булевой константы p существует базис, состоящий из булевой функции от $\max(k+1;3)$ переменных и отрицания одной переменной (существует базис, состоящий из булевой функции от не более чем $2{,}5k+2$ переменных и отрицания этой функции), в котором любую булеву функцию, кроме константы p, можно реализовать схемой из функциональных элементов, неизбыточной и допускающей проверяющий (соответственно, диагностический) тест длины не более 2 относительно не более k однотипных константных неисправностей типа p на входах и выходах элементов. Показано, что при рассмотрении только однотипных константных неисправностей типа p на входах элементов указанные оценки длин тестов можно понизить до 1.

Ключевые слова: схема из функциональных элементов, однотипная константная неисправность, проверяющий тест, диагностический тест.

1. Введение

В работе рассматривается задача синтеза легкотестируемых схем, реализующих заданные булевы функции. Логический подход к тестированию электрических схем предложен С. В. Яблонским и И. А. Чегис в [1]; этот подход также применим к тестированию схем из функциональных элементов (см. [2, 3, 4]). Пусть имеется схема из функциональных элементов S с одним выходом, реализующая булеву функцию $f(\tilde{x}^n)$, где $\tilde{x}^n = (x_1, \ldots, x_n)$. Под воздействием некоторого источника неисправностей один или несколько элементов схемы S могут перейти в неисправное состояние. В результате схема S вместо исходной функции $f(\tilde{x}^n)$ будет

реализовывать некоторую булеву функцию $g(\tilde{x}^n)$, вообще говоря, отличную от f. Все такие функции $g(\tilde{x}^n)$, получающиеся при всевозможных допустимых для рассматриваемой задачи неисправностях элементов схемы S, называются *функциями неисправности* данной схемы.

Введём следующие определения [2, 3, 4]. *Проверяющим тестом* для схемы S называется такое множество T наборов значений переменных $x_1, \ldots, x_n$, что для любой отличной от $f(\tilde{x}^n)$ функции неисправности $g(\tilde{x}^n)$ схемы S в T найдётся набор $\tilde{\sigma}$, на котором $f(\tilde{\sigma}) \neq g(\tilde{\sigma})$. *Диагностическим тестом* для схемы S называется такое множество T наборов значений переменных $x_1, \ldots, x_n$, что T является проверяющим тестом и, кроме того, для любых двух различных функций неисправности $g_1(\tilde{x}^n)$ и $g_2(\tilde{x}^n)$ схемы S в T найдётся набор $\tilde{\sigma}$, на котором $g_1(\tilde{\sigma}) \neq g_2(\tilde{\sigma})$. Число наборов в T называется *длиной* теста. В качестве тривиального диагностического (и проверяющего) теста длины 2^n для схемы S всегда можно взять множество, состоящее из всех двоичных наборов длины n. Тест называется *полным*, если в схеме могут быть неисправны сколько угодно элементов, и *единичным*, если в схеме может быть неисправен только один элемент. Единичные тесты обычно рассматривают для неизбыточных схем (см. [4, с. 110–111]), т. е. для таких схем, в которых любая допустимая неисправность любого одного элемента приводит к функции неисправности, отличной от исходной функции, реализуемой данной схемой (такие функции неисправности называют *нетривиальными*).

Назовём проверяющий (диагностический) тест *k-проверяющим* (*k-диагностическим*), если в схеме могут быть неисправны не более k элементов, где $k \in \mathbb{N}$. Будем рассматривать такие тесты только для *k-неизбыточных схем* (см. [5, с. 68]), в которых любая допустимая неисправность не менее одного и не более k элементов приводит к нетривиальной функции неисправности. Очевидно, что понятия 1-проверяющего и 1-диагностического тестов совпадают с понятиями единичного проверяющего и единичного диагностического тестов соответственно.

Любое множество булевых функций будем называть *базисом*.

Пусть зафиксирован вид неисправностей элементов, B — произвольный функционально полный базис и T — единичный проверяющий тест для некоторой схемы из функциональных элементов S в базисе B. Введём следующие обозначения: пусть $D^B_{\text{ЕП}}(T)$ — длина теста T; $D^B_{\text{ЕП}}(S) = \min D^B_{\text{ЕП}}(T)$, где минимум берётся по всем единичным проверяющим тестам T для схемы S; $D^B_{\text{ЕП}}(f) = \min D^B_{\text{ЕП}}(S)$, где минимум берётся по всем неизбыточным схемам S в базисе B, реализующим функцию f; $D^B_{\text{ЕП}}(n) = \max D^B_{\text{ЕП}}(f)$, где максимум берётся по всем булевым функ-

циям f от n переменных, для которых определено значение $D^B_{\text{ЕП}}(f)$. Функция $D^B_{\text{ЕП}}(n)$ называется *функцией Шеннона* длины единичного проверяющего теста. По аналогии с функциями $D^B_{\text{ЕП}}$ можно ввести функции $D^B_{\text{ПП}}$, $D^B_{k\text{-П}}$, $D^B_{\text{ЕД}}$, $D^B_{\text{ПД}}$ и $D^B_{k\text{-Д}}$ для соответственно полного проверяющего, k-проверяющего, единичного и полного диагностического и k-диагностического тестов, зависящие от T, от S, от f и от n (в определениях функций $D^B_{\text{ПП}}(f)$ и $D^B_{\text{ПД}}(f)$ не предполагается неизбыточность схем, а в определениях функций $D^B_{k\text{-П}}(f)$ и $D^B_{k\text{-Д}}(f)$ предполагается k-неизбыточность схем). Так, например, $D^B_{k\text{-Д}}(n)$ — функция Шеннона длины k-диагностического теста.

Перечислим основные результаты, касающиеся тестирования схем из функциональных элементов. Класс допустимых неисправностей функциональных элементов ограничим константными неисправностями на входах и выходах элементов, а также только на входах элементов, при которых значение на неисправном входе (выходе) любого элемента становится равно некоторой булевой константе. Неисправности на входах и выходах элементов называются однотипными константными типа p, если эта константа одна и та же для каждого неисправного элемента и равна p, и произвольными константными, если эта константа может быть равна как 0, так и 1 для каждого неисправного элемента независимо от неисправностей других элементов. Для удобства над буквой D после символов, обозначающих базис, через точку с запятой будем ставить символы «$0, 1$» или «p» в случаях, когда в схемах допускаются соответственно произвольные константные неисправности или однотипные константные неисправности типа p, $p \in \{0, 1\}$, на входах/выходах элементов, а под буквой D после символов, обозначающих вид функции, — символы «(IO)» или «(I)» в случаях, когда в схемах допускаются неисправности соответственно на входах и выходах элементов или только на входах элементов. Вполне разумно предполагать, что если в базисе содержится булева константа α, то у элемента, её реализующего, не может быть неисправности типа α на его выходе.

