УДК 004.89

ОБ УПРАВЛЕНИИ В ЛОГИЧЕСКОМ МОДЕЛИРОВАНИИ

Попов С.В.

ООО «Научно-внедренческая фирма БП+», Москва, *e-mail:s-v-popov@yandex.ru*

Активное использование логических методов для моделирования различных предметных областей в интересах создания их т.н. «цифровых двойников», требует анализа процесса формирования понятий предметной области. Последние есть базис для описания логических моделей, выступающих в качестве «цифровых двойников». В таком случае теория понятий становится прикладной наукой, но уже не в смысле исследования семантики, а в смысле порядка порождения объектов, являющихся представителями понятий предметной области. Этот процесс носит характер сложного взаимодействия различных частей модели, т.к. понятия сложной предметной области, в общем случае, частично-упорядочены и этот порядок нельзя назвать простым. А так как для моделирования необходимы точность и полнота формирования понятийной базы, то требуется достаточно сложный аппарат управления порождением объектов-представителей понятий. Тем самым, можно утверждать, что программа создания логической модели («цифрового двойника») с необходимостью представляется как состоящая из двух компонентов, выполняющих различные функции. Первый компонент – это операционная часть. Ее назначение в том, чтобы на основании уже имеющихся данных, порождать новые объекты, которые суть представители понятий. Но только порождение без управления этим процессом не позволяет выдержать свойство полноты, т.к. частичный порядок на множестве понятий предметной области может содержать рекурсивные конструкции. Т.е., когда в частичном порядке присутствуют циклы. В этом случае необходима система управления, которая контролировала бы порождение новых представителей понятий в соответствии с их частичным порядком. В противном случае возможна потеря полноты модели, что не допустимо для «цифровых двойников». В статье обосновывается концепция такого двухуровневого метода проектирования логических моделей, когда один компонент отвечает за создание объектов – представителей понятий, а другой – за последовательность их порождения. В этом случае выполняется главное условие – корректность и полнота модели.

Ключевые слова: логическая модель, понятие, частичный порядок, предметная область, программа, модель, цифровой двойник.

ABOUT CONTROL IN LOGICAL MODELING

Popov S. V.

LLC "Nauchno-vnedrencheskaya firma BP+", Moscow, e-mail: s-v-popov@yandex.ru

The active use of logical methods for modeling various subject areas in the interests of creating their so-called "digital counterparts" requires an analysis of the process of forming concepts of the subject area. The latter are the basis for describing logical models acting as "digital doubles". In this case, the theory of concepts becomes an applied science, but no longer in the sense of the study of semantics, but in the sense of the order of generation of objects that are representatives of the concepts of the subject area. This process has the character of a complex interaction of various parts of the model, because the concepts of a rather complex subject area are, in general, partially ordered and this order cannot be called simple. And since the accuracy and completeness of the formation of the conceptual base are necessary for modeling, a rather complex apparatus for controlling the generation of objects-representatives of concepts is required. Thus, it can be argued that the program for creating a logical model ("digital twin") is necessarily represented as consisting of two components performing different functions. The first component is the operating part. Its purpose is to generate new objects based on already available data, which are representatives of concepts. But only generation without control of this process does not allow to maintain the property of completeness, because a partial order on a set of concepts of the subject area may contain recursive constructions. I.e., when there are cycles in the partial order. In this case, it is necessary to involve a management system that controlled the generation of new representatives of concepts in accordance with their partial order. Otherwise, the completeness of the model may be lost, which is not acceptable for "digital doubles". The article substantiates the concept of such a two–level method of designing logical models, when one component is responsible for creating objects representing concepts, and the other for the sequence of their generation. In this case, the main condition is fulfilled – the correctness and completeness of the model.

Keywords: logical model, concept, partial order, subject area, program, model, digital twin.

Введение. Опыт разработок в области ИИ, связанных с построением логических моделей различных предметных областей, приводит к однозначному выводу о специфичности их построения [1, 2, 3]. Поясним, что имеется в виду. Всякая сложная информационная система должна обладать адекватным теоретическим базисом, для обоснования приемов переработки информации. Так в качестве базиса информационных систем объектов управления используются конкретные теории: экономические, математические, физические и т.п., а базисом для разработки баз данных выступает реляционная алгебра. С другой стороны, построение логических моделей, связанных с формированием понятий (классов объектов) базируется на представлениях о формализме, с одной стороны, включающем логико-математические преобразования, а с другой, - основывающемся на результатах теоретического программирования. В результате получаем то, что называется *логическими моделями*.

