Серия «Математика»
Том 2 (2008), № 2, С. 1—10

Онлайн-доступ к журналу:
http://isu.ru/izvestia

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 510.62:004.82

Бесконечные ленивые маркированные деревья *

В.С. Ульянов
Иркутский государственный университет

Аннотация. В работе исследуется проблема построения интерфейсов к системам описаний предметных областей в формате логических онтологий. Элитность логических средств, лежащих в основе этого подхода, делает задачу построения интерфейсов, ориентированных на массового пользователя, весьма нетривиальной задачей. В данной работе предлагается подход, основанный на конструкции бесконечного ленивого дерева, исследуются вопросы корректности данного подхода, и способы его реализации.

Ключевые слова: онтологии, Онтобокс, интерфейс, деревья, семантический веб

1. Введение

В работе рассматривается проблема создания интерфейсов для систем представления знаний, основанных на онтологиях, включая семантический веб [4]. Обладая высокой выразительностью и гибкостью, данный подход с большим трудом пробивает себе дорогу к массовому пользователю. Среди ряда причин, одной из ключевых является сложность логических формализмов, лежащих в основе онтологий. В данной работе рассматриваются способы решения данной проблемы через создание интерфейсов, понятных и привычных массовому пользователю. С другой стороны, инкапсулируя сложные логические структуры, такие интерфейсы должны в максимальной степени сохранять преимущества логических описаний. Основными требованиями к интерфейсу системы управления онтологиями для клиента можно назвать следующие:

1. удобство и привычность для пользователя;
2. широкие возможности для редактирования богатых онтологических структур;
3. сохранение целостности онтологии;
4. поддержка мировых форматов представления онтологий (OWL).

* Работа выполнена при частичной финансовой поддержке программ «Фундаментальные исследования и высшее образование» (проект НОЦ-017 «Байкал») и «Развитие научного потенциала высшей школы (2009-2010 гг.)» (проекты РНП.2.2.1.1/5901 и 3.2.3/3488).

Удобство и привычность работы пользователя может обеспечиваться использованием комбинаций стандартных элементов интерфейса (таких как деревья, списки, таблицы, кнопки и др.) вместо логических формул и выражений. Простота использования системы может быть достигнута только посредством скрывания от пользователя онтологической "начинки в идеале он даже не должен знать, что он работает с онтологией и что это такое. Наибольший интерес вызывает возможность представления графа семантической сети в виде деревьев. Для этого необходим общий гибкий и эффективный механизм, расчитанный на обычного пользователя ПК.

2. Представление семантических сетей в виде деревьев

Как известно, любую иерархию можно представить в виде дерева. Это довольно типичный элемент интерфейса, присутствующий в большинстве операционных систем и, в частности, в Windows. Поэтому, для большинства пользователей деревья являются привычным способом представления иерархии, что важно для построения интуитивно понятного интерфейса. Заметим, что иерархия классов - не единственный вид дерева, который можно построить над онтологией. В общем случае, описание предметной области моделируется ориентированным графом, нагруженным информацией в вершинах и на ребрах. Это с одной стороны усложняет задачу выделения деревьев из графа, ввиду многообразия вариантов построения. С другой стороны, по той же самой причине это значительным образом увеличивает спектр возможностей по навигации и управлению онтологическими базами знаний. В частности можно строить иерархии классов и объектов (по принадлежности объектов определенному классу), представлять в виде дерева связи объектов через их свойства. Наличие циклов в семантических графах приводит к тому, что моделирующие их деревья становятся бесконечными, что делает невозможным их актуализацию в ограниченном пространстве компьютера.

Нами был предложен и реализован подход к построению деревьев с помощью т.н. правил. Суть подхода состоит в следующем. Сначала задается типизация узлов дерева в соответствии с типом сущности. Имеются узлы следующих типов: классы, объекты, свойства, типы данных и онтологии. Затем задается набор правил, которые бывают тех же типов, что и узлы: правила для классов, правила для объектов, правила для свойств и т.д.

Правила определяют, в каких условиях они выполняются, а также типизацию и принцип построения дочерних элементов узла. Данный механизм предоставляет следующие возможности представления иерархий в виде дерева:

- в качестве корня дерева может быть выбрана любая сущность базы знаний;
- дочерние элементы создаются по мере открывания узлов (так называемая "ленивая"обработка);
- в качестве дочерних элементов могут выступать объекты любых типов узлов;
- правило может задаваться как для определенного типа узла, так и для конкретной сущности с уникальным URI;
- можно вводить дополнительные (собственные) типы узлов и правила.