В [4, с. 116] для базиса Жегалкина $B_1 = \{\&, \oplus, 1, 0\}$ показано, что $D^{B_1;\,0,1}_{\text{ЕП (IO)}}(n) \leqslant n + 3$; при этом используется метод построения схем из работы [6]. К. К. Салуджа и С. М. Редди в [7] получили оценку $D^{*\,B_1;\,0,1}_{k\text{-П (IO)}}(n) \leqslant 4 + \sum\limits_{i=1}^{\lfloor \log_2 2k \rfloor} C^i_n$; наличие звёздочки над буквой D обусловлено тем, что в указанной работе рассматривались схемы, содержащие, помимо входных переменных $x_1, \ldots, x_n$, дополнительную вход-

ную переменную h_0, вместо которой при реализации функций подавалась булева константа, но которая принимала значения как 0, так и 1 на наборах из теста. Д. С. Романовым и Е. Ю. Романовой в [8] для базисов $B_1' = \{\&, \oplus, 1\}$, $B_1'' = \{\&, \oplus, \sim\}$ установлены неравенства $D^{B_1';\,0,1}_{\text{ЕП (IO)}}(n) \leqslant 16$ и $D^{B_1'';\,0,1}_{\text{ЕП (IO)}}(n) \leqslant 16$; в частности, улучшен упомянутый результат из [4] (любая схема в базисе B_1' является также схемой в базисе B_1). Н. П. Редькин в [9–11] для базиса $B_2 = \{\&, \vee, \neg\}$ получил оценки $D^{B_2;\,p}_{\text{ПП (I)}}(n) \lesssim 4\left(2^{\lfloor \frac{n}{2} \rfloor} + 2^{\lceil \frac{n}{2} \rceil - 1}\right)$, $D^{B_2;\,p}_{\text{ЕД (I)}}(n) \lesssim 4\left(2^{\lfloor \frac{n}{2} \rfloor} + 2^{\lceil \frac{n}{2} \rceil - 1}\right)$ и $D^{B_2;\,0,1}_{\text{ПП (I)}}(n) \lesssim \frac{2^n}{\sqrt{\log_2 n}}$ соответственно, где $p = 0$ или 1.

В данной работе будут рассматриваться k-проверяющие и k-диагностические тесты при $k \in \mathbb{N}$, а в качестве неисправностей функциональных элементов — однотипные константные неисправности типа p, $p \in \{0, 1\}$, на входах и выходах элементов, а также только на входах элементов. Будут определены базисы B_3 и B_4, состоящие из двух булевых функций от не более чем $\max(k + 1; 3)$ переменных и от не более чем $2{,}5k + 2$ переменных соответственно, для которых в случае $p = 0$, в частности, будут доказаны равенства $D^{B_3;\,0}_{k\text{-П (IO)}}(n) = D^{B_4;\,0}_{k\text{-Д (IO)}}(n) = 2$ и $D^{B_3;\,0}_{k\text{-П (I)}}(n) = D^{B_4;\,0}_{k\text{-Д (I)}}(n) = 1$ (следствия 1–4). Также будет рассмотрен случай $p = 1$.

Введём обозначения $\tilde{0}^r = \underbrace{0, \ldots, 0}_{r}$, $\tilde{1}^r = \underbrace{1, \ldots, 1}_{r}$, где $r \in \mathbb{N} \cup \{0\}$.

Будем говорить, что функциональный элемент E' расположен в схеме S *ниже* функционального элемента E, если в этой схеме существует ориентированный путь от E к E'.

2. Проверяющие тесты

Пусть $p = 0$, т. е. в качестве неисправностей функциональных элементов допускаются только однотипные константные неисправности типа 0 на входах и, возможно, на выходах элементов. Два двоичных набора будем называть *k-соседними*, если они различаются не более, чем в k компонентах. Пусть $\omega(\tilde{x}^t)$, где $t \geqslant 2$, — произвольная булева функция, принимающая значение 1 на наборе $(\tilde{1}^t)$ и значение 0 на всех наборах, k-соседних с набором $(\tilde{1}^t)$, кроме него самого.

Лемма 1. *Любая схема S, состоящая только из входных переменных $x_1, \ldots, x_n$ и функциональных элементов, реализующих функцию вида $\omega(\tilde{x}^t)$, выход каждого из которых, кроме выходного, соединён ров-*

но с одним входом ровно одного элемента, k-неизбыточна, и множество $\{(\tilde{1}^n)\}$ является для неё k-проверяющим тестом относительно неисправностей на входах и выходах элементов.

Доказательство. На наборе $(\tilde{1}^n)$ на всех t входах и на выходе каждого элемента схемы S возникнет значение 1, поскольку $\omega(\tilde{1}^t) = 1$. Предположим, что среди всех входов и выходов элементов схемы S есть не менее одного и не более k неисправных. Из всех элементов этой схемы, у которых хотя бы один вход и/или выход неисправны, выберем произвольный «нижний» элемент E, ниже которого в схеме S не существует элемента с указанным свойством (это можно сделать, так как схема S конечна и не содержит ориентированных циклов).

Докажем, что значение на выходе элемента E на наборе $(\tilde{1}^n)$ в схеме S равно 0. Если неисправен выход этого элемента, то утверждение очевидно. Если же неисправен хотя бы один из входов элемента E, а его выход исправен, то на наборе $(\tilde{1}^n)$ значения не менее чем на одном и не более чем на k входах этого элемента в схеме S отличны от «правильных», т. е. от единиц, поскольку всего в этой схеме неисправны не более k входов/выходов элементов, а выходы элементов в ней не ветвятся. Тогда в силу определения функции ω значение на выходе элемента E на наборе $(\tilde{1}^n)$ в схеме S равно 0, что и требовалось доказать.

