Санкт-Петербургский государственный университет

Направление Фундаментальная информатика и информационные технологии

Петров Дмитрий Андреевич

Разработка и реализация алгоритмов для построения многоуровневого описания классов при создании самокорректирующейся сети

Бакалаврская работа

Научный руководитель:
д. ф.-м. н., профессор Косовская Т. М.

Рецензент:
ст. преп. Петухова Нина Дмитриевна

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Fundamental Computer Science and Information Technology

Dmitrii Petrov

Development and implementation of algorithms for construction of multi-level class description in creating self-modificated network

Graduation Thesis

Scