УДК 316.62

СОЦИАЛЬНОЕ ПРОГРАММИРОВАНИЕ

В. И. Шалак
Институт философии РАН, Москва (Россия)

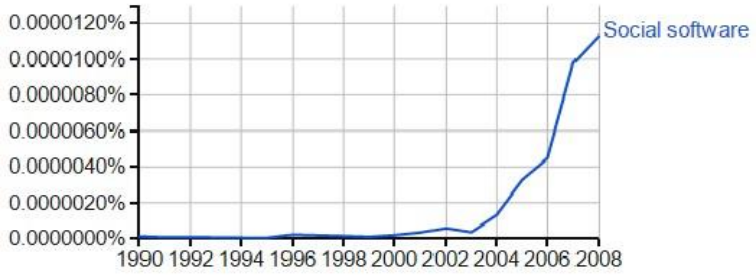
Социальное программирование – новая область междисциплинарных исследований на стыке информатики, теории игр, логики и социальных наук. Основная цель исследования – анализ и создание новых эффективных социальных процедур. Для этого требуется решение задач проверки корректности процедур, анализа и оптимизации информационного обмена, синтеза новых процедур, контроля за многосубъектным социальным взаимодействием.

Ключевые слова: социальное программирование, социальные процедуры, теория игр, логика, информатика

Задача настоящей статьи – привлечь внимание заинтересованных читателей к новому междисциплинарному направлению исследований, объединяющему достижения компьютерных и социальных наук [5]. Это направление возникло совсем недавно, можно даже назвать год, когда была высказана идея, получившая вскоре отклик со стороны многочисленных исследователей.

Автором идеи является Рохит Парик (Rohit Parikh), американский логик индийского происхождения. В 1995 году на Международном конгрессе по логике, методологии и философии науки он выступил с докладом «*Language as social software*» [9, 10], в котором провел аналогию между современными языками программирования и естественным языком как одним из инструментов социального взаимодействия и предложил использовать достижения теоретического программирования для решения определенных задач, стоящих перед обществом.

Идея привлекла внимание, начали появляться диссертации [13], а после того, как в 2002 году была опубликована долгое время ходившая в рукописи статья Парика «*Social software*» [11], произошел взрывной рост числа исследований, относящихся к данной теме. Стали публиковаться многочисленные статьи, монографии, защищаться диссертации [8, 13], проводиться конференции. На диаграмме представлены частоты упоминания данного термина в англоязычной литературе с 1990 по 2008 годы.



Термин «*Social Software*» дословно переводится как «*социальные программы*», но мы будем переводить его как «*социальное программирование*», поскольку речь идет не о конечных продуктах, а о теоретическом исследовании проблемы, разработке методов, которые в последующем должны вылиться в конкретные практические приложения.

Прежде чем уточнить смысл данного термина, попробуем ответить на вопрос о том, насколько вообще правомерна аналогия между алгоритмическим функционированием компьютеров и функционированием общества.

Возьмем отдельного человека. Одно из важных требований, предъявляемых к нему обществом, – предсказуемость, поведение в соответствии с определенными правилами. Если человек непредсказуем, его изолируют. Основаниями для предсказуемости являются естественные потребности человека, нормы морали, права, обязательства перед другими людьми, системы общепринятых ценностей и пр.

Требование предсказуемости предъявляется не только по отношению к людям, но и к государственным структурам. Например, согласно закону, обязанностью Центрального Банка является защита и обеспечение устойчивости рубля. Недавнее обесценивание рубля вызвало справедливые претензии к его деятельности, поскольку он ничего не сделал для стабилизации национальной валюты.

Подразумевается, что существует определенный набор правил, согласно которым должны функционировать те или иные социальные институты. Отступление от правил рассматривается как нарушение. Аналогия с неправильно выполняемой компьютерной программой достаточно очевидна.

Есть и другие основания для аналогии. Отдельные люди, коммерческие и государственные структуры постоянно ставят перед собой те или иные цели и планируют действия по их достижению. У всех на слуху принятые программы по обеспечению жильем, импортозамещению, реформированию армии и образования. После того как цель поставлена, назначаются главные исполнители, которые, в свою очередь, детализируют план действий и назначают других исполнителей. Когда истекает срок выполнения программы, проверяют, все ли сделано и достигнуты ли именно те результаты, которые были заявлены в самом начале. Если желаемое расходится с действительностью, появляется оправдание «*хотели, как лучше, а получилось, как всегда*». В терминах программирования это называется некорректностью программы.

Ровно 80 лет назад Алан Тьюринг предложил математическую модель вычислений, которая впоследствии получила название «*машины Тьюринга*». Считать

умели и до Тьюринга, но именно построение математической модели вычислений дало сильнейший толчок исследованиям в этом направлении, которые вылились в практические приложения, буквально преобразившие всю нашу жизнь.

Высказанная Париком *идея социального программирования* заключается в том, чтобы распространить достижения теории игр, логики и информатики на изучение и разработку новых социальных процедур, которые сделают функционирование общества более эффективным. Тремя составными частями предложенного направления исследований являются моделирование социальных ситуаций, проверка корректности социальных процедур и проектирование новых. Развитие и внедрение идей социального программирования можно сравнить с переходом от кустарного к промышленному производству, но на этот раз в вопросах социального управления.

Корректность программ

В 1969 году специалистом в области логики и computer science Чарльзом Хоаром была построена специальная логическая система, предназначенная для анализа корректности программ [6]. Ее базисные выражения имели вид $\{A\}\alpha\{B\}$, где A – описание *предусловий* выполнения программы α, а B – описание *постусловий*. Прочитать данное выражение можно следующим образом: «*если в состоянии A начать выполнять программу* α*, то по ее окончании будет совершен переход в состояние B*». С помощью дополнительных правил комбинации таких выражений можно описать выполнение любой программы.

Допустим, у нас есть конкретная компьютерная программа α, описание условий A, в которых мы хотим ее применять, и описание B результата, который ожидается. Например, в конце каждого месяца программа α должна для каждого работника предприятия (в зависимости от занимаемой должности, оклада, реально отработанного времени, различных льгот и отчислений) определить его заработную плату. Программист, написавший программу, живой человек, который может ошибаться. Вполне закономерен вопрос: действительно ли программа правильно начисляет заработную плату? Как это определить? Можно попробовать протестировать программу на большом количестве примеров, и если недостатков не будет выявлено, то считать, что программа работает правильно. К сожалению, такое тестирование не дает ответа на вопрос о корректности программы, а может лишь, если повезет, обнаружить ошибку на каком-то частном примере. А как быть в случае, если речь идет об управлении ядерным реактором или космической станцией, когда каждая ошибка уникальна и стоит слишком дорого? Именно для этого и понадобилось создать теорию корректности программ.

Часто уязвимость компьютерных программ проявляется, когда изменяемые входные параметры принимают экстремальные значения. Этим пользуются хакеры. Достаточно подать на вход программы специально подобранные значения, чтобы она начала выполняться неправильно и привела к ситуации, в которой возможна утечка информации или перехват управления. Но то же самое относится и к различным социальным и государственным структурам. Их функционирование происходит по определенным правилам, которые, в большинстве своем, ориентируются на устойчивое развитие и плохо учитывают ситуации нестабильности. На помощь

может прийти адаптированная теория корректности программ в применении к социальным процедурам, которая позволит проанализировать их, выявить слабые места и внести те или иные изменения. В противном случае в самый неподходящий момент они могут оказаться уязвимыми. Примеров техногенных и природных катастроф, когда все перестает работать и люди вдруг оказываются совершенно незащищенными, более чем достаточно. Единственное, что остается, – это уповать на помощь МЧС. Можно вспомнить недавние случаи с валютными заёмщиками, когда все стороны конфликта действовали в полном соответствии с имеющимися правилами, но в результате сложилась тупиковая ситуация, разрешить которую без директивного вмешательства государства оказалось невозможным. Все это иллюстрации некорректных социальных процедур.

Нет никакой гарантии, что такие ситуации не могут быть созданы искусственно.

Проблема перевода

Еще одна тонкость с корректностью компьютерных и социальных программ связана с различием между дескриптивной и операциональной семантикой языков. С помощью языка программирования высокого уровня можно написать сложную программу, удовлетворяющую всем критериям корректности. После этого она поступает на вход компилятора, который осуществляет перевод с языка высокого уровня на язык машинных команд, занятый лишь изменением содержания отдельных регистров памяти и пересылкой значений между ними. Если перевод будет выполнен некорректно, то последствия выполнения программы могут быть самыми неожиданными.