Далее изменение значения на выходе элемента E на наборе $(\tilde{1}^n)$ в схеме S с «правильного» значения 1 на 0 пройдёт по цепочке до выхода схемы S (здесь снова используются тот факт, что всего в этой схеме неисправны не более k входов/выходов элементов, а выходы элементов в ней не ветвятся, и определение функции ω). Таким образом, неисправность схемы S будет обнаружена на наборе $(\tilde{1}^n)$. Отсюда следует, что данная схема k-неизбыточна и множество $\{(\tilde{1}^n)\}$ является для неё k-проверяющим тестом. Лемма 1 доказана. □

Положим для удобства $m = \max(k+1; 3)$, $\tilde{x}^m = (x_1, \ldots, x_m)$ и $\varphi(\tilde{x}^m) = x_1 \ldots x_m \vee \overline{x}_1 \ldots \overline{x}_m$. Рассмотрим базис $B_3 = \{\varphi(\tilde{x}^m), \overline{x}\}$. Любой функциональный элемент, реализующий функцию вида $\varphi(\tilde{x}^m)$ (вида $\overline{x}$), будем называть *φ-элементом* (соответственно, *инвертором*).

Лемма 2. *Не существует схем в базисе B_3, реализующих константу 0 и k-неизбыточных относительно неисправностей на входах и выходах элементов.*

Доказательство. Выход любой схемы в базисе B_3, реализующей константу 0, очевидно, не может совпадать ни с одним из её входов, по-

этому он является выходом некоторого функционального элемента. Тогда при неисправности на выходе этого элемента получающаяся схема по-прежнему будет реализовывать константу 0, т. е. исходная схема k-избыточна. Лемма 2 доказана. □

Выделим возможное представление функции $f(\tilde{x}^n)$:

$$f(\tilde{x}^n) = x_i, \tag{1}$$

где $i \in \{1, \ldots, n\}$.

Лемма 3. *Любую булеву функцию $f(\tilde{x}^n)$ вида* (1) *можно реализовать k-неизбыточной схемой в базисе B_3, допускающей k-проверяющий тест длины* 0 *относительно неисправностей на входах и выходах элементов.*

Доказательство. Функцию f, очевидно, можно реализовать схемой, не содержащей функциональных элементов. У такой схемы нет ни одной функции неисправности, поэтому пустое множество является для неё k-проверяющим тестом. Лемма 3 доказана. □

Лемма 4. *Любую булеву функцию $f(\tilde{x}^n)$, не представимую в виде* (1), *можно реализовать k-неизбыточной схемой в базисе B_3, допускающей k-проверяющий тест из одного набора $(\tilde{1}^n)$ относительно неисправностей на входах и выходах элементов, при которых — в случае $f(\tilde{1}^n) = 0$ — выход выходного элемента схемы исправен.*

Доказательство. Пусть $A = [\{\varphi(\tilde{x}^m)\}]$ — замыкание множества $\{\varphi(\tilde{x}^m)\}$, тогда $A \subseteq T_1$, поскольку $\varphi(\tilde{1}^m) = 1$ (определения замыкания и замкнутого класса T_1 можно найти, например, в [12] на с. 14 и 17 соответственно). Из определения функции φ нетрудно получить, что $\varphi(\underbrace{x, \ldots, x}_{m}) \equiv 1$, $\varphi(\underbrace{x, \ldots, x}_{m-1}, y) = x \sim y$, поэтому $1 \in A$ и $x \sim y \in A$. Далее, $xy = \varphi(x, y, \underbrace{1, \ldots, 1}_{m-2}) \in A$ (поскольку $m \geqslant 3$) и $\overline{x} \vee y = xy \sim x \in A$. Таким образом, $\{\overline{x} \vee y, xy\} \subseteq A$, следовательно, $T_1 = [\{\overline{x} \vee y, xy\}] \subseteq A$ (равенство $T_1 = [\{\overline{x} \vee y, xy\}]$ установлено, например, в [12, с. 37]). Отсюда и из соотношения $A \subseteq T_1$ получаем, что $A = T_1$, т. е. любую булеву функцию $h(\tilde{x}^n)$ из класса T_1 можно выразить формулой ϕ_h над множеством $\{\varphi(\tilde{x}^m)\}$. Тогда существует схема S_h в базисе B_3, состоящая только из входных переменных $x_1, \ldots, x_n$ и φ-элементов, выход каждого из которых, кроме выходного, соединён ровно с одним входом ровно одного элемента, моделирующая формулу ϕ_h.

Отметим, что функция $\varphi(\tilde{x}^m)$ принимает значение 1 только на наборах $(\tilde{0}^m)$ и $(\tilde{1}^m)$. Отсюда и из неравенства $k < m$ вытекает, что она принимает значение 0 на всех наборах, k-соседних с набором $(\tilde{1}^m)$, кроме него самого. Поэтому к схеме S_h можно применить лемму 1, из которой следует, что данная схема k-неизбыточна и множество $\{(\tilde{1}^n)\}$ является для неё k-проверяющим тестом. Рассмотрим два случая.

1. Пусть $f(\tilde{1}^n) = 1$. Тогда $f(\tilde{x}^n) \in T_1$, схема S_f реализует функцию f, k-неизбыточна и множество $\{(\tilde{1}^n)\}$ является для неё k-проверяющим тестом длины 1.

2. Пусть $f(\tilde{1}^n) = 0$. Тогда $\overline{f}(\tilde{x}^n) \in T_1$, схема $S_{\overline{f}}$ реализует функцию $\overline{f}$, k-неизбыточна и множество $\{(\tilde{1}^n)\}$ является для неё k-проверяющим тестом. Выход схемы $S_{\overline{f}}$ соединим со входом инвертора I, выход которого объявим выходом полученной схемы (обозначим её S). Очевидно, что схема S реализует функцию f, а множество $\{(\tilde{1}^n)\}$ позволяет обнаружить любую неисправность этой схемы, при которой вход и выход инвертора I исправны. Если вход этого инвертора неисправен, а выход исправен, то на выходе схемы S реализуется константа 1, которую можно отличить от функции f на наборе $(\tilde{1}^n)$. Поэтому схема S является k-неизбыточной относительно неисправностей на входах и выходах элементов, при которых выход её выходного элемента исправен, и множество $\{(\tilde{1}^n)\}$ является для неё k-проверяющим тестом длины 1 относительно неисправностей указанного вида. Лемма 4 доказана. □

Лемма 5. *Для любой k-неизбыточной схемы в базисе B_3, реализующей булеву функцию $f(\tilde{x}^n)$, не представимую в виде* (1)*, любой k-проверяющий тест относительно неисправностей на входах элементов содержит хотя бы один набор.*

Доказательство. Выход любой k-неизбыточной схемы, реализующей функцию f, не может совпадать ни с одним из её входов, поэтому он является выходом некоторого функционального элемента. Тогда при неисправности любого входа этого элемента получающаяся схема будет реализовывать нетривиальную функцию неисправности, которую надо отличить от функции f хотя бы на одном наборе. Лемма 5 доказана. □

Теорема 1. *Для любой булевой функции $f(\tilde{x}^n)$, отличной от константы 0, справедливо равенство*

$$D^{B_3;\,0}_{k\text{-П (IO)}}(f) = \begin{cases} 0, & \text{если } f \text{ представима в виде (1)}, \\ 1, & \text{если } f \text{ не представима в виде (1) и } f(\tilde{1}^n) = 1, \\ 2, & \text{если } f(\tilde{1}^n) = 0. \end{cases}$$

Если же $f \equiv 0$, то значение $D^{B_3;\,0}_{k\text{-П (IO)}}(f)$ не определено.