В данной работе строится математическая модель данного вида интерфейсов, и исследуются вопросы ее корректности.

3. Ленивые бесконечные деревья

Определение 1. [Дерево] Пусть $V = \{v_1, v_2, \dots\}$ – счетное множество имен, которые будем называть вершинами. Деревом будем называть множество $T \subset V_T \times V_T$ с корнем v_{top} и множеством вершин $V_T \subset V$, что

1. $\forall v_i : \langle v_i, v_{top}\rangle \notin T$ (корень дерева).
2. $\forall u \in V_T \setminus \{v_{top}\}\ \exists v_1, \dots, v_k : \langle v_{top}, v_1\rangle \in T, \dots, \langle v_k, u\rangle \in T$ (связность).
3. $\forall v_1 \dots \forall v_k : \neg(\langle v_1, v_2\rangle, \dots, \langle v_k, v_1\rangle \in T)$ (отсутствие циклов).

Вершину v назовем *листом* дерева T, если $\forall v' \in V_T : \langle v, v'\rangle \notin T$. Вершину v назовем *непосредственной подвершиной* вершины w, если $\langle w, v\rangle \in T$. Вершина v является *подвершиной* w, если она либо непосредственная подвершина w, либо существует v' такая, что v непосредственная подвершина v' и v' подвершина w. Аналогичным образом определяются непосредственные и произвольные надвершины.

Пусть $\mathcal{W} = \langle \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{O}\rangle$ – словарь онтологии, где $\mathcal{C}$ – множество именованных концептов, $\mathcal{R}$ – множество ролей, $\mathcal{P}$ – множество атрибутов, $\mathcal{O}$ – множество имен объектов [3]. Введем отображение $\lambda : V_T \to \mathcal{O}$ такое, что $\forall v \in V_T \exists o \in \mathcal{O} : \lambda(v) = o$. Такое o будем называть меткой, и тот факт, что вершина v помечена объектом o будем обозначать v^o. Различным элементам из V_T могут соответствовать одни и те же элементы из $\mathcal{O}$.

В деревьях, используемых в интерфейсах, подвершины вершин всегда упорядочены. Чтобы отразить этот факт в нашей модели, будем считать, что множество V линейно упорядочено, и для двух подвершин $v_i^{o_i}, v_j^{o_j}$ вершины v^o $\langle v^o, v_i^{o_i}\rangle < \langle v^o, v_j^{o_j}\rangle$ тогда и только тогда, когда $v_i < v_j$.

Определение 2. [Маркированное дерево] Четверку $\mathbb{T} = \langle V_T, T, \lambda, <\rangle$ будем называть маркированным деревом.

В пункте 4 мы рассмотрим способ задания маркированных деревьев. Отметим, что именно маркированные деревья будут использоваться нами в качестве основного элемента интерфейса онтологий. Хотя область их применения не ограничивается онтологиями. Механизм построения деревьев может применяться и к другой структурированной информации, которую можно представить в виде иерархий или ориентированных графов. В последнем случае для работы с циклами используются бесконечные ленивые деревья.

Сложность задачи состоит в том, что онтологии могут иметь самую разнообразную структуру и описывать самые разнообразные предметные области. Поэтому система построения деревьев должна обладать гибкими возможностями, достаточными для того, чтобы деревья могли адекватно отображать суть описываемой предметной области, и легко настраиваться на различные способы навигации по онтологии. Поскольку деревья работают в рамках логических формализмов, то наиболее естественным подходом представляется использование логических средств для формирования самих деревьев, для того, чтобы они стали органичной частью логической системы. Ниже представлен подход, реализующий эту общую идею.

