МАТЕМАТИЧЕСКАЯ ЛОГИКА

КРЕАТИВНАЯ НЕДЕТЕРМИНИРОВАННАЯ ВЫЧИСЛИМОСТЬ*

А.М. Анисов

Сектор логики
Институт философии РАН
ул. Волхонка, 14, Москва, Россия, 199991

Как показано в данной работе, любым реализуемым в стандартных теориях детерминированной вычислимости процессам присущ ряд принципиальных ограничений, существенно затрудняющих их применение в анализе философских проблем, связанных с протеканием явлений во времени. Предлагается метод, основанный на нестандартном обобщении идеи вычислимости, позволяющий осуществить адекватное исследование таких философских проблем. Эта цель достигается за счёт отказа от детерминизма в вычислительных процессах.

Ключевые слова: недетерминированная вычислимость, вычислительные концепции, математическая логика, языки программирования.

В последнее время предпринимаются усилия по созданию вычислительных концепций тех или иных сфер реальности или даже всего универсума в целом. Если при этом пользоваться только стандартной теорией детерминированной вычислимости, картина реальности также окажется полностью детерминированной, со всеми вытекающими отсюда неприятными философскими следствиями. В таких условиях задача построения и логико-философского исследования альтернативных теорий вычислимости, допускающих в той или иной форме элементы индетерминизма, становится особенно актуальной для науки и философии.

Следуя идеям Марио Бунге [1], определим **детерминизм** как принцип, согласно которому всё существующее, во-первых, чем-то *однозначно обусловлено*, и, во-вторых, во всём *полностью определённо*. Отсюда **недетерминизм** означает либо неоднозначную обусловленность, либо неполную определённость, или и то и другое вместе. Крайней формой недетерминизма является **индетерминизм** — допущение существования чего-либо ничем не обусловленного или полностью неопределённого.

* Работа выполнена при поддержке РГНФ, проект № 07-03-00203а.

В основе предлагаемого подхода к проблеме недетерминированной вычислимости лежит осознание того факта, что исторически сложившиеся представления о вычислимости являются слишком узкими для успешного применения в философии и в науке. Например, во всех имеющихся теориях вычислимости (как детерминированной, так и недетерминированной) всякое конкретное вычисление непременно имеет первый шаг выполнения. Тем самым любой гипотетический не имеющий начала процесс автоматически оказывается за рамками возможности его моделирования средствами этих теорий.

Выделенные в литературе типы недетерминированной вычислимости также оказываются ограниченными и неполными. По сути, всё сводится к трём разновидностям выходов за пределы детерминизма: вычислимость на отношениях (вместо функций), вероятностные правила вычислений и вычислимость с использованием оракулов. При этом применяются стандартная математика и логика, а философские основания феномена недетерминированности вообще не обсуждаются [2].

Между тем в ходе философского осмысления человеческого опыта и результатов наук накапливается всё больше проблем, обсуждение которых требует привлечения альтернативных теорий вычислимости. Возьмём старую философскую проблему причинности. Победоносное шествие юмовского подхода к причинности как функции, при котором проблема производительной причинности (в смысле «причина *производит* следствие») объявляется псевдопроблемой, обусловлено, помимо прочих обстоятельств, отсутствием надлежащего концептуального аппарата. В подходящей альтернативной теории вычислимости понятию «производительной причинности» можно придать точный смысл. Редуцирование проблемы времени к геометрическим вопросам, как это делается в современной физике, не позволяет даже поставить проблему течения времени. Это можно сделать, понимая течение времени как осуществление особого вычислительного процесса, однако средств традиционной теории вычислимости здесь недостаточно хотя бы в силу того, что их применение предрешает ответ на вопрос о начале времени.

Ещё одна важная проблема связана с моделированием интеллекта. Естественная идея трактовать интеллектуальные операции как вычисления наталкивается на обоснованные контраргументы именно в связи с тем, что исходная теория вычислимости оказывается слишком бедной для построения таких моделей. Наконец, множатся попытки весь мир представить в виде совокупности вычислительных процессов. Если при этом базироваться на имеющихся теориях вычислимости, полученные результаты останутся на уровне метафор или игровых моделей (типа игры Конвея «Жизнь»).