Следствие 1. *Для любого $n \geqslant 1$ справедливо равенство $D^{B_3;\,0}_{k\text{-П (IO)}}(n) = 2$.*

Доказательство теоремы 1. Вместо $D^{B_3;\,0}_{k\text{-П (IO)}}(f)$ для краткости будем писать $D(f)$. В случае $f \equiv 0$ значение $D(f)$ не определено в силу леммы 2. Равенство $D(f) = 0$, если функция f представима в виде (1), следует из леммы 3. Неравенство $D(f) \leqslant 1$ в случае, когда функция f не представима в виде (1) и $f(\tilde{1}^n) = 1$, следует из леммы 4.

Пусть $f \not\equiv 0$ и $f(\tilde{1}^n) = 0$. В силу леммы 4 функцию f можно реализовать k-неизбыточной схемой в базисе B_3, допускающей k-проверяющий тест из одного набора $(\tilde{1}^n)$ относительно неисправностей на входах и выходах элементов, при которых выход выходного элемента схемы исправен. Добавим к этому набору любой двоичный набор $\tilde{\sigma}$ длины n, на котором функция f принимает значение 1. Тогда любую неисправность, при которой неисправен выход выходного элемента указанной схемы, можно обнаружить на наборе $\tilde{\sigma}$. Поэтому данная схема k-неизбыточна относительно неисправностей на входах и выходах элементов и множество $\{(\tilde{1}^n), \tilde{\sigma})\}$ является для неё k-проверяющим тестом длины 2, откуда следует, что $D(f) \leqslant 2$.

Неравенство $D(f) \geqslant 1$ в случае, когда функция f отлична от константы 0 и не представима в виде (1), вытекает из леммы 5 (неисправности на входах элементов являются частным случаем неисправностей на входах и выходах элементов).

Докажем неравенство $D(f) \geqslant 2$ в случае $f \not\equiv 0$ и $f(\tilde{1}^n) = 0$. Предположим, что оно неверно, т. е. $D(f) \leqslant 1$. Выше было показано, что $D(f) \geqslant 1$, поэтому $D(f) = 1$. Значит, функцию $f(\tilde{x}^n)$ можно реализовать k-неизбыточной схемой S, допускающей k-проверяющий тест из какого-то одного набора $\tilde{\pi}$. Пусть данная функция существенно зависит от s переменных; из соотношений $f \not\equiv 0$ и $f(\tilde{1}^n) = 0$ следует, что $1 \leqslant s \leqslant n$. Без ограничения общности это переменные $x_1, \ldots, x_s$. Очевидно, что каждая переменная x_i, $i = 1, \ldots, s$, обязана подаваться хотя бы на один вход

хотя бы одного элемента схемы S. Тогда при неисправности этого входа получающаяся функция неисправности g_i данной схемы не совпадает с f и её надо отличить от функции f хотя бы на одном наборе, причём функция g_i, очевидно, может отличаться от функции f только на тех наборах, i-я (слева) компонента которых равна 1. Отсюда получаем, что первые s компонент набора $\tilde{\pi}$ равны единице. Функция $f(\tilde{x}^n)$ не зависит существенно от переменных $x_{s+1}, \ldots, x_n$ (при $s < n$), поэтому $f(\tilde{\pi}) = f(\tilde{1}^n) = 0$. Но тогда неисправность на выходе выходного элемента схемы S нельзя обнаружить на наборе $\tilde{\pi}$, т. е. множество $\{\tilde{\pi}\}$ не может являться k-проверяющим тестом для схемы S. Полученное противоречие означает, что $D(f) \geqslant 2$. Теорема 1 доказана. □

Теорема 2. *Для любой булевой функции $f(\tilde{x}^n)$ справедливо равенство*

$$D^{B_3;\,0}_{k\text{-П (I)}}(f) = \begin{cases} 0, & \text{если } f \text{ представима в виде (1)}, \\ 1, & \text{если } f \text{ не представима в виде (1)}. \end{cases}$$

Следствие 2. *Для любого $n \geqslant 0$ справедливо равенство $D^{B_3;\,0}_{k\text{-П (I)}}(n) = 1$.*

Доказательство теоремы 2. Вместо $D^{B_3;\,0}_{k\text{-П (I)}}(f)$ для краткости будем писать $D(f)$. Равенство $D(f) = 0$ в случае, когда функция f представима в виде (1), следует из леммы 3. Если же функция f не представима в виде (1), то неравенство $D(f) \leqslant 1$ следует из леммы 4, а неравенство $D(f) \geqslant 1$ — из леммы 5. Теорема 2 доказана. □

Используя теоремы 1–2, следствия 1–2 и принцип двойственности (см., например, [12, с. 19, утверждение 3]), а именно, рассматривая схемы, получающиеся заменой всех элементов в схемах из доказательства теорем 1–2 на двойственные, нетрудно получить двойственные им результаты для случая $p = 1$ и базиса $B_3^* = \{\overline{x_1 \ldots x_m \vee \overline{x}_1 \ldots \overline{x}_m}, \overline{x}\}$. В частности, при $n \geqslant 1$ справедливо равенство $D^{B_3^*;\,1}_{k\text{-П (IO)}}(n) = 2$, а при $n \geqslant 0$ — равенство $D^{B_3^*;\,1}_{k\text{-П (I)}}(n) = 1$.