4. Правила построения дерева

Определим дерево как выводимый объект нашей логической системы. Для этого введем общую структуру правила построения дерева. Пусть G – одноместный предикат, действующий на объектах $\mathcal{O}$, H – одноместная функция, действующая на объектах $\mathcal{O}$ и возвращающая последовательность объектов. Тогда общая структура правила для маркированного дерева $\mathbb{T} = \langle V_T, T, \lambda, <_T \rangle$ будет иметь следующий вид:

$$r[G,H]\frac{\mathbb{T}, v^o \in V_T \qquad G_r(o) = true \quad \langle o_1, \ldots, o_k \rangle = H_r(o)}{\mathbb{T}'}$$

где $\mathbb{T}' = \langle V'_T, T', \lambda', < \rangle$ дерево такое, что

1. $V'_T = V_T \cup \{v_1, \ldots, v_k\}$ и $\{v_1, \ldots, v_k\} \subset V \setminus V_T$;
2. $\lambda'(v) = \begin{cases} o_i, & \text{если } v = v_i \in \{v_1, \ldots, v_k\} \\ \lambda(v), & \text{если } v \in V_T \end{cases}$
3. $T' = T \cup \{\langle v^o, v_1^{o_1} \rangle, \ldots, \langle v^o, v_k^{o_k} \rangle\}$

С содержательной точки зрения, правило «раскрывает» вершину v^o дерева, добавляя к ней подвершины $v_1^{o_1}, \ldots, v_k^{o_k}$ в следующей последовательности:

1. сначала проверяется условие $G_r(o)$ на применимость правила к вершине v^o, и если оно выполняется, то
2. с помощью $H_r(o)$ генерируется последовательность «маркеров» для подвершин $\langle o_1, \ldots, o_k \rangle$;

3. для каждого o_i из счетного множества V выбирается новая не задействованная в дереве $\mathbb{T}$ вершина v_i с таким расчетом, что если $i < j$, то $v_i < v_j$ в V (чтобы обеспечить порядок подвершин);
4. каждая $v_i, i = \overline{1,k}$, маркируется сущностью o_i и добавляется к множеству V_T; а в T добавляется пара инцидентности $\langle v^o, v_i^{o_i} \rangle$;

Отметим, что правило r[G,H] полностью определяется предикатом G и функцией H, в то время как $\mathbb{T}$ и v^o являются его параметрами.

Пусть имеется вершина v_{top} и упорядоченная последовательность правил $\langle r_1[G_1, H_1], \ldots, r_n[G_n, H_n] \rangle$. Покажем, что этого достаточно для формирования ленивого бесконечного дерева. Для этого, начиная с вершины v_{top}, производится последовательное разворачивание вершин дерева. На каждом очередном шаге происходит выбор листовой вершины для «разворачивания», что моделирует процесс «кликанья» мышью по этой вершине пользователем. Процесс «разворачивания» состоит в последовательном применении к вершине всех правил $r_1, \ldots, r_n$. При этом последовательность подвершин разворачиваемого узла v^o, соответствующих правилу r_i, вычисляется с помощью $F_i(o) =$ if $G_i(o)$ then $H_i(o)$ else $\emptyset$. Результатом применения всех правил тогда будет

$$F(o) = F_1(o) \oplus \ldots \oplus F_n(o),$$

где $\oplus$ обозначает конкатенацию последовательностей. Очевидно, что при наличии дерева $\mathbb{T}$ и вершины v^o в нем, функция F однозначно определяет переход от $\mathbb{T}$ к дереву $\mathbb{T}'$, в котором разворачивается вершина v^o. Такой переход обозначим $\mathbb{T}' = \mathbb{F}(\mathbb{T}, v^o)$.

Нажатием вершины v^o назовем результат применения правила:

$$\frac{v^o = click(V_T), \mathbb{T}}{\mathbb{T}'} \tag{4.1}$$

где $click(V_T)$ - функция, возвращающая вершину, к которой применяется операция «нажатие», и $\mathbb{T}' = \mathbb{F}(\mathbb{T}, v^o)$. Без ограничения общности будем считать, что функция $click(V_T)$ возвращает только листья дерева, поскольку легко проверить, что если вершина не является листом дерева, то к ней уже было применено правило 4.1. В этом случае $\mathbb{T}$ и $\mathbb{T}'$ совпадают.

С точки зрения построения интерфейсов ключевым свойством является индеферентность процедуры построения дерева к последовательности вершин, «нажатых» пользователем. Другими словами, неважно, в какой последовательности пользователь нажимал вершины: если в результате были нажаты одни и те же вершины, то и получившиеся деревья будут одинаковыми. Доказательство данного факта для деревьев, построенных с помощью правила (4.1), требует определенных рассмотрений.

5. Изоморфизм деревьев

В этом пункте будет показано, что два дерева, построение которых начинается с «одинаковых» вершин, и у которых, хоть и в разной последовательности, но нажимались «одни и те же» вершины, всегда «совпадают». Формально это определяется следующим образом.