Таким образом, исходным пунктом нашего подхода является следующий тезис: ряд важных старых и новых философских проблем требует для точной постановки и обсуждения применения вычислительных методов, однако имеющихся в нашем распоряжении теорий вычислимости для этого совершенно недостаточно. Отсюда возникает задача построения и изучения альтернативных теорий вычислимости, более адекватных соответствующей философской проблематике. Как планируется показать, важнейшей характеристикой полученных в результате моделей вычислимости будет недетерминизм.

Поставленная задача потребует, в первую очередь, применения методов и понятийного аппарата современной математической логики. Но главное внимание будет обращено не на математические подробности, а на концептуальную сторону дела. Проблемы недетерминированной вычислимости в их логической постановке должны рассматриваться в тесной связи с философскими концепциями. Сама исходная постановка этих проблем мотивирована философскими соображениями, а не нуждами математических теорий вычислимости. Из этого вытекает, что в центре исследования будут находиться вопросы логико-философского обоснования концепции недетерминированной вычислимости.

Более конкретно исследование недетерминированной вычислимости предполагается вести на базе абстрактных вычислительных устройств (машин или компьютеров). Теорию таких устройств крайне затруднительно представить в аксиоматической форме. Остаётся применить метод генетического построения теории. Вычислительные устройства, допустимые для них предписания (программы) и типы данных, которые они используют, будут точным образом описываться. Затем путём проведения строгих мысленных экспериментов с работающими под управлением программ устройствами будет осуществляться развёртывание теории и постижение свойств получаемых в результате абстрактных вычислительных процессов. Кстати говоря, классическая теория машинной вычислимости (в виде машины Тьюринга, машины с неограниченными регистрами и т.д.) задаётся именно генетическим способом (хотя попытки её аксиоматизации имеются).

Наиболее удобной для изложения и изучения формой вычислительного устройства является обобщенно понимаемый компьютер, обладающий двумя главными компонентами — процессором (осуществляющим очередной шаг вычисления в соответствии с программой) и памятью (хранилищем промежуточных или итоговых данных). Будут рассмотрены различные типы недетерминированных компьютеров. Тип компьютера будет определяться разновидностями абстрактных программных команд, которые он способен выполнять. Для этого потребуется формально строго задать синтаксис и семантику соответствующих абстрактных языков программирования. Семантику предполагается формулировать в терминах пред- и постусловий.

Абстрактные компьютеры, языки программирования и недетерминированные вычисления интересны не сами по себе, а в связи с возможностью точной постановки и обсуждения (в идеале — и решения, пусть не окончательного) допускающих вычислительный подход аспектов следующих философских проблем: детерминизма, причинности, движения, течения времени, креативности (проблемы появления нового и его «стыковки» со старым), искусственного интеллекта (ИИ), изменяющегося знания. Эти проблемы, повторим, должны браться не во всей сложности и глобальности, а лишь в аспекте их вычислительной трактовки в рамках дихотомии детерминизм — недетерминизм.

Однако в философских исследованиях в нашей стране и за рубежом идея недетерминированной вычислимости не находит заметного применения. Причина, по-видимому, заключается в том, что философы либо довольствуются традицион-

ной теорией детерминированной вычислимости, либо объявляют решаемые ими задачи принципиально невычислимыми. Так, в философии ИИ уже долгое время идёт дискуссия между сторонниками и противниками вычислительного подхода. Первые возлагают надежды на компьютеры нетрадиционной архитектуры (параллельные машины, клеточные автоматы, квантовые компьютеры и др.), вторые не без оснований указывают, что подобные компьютеры не выводят нас за границы классической вычислимости. Например, хотя квантовые вычисления в ряде случаев оказываются эффективнее вычислений по Тьюрингу, они не выходят за границы класса стандартных вычислимых функций.

Введем в рассмотрение идеальные (в противоположность реальным) вычислительные устройства — абстрактные компьютеры. Каждый абстрактный компьютер @ представляет из себя упорядоченную пару вида <Mm, Pr>, где Mm — память компьютера @, в которой размещаются результаты вычислений, и Pr — процессор, осуществляющий необходимые вычисления. Поскольку термин «вычисление» нами трактуется предельно широко, на размеры памяти Mm и возможности процессора Pr не накладывается никаких ограничений, связанных с требованиями финитности, конструктивности, алгоритмичности и т.п. Вместо этого будем считать, что абстрактные компьютеры способны совершать любые преобразования, допустимые в рамках теории множеств и теории моделей, и именно в этом смысле понимать термин «вычисление» применительно к абстрактным компьютерам. Важно, однако, чтобы последовательность таких преобразований была *линейной дискретной последовательностью шагов*.