Чтобы перевести эту проблему на более понятный язык, представим задачу построить дом по прекрасному проекту. Одного этого еще мало, чтобы он появился. Должны быть специальные люди, мастера, прорабы, которые переведут этот план с языка описания на язык конкретных работ, которые должны быть выполнены в определенной последовательности и в определенное время. Может случиться так, что дом будет построен, но через некоторое время в нем станут проявляться недостатки, связанные с нарушением технологий. Эти нарушения никак не отражены в исходном проекте дома. Они возникли на уровне его операциональной реализации мастерами, прорабами и строительными рабочими. Иногда такие нарушения оправдывают тем, что «*не так поняли*», и это действительно может быть правдой, а не злым умыслом.

Анализ протоколов

Трудно найти компьютер, который работал бы в полной изоляции от других и не был подключен ни к какой сети. Практически в каждой организации компьютеры объединены в локальные сети, которые, в свою очередь, имеют выходы и связи с другими сетями. Глобальная сеть компьютеров называется Интернет.

Обмен информацией между компьютерами происходит по жестко заданным правилам, объединяемым в протоколы. При этом сами протоколы образуют целую иерархию, надстраиваясь одни над другими. К протоколам предъявляется ряд требований. Они должны обеспечивать передачу именно той информации, которая необходима. Они должны быть надежны, то есть гарантировать, что информация

дойдет до адресата. Если в ходе передачи информации допущена ошибка, протоколы должны предоставлять возможность исправить ее. В случае необходимости они должны обеспечивать защищенность передаваемой информации. Отдельные протоколы существуют для пересылки электронной почты, для электронных платежей, для просмотра страниц Интернет и т.д. Анализ и разработка протоколов с точки зрения требований, которым они должны удовлетворять, является чрезвычайно важной деятельностью. Созданы специальные теории анализа протоколов и программные средства, помогающие в этом.

Теперь с этой точки зрения посмотрим на общество. Оно тоже пронизано обменом пакетами информации между различными структурными элементами. Всякий информационный контакт между отправителем и получателем подчиняется определенным правилам. Иски в суд, заявки на гранты, налоговая отчетность, отчеты о проделанной работе, обращения в органы власти и многое другое требуют сбора и оформления определенных документов, которые передаются адресату. После этого адресат подтверждает получение документов. В случае, если каких-то документов не хватает, об этом сообщают. Через определенный промежуток времени приходит ответ по существу полученной информации. Это может потребовать сбора нового пакета документов и нового акта коммуникации. В зависимости от содержания информации способы ее передачи могут различаться.

В информатике одними из требований к протоколам являются оптимальность и полнота, позволяющие избегать передачи избыточной информации и в то же время обеспечивающие передачу всей необходимой информации. Такие же требования вполне естественно предъявить к документообороту в обществе. Это позволило бы значительно повысить эффективность функционирования социальных структур.

Синтез программ

В СССР экономика была плановой. Каждые пять лет составлялся план ее будущего развития, который затем принимался на Съезде и по частям спускался подчиненным организациям, которые также занимались планированием, но на более низком уровне. Сегодня государственная структура изменилась, но хаос не наступил, планирование как было, так и осталось. Оно затрагивает не только экономику, но и политику, культуру, демографию, военную и идеологическую сферы. Это же справедливо и для других стран.

В теоретическом и практическом программировании изучается задача автоматического синтеза программ. Возьмем базовое выражение логики Хоара $\{A\}\alpha\{B\}$. При анализе корректности программ известны все три его параметра – A, B и α. Нужно лишь удостовериться, что программа α действительно совершает переход от предусловий A к постусловиям B. При решении задачи синтеза известны лишь предусловия A и постусловия B, а программа α является неизвестной величиной, которую требуется найти. Задача кажется трудновыполнимой, но ее научились решать. Один из методов заключается в построении логического вывода B из A. Каждому элементарному шагу вывода соответствует заранее известное элементарное действие. Последовательным шагам вывода соответствует последовательное выполнение действий, в результате чего на выходе получается

программа, которая гарантированно является корректной. Это всего лишь общая схема, многие детали которой опущены. Есть и другие методы. Ничто не мешает попробовать перенести их из области компьютерных вычислений в социальную, чтобы на выходе получать практически реализуемые социальные программы.

Если под нужным углом присмотреться к поведению людей, то окажется, что мы всю жизнь синтезируем микросоциальные программы. Приведем простой житейский пример логико-семантического подхода к планированию действий, который может быть формализован в виде специального исчисления и реализован на компьютерах.