Определение 3. [Изоморфизм деревьев] Пусть $\mathbb{T}$ и $\mathbb{T}'$ – два дерева, определенных в алфавите V. Деревья $\mathbb{T}$ и $\mathbb{T}'$ изоморфны, если существует взаимооднозначное отображение ϕ с множества V_T на V'_T такое, что для любых v, v_i и v_j:
1. $\lambda(v) = \lambda'(\phi(v))$;
2. $\langle v_i, v_j\rangle \in \mathbb{T} \Leftrightarrow \langle \phi(v_i), \phi(v_j)\rangle \in \mathbb{T}'$;
3. $v_i < v_j \Leftrightarrow \phi(v_i) < \phi(v_j)$.

Определение изоморфизма уточняет, что мы понимаем под «совпадающими» деревьями.

Определение 4. [Изначальное дерево] Дерево $\mathbb{T}^0 = \langle V_T^0, T^0, \lambda^0, <\rangle$ будем называть изначальным, если оно состоит из одной вершины v_{top}^o и $V_T^0 = \{v_{top}^o\}, T^0 = \emptyset, \lambda^0 = \{v_{top} \to o\}$ для некоторого $o \in \mathcal{O}$.

Очевидно, что два изначальных дерева $\mathbb{T}^0$ и $\mathbb{T}'^0$ изоморфны тогда и только тогда, когда для их единственных вершин v_{top} и v'_{top} выполняется $\lambda(v_{top}) = \lambda'(v'_{top})$.

Назовем последовательностью нажатий $S = \langle v_1^{o_1}, \ldots, v_k^{o_k}\rangle$ вывод дерева, начинающийся с некоторого изначального дерева и состоящий из применений правила (4.1) такой, что на i-м шаге функция $click(V_T)$ возвращается вершина $v_i^{o_i}$. Чтобы не перегружать запись в дальнейшем в последовательностях нажатий метки вершин o_i будут опускаться (они однозначно восстанавливаются с помощью λ). Напоминаем, что, по условию применимости правила (4.1), вершина v_i должна быть листом в построенном к i-ому шагу дереве. Будем говорить, что последовательности нажатий $S = \langle v_1, \ldots, v_k\rangle$ и $S' = \langle w_1, \ldots, w_k\rangle$ приводят к одинаковым деревьям, если существует взаимоодназначное отображение между $\{v_1, \ldots, v_k\}$ и $\{w_1, \ldots, w_k\}$, которое индуцирует изоморфизм деревьев, получившихся в результате применения S и S'.

Для того, чтобы более полно охарактеризовать ситуацию, введем понятие канонической последовательности нажатий, которая интуитивно соответствует обходу дерева «в глубину слева направо». Для этого определим линейный порядок на вершинах дерева, который индуцируется частичным порядком подвершин.

Определение 5. Считаем, что $v \ll w$ если
1. v и w являются подвершинами одной вершины и $v < w$, либо

2. v является надвершиной w, либо
3. существует надвершина v' вершины v и надвершина w' вершины w такие, что v' и w' являются непосредственными подвершинами одной вершины и $v' < w'$.

Определение 6. Последовательность нажатий $S = \langle v_1, \ldots, v_k \rangle$ называется *канонической*, если для любых $0 \leq i, j \leq k$, неравенство $i < j$ влечет неравенство $v_i \ll v_j$.

Очевидным образом из любой последовательности нажатий $S = \langle v_1, \ldots, v_k \rangle$ можно получить каноническую последовательность $S_{con} = \{v_1^*, \ldots, v_k^*\}$, переупорядочив v_i по возрастанию в соответствии с порядком $\ll$. Заметим, что вершина v_{top} всегда является наименьшей в соответствии со свойствами порядка $\ll$. Следующая лемма делает каноническое упорядочивание корректным:

Лемма 1. *Для любой последовательности S деревья, порожденные S и S_{con} изоморфны.*

Доказательство строится рассмотрением утверждения для произвольного k. Если $k = 0$, то результат тривиально верен. Для $k > 0$ доказательство проводится индукцией по первому рассогласованному нажатию i, когда $v_i \neq v_i^*$. Укажем алгоритм, позволяющий нам преодолеть рассогласованность последовательностей:

Алгоритм.