В качестве памяти абстрактных компьютеров разрешается использовать любые непустые множества произвольной мощности. В частности, память Mm компьютера @ = <Mm, Pr> может иметь несчётную мощность.

По определению, Mm(S) — подмножество множества Mm, указывающее, как много регистров или ячеек памяти (элементов Mm) ушло на размещение объекта (множества) S:

$$\mathrm{Mm}(S) \subset \mathrm{Mm}.$$

А если в действительности объект S не был размещён в памяти Mm? Тогда естественно считать, что для размещения S не была использована ни одна из ячеек памяти, т.е. что $\mathrm{Mm}(S) = \varnothing$. Короче говоря, объект S размещён в памяти Mm, если и только если $\mathrm{Mm}(S) \neq \varnothing$.

Последнее условие, налагаемое на множества вида Mm(S), касается проблемы размещения в памяти двух и более объектов. Если необходимо поместить в память Mm множества S и S′ (за один шаг или последовательно, множество за множеством), будем считать, что они займут непересекающиеся области памяти Mm, если только эти множества различны:

$$S \neq S' \rightarrow \mathrm{Mm}(S) \cap \mathrm{Mm}(S') = \varnothing.$$

Если же S = S′, то, само собой разумеется, Mm(S) = Mm(S′).

Размещением теоретико-множественных объектов в памяти, равно как и их удалением, управляет выполняемая процессором Pr программа, написанная на спе-

циальном языке АВТ — абстрактном языке программирования. Компьютеры, способные выполнять АВТ-программы, будем называть АВТ-компьютерами. Сформулируем постулат, касающийся АВТ-программ и АВТ-компьютеров, который ввиду его принципиальной важности выделим особо.

Постулат существования: *любой объект может появиться в памяти* Mm *или исчезнуть из нее только в результате выполнения процессором* Pr *соответствующего оператора языка программирования* АВТ.

Программы на языке АВТ являются, по определению, конечной последовательностью инструкций

$$\begin{array}{c} \mathrm{I}_{i_0} \\ \mathrm{I}_{i_1} \\ \vdots \\ \mathrm{I}_{i_n} \end{array}$$

(где i_0, i_1, ..., i_n — натуральные числа и $i_j < i_k$, если $j < k$), которые выполняются одна за другой сверху вниз, если только нет команды изменить порядок их выполнения.

Каждая инструкция порождает элементарный процесс и содержит либо единственный оператор языка АВТ, либо представлена в виде составного оператора

IF *условие* THEN *оператор*,

где IF ... THEN имеет обычный смысл (как, например, в языке BASIC).

Подчеркнём, что и этот составной оператор выполняется за один шаг и, таким образом, порождает элементарный процесс. В качестве *условий* можно брать любые теоретико-множественные и теоретико-модельные формулы.

Оператор GOTO. Хорошо известный оператор безусловного перехода. Используется в АВТ-программах в виде конструкции

GOTO I_j,

где I_j — одна из инструкций соответствующей АВТ-программы. Его действие ничем не отличается от поведения аналогичных операторов в обычных языках программирования.

Оператор завершения АВТ-программ END. Если выполнен оператор END, процесс выполнения соответствующей АВТ-программы заканчивается. При этом в памяти АВТ-компьютера сохраняются все объекты, размещённые там в ходе выполнения программы.

Следующие два оператора специфичны, поэтому их характеристика будет более подробной.

Оператор выбора CHOOSE. Применяется в АВТ-программах в следующей форме.

CHOOSE *список переменных* | *условие*

В этой записи *условие* означает то же самое, что и в случае оператора IF ... THEN, за исключением того, что *условие* должно содержать **все** переменные из *списка переменных*, причем переменные не должны быть **связанными** (т.е. в условии не должно быть кванторов по этим переменным). На *список переменных* также накладываются ограничения: он не должен содержать **повторных** вхождений одной и той же переменной, и в него не могут входить переменные, значения которых **уже** размещены в памяти Mm. Поскольку вопрос о том, значения каких переменных размещены в памяти Mm, требует анализа хода выполнения соответствующей АВТ-программы, последнее ограничение имеет не синтаксический, а семантический характер.