Если логическая модель сложная и связана с порождением новых объектов в рамках определенной понятийной системы, то следует представлять и механизм, с помощью которого порождаются объекты - содержание этих понятий. Если такой механизм, по тем или иным причинам, отсутствует, например, из-за недостаточной проработки предметной области, то это выливается в неуправляемый перебор. Тогда модель громоздка и трудна для использования. Типичным примером являются модели, связанные с задачами на графах. Если не ограничить перебор условиями или стратегиями, то их решение весьма быстро исчерпывают все доступные ресурсы, т.к. основным приемом становится полный перебор.

С другой стороны, если в модели предполагается построение объектов - представителей понятий предметной области, то их не управляемое порождение, в большинстве случаев не приводит к решению задачи. Рассмотрим пример, который иллюстрирует, что получение непосредственных следствий недостаточно, и требуется совершить еще несколько логических преобразований для получения общей картины. Пусть отрезок *AB* пересекают два отрезка *CD* и *EF*, в точках соответственно *G* и *H*. Очевидно, что в этом случае появляются восемь углов, центральными вершинами которых являются *G* и *H*, а концевыми *A*, *B*, *C*, *D*, *E*, *F*. Однако, появление точек *G* и *H* на отрезке *AB* приводят к появлению пяти новых отрезков *AF*, *AG*, *FG*, *BF*, *BG*. Что влечет появление четырех новых углов: *BFG*, *CFG*, *DGF*, *EGF*. Следовательно, рассмотрение только непосредственных следствий из пересечения отрезков *AB*, *CD* и *EF* в виде появления двух точек *G* и *H* и первых восьми углов недостаточно. Надо вывести еще одно следствие в виде новых отрезков от пересечения, и новых четырех углов.

Подобное явление наводит на мысль о необходимости создания механизма управления порождением объектов - представителей понятий в логических моделях. Человек, в своих рассуждениях, управляет построением новых объектов, исходя из содержательных

представлений о цели сформулированной задачи. Программа, решающая интеллектуальную задачу, не основывается на содержательных представлениях, поэтому в нее необходимо заложить знания, извлеченные из предметной области. Только тогда ее решения будут обладать полнотой. В противном случае возможна потеря решений из-за отсутствия в логических построениях некоторых объектов, присутствующих в предметной области.

Исходя из этого, мы рассмотрим основные приемы формирования понятий и предложим механизм, т.н. *Супервизор*, который позволит управлять этим процессом. В результате будут порождены все необходимые для решения объекты логической модели. Все рассуждения будут проиллюстрированы на примерах, взятых из разработанной автором системы «Живая математика», которая создана для поддержки решения математических задач школьного курса. Аналогичные идеи можно обнаружить в [4, 5].

1. ***Понятия***. Так как основная цель логической модели предметной области состоит в моделировании преобразований над объектами, то, естественно, что такая модель опирается на конкретный понятийный базис. В каждой моделируемой предметной области используется множество ***C*** понятий, частично-упорядоченное отношением предок-потомок. Этот частичный порядок допускает циклы, когда понятие определяется через само себя, используя линейный порядок некоторого параметра. Такое определение предполагает использование в качестве понятия-предка с меньшим значением этого параметра. Это т.н. *рекурсивное определение*.

Каждое используемое понятие предметной области в модели представлено совокупностью объектов, являющихся представителями этого понятия. Так понятие *точка* имеет конкретные реализации в виде точек на чертеже с конкретными именами и координатами. То же относится к понятиям *отрезок*, *треугольник*, *многочлен*, *квадратный трехчлен* и т.д. *Отрезок* есть пара точек, обладающая дополнительными параметрами: длина, угол наклона и свободный член уравнения. В объектно-ориентированной программной реализации каждое понятие представляет собой класс с соответствующим содержанием. Тогда каждый представитель этого класса – конкретный представитель понятия. Это может быть фигура чертежа, алгебраическое или тригонометрическое выражение, график конкретной функции и т.п.