Шаг 0. $i := 1$
Шаг 1. Если $v_i = v_i^*$, тогда перейти на шаг 2, иначе перейти на шаг 3.
Шаг 2. $i := i + 1$. Если $i > k$, СТОП! иначе перейти на шаг 1.
Шаг 3. Найти j такое, что $v_i^* = v_j$.
Шаг 4. $S := \langle v_1, \ldots, v_{i-1}, v_j, v_i, \ldots, v_{j-1}, v_{j+1}, \ldots, v_k \rangle$. Перейти на шаг 2.

База индукции очевидна. Пусть элементы последовательности совпадают для каждого i, т.е. $v_i = v_i^*, i = \overline{1,k}$. Тогда можно построить тождественный изоморфизм

$$\iota(v_i) = v_i^*.$$

Очевидно, что так как набор правил совпадает, то последовательное применение к одинаковым вершинам приводит к одинаковым деревьям.

Шаг индукции. Пусть доказано для $n - 1$ рассогласований, что с помощью последовательности S получается дерево $\mathbb{T}$ изоморфное дереву $\mathbb{T}'$, полученного с помощью S^*. Тогда достаточно доказать что шаг 4 сохраняет изоморфность деревьев. Очевидно, мы вправе сделать такую перестановку, сохранив допустимость последовательности, т.к. из допустимости последовательности в канонической форме следует

$v_j = v_i^* \ll v_i, v_j \ll v_{i+1}, \ldots, v_j \ll v_{j-1}$. Т.е. переставляемая вершина v_j не зависит от вершин $v_{i+1}, \ldots, v_{j-1}$. Тогда порядок применения правил к вершине v_j не влияет на изоморфизм деревьев. Очевидно, что при такой перестановке сохраняется и порядок дочерних подвершин, т.к. правило «нажатие» детерминировано меткой узла. Значит применение правила к одному и тому же узлу приводит к получению одинаковых пар инцидентности $\langle v_j, v \rangle$, с точностью до метки, вне зависимости от очередности в последовательности нажатий S. □

Из данной леммы напрямую следует теорема, обосновывающая тот факт, что разный порядок нажатий не влияет не структуру получившегося дерева:

Теорема 1. *Пусть последовательности нажатий S и S' обладают каноническими последовательностями S_{con} и S'_{con}, соответственно, причем S_{con} и S'_{con} порождают изоморфные деревья. Тогда S и S' также порождают изоморфные деревья.*

Доказательство теоремы 1 следует из коммутативности следующей тривиальной диаграммы:

$$\begin{array}{ccc} \mathbb{T} & \xrightarrow{\phi} & \mathbb{T}' \\ \iota \downarrow & & \uparrow \iota'^{-1} \\ \mathbb{T}^* & \xrightarrow{\phi} & \mathbb{T}'^* \end{array}$$

Здесь ι и ι' – тождественные изоморфизмы между деревьями, порождаемыми последовательностями и их каноническими вариантами, а ϕ – изоморфизм между деревьями, полученными с помощью канонических последовательностей. Таким образом изоморфизм переводящий из $\mathbb{T}$ в $\mathbb{T}'$ будет иметь следующий вид: $\iota\phi(\iota')^{-1} = \phi$. □

6. Примеры

Рассмотрим структуру правил. Для начала возьмем один из простейших видов правил, где переход от родительского узла к дочернему происходит по роли из множества $\mathcal{R}$ словаря $\mathcal{W}$, то есть имеет вид $r_R[G_R, H_R]$, где

$$G_R(o) \equiv o \in Dom(R)$$
$$H_R(o) = \{o_i : R(o, o_i)\}$$

Пример.

$$G_{hasChild}(o) = o \in Person, H_{hasChild}(o) = \{o_i : hasChild(o, o_i)\}$$

Тогда для построения дерева всех потомков некоторого человека, мы указываем его в качестве вершины дерева и в качестве множества правил – правило $[G_{hasChild}, H_{hasChild}]$.

Тогда при нажатии вершины некоторой персоны откроются вершины его детей.

Очень важной особенностью данного подхода является ленивый характер построения дерева. То есть дерево не строится сразу полностью, а оно постепенно «вырастает» путем «подгрузки» данных из базы знаний. Причем рост дерева происходит только в тех направлениях, которые нужны пользователю. Во-первых, это сокращает объем памяти, необходимый для хранения дерева. Во-вторых, это позволяет работать с бесконечными деревьями. Как упоминалось выше, преобразованием семантической сети онтологии в дерево может привести к появлению бесконечных деревьев ввиду циклов. И тогда построение полного бесконечного дерева невозможно. В нашем же подходе мы имеем актуально конечное состояние бесконечного дерева.