Более формально синтаксическую форму оператора CHOOSE можно представить в виде записи

$$\mathsf{CHOOSE}\ X_0, X_1, X_2, ..., X_n \mid \textit{условие}\,(X_0, X_1, X_2, ..., X_n),$$

где X_i — некоторая переменная, причем переменные X_i и X_j различны, если $i \neq j$. Все выражение может быть прочитано как «Выбрать объекты (множества) $X_0, X_1, X_2, ..., X_n$ такие, что выполняется предикат *условие* $(X_0, X_1, X_2, ..., X_n)$».

Сформулируем условия выполнимости оператора CHOOSE в общем виде. Если процессор Pr АВТ-компьютера @ = <Mm, Pr> выполняет синтаксически правильную инструкцию I вида

$$\mathsf{CHOOSE}\ X_0, X_1, X_2, ..., X_n \mid \textit{условие}\,(X_0, X_1, X_2, ..., X_n)$$

и *предусловие* P

$$\mathrm{Mm}(X_0) = \varnothing\ \&\ \mathrm{Mm}(X_1) = \varnothing\ \&\ \mathrm{Mm}(X_2) = \varnothing\ \&\ ...\ \&\ \mathrm{Mm}(X_n) = \varnothing$$

ложно, выполнение завершается аварийно: произойдет **авост**.

Если P **истинно**, процессор Pr пытается найти (выбрать) такие объекты (множества) $S_0, S_1, S_2, ..., S_n$, которые, будучи присвоены в качестве значений переменным $X_0, X_1, X_2, ..., X_n$ соответственно, обеспечивают *истинность условия* инструкции I. Затем процессор Pr пытается *разместить в памяти* Mm объекты $S_0, S_1, S_2, ..., S_n$.

Если объектов (множеств) $S_0, S_1, S_2, ..., S_n$, удовлетворяющих *условию* инструкции I и способных поместиться в свободной области памяти Mm, **не существует**, выполнение I завершается **авостом**. В противном случае (т.е. если требуемые объекты **существуют** и памяти для их размещения **достаточно**) выполнение I завершается успешно в состоянии, в котором **истинны** следующие *постусловия*:

$$\mathrm{Mm}(S_i) \neq \varnothing \text{ для всех } i,\ 0 \leq i \leq n;$$
$$\textit{условие}\,(S_0, S_1, S_2, ..., S_n).$$

Приведём пример конкретной АВТ-программы. Пусть T — какая-либо теория в не более чем счётном языке первопорядкового исчисления предикатов. Рассмотрим синтаксически правильную программу

$$I_1\ \mathsf{CHOOSE}\ X \mid (X \mid\!=\mathrm{T})$$
$$I_2\ \mathsf{GOTO}\ I_1$$

Выполнение первой инструкции состоит в нахождении модели теории T. Но если теория T противоречива, она не имеет модели и выполнение I_1 в соответствии с семантикой оператора CHOOSE завершится аварийно. Однако и в том случае, если теория T имеет модель, это не гарантирует успешности выполнения инструкции I_1. Например, если память АВТ-компьютера, на котором выполняется данная программа, конечна и теория T не имеет конечных моделей, попытка выполнить I_1 приведет к авосту.

Пусть теперь память Mm счётна (т.е. $|Mm| = \omega$). Если теория T непротиворечива, то в соответствии с теоремами логики существуют счётные модели теории T. Одна из таких моделей будет найдена процессором Pr и размещена в памяти Mm. А если память Mm несчётна и T имеет бесконечную модель, то процессор Pr мог бы выбирать между неизоморфными моделями теории T, так как наряду со счётными моделями теория T имела бы и несчётные модели. Но сказать, какой из возможных исходов будет иметь место *до* выполнения инструкции I_1, невозможно в принципе, так что в общем случае при использовании оператора CHOOSE мы имеем дело с ситуацией ***недетерминированного выбора***. В некотором роде оператор выбора CHOOSE близок к аксиоме выбора: их объединяет неконструктивный (в смысле математического конструктивизма) характер получения результатов.

При условии успешного выполнения инструкции I_1 рассматриваемой АВТ-программы процессор Pr приступит к выполнению инструкции I_2, в соответствии с которой произойдет возврат к инструкции I_1. Как только осуществится этот переход по GOTO, возникнет авост. Почему? В силу того обстоятельства, что $Mm(X) \neq \varnothing$ после первого выполнения инструкции I_1. Но оператор выбора CHOOSE в соответствии с определением не может применяться к переменной, в отношении значения которой выбор был уже сделан, а само это значение было размещено в памяти Mm. Таким образом, независимо от того, противоречива теория T или нет, все равно выполнение данной АВТ-программы завершится аварийно.

Очевидно, наряду с оператором, выбирающим объекты и размещающим их в памяти АВТ-компьютера, необходим также оператор, аннулирующий результаты предшествующих актов выбора и освобождающий память для размещения новых объектов.

Оператор уничтожения DELETE. Его синтаксис предельно прост:

DELETE *список переменных*,

где *список переменных* не должен содержать **повторных** вхождений одной и той же переменной (ограничение не очень принципиальное, но упрощающее синтаксис и сохраняющее преемственность с аналогичным ограничением оператора CHOOSE). То же самое можно представить в другой форме.

$$\text{DELETE } X_0, X_1, X_2, \ldots, X_n.$$

Теперь определим семантику рассматриваемого оператора.

Если процессор Pr АВТ-компьютера @ = <Mm, Pr> выполняет синтаксически правильную инструкцию I вида

$$\mathsf{DELETE}\ X_0, X_1, X_2, ..., X_n,$$

и *предусловие* P

$$\mathrm{Mm}(X_0) \neq \varnothing \ \&\ \mathrm{Mm}(X_1) \neq \varnothing \ \&\ \mathrm{Mm}(X_2) \neq \varnothing \ \&\ ... \ \&\ \mathrm{Mm}(X_n) \neq \varnothing$$

ложно, выполнение завершается аварийно: произойдет **авост**.

Если P **истинно**, процессор Pr завершит выполнение инструкции I в состоянии, в котором будет **истинным** следующее постусловие:

$$\mathrm{Mm}(X_i) = \varnothing \text{ для всех } i,\ 0 \le i \le n.$$

Воспользуемся оператором DELETE для модификации рассматриваемого примера АВТ-программы в предположении, что теория T имеет модель и память Mm бесконечна.

Расположить инструкцию с оператором DELETE в данной программе, содержащей всего две инструкции, можно тремя следующими способами:

(π1)	(π2)	(π3)
I_1 CHOOSE X| X |= T	I_1 CHOOSE X| X |= T	I_1 DELETE X
I_2 GOTO I_1	I_2 DELETE X	I_2 CHOOSE X| X |= T
I_3 DELETE X	I_3 GOTO I_1	I_3 GOTO I_1

Очевидно, АВТ-программа π1 успешно работать не будет по той же самой причине, что и исходная программа. Зато с АВТ-программой π2 все в порядке: осуществив выбор модели теории T в соответствии с инструкцией I_1, процессор Pr перейдет к выполнению инструкции I_2. Так как на этот момент предусловие $\mathrm{Mm}(X) \neq \varnothing$ истинно, процессор Pr завершит выполнение I_2 в состоянии $\mathrm{Mm}(X) = \varnothing$ и, выполняя инструкцию I_3, перейдет по GOTO к I_1. Поскольку предусловие $\mathrm{Mm}(X) = \varnothing$ истинно, инструкция I_1 будет вновь выполнена и т.д. — процесс выполнения программы π2 никогда не завершится.

Осталось проанализировать третью альтернативу. Для того чтобы выполнить АВТ-программу π3, процессор Pr должен *вначале* выполнить инструкцию I_1, что возможно лишь в том случае, если $\mathrm{Mm}(X) \neq \varnothing$. Но в соответствии с постулатом существования объект X может появиться в памяти АВТ-компьютера только в результате действия оператора CHOOSE, который должен выполняться *после* команды DELETE, так как выполнение инструкции I_1 с оператором DELETE *предшествует* выполнению инструкции I_2 с оператором CHOOSE в программе π3.

Казалось бы, из сказанного следует однозначный вывод: попытка выполнить АВТ-программу π3 тут же завершится авостом. Однако это так только при условии принятия допущения о том, что процесс выполнения АВТ-программ *обязательно* должен иметь начало. Применительно к обычным компьютерам и языкам программирования правомерность и даже неизбежность принятия данного допу-

щения не вызывает сомнений. Но в случае АВТ-компьютеров и АВТ-программ оно не выглядит столь необходимым.