3. Диагностические тесты

Рассмотрим случай $p = 0$. Положим для удобства $q = 2k + \left\lfloor \frac{k}{2} \right\rfloor + 2$ и $\tilde{x}^q = (x_1, \ldots, x_q)$. Отметим, что $q \leqslant 2{,}5k + 2$, а в силу соотношения $\left\lfloor \frac{k}{2} \right\rfloor + 1 = \left\lceil \frac{k+1}{2} \right\rceil$ имеет место равенство $q = 2k + \left\lceil \frac{k+1}{2} \right\rceil + 1$. Пусть $\psi(\tilde{x}^q)$ — булева функция, удовлетворяющая следующим условиям:

(i) $(\tilde{1}^q) = 1$;

(ii) на всех наборах, k-соседних с набором $(\tilde{1}^q)$, кроме него самого, функция ψ принимает значение 0;

(iii) на всех наборах, k-соседних с набором $\tilde{\sigma}_1 = (\tilde{0}^{k+1}, \tilde{1}^{k+\lceil \frac{k+1}{2} \rceil})$, функция ψ принимает значение 0;

(iv) на всех наборах, k-соседних с набором $\tilde{\sigma}_2 = (\tilde{1}^{\lceil \frac{k+1}{2} \rceil}, \tilde{0}^{2k+1})$, функция ψ принимает значение 1.

На всех остальных двоичных наборах длины q функция ψ может принимать произвольные значения.

Покажем, что данная функция определена корректно, т. е. множества наборов, на которых она принимает значения 0 и 1, не пересекаются. Заметим, что набор $\tilde{\sigma}_2$ отличается от набора $(\tilde{1}^q)$ в $2k+1$ компоненте, а от набора $\tilde{\sigma}_1$ — в

$$\left(\left\lceil \frac{k+1}{2} \right\rceil\right) + \left(k + \left\lceil \frac{k+1}{2} \right\rceil\right) = k + 2\left\lceil \frac{k+1}{2} \right\rceil \geqslant 2k+1$$

компоненте, поэтому любой набор, k-соседний с набором $\tilde{\sigma}_2$, отличается от каждого из наборов $(\tilde{1}^q)$, $\tilde{\sigma}_1$ по крайней мере в $k+1$ компоненте, т. е. не может быть k-соседним ни с одним из этих наборов. Далее, набор $(\tilde{1}^q)$ отличается от набора $\tilde{\sigma}_1$ в $k+1$ компоненте, поэтому не является k-соседним с этим набором. Тем самым показано, что множества наборов, на которых функция $\psi(\tilde{x}^q)$ принимает значения 0 и 1, не пересекаются, значит, она определена корректно.

Рассмотрим базис $B_4 = \{\psi, \overline{\psi}\}$. Любой функциональный элемент, реализующий функцию вида $\psi(\tilde{x}^q)$ (вида $\overline{\psi}(\tilde{x}^q)$), будем называть *$\psi$-элементом* (соответственно, *$\overline{\psi}$-элементом*).

По аналогии с леммами соответственно 2, 3 и 5 доказываются следующие утверждения.

Лемма 6. *Не существует схем в базисе B_4, реализующих константу 0 и k-неизбыточных относительно неисправностей на входах и выходах элементов.*

Лемма 7. *Любую булеву функцию $f(\tilde{x}^n)$ вида (1) можно реализовать k-неизбыточной схемой в базисе B_4, допускающей k-диагностический тест длины 0 относительно неисправностей на входах и выходах элементов.*

Лемма 8. *Для любой k-неизбыточной схемы в базисе B_4, реализующей булеву функцию $f(\tilde{x}^n)$, не представимую в виде* (1)*, любой k-диагностический тест относительно неисправностей на входах элементов содержит хотя бы один набор.*

Выделим ещё одно возможное представление функции $f(\tilde{x}^n)$:

$$f(\tilde{x}^n) = x_{i_1} \& \ldots \& x_{i_s}, \tag{2}$$

где $s \in \{1, \ldots, n\}$ и $i_1, \ldots, i_s$ — попарно различные индексы от 1 до n.

Отметим, что представление (1) является частным случаем представления (2).

Лемма 9. *Любую булеву функцию $f(\tilde{x}^n)$, не представимую в виде* (1)*, можно реализовать k-неизбыточной схемой S в базисе B_3, допускающей k-диагностический тест длины 1 относительно неисправностей на входах и выходах элементов, при которых — в случае, когда функция f не представима в виде* (2) *— выход выходного элемента схемы исправен; при этом в указанном случае единственной функцией неисправности схемы S является функция $f \oplus x_1 \ldots x_n$.*

Доказательство. Пусть $A' = [\{\psi(\tilde{x}^q)\}]$ — замыкание множества $\{\psi(\tilde{x}^q)\}$, тогда $A' \subseteq T_1$ в силу условия (i). Из условий (i)–(iv) нетрудно получить, что $\psi(\underbrace{x, \ldots, x}_{q}) \equiv 1$, $\psi(\tilde{1}^{q-2}, x, y) = xy$ и $\psi(\underbrace{y, \ldots, y}_{k+1}, \underbrace{x, \ldots, x}_{k+\lceil \frac{k+1}{2} \rceil}) = \overline{x} \vee y$, поэтому $\{\overline{x} \vee y, xy\} \subseteq A'$ и $T_1 = [\{\overline{x} \vee y, xy\}] \subseteq A'$. Отсюда и из соотношения $A' \subseteq T_1$ получаем, что $A' = T_1$, т. е. любую булеву функцию $h(\tilde{x}^n)$ из класса T_1 можно выразить формулой ϕ'_h над множеством $\{\psi(\tilde{x}^q)\}$. Тогда существует схема S'_h в базисе B_4, состоящая только из входных переменных $x_1, \ldots, x_n$ и ψ-элементов, выход каждого из которых, кроме выходного, соединён ровно с одним входом ровно одного элемента, моделирующая формулу ϕ'_h. С учётом условий (i), (ii) получаем, что к схеме S'_h можно применить лемму 1, из которой следует, что данная схема k-неизбыточна и множество $\{(\tilde{1}^n)\}$ является для неё k-проверяющим тестом.

Введём для удобства обозначение $a = \overline{f}(\tilde{1}^n)$. Построим схему S в базисе B_4, реализующую функцию $f(\tilde{x}^n)$ (см. рисунок). Схема S состоит из q подсхем S_1–S_q и элемента E, являющегося ψ-элементом при $a = 0$ и $\overline{\psi}$-элементом при $a = 1$, входы которого соединяются с выходами этих подсхем (1-й вход — с выходом подсхемы S_1, …, q-й вход — с выходом

подсхемы S_q). Выходом схемы S является выход элемента E. Каждая из подсхем $S_1, \ldots, S_{\lceil \frac{k+1}{2} \rceil}$ представляет собой копию схемы $S'_{f \oplus a}$, каждая из подсхем $S_{\lceil \frac{k+1}{2} \rceil + 1}, \ldots, S_{k+1}$ — копию схемы $S'_{h_\&}$, где $h_\&(\tilde{x}^n) = x_1 \& \ldots \& x_n$, а каждая из подсхем $S_{k+2}, \ldots, S_q$ — копию схемы $S'_{f \oplus a \oplus h_\& \oplus 1}$ (отметим, что каждая из булевых функций $f \oplus a$, $h_\&$ и $f \oplus a \oplus h_\& \oplus 1$ принадлежит классу T_1: например, $(f \oplus a)(\tilde{1}^n) = f(\tilde{1}^n) \oplus a = \overline{a} \oplus a = 1$).