2. ***Операции над понятиями, определяющие свойства понятий*** (***R: C→ P***), которые суть значения этих операций. Для их определения вводится множество функций с областью определения ***C***. Например, *длина* отрезка, *площадь* треугольника, *медиана* треугольника, *общий множитель* многочлена, множество *множителей* полинома, конкретные *преобразования* тригонометрических объектов и пр. Если исследуется значение

этих функций, то говорим о конкретных свойствах объектов, ими определяемых. Например, длина отрезка равна 5, синус угла равен 0,3 и т.п. Эти свойства используются для определения других понятий. Так *медиана из угла А – это отрезок, проведенный из вершины А к противоположной стороне, который делит последнюю пополам*.

3. ***Средства конструирования новых понятий*** (***F*: *C* → *C***) *используются для формирования новых понятий, но сами не принадлежат совокупности базисных понятий логической модели*. Например, отрезок есть *пара* точек, треугольник – *три* попарно касающихся и не коллинеарных отрезка, многочлен – *совокупность* одночленов, синус угла – *отношение* длин определенных отрезков и пр. При реализации модели эти средства должны быть вынесены в отдельную программную часть, определенную над совокупностью базисных понятий. Таким образом, для преобразований совокупности ***F*** словарь ***C*** понятий выступает в роли области определения. Отметим, что без управления процессом образования новых понятий, этот процесс становится малоинформативным. Поэтому управление порождением новых объектов должно быть создано в полном согласовании с частичным порядком над множеством ***C***, и требованием *полноты* – т.е. возможности порождения всех объектов понятий, которые следуют из условия задачи. Тем самым, определяется частичный порядок на понятиях, задаваемый отношением предок-потомок.

Необходимость предварительной проработки всего множества понятий с целью получения непротиворечивого частично-упорядоченного множества ***C*** следует из того, что при объектно-ориентированном программировании каждое понятие определяет класс объектов, и наличие противоречия в такой структуре недопустимо. Это приведет в ошибках модели. Исходя из этого, все понятия логической модели должны быть проанализированы и соответствующим образом упорядочены отношением предок-потомок.

4. ***Логическая система*** (***L***) доказательств свойств объектов представляет интерес, т.к. включает логические средства как прямого вывода, так и обратного. Поясним, что имеется в виду в данном случае. Например, доказательство равенства отрезка конкретному значению или равенства площадей треугольников требует логического построения из совокупности аксиом и известных теорем геометрии. Каждую теорему предметной области можно представить в виде логического следования: из набора посылок следует заключение. Тогда применение теоремы представляет собой проверку посылок и, в случае их истинности, присоединение к известным фактам заключения. Тем самым расширяется совокупность данных, которая выводится из исходной формулировки. То же относится к алгебраическим и тригонометрическим преобразованиям. Однако, в последних случаях используются *правила переписывания*, т.е. замены равного равным. Таким образом, прямой вывод – это движение от условия задачи к его заключению.

На этом основано построение т.н. *неподвижной точки* – прием построения модели, когда на каждом шаге выводятся новые данные, присоединяемые к уже известным. Рано или поздно происходит насыщение, т.е. выводимые данные уже не являются новыми и процесс поиска можно останавливать. Это состояние модели и называется *неподвижной точкой*. Именно оно описывает искомое состояние логической модели, которое следует из исходных данных.

Человек, как правило, не может использовать такой прием т.к. возникает большой объем данных, из которых к искомому заключению относится лишь небольшая часть. Компьютер может эффективно оперировать большими массивами информации, и поэтому для него метод неподвижной точки вполне приемлем. Более того, в большинстве случаев, это единственный способ автоматически найти решение.

Обратное движение наиболее наглядно видно на примере задач, в которых требуется совершить ограниченный перебор. Тогда решение задачи сводится к поиску и обоснованию гипотез, из которых выводится требуемое заключение. Например, в задаче: подобрать цифры *a, b, c*, чтобы выполнялось равенство $abc = a! + b! + c!$ однозначно вывести эти значения из условия невозможно. Требуется выдвижение и проверка ряда гипотез, касающихся значений чисел *a, b, c*. К такому же классу решений можно отнести известные логические задачи, числовые ребусы и комбинаторные задачи.