Действительно, предположим, что процесс выполнения АВТ-программы π3 не имел начала, т.е. всякому очередному выполнению любой инструкции программы π3 предшествовало бесконечное число реализаций этой инструкции. Такое предположение непротиворечиво и потому вполне допустимо. В самом деле, перед тем как в очередной раз выполнить инструкцию I_1, процессор Pr выполнил инструкцию I_3, а перед этим — инструкцию I_2, после чего АВТ-компьютер перешёл в состояние с $Mm(X) \neq \varnothing$. Переход по GOTO к I_1 сохранил это состояние, так что истинность предусловия оператора DELETE была обеспечена. После успешного выполнения I_1 стало истинным утверждение $Mm(X) = \varnothing$, необходимое для выполнения I_2 и т.д.

Наглядно описанный процесс можно изобразить следующей схемой:

$$..., I_1, I_2, I_3, I_1, I_2, I_3, I_1, ...$$

Таким образом понятый процесс выполнения программы π3 не имеет ни начала, ни конца, в отличие от традиционных вычислительных процессов, которые непременно когда-либо начинаются. Тем не менее, будет ли выполняться программа π3? Утвердительный ответ вытекает из принятия следующего постулата.

Постулат реализуемости: *если предположение о том, что* АВТ-*программа* π *выполнима, непротиворечиво, то программа* π *выполняется.*

Интересное, на наш взгляд, различие между АВТ-программами π2 и π3 заключается в том, что π3 можно выполнить только при условии отсутствия начала процесса выполнения, тогда как π2 выполнима независимо от того, имел процесс её выполнения начало или нет. Гипотетический процесс выполнения π2, имеющий первый шаг, был описан выше. Что касается описания воображаемого выполнения π2 в ходе не имеющего начала процесса, то оно практически полностью повторяет соответствующее описание выполнения π3. Мы говорим о гипотетических или воображаемых процессах выполнения π2 потому, что если допустить наличие не имеющих начала процессов наряду с «нормальными», то на вопрос о том, процесс какого типа осуществляется при выполнении π2 на данном АВТ-компьютере, нельзя ответить однозначно. С равным успехом это может быть как первая, так и вторая разновидность процессов.

Обсуждаемое различие важно для приложений в философии. Так, проблема начала времени не имеет устраивающего всех исследователей единственного решения. Если принимается тезис о том, что эта проблема неразрешима, то для моделирования течения времени больше подходит конструкция, аналогичная программе π2; принятие тезиса об отсутствии начала течения времени заставит прибегнуть к программам типа π3. Наконец, на языке АВТ-программ нетрудно выразить и идею начала времени. Для этого достаточно перед выполнением бесконечного цикла выполнить инструкцию, которая больше уже выполняться

не будет. Например, применительно к программе $\pi 2$ достаточно добавить к списку её инструкций команду GOTO I_1.

($\pi 4$)

I_0 GOTO I_1

I_1 CHOOSE $X|\, X\, |= T$

I_2 DELETE X

I_3 GOTO I_1

Полученная АВТ-программа $\pi 4$ может быть выполнена только в ходе процесса, имеющего начало. Действительно, первой будет выполнена инструкция I_0, а дальше возникнет бесконечный цикл. Схематически

$$I_0, I_1, I_2, I_3, I_1, I_2, I_3, I_1, \ldots$$

В заключительной части кратко опишем более общую, чем АВТ, теорию абстрактной недетерминированной АВТС-вычислимости, которая позволяет в прямом смысле *творить новое*. Несколько лет назад тема креативности уже обсуждалась нами [3], но это были интуитивно-содержательные рассмотрения, которым теперь мы можем придать формально точный смысл.

Прежде всего, отметим, что оператор недетерминированного выбора CHOOSE в определенном смысле уже является формальным средством моделирования новизны. Что же это за смысл? Поясним, обратившись к великолепной идее Творения по Г. Лейбницу. Лейбниц считал, выражаясь нашим языком, что акт Творения Мира состоял в *выборе* Богом одного из возможных (существующих в его уме) миров в качестве действительного. Правда, этот выбор нельзя назвать недетерминированным, поскольку Бог выбрал *наилучший* из всех возможных миров, т.е. *выбора как такового у него не было*: коль скоро наилучший мир единственный, всеблагой Создатель был *вынужден* взять именно этот мир. Но это издержки концепции Лейбница. Вполне можно допустить, что «хороших» миров много, и среди них нет выделенного, наилучшего. Так что действительно было из чего выбирать, причём недетерминированным образом.

Другой, более радикальный смысл понятия новизны, связан с отказом от самой идеи выбора, пусть и недетерминированного, в акте творения нового. В самом деле, раз есть выбор, то, стало быть, есть и те возможности, из которых выбирают. Они *уже существуют* ***до*** *акта выбора*, вот в чём суть. В этом смысле они никакие не новые. *Подлинная новизна появляется ниоткуда, из ничего*. А из ничего и выбрать ничего нельзя.

Идеи Лейбница нашли формальное выражение в модальной логике, в семантике возможных миров. В этой семантике в каждой модели модального исчисления имеется определённое множество возможных миров, в котором один мир выделен в качестве действительного (правда, уже без атрибута «наилучший»). Что касается идеи радикальной новизны, то никаких формальных аналогов для неё не существует. Более того, согласно распространённому мнению, идея о творении

из ничего носит мистический характер и рационально непостижима, не говоря уже о том, чтобы её формализовать.

С нашей позиции, проблема тут есть, и она не столь проста. Суть её в следующем. С точки зрения математики желательно, чтобы любые формальные структуры возникали закономерным образом. В идеале построение теории множеств, которую можно рассматривать как источник практически всех математических объектов, начинается с постулирования существования пустого и бесконечного множеств, из которых с помощью разрешенных операций получаются все другие множества. Правда, в жизни идеал оказался неосуществимым по причине отсутствия единого универсума множеств. Даже натуральный ряд оказался не единственным в силу наличия нестандартных моделей арифметики. Но это не отменяет главного: любой альтернативный универсум, коль скоро он считается заданным, устроен регулярным и предсказуемым образом. Так что никакие самые экзотические объекты из альтернативных универсумов не могут выступать в качестве примеров творения нового из ничего. Все их свойства предопределены, никакой их атрибут не может вдруг появиться или, напротив, быть утрачен.

Но в реальности свойства появляются и исчезают! Например, когда-то не существовало свойства *Разумное животное*, но ныне это свойство существует. Мы можем быть также уверены, что существует соответствующее этому свойству конечное множество разумных животных. Однако это вовсе не означает, что мы должны быть готовы моделировать такое множество посредством некоторого построения, начинающегося с пустой совокупности. Натуральные числа, допустим, мы так и строим: объявляем, что $0 =_{Df} \varnothing$, $1 =_{Df} \{\varnothing\}$, $2 =_{Df} \{\varnothing, \{\varnothing\}\}$ и т.д. Поведение получаемых объектов регулярно, закономерно и предсказуемо. Но не будет ли бессмысленным предположение, что подобным путем можно получить множество разумных животных? Нам представляется, что будет. Абсурдно полагать, что множество разумных животных возникнет по правилам теории множеств на каком-то этапе порождения множеств из пустой совокупности.

Не означает ли сказанное выше, что похоронена надежда на использование логики и математики в построении структур, которые можно было бы обоснованно считать способными выступать в роли появляющихся из ничего? Ведь логика и математика действительно имеет дело с регулярными, закономерными и предсказуемыми структурами. Или это не всегда так?

И всё же имеется исключение из общего правила. На роль иррегулярных объектов теории множеств мы предлагаем *праэлементы* или *атомы*. Атомы являются праэлементами потому, что они исходные объекты в том смысле, что не получены из каких-то ранее построенных множеств. Праэлементы являются атомами (неделимыми) потому, что им, как и пустому множеству $\varnothing$, ничего не принадлежит в качестве элемента. Тем не менее, они не равны пустому множеству. Атом привлекателен тем, что с чисто математической точки зрения он почти ничего из себя не представляет. Атомы настолько свободны от математических свойств, насколько это вообще представляется возможным. Тем не менее, они — чисто формальные объекты, которые могут быть введены в теорию множеств посредст-

вом соответствующих аксиом [4]. Именно это обстоятельство даёт нам шанс для построения формальной модели творения нового из ничего.

Язык абстрактного программирования АВТС получается из языка АВТ добавлением оператора CREATE. Формально синтаксическую форму оператора CREATE можно представить в виде записи

$$\mathsf{CREATE}\ X_0, X_1, X_2, ..., X_n \mid \textit{условие}\,(X_0, X_1, X_2, ..., X_n),$$

где X_i — некоторая переменная, причем переменные X_i и X_j различны, если $i \neq j$.