В качестве примера можно привести роль «родственник».

$$G_{hasRelative} = C(o) \subseteq Person, H_{hasRelative} = \{o_i : hasRelative(o, o_i)\}$$

Если A является родственником B, то и B является родственником A. И применяя это правило, даже для двух человек получится бесконечное дерево. В нем переход от родительского узла к дочернему может потенциально происходить сколь угодно долго. Но реально оно ограничено количеством кликов мыши пользователем.

Очевидно, что общий вид правила позволяет нам использовать всю мощь языка OntoBox QL как в качестве процедур получения объектов H, так и для проверки предусловия G. Так как язык запросов OntoBox QL в общем случае возвращает коллекцию, предусловием правила может послужить проверка не пустоты результата.

$$r_{ql} = \langle G_{ql}(o), H_{ql}(o) \rangle, G_{ql}(o) = \exists H_{ql}(o), H_{ql}(o) = QL(o)$$

где $QL(o)$ - запрос на языке OntoBox QL, первым шагом пути которого является o.

В частности для описанного выше примера с потомками запрос будет выглядеть следующим образом:

$$o/hasChild$$

где o - полное имя объекта онтологии.

7. Заключение

Рассмотренный метод построения бесконечных ленивых маркированных деревьев был реализован в системе META-2, которая является клиентской программой для сервера управления онтологиями Ontobox. Механизм построения деревьев стал играть ключевую роль при работе с элементами семантической сети. Гибкость построения деревьев на основе правил, в том числе с использованием языка OntoBox QL, показала свою эффективность на ряде проектов (реализованных на базе системы) для различных междисциплинарных исследований озера Байкал, задачи формализации спецификаций, а также для создания интеллектуализированной системы видео-лекций.

Автор благодарен А.В. Манциводе за поддержку в проведении данного исследования.

Список литературы

1. Baader, F. The Description Logic Handbook: Theory, Implementation, Applications / F. Baader, D. Calvanese , D.L. McGuinness, D. Nardi, P.F. Patel-Schneider. – Cambridge. – 2003. – P.574.
2. Schmidt-Schauss, M. Attributive concept descriptions with complements / M. Schmidt-Schauss, G. Smolka // Artificial Intelligence. – 1991. – V.48. – P.1–26.
3. Малых, А.А. Логические архитектуры и объектно-ориентированный подход / А.А. Малых, А.В. Манцивода, В.С. Ульянов // Вестник НГУ. Серия: Математика, механика, информатика. – 2009. – Т9. – №3. – С. 64-85.
4. The Semantic Web [Электронный ресурс]. – Режим доступа: http://www.w3.org/2001/sw/ – Загл. с экрана.
5. The NCBI Entrez Taxonomy. [Электронный ресурс]. – Режим доступа: http://www.ncbi.nlm.nih.gov/sites/entrez?db=taxonomy – Загл. с экрана.
6. Web Ontology Language (OWL). [Электронный ресурс]. – Режим доступа: www.w3.org/2004/OWL – Загл. с экрана.
7. Malykh, A. A Query Language for Logic Architectures. / A. Malykh, A. Mantsivoda // Accepted for PSI'09. – [Электронный ресурс]. – Режим доступа: http://meta2project.org/ru/psi09.pdf – Загл. с экрана.
8. Berners-Lee, T. The Semantic Web / T. Berners-Lee, J. Hendler ,O. Lassila. // Scientific American. – 2001. – V.5. – P.34-43.

V. S. Ulyanov

Infinite Lazy Marked Trees

Abstract. This paper is focused on the construction of interfaces to knowledge description systems based on ontologies. The 'elite' nature of logic, on which knowledge management systems are based, complicates the development of interfaces for the mass of users. In this paper an approach is proposed, which is based on the concept of infinite lazy marked trees. Soundness and implementation issues of the approach are also considered.

Keywords: ontologies, Ontobox, trees, semantic web

Ульянов Владимир Сергеевич, Институт математики,экономики и информатики, Иркутский государственный университет, 664003, Иркутск, ул. К. Маркса, 1 тел.: (3952)242210 (ulyanov@baikal.ru)

Vladimir Ulyanov, Irkutsk State University, 1, K. Marks St., Irkutsk, 664003 Phone: (3952)242210 (ulyanov@baikal.ru)