Отметим, что логическая система не вводит новых объектов, а оперирует с ними для доказательства конкретных соотношений, сформулированных в виде логических выражений с использованием трех предыдущих словарей, а также логических правил. Прямой логический вывод может быть представлен: умозаключениями, формирующими древесную конструкцию построения заключения из посылок, линейной последовательностью преобразований при использовании переписывания или комбинацией этих средств. Обратный логический вывод, как правило, образует древесную конструкцию, причем от условия задачи мы движемся к гипотезам, из которых эта задача логически следует, до тех пор, пока не будут обнаружено полное множество утверждений, из которых следует заключение исходной задачи.

5. ***Необходимость Супевизора***. Логическая модель строится по исследуемой предметной области, в которой выделяются определенные понятия, закладываемые в модель. При программированию, каждому понятию соответствует класс, члены которого суть представители соответствующего понятия. Если порядок на множестве $\boldsymbol{C}$ не имеет циклов, то порождение объектов модели простое: из базисных понятий переходим далее в соответствии с порядком. В таком случае, в первом приближении, можно считать, что нет необходимости в особом инструменте для порождения объектов. Но в общем случае частичный порядок на множестве $\boldsymbol{C}$ имеет циклы. Например, рассмотрим критерий равенства треугольников:

равенство пары отрезков и прилегающих углов приводит к равенству треугольников, из которого следует равенство двух пар других соответствующих отрезков, являющихся сторонами треугольников. Здесь отношение на объектах приводит к порождению одноименного отношения на других однотипных объектах. Другой пример – это транзитивное замыкание отношения равенства или параллельности отрезков.

В общем случае, порождение некоторого объекта предметной области вызывает порождения других объектов, если соответствующие понятия находятся в отношении предок-потомок. И тут возникает естественный вопрос: на сколько необходимо разделять структуру описания понятий предметной области в виде классов от процедуры последовательного порождения объектов – представителей понятий. Как отмечалось, если частично-упорядоченное множество понятий предметной области простое, то выделять отдельно процедуру порождения не рационально. Она вполне может быть реализована, используя средства объектно-ориентированного программирования. Однако, если структура понятий предметной области достаточно сложная, то приходится выделять отдельную часть программы (т.н. *Супервизор*), чтобы реализовать последовательность порождения объектов с учетом частичного порядка на множестве понятий.

6. ***Конструкция Супервизора***. *Супервизором* назовем ту часть программы построения логической модели, которая определяет последовательность построения объектов (представителей понятий) в зависимости от формулировки задачи и описания предметной области. Каждое понятие множества $\boldsymbol{C}$ определяет тип объектов, которые суть представители этого понятия. Если $\boldsymbol{o}$ есть объект, то $type(\boldsymbol{o})$ есть то понятие, представителем которого является объект $\boldsymbol{o}$. Так как все понятия предметной области частично упорядочены, то в супервизоре имеется таблица $\boldsymbol{T_C}$, представляющая этот порядок: каждая ее строка (t_i, t_j) представляет пару понятий, упорядоченных так: $t_i \le t_j$. Из этого следует, что когда порождается объект $\boldsymbol{o}$, то Супервизор запускает порождение всех объектов типа t_j, для которых выполняется отношение $type(\boldsymbol{o}) \le t_j$.

Однако, чтобы запустить процесс порождения объектов понятий-потомков, Супервизор должен получить сигнал, который формируется при порождении объектов - представителей понятий-предков. Эти сигналы формируются модулями, осуществляющими собственное порождение объектов – представителей понятий. Рассмотрим для примера работу произвольного модуля $\boldsymbol{M_c}$, порождающего объекты, принадлежащие понятию $\boldsymbol{c} \in \boldsymbol{C}$. При порождении объекта $\boldsymbol{o}$, который есть представитель понятия $\boldsymbol{c}$, вырабатывается сигнал $\boldsymbol{h_c}$, однозначно определяющий порожденный объект $\boldsymbol{o}$. После выполнения модуля $\boldsymbol{M_c}$ управление передается Супервизору, который получает на вход сигнал $\boldsymbol{h_c}$. Т.к. сигнал $\boldsymbol{h_c}$ однозначно определяет тип $type(\boldsymbol{o})$ объекта $\boldsymbol{o}$, последний, используя таблицу $\boldsymbol{T_C}$, однозначно определяет

все понятия, являющиеся потомками понятия *type*(***o***). Тем самым, решен вопрос о запуске соответствующих модулей для порождения новых объектов – представителей понятий.