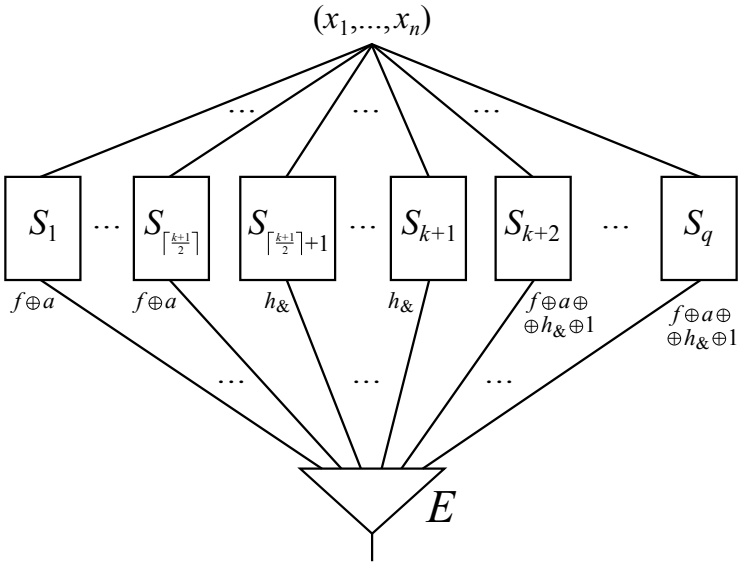


Схема S

Докажем, что построенная схема S в случае отсутствия в ней неисправностей реализует функцию $f(\tilde{x}^n)$. Прежде всего заметим, что элемент E по определению реализует функцию вида $\psi \oplus a$ от своих входов. На любом двоичном наборе $\tilde{\tau}_a$ длины n, на котором функция f принимает значение a, на выходах подсхем S_1–$S_{\lceil \frac{k+1}{2} \rceil}$, $S_{\lceil \frac{k+1}{2} \rceil + 1}$–$S_{k+1}$, S_{k+2}–S_q возникнут значения 0, 0, 1 соответственно (здесь используется тот факт, что $\tilde{\tau}_a \neq (\tilde{1}^n)$, поскольку $f(\tilde{\tau}_a) = a = \overline{f}(\tilde{1}^n)$). Тогда на входы элемента E будет подан в точности набор $\tilde{\sigma}_1$, а на его выходе, т. е. на выходе схемы S, возникнет значение $\psi(\tilde{\sigma}_1) \oplus a = a = f(\tilde{\tau}_a)$ в силу условия (iii). Далее, на любом двоичном наборе $\tilde{\tau}_{\overline{a}}$ длины n, на котором функция f принимает значение $\overline{a}$, отличном от набора $(\tilde{1}^n)$, на выходах подсхем S_1–$S_{\lceil \frac{k+1}{2} \rceil}$, $S_{\lceil \frac{k+1}{2} \rceil + 1}$–$S_{k+1}$, S_{k+2}–S_q возникнут значения 1, 0, 0 соответственно. Тогда на входы элемента E будет подан в точности набор $\tilde{\sigma}_2$, а на его выходе,

т. е. на выходе схемы S, возникнет значение $\psi(\tilde{\sigma}_2) \oplus a = \overline{a} = f(\tilde{\tau}_{\overline{a}})$ в силу условия (iv). Наконец, на наборе $(\tilde{1}^n)$ на выходах подсхем S_1–$S_{\lceil \frac{k+1}{2} \rceil}$, $S_{\lceil \frac{k+1}{2} \rceil + 1}$–$S_{k+1}$, S_{k+2}–S_q возникнут значения 1, 1, 1 соответственно. Тогда на входы элемента E будет подан набор $(\tilde{1}^q)$, а на его выходе, т. е. на выходе схемы S, возникнет значение $\psi(\tilde{1}^q) \oplus a = \overline{a} = f(\tilde{1}^n)$ в силу условия (i). Таким образом, на выходе схемы S реализуется в точности функция $f(\tilde{x}^n)$.

Найдём все возможные функции неисправности схемы S. Пусть выход элемента E исправен. При произвольной неисправности не менее одного и не более k входов/выходов элементов подсхем S_1–S_q и/или входов элемента E на любом входном наборе схемы S могут измениться значения не более чем на k входах элемента E. Поэтому на любом наборе $\tilde{\tau}_a$, на котором функция f принимает значение a, на входы элемента E поступит набор, k-соседний с набором $\tilde{\sigma}_1$, а на его выходе, т. е. на выходе схемы S, возникнет значение $0 \oplus a = a = f(\tilde{\tau}_a)$ в силу условия (iii). Аналогично на любом наборе $\tilde{\tau}_{\overline{a}}$, на котором функция f принимает значение $\overline{a}$, отличном от набора $(\tilde{1}^n)$, на входы элемента E поступит набор, k-соседний с набором $\tilde{\sigma}_2$, а на его выходе, т. е. на выходе схемы S, возникнет значение $1 \oplus a = \overline{a} = f(\tilde{\tau}_{\overline{a}})$ в силу условия (iv). Наконец, на наборе $(\tilde{1}^n)$ на входы элемента E поступит набор, k-соседний с набором $(\tilde{1}^q)$. При этом хотя бы одна компонента указанного набора будет равна 0, поскольку множество $\{(\tilde{1}^n)\}$ является k-проверяющим тестом для каждой из подсхем S_1–S_q, а в случае исправности всех элементов каждая из этих подсхем на наборе $(\tilde{1}^n)$ выдаёт единицу. Следовательно, на выходе элемента E, т. е. на выходе схемы S, возникнет значение $0 \oplus a = a = \overline{f}(\tilde{1}^n)$ в силу условия (ii). Таким образом, на выходе схемы S возникнет функция неисправности g_1, отличающаяся от функции f только на наборе $(\tilde{1}^n)$ (её можно записать в виде $g_1 = f \oplus x_1 \dots x_n$).