Представим себе, что ребенок, увидев выходящую из дома маму, просит ее *принести что-нибудь вкусненького*. Слово *принести* указывает на некоторую программу, а слово *вкусненького* указывает на постусловия ее выполнения. Предусловия пока что неизвестны. Средствами логики Хоара это можно записать следующим образом

{A(x)}*Принести*(x){*Вкусненькое*(x)}

Мама должна уточнить для себя, что значит *Вкусненькое*? В терминах логики это означает дать определение термину. Вкусненькое, по мнению мамы, – это продукт питания, который нравится ее ребенку.

Вкусненькое(x) =$_{def}$ *Продукт_питания*(x) и *Нравится_ребенку*(x).

В правой определяющей части появились два термина, которые требуют уточнения.

Продукт_питания(x) =$_{def}$ [*Находится_на_улице*(x) и *Съедобное*(x)]

или

Находится_в_продуктовом_магазине(x).

Нравится_ребенку(x) =$_{def}$ *Любит_есть*(x).

Поскольку дело происходит в городе, а не на даче, мама отбрасывает вариант [*Находится_на_улице*(x) и *Съедобное*(x)] и в результате получает:

Вкусненькое(x) =$_{def}$ *Находится_в_продуктовом_магазине*(x) и *Любит_есть*(x).

Теперь необходимо уточнить, что *Любит_есть* ребенок? По своему опыту мама знает, что ребенок любит *сосиски*, *йогурты* и *шоколад*:

Любит_есть(x) =$_{def}$ *С*(x) или *Й*(x) или *Ш*(x).

Задача принести что-нибудь вкусненькое сводится к простой задаче *посещения продуктового магазина* и покупке *сосисок, йогурта* или *шоколада*. В магазине есть все. В игру вступают предпочтения по цене и пользе для здоровья. Мама решает, что *йогурт* лучше, чем *шоколад*, а *шоколад* лучше, чем *сосиски*, и покупает *йогурт*. Исходная задача {A(x)}*Принести*(x){*Вкусненькое*(x)} трансформируется в программу

{*В_продуктовом_магазине*(x) и [*С*(x) или *Й*(x) или *Ш*(x)]} *Купить*(x) {*Вкусненькое*(x)}.

Это выражение можно прочитать следующим образом: «*Всякий раз, когда в продуктовом магазине есть сосиски, йогурт или шоколад, мама может купить что-нибудь одно и это будет вкусненьким для ее ребенка*».

Последовательный анализ целевого условия позволил свести исходную нечетко поставленную задачу к более простым и четким подзадачам, решение которых не

представляет труда. Кроме этого, были найдены предусловия задачи, ограничивающие применение программы определенной предметной областью. В будущем на случай повторной просьбы у мамы уже есть готовая программа для ее выполнения.

Несмотря на простоту примера, он содержит все основные элементы, которые могут быть использованы при решении самых разных задач: от планирования летнего отпуска до постройки железных дорог, планирования военных операций и составления образовательных программ. Использование вычислительной техники позволяет алгоритмизировать процесс планирования и сделать его доступным для массового использования.

Многосубъектность социальных взаимодействий

Основные достоинства социального программирования как нового направления исследований раскрываются, когда мы обращаем внимание на многосубъектность социальных взаимодействий. Это тоже имеет аналог в компьютерных системах в виде параллельных и распределенных вычислительных систем. В случае параллельных вычислений задача разбивается на несколько подзадач, которые решаются отдельными процессорами и должны быть определенным образом синхронизированы. Это достигается не только общими часами, но и обменом информацией между процессами. Для успешного решения задач они должны «знать», что делают другие процессы, каково состояние их вычислений. В несколько иной форме это справедливо и для распределенных вычислений, реализуемых компьютерами сети Интернет, которые также обмениваются информацией.

В то же время приведенная аналогия не является полной. Если сравнить функционирование компьютера и поведение отдельного человека, обнаружится много различий. Каждый компьютер работает по жестко заданной программе. Если известны входные параметры и не случится никакого сбоя, результат его функционирования предсказуем. На поведение людей влияют не только входные условия, но и множество других факторов, которые могут приводить к плохо предсказуемым результатам.

Для объяснения экономического поведения людей была создана теория игр. Она получила математическое оформление в 1944 году благодаря работам Джона фон Неймана и Оскара Моргенштерна. Затем теорию игр распространили на другие виды поведения. Однако оказалось, что реальное человеческое поведение не всегда хорошо описывается теорией игр и требует учета не только «пользы» от выигрыша, но и других факторов. Например, если применить игру, известную под названием «Дилемма заключенного», к противостоянию США и СССР времен холодной войны, оптимальным решением было первыми нанести ядерный удар по противнику, но этого не произошло. Другие факторы помешали принятию такого решения. Что это за факторы, спорят до сих пор.

На принятие решений влияют *знания*, которыми располагают люди или группы людей. При этом знания по их функциональной роли бывают разные. Есть индивидуальное знание отдельного человека, и есть знание, которым суммарно обладает группа людей, но которым может не обладать ни один из ее членов. Есть

общее знание группы людей, которым обладает каждый ее член. Есть знание о знании о знании… других людей. Этот вид знания очень важен и используется в рефлексивном управлении [1]. Еще один важный вид знания в англоязычной литературе именуется *common knowledge*. Трудно подобрать адекватный перевод этого термина на русский язык, но смысл заключается в том, что это знание, исходя из которого должны (!) строить свое поведение все взаимодействующие субъекты. Например, если глава государства делает важное заявление, то назначение этого заявления – стать *common knowledge*, игнорировать которое в будущем уже нельзя. Можно сказать, что после таких заявлений мир меняется и люди начинают жить в новых условиях.

Но знания – это всего лишь один и далеко не единственный фактор, влияющий на поведение людей. Очень часто люди считают знаниями то, что на самом деле является лишь их *убеждениями*. Одни убеждены, что Бог существует, другие, что его нет. Одни верят гороскопам, другие отметают любую эзотерику. Убеждения более разнообразны, чем знания, но процессы их формирования и функционирования также могут быть подвергнуты строгому анализу логико-математическими средствами.

Но и это еще не все. На наше поведение влияют *потребности* в пище, крыше над головой, социальной защищенности и продолжении рода. У каждого из нас есть *привычки*, от которых мы не хотим отказываться. Мы находимся под воздействием различных *запретов* со стороны законов и общественной морали. На основе всего этого формируются *предпочтения*. Поведение отдельного человека – это результирующая многовекторного воздействия на него этих и многих других факторов, это сложная траектория между многочисленными точками притяжения и отталкивания.

Несмотря на все сложности, ситуации социального взаимодействия могут быть подвергнуты строгому анализу с использованием математики и логики. Появился симбиоз теории игр и логики [3], и тем самым был приоткрыт ящик Пандоры.

На определенном уровне абстракции поведение человека можно представить как последовательное прохождение ситуаций принятия решений. Брать с собой зонтик или нет? Перебежать улицу или дождаться зеленого света светофора? Какую книгу купить по интересующей тематике? В каждой из этих ситуаций человек совершает выбор, который задает направление будущего развития событий. Ситуации принятия решений являются базовыми для теории игр и специально изучаются в теории принятия решений.

В социальном программировании изучают не только то, как активные субъекты совершают выбор, но и динамику изменения и формирования предпочтений [4, 5, 7]. Какие информационные условия требуется создать, чтобы выбор стал предсказуем? Это выходит за рамки вопросов, относящихся к классической теории игр и теории принятия решений. Понятно, какое важное значение имеют подобные исследования. До сих пор на ниве создания условий для принятия требуемых решений работали рекламные и PR-службы. От таланта их сотрудников зависел успех продвижения конкретных товаров и услуг. Теперь на смену талантливым одиночкам приходит

наука, создавая теорию и технологии целенаправленного изменения поведения людей. Мы говорим об отдельных людях, но нужно учесть, что это же относится к их группам и вообще любым субъектам социальных взаимодействий. Отсюда один шаг до целенаправленных манипуляций.

Наблюдая за происходящими событиями, можно заметить признаки того, что эти технологии уже начали применяться на практике и демонстрируют свою эффективность. Хотелось бы обратить внимание на так называемые Nudgetехнологии, много публикаций о которых можно найти в сети Интернет [2]. Эти технологии уже содержат элементы, активно изучаемые в социальном программировании.