Однако, если Супервизор будет реагировать на каждый вновь порожденный объект, то ресурсы, затрачиваемые на управление этим процессом, будут излишне большими, т.к. объектов много. Для исключения этого предлагается ввести дополнительный элемент, который есть надстройка над частично-упорядоченным множеством $\boldsymbol{C}$ и позволяет описать семантику модели в более содержательных терминах. Представим частично-упорядоченное множество $\boldsymbol{C}$ в виде ор-графа $\boldsymbol{G_C}$, узлы которого соответствуют понятиям из $\boldsymbol{C}$, а дуги – отношению порядка ≤. Введет следующее определение.

Пусть $\boldsymbol{D} \subseteq \boldsymbol{G_C}$ есть связный подграф, содержащий множество $\boldsymbol{N_D}$ узлов и $\boldsymbol{E_D}$ дуг. При этом $\boldsymbol{In_D}$ есть множество дуг дополнения $\boldsymbol{G_C / D}$, ведущих в $\boldsymbol{D}$, и $\boldsymbol{Out_D}$ – выходящих из $\boldsymbol{D}$. Начала дуг множества $\boldsymbol{In_D}$ определяют те понятия, которые суть предки понятий, соответствующих узлам подграфа $\boldsymbol{D}$. Назовем их *входными* для $\boldsymbol{D}$ и полагаем, что эти понятия представлены объектами – их примерами. Последние пусть образуют множество $\boldsymbol{O}(\boldsymbol{In_D})$. Аналогично, начала дуг из $\boldsymbol{Out_D}$ определяют множество понятий, которые суть потомки понятий множества, определяемых множеством $\boldsymbol{In_D}$. Из вида частичного порядка и операционной семантики, определенной им, следует, что варьирование множества $\boldsymbol{O}(\boldsymbol{In_D})$ объектов влечет изменение объектов $\boldsymbol{O}(\boldsymbol{Out_D})$, определяемых началами дуг множества $\boldsymbol{Out_D}$. Введем такое определение.

Назовем *неподвижной точкой* подграфа $\boldsymbol{D}$ при фиксированном множестве $\boldsymbol{O}(\boldsymbol{In_D})$ такую совокупность $\boldsymbol{O}(\boldsymbol{Out_D})$ объектов, которая не может быть расширена при введенной операционной семантике, если не менять множество $\boldsymbol{O}(\boldsymbol{In_D})$. В данном случае ничего не говорится о конечности или бесконечности множеств $\boldsymbol{O}(\boldsymbol{In_D})$ и $\boldsymbol{O}(\boldsymbol{Out_D})$, интерес представляет только не расширяемость множества $\boldsymbol{O}(\boldsymbol{Out_D})$ при фиксированном $\boldsymbol{O}(\boldsymbol{In_D})$. В качестве подграфа $\boldsymbol{D}$ может выступать и весь граф $\boldsymbol{G_C}$, который обладает пустым множеством $\boldsymbol{In}$, а множество $\boldsymbol{Out}$ определяется условиями предметной области.

Справедлива

***Теорема* 1**. *Неподвижная точка, определяемая графом $\boldsymbol{G_C}$, влечет, что каждый подграф $\boldsymbol{D} \subseteq \boldsymbol{G_C}$ обладает неподвижной точкой.*

С другой стороны, справедлива

***Теорема* 2**. *Если каждый подграф $\boldsymbol{D} \subseteq \boldsymbol{G_C}$ обладает неподвижной точкой, то и граф $\boldsymbol{G_C}$ обладает неподвижной точкой.*

Рассмотрим практические аспекты использования понятия неподвижной точки. Как уже было сказано, если управление поиском решения осуществляется в соответствии со стратегией, когда каждый объект по мере его порождения необходимо проверять на возможность порождения объектов-потомков, то это требует много ресурсов. Поэтому более

рациональной представляется, когда выделяются фрагменты графа G_C, обладающие *содержательной семантикой*, и для каждого из них строится неподвижная точка, определяемая входными понятиями. Например, равенство углов в геометрической задаче может быть получено в результате транзитивного замыкания, выполнения условий параллельности, коллинеарности или перпендикулярности соответствующих сторон и т.д. Тем самым, понятие равенства углов, которое вполне содержательное, выделяет фрагмент графа G_C, для которого строится неподвижная точка. И ее построение может осуществляться одним модулем программы поиска решения.