Если функция f не представима в виде (2), то по условию леммы выход элемента E исправен и, тем самым, g_1 — единственная функция неисправности схемы S. Данную функцию можно отличить от функции f на наборе $(\tilde{1}^n)$, поэтому схема S является k-неизбыточной относительно неисправностей на входах и выходах элементов, при которых выход её выходного элемента исправен, и допускает k-диагностический тест из одного набора, что и требовалось доказать.

Если функция f представима в виде (2) при $s = n$, а выход элемента E неисправен, то на выходе схемы S возникнет функция неисправно-

сти $g_2 \equiv 0$, однако

$$g_1 = f \oplus x_1 \dots x_n = x_1 \dots x_n \oplus x_1 \dots x_n \equiv g_2.$$

Значит, g_1 — единственная функция неисправности схемы S. Данную функцию можно отличить от функции f на наборе $(\tilde{1}^n)$, поэтому схема S является k-неизбыточной и допускает k-диагностический тест из одного набора, что и требовалось доказать. Тем самым установлено, что функцию $x_1 \dots x_n$ можно реализовать k-неизбыточной схемой в базисе B_4, допускающей k-диагностический тест длины 1 относительно неисправностей на входах и выходах элементов. В случае же, когда функция f имеет вид (2) и $s < n$, можно без ограничения общности считать, что $i_1 = 1$, $\dots$, $i_s = s$, и применить утверждение, сформулированное в предыдущем предложении, с заменой n на s (на входы элементов схемы в этом случае будут подаваться только переменные $x_1, \dots, x_s$), а затем добавить к единственному тестовому набору справа произвольные $n - s$ компонент, чтобы его длина стала равна n и множество, состоящее из одного этого набора, удовлетворяло определению диагностического теста. Лемма 9 доказана. □

Теорема 3. *Для любой булевой функции $f(\tilde{x}^n)$, отличной от константы 0, справедливо равенство*

$$D^{B_4;\,0}_{k\text{-Д (IO)}}(f) = \begin{cases} 0, & \text{если } f \text{ представима в виде (1)}, \\ 1, & \text{если } f \text{ представима в виде (2), но не в виде (1)}, \\ 2, & \text{если } f \text{ не представима в виде (2)}. \end{cases}$$

Если же $f \equiv 0$, то значение $D^{B_4;\,0}_{k\text{-Д (IO)}}(f)$ не определено.

Следствие 3. *Для любого $n \geqslant 0$ справедливо равенство $D^{B_4;\,0}_{k\text{-Д (IO)}}(n) = 2$.*

Доказательство теоремы 3. Вместо $D^{B_4;\,0}_{k\text{-Д (IO)}}(f)$ для краткости будем писать $D(f)$. В случае $f \equiv 0$ значение $D(f)$ не определено в силу леммы 6. Далее будем считать, что $f \not\equiv 0$. Равенство $D(f) = 0$, если функция f представима в виде (1), следует из леммы 7. Неравенство $D(f) \leqslant 1$ в случае, когда функция f представима в виде (2), но не в виде (1), следует из леммы 9.

Пусть функция f не представима в виде (2). В силу леммы 9 её можно реализовать схемой S в базисе B_4, единственной функцией неисправности которой при неисправностях на входах и выходах элементов, за исключением выхода выходного элемента, является функция

$g_1 = f \oplus x_1 \dots x_n$. Если же неисправен выход выходного элемента схемы S, то она станет реализовывать функцию $g_2 \equiv 0$. Пусть $\tilde{\sigma}$ — произвольный двоичный набор длины n, отличный от набора $(\tilde{1}^n)$, на котором функция f принимает значение 1; такой набор найдётся, поскольку $f \not\equiv 0$ и $f \not\equiv x_1 \dots x_n$. Тогда функцию f можно отличить от функции g_1 на наборе $(\tilde{1}^n)$, а функцию g_2 можно отличить от каждой из функций f, g_1 на наборе $\tilde{\sigma}$. Поэтому схема S является k-неизбыточной относительно неисправностей на входах и выходах элементов и допускает k-диагностический тест из наборов $(\tilde{1}^n)$ и $\tilde{\sigma}$, откуда следует, что $D(f) \leqslant 2$.

Неравенство $D(f) \geqslant 1$ в случае, когда функция f не представима в виде (1), вытекает из леммы 8.

Докажем неравенство $D(f) \geqslant 2$ в случае, когда функция f не имеет вид (2). Пусть S — произвольная k-неизбыточная схема, реализующая функцию f, и на входы её элементов подаются s переменных из числа $x_1, \dots, x_n$. Поскольку функция f не представима в виде (2), а значит, и в виде (1), то $s \geqslant 1$ и, кроме того, в схеме S содержится выходной элемент. Без ограничения общности на входы элементов схемы S подаются переменные $x_1, \dots, x_s$. При неисправности на выходе выходного элемента данной схемы возникнет функция неисправности $g_1 \equiv 0$.

Очевидно, что функция $f(\tilde{x}^n)$ может зависеть существенно только от переменных $x_1, \dots, x_s$. Если на всех двоичных наборах длины n, хотя бы одна из первых s (слева) компонент которых равна 0, функция f принимает значение 0, то $f \equiv 0$ или $f \equiv x_1 \& \dots \& x_s$ (т. е. имеет вид (2)), что невозможно по предположению. Поэтому на некотором наборе $\tilde{\pi}$ длины n, хотя бы одна из первых s компонент которого равна 0, функция f принимает значение 1. Пусть i-я компонента набора $\tilde{\pi}$ равна 0, где $i \in \{1, \dots, s\}$. Переменная x_i подаётся хотя бы на один вход хотя бы одного элемента схемы S. При неисправности этого входа получающаяся функция неисправности g_2 данной схемы в силу её k-неизбыточности не совпадает с f, однако указанную неисправность, очевидно, нельзя обнаружить на наборе $\tilde{\pi}$ (на этом наборе $x_i = 0$), поэтому $g_2(\tilde{\pi}) = f(\tilde{\pi}) = 1$. Таким образом, функции f, g_1 и g_2 попарно различны. Чтобы отличить эти функции друг от друга, необходимы по крайней мере два набора, откуда следует неравенство $D(f) \geqslant 2$. Теорема 3 доказана. □

Теорема 4. *Для любой булевой функции $f(\tilde{x}^n)$ справедливо равенство*

$$D^{B_4;\,0}_{k\text{-Д (I)}}(f) = \begin{cases} 0, & \text{если } f \text{ представима в виде (1)}, \\ 1, & \text{если } f \text{ не представима в виде (1)}. \end{cases}$$

Следствие 4. *Для любого* $n \geqslant 0$ *справедливо равенство* $D^{B_4;\,0}_{k\text{-Д (I)}}(n) = 1$.