Перспективы социального программирования

То, что современное общество слишком сложно устроено, очевидно всем. Небольшой сбой в функционировании или синхронности одного из его структурных элементов может вызвать по цепочке общий коллапс. Ручного управления, опирающегося на прежний опыт, уже недостаточно. Поэтому вполне естественно не только широкое использование компьютерной техники, что уже происходит, но и разработка новых эффективных социальных процедур. Именно эту задачу и пытается решать социальное программирование. Мы отстаем на десять лет, но еще есть время наверстать упущенное

Список литературы

1. Лефевр В. Алгебра совести. М.: Когито-Центр, 2003. 426 с.
2. Санстейн К. Надж: краткое руководство. URL: http://hrazvedka.ru/guru/nadzh-kratkoerukovodstvo.html.
3. Benthem J. Logic in Games. Cambridge, 2014. 547 p.
4. Ditmarsch H., Hoek W., Kooi B. Dynamic Epistemic Logic. Berlin, 2007. 296 p.
5. Eijck J., Verbrugge R. (eds.) Discourses on Social Software. Amsterdam, 2009. 248 p.
6. Hoare C. An axiomatic basis for computer programming // Communications of the ACM. 1969. Vol. 21. P. 576-580, 583.
7. Liu F. Reasoning about Preference Dynamics // Synthese Library. Vol. 354. Dordrecht, 2011. 203 p.
8. Pacuit E. Topics in Social Software: Information in Strategic Situations // PhD. thesis, CUNY. 2005. 85 p.
9. Parikh R. Language as social software // International Congress on Logic, Methodology and Philosophy of Science, 1995. P. 417.
10. Parikh R. Language as social software // Future Pasts: the Analytic Tradition in Twentieth Century Philosophy / ed. J. Floyd and S. Shieh. Oxford, 2001. P. 339-350.
11. Parikh R. Social software // Synthese. 2002. Vol. 132. P. 187–211.
12. Parikh R. Towards a Theory of Social Software // Technical Report. CUNY, 2002. 13 p.
13. Pauly M. Logic for Social Software // Ph.D. thesis. University of Amsterdam, 2001. 173 p.
14. Pauly M. Programming and Verifying Subgame-Perfect Mechanisms // Journal of Logic and Computation. 2005. Vol. 15, No. 3. P. 295-316.

Дата поступления в редакцию: 24.12.15

Авторская справка
Шалак Владимир Иванович, доктор философских наук, ведущий научный сотрудник сектора логики Института философии РАН. E-mail: shalack@mail.ru

UDC 316.62

SOCIAL SOFTWARE

V. I. Shalack
Institute of Philosophy, Russian Academy of Sciences, Moscow (Russia)

Abstract. Social software is new field of interdisciplinary research at the boundary of computer science, game theory, logic and social sciences. The main purpose of research is an analysis and development of new effective social procedures. To do this, we must solve the problem of testing the correctness of social procedures, analysis and optimization of information exchange, synthesis of new procedures, monitoring of many-agent social interaction.

Keywords: social software, social procedures, game theory, logic, computer science

References

1. Lefevr V. Algebra sovesti. M.: Kogito-Tsentr, 2003. 426 s.
2. Sanstein K. Nadzh: kratkoe rukovodstvo. URL: http://hrazvedka.ru/guru/nadzh-kratkoerukovodstvo.html
3. Benthem J. Logic in Games. Cambridge, 2014. 547 p.
4. Ditmarsch H., Hoek W., Kooi B. Dynamic Epistemic Logic. Berlin, 2007. 296 p.
5. Eijck J., Verbrugge R. (eds.) Discourses on Social Software. Amsterdam, 2009. 248 p.
6. Hoare C. An axiomatic basis for computer programming // Communications of the ACM. 1969. Vol. 21. P. 576-580, 583.
7. Liu F. Reasoning about Preference Dynamics // Synthese Library. Vol. 354. Dordrecht, 2011. 203 p.
8. Pacuit E. Topics in Social Software: Information in Strategic Situations. // PhD. thesis, CUNY. 2005. 85 p.
9. Parikh R. Language as social software // International Congress on Logic, Methodology and Philosophy of Science. 1995. P. 417.
10. Parikh R. Language as social software // Future Pasts: the Analytic Tradition in Twentieth Century Philosophy / ed. J. Floyd and S. Shieh. Oxford, 2001. P. 339-350.
11. Parikh R. Social software // Synthese. 2002. Vol. 132. P. 187–211.
12. Parikh R. Towards a Theory of Social Software // Technical Report. CUNY, 2002. 13 p.
13. Pauly M. Logic for Social Software // Ph.D. thesis. University of Amsterdam, 2001. 173 p.
14. Pauly M. Programming and Verifying Subgame-Perfect Mechanisms // Journal of Logic and Computation. 2005. Vol. 15, No. 3. P. 295-316.

Author's Bio
Shalack Vladimir Ivanovich, doctor of philosophy, leading researcher of the department of Logic, Institute of Philosophy of the Russian Academy of Sciences.
E-mail: shalack@mail.ru