Все выражение может быть прочитано как «Создать атомы или множества атомов $X_0, X_1, X_2, ..., X_n$ такие, что выполняется предикат *условие* $(X_0, X_1, X_2, ..., X_n)$».

Сформулируем условия выполнимости оператора CREATE в общем виде. Если процессор Pr АВТ-компьютера @ = <Mm, Pr> выполняет синтаксически правильную инструкцию I вида

$$\mathsf{CREATE}\ X_0, X_1, X_2, ..., X_n \mid \textit{условие}\,(X_0, X_1, X_2, ..., X_n)$$

и *предусловие* P

$$\mathrm{Mm}(X_0) = \varnothing\ \&\ \mathrm{Mm}(X_1) = \varnothing\ \&\ \mathrm{Mm}(X_2) = \varnothing\ \&\ ...\ \&\ \mathrm{Mm}(X_n) = \varnothing$$

ложно, выполнение завершается аварийно: произойдет **авост**.

Если P **истинно**, процессор Pr пытается создать (сотворить из ничего) такие атомы или множества атомов $S_0, S_1, S_2, ..., S_n$, которые, будучи присвоены в качестве значений переменным $X_0, X_1, X_2, ..., X_n$ соответственно, обеспечивают *истинность условия* инструкции I. Затем процессор Pr пытается *разместить в памяти* Mm объекты $S_0, S_1, S_2, ..., S_n$.

Если атомов или множеств атомов $S_0, S_1, S_2, ..., S_n$, удовлетворяющих *условию* инструкции I и способных поместиться в свободной области памяти Mm, **логически не может существовать**, выполнение I завершается **авостом**. В противном случае (т.е. если *условие* $(X_0, X_1, X_2, ..., X_n)$ **непротиворечиво** и памяти для размещения новых объектов $S_0, S_1, S_2, ..., S_n$ **достаточно**) выполнение I завершается успешно в состоянии, в котором **истинны** следующие *постусловия*:

$$\mathrm{Mm}(S_i) \neq \varnothing \text{ для всех } i,\ 0 \leq i \leq n;$$
$$\textit{условие}\,(S_0, S_1, S_2, ..., S_n).$$

Завершим наши усилия сотворением нового атома из ничего в предположении, что памяти достаточно для размещения одного атома. Рассмотрим следующую элементарную АВТС-программу.

$$I_1\ \mathsf{CREATE}\ X \mid (X \neq \varnothing)\ \&\ \forall Z\,(Z \notin X)$$

В соответствии с постулатом существования, до выполнения этой программы $\mathrm{Mm}(X) = \varnothing$. Условие $(X \neq \varnothing)\ \&\ \forall Z\,(Z \notin X)$ указывает, что создаваемый X будет *атомом* (а не множеством атомов). В ходе выполнения (согласно постулату реализуемости) оператора CREATE атом S, присваиваемый переменной X, *появ-*

ляется ниоткуда, возникает в истинном смысле из ничего. Но до выполнения оператора CREATE атом S нигде не существовал ни в каком качестве, т.е. S будет *совершенно новым*. После выполнения программы имеем $Mm(S) \neq \varnothing$ и $(S \neq \varnothing)$ & $\forall Z\,(Z \notin S)$.

ЛИТЕРАТУРА

[1] *Бунге М.* Причинность. Место принципа причинности в современной науке. — М., 1962.
[2] *Анисов А.М.* Недетерминированная вычислимость: Философские основания // Логические исследования. — Вып. 15. — М., 2008. — С. 5—30.
[3] *Анисов А.М.* Креативность // Credo new. — 2002. — № 1. — С. 103—116.
[4] *Йех Т.* Теория множеств и метод форсинга. — М., 1973.

A CREATIVE NONDETERMINISTIC COMPUTABILITY

A.M. Anisov

Department of Logic
Institute of Philosophy RAS
Volhonka, 14, Moscow, Russia, 199991

This paper shows a range of fundamental limitations inherent to any process realized in a standard theory of deterministic computability that significantly hamper its application to an analysis of philosophical problems concerning time-flow phenomena. Here a method based on a non-standard generalization of a computability concept is proposed that allows an adequate investigation of such philosophical problems. This goal is achieved by means of rejecting determinism in computational processes.

Key words: nondeterministic computability, computability concepts, mathematical logic, programming languages.