Доказательство теоремы 4. Вместо $D^{B_4;\,0}_{k\text{-Д (I)}}(f)$ для краткости будем писать $D(f)$. Равенство $D(f) = 0$ в случае, когда функция f представима в виде (1), следует из леммы 7. Если же функция f не представима в виде (1), то неравенство $D(f) \geqslant 1$ следует из леммы 8, а неравенство $D(f) \leqslant 1$ — из леммы 9. Теорема 4 доказана. □

Используя теоремы 3–4, следствия 3–4 и принцип двойственности, нетрудно получить двойственные им результаты для случая $p = 1$ и базиса $B_4^* = \{\psi^*, \overline{\psi^*}\}$, где $\psi^*(\tilde{x}^q)$ — двойственная к $\psi(\tilde{x}^q)$ булева функция. В частности, при $n \geqslant 0$ справедливы равенства $D^{B_4^*;\,1}_{k\text{-Д (IO)}}(n) = 2$ и $D^{B_4^*;\,1}_{k\text{-Д (I)}}(n) = 1$.

Работа выполнена при поддержке Программы Президиума РАН № 01 «Фундаментальная математика и ее приложения» (грант PRAS-18-01).

Список литературы

[1] Чегис И. А., Яблонский С. В. Логические способы контроля работы электрических схем // Труды МИАН. — 1958. — **51**. — С. 270–360.

[2] Яблонский С. В. Надежность и контроль управляющих систем // Материалы Всесоюзного семинара по дискретной математике и её приложениям (Москва, 31 января–2 февраля 1984 г.) / Под ред. О. Б. Лупанова. — М.: Изд-во МГУ, 1986. — С. 7–12.

[3] Яблонский С. В. Некоторые вопросы надёжности и контроля управляющих систем // Матем. вопросы киберн. Вып. 1. — М.: Наука, 1988. — С. 5–25.

[4] Редькин Н. П. Надёжность и диагностика схем. — М.: Изд-во МГУ, 1992.

[5] Коляда С. С. Верхние оценки длины проверяющих тестов для схем из функциональных элементов. — Дисс. на соиск. уч. ст. к.ф.-м.н. — М., 2013. — 77 с.

[6] Reddy S. M. Easily testable realizations for logic functions // IEEE Trans. Comput. — 1972. — **C-21**, No. 11. — P. 1183–1188.

[7] Saluja K. K, Reddy S. M. Fault detecting test sets for Reed-Muller canonic networks // IEEE Trans. Comput. — 1975. — **C-24**, No. 10. — P. 995–998.

[8] Романов Д. С., Романова Е. Ю. Метод синтеза неизбыточных схем, допускающих единичные проверяющие тесты константной длины // Дискретн. матем. — 2017. — **29**, вып. 4. — С. 87–105.

[9] Редькин Н. П. О проверяющих тестах для схем при однотипных константных неисправностях на входах элементов // Изв. вузов. Матем. — 1988. — № 7. — С. 57–64.

[10] Редькин Н. П. О схемах, допускающих короткие единичные диагностические тесты // Дискретн. матем. — 1989. — **1**, вып. 3. — С. 71–76.

[11] Редькин Н. П. О проверяющих тестах для схем при константных неисправностях на входах элементов // Вестн. Моск. ун-та. Матем. Механ. — 1997. — № 1. — С. 12–18.

[12] Угольников А. Б. Классы Поста. Учебное пособие. — М.: Изд-во ЦПИ при мех.-мат. ф-те МГУ, 2008.

Synthesis of easily testable logic networks under one-type stuck-at faults at inputs and outputs of gates

Popkov K.A.

The following assertions are proved: for each natural k and each Boolean constant p, there exists a basis consisting of a Boolean function on $\max(k+1;\,3)$ variables and negation of one variable (there exists a basis consisting of a Boolean function on not more than $2{,}5k+2$ variables and negation of this function), in which one can implement any Boolean function except a Boolean constant p by a logic network which is irredundant and allows a fault detection test (a diagnostic test, respectively) with a length not exceeding 2 under not more than k stuck-at-p faults at inputs and outputs of gates. It is shown that, when considering only stuck-at-p faults at inputs of gates, one can reduce the mentioned bounds on lengths of tests to 1.

Keywords: logic network, one-type stuck-at fault, fault detection test, diagnostic test.

К сведению авторов публикаций в журнале «Интеллектуальные системы. Теория и приложения»

В соответствии с требованиями ВАК РФ к изданиям, входящим в перечень ведущих рецензируемых научных журналов и изданий, в которых могут быть опубликованы основные научные результаты диссертаций на соискание ученой степени доктора и кандидата наук, статьи в журнал «Интеллектуальные системы. Теория и приложения» предоставляются авторами в следующей форме:

1. Статьи, набранные в пакете LaTeX, предоставляются к загрузке через WEB-форму http://intsysjournal.org/generator_form .
2. К статье прилагаются файлы, содержащие название статьи на русском и английском языках, аннотацию на русском и английском языках (не более 50 слов), список ключевых слов на русском и английском языках (не более 20 слов), информация об авторах: Ф.И.О. полностью, место работы, должность, ученая степень и/или звание (если имеется), контактные телефоны (с кодом города и страны), e-mail, почтовый адрес с индексом города (домашний или служебный).
3. Список литературы оформляется в едином формате, установленном системой Российского индекса научного цитирования.
4. За публикацию статей в журнале «Интеллектуальные системы. Теория и приложения» с авторов (в том числе аспирантов высших учебных заведений) статей, рекомендованных к публикации, плата не взимается. Оттиски статей авторам не предоставляются. Журнал распространяется по подписке, экземпляры журнала рассылаются подписчикам наложенным платежом. Условия подписки публикуются в каталоге НТИ «Роспечать», индекс журнала 64559.
5. Доступ к электронной версии последнего вышедшего номера осуществляется через НЭБ «Российский индекс научного цитирования». Номера, вышедшие ранее, размещаются на сайте http://intsysjournal.org, и доступ к ним бесплатный. Там же будут размещены аннотации всех публикуемых статей.

Подписано в печать: 20.09.2018
Дата выхода: 28.09.2018
Тираж: 200 зкз.
Цена свободная
Свидетельство о регистрации СМИ: ПИ № ФС77–58444 от 25 июня 2014 г., выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций(Роскомнадзор).