Следовательно, функционирование Супервизора уже не сводится к попытке построения объекта-потомка каждый раз при порождении одного объекта-предка. Супервизор отвечает за порождением частичных неподвижных точек, определяемых содержательно значимыми фрагментами модели понятий предметной области.

7. ***Моделирование временных зависимостей***. Но рассмотрим еще один аспект построения логических моделей, который связан с порядком порождения объектов-представителей понятий в соответствии с частичным порядком множества $\boldsymbol{C}$. Нас интересует сложный частичный порядок, в котором имеются циклы, т.к. простая иерархия не вызывает особых трудностей при порождении объектов. Если оба понятия $\boldsymbol{c}_0$, $\boldsymbol{c}_1$ связаны отношением предок-потомок, т.е. в частичном порядке над множеством $\boldsymbol{C}$ встречается цикл, содержащий оба эти понятия, то в программной реализации модели объекты понятия $\boldsymbol{c}_1$ будут порождаться несколько раз по мере появления новых объектов-представителей понятия $\boldsymbol{c}_0$. Если не управлять этим процессом, то возникает ситуация, когда один и тот же объект-представитель понятия $\boldsymbol{c}_1$ будет порождаться несколько раз – по числу обращений к соответствующему модулю. Это не приводит к появлению новой информации в модели, но очень ресурсозатратно, т.к. часть объектов порождается несколько раз. Чтобы избежать такого нерационального использования ресурсов, в логическую модель вводится параметр *абсолютное время*, представляющее собой значение одной глобальной переменной, значение которой монотонно увеличивается независимо от выполнения программы построения модели. Тем самым абсолютное время моделирует обычное время. Его назначение состоит в фиксации времени появления объектов мидели и времени выполнения модулей, которые эти объекты порождают. Поясним это примером.

Если объект $\boldsymbol{o}$ порождается модулем $\boldsymbol{M}$ при наличии объектов $\boldsymbol{o}_1$, $\boldsymbol{o}_2$, …, $\boldsymbol{o}_\text{m}$, то естественно допустить, что, хотя бы один из этих объектов $\boldsymbol{o}_1$, $\boldsymbol{o}_2$, …, $\boldsymbol{o}_m$ должен появиться после последнего применения модуля $\boldsymbol{M}$, т.к. в противном случае произойдет повторное порождение объекта $\boldsymbol{o}$. Последнее приводит к нерациональному использованию вычислительных ресурсов. Поэтому введение времени появления каждого объекта модели и времени срабатывания модулей для

управления процессом порождения объектов, оправдано. Технически это реализуется сравнительно просто, что и позволяет достаточно эффективно управлять этим процессом.

В результате получаем, что в случае конечных моделей неподвижные точки будут получены практически при любом способе запуске модулей порождения объектов модели за конечное число шагов. В случае бесконечных моделей говорить о получении неподвижной точки не приходится в принципе, там приходится реализовывать более сложные стратегии построения модели.

Литература

1. Искусственный интеллект и принятие решений: Интеллектуальный анализ данных. Моделирование поведения. Когнитивное моделирование. Моделирование и управление / Под ред. С.В. Емельянова. - М.: Ленанд, 2017. - 108 с.
2. Jeff Heflin, James Hendler, and Sean Luke. Applying Ontology to the Web: A Case Study, In International WorkConference on Artificial and Natural Neural Networks, IWANN'99. 2019.
3. Michael Kifer, Georg Lausen, James Wu. Logical Foundations of Object Oriented and Frame Based Languages. Journal of ACM 2015, vol. 42, p. 741-843
4. Tim Berners-Lee, The Semantic Web and Research Challenges, http://www.w3.org/2013/Talks/01-sweb-tbl/slide1- 0.html
5. Попов С.В. Логическое моделирование. М.: Тровант, 2016. – 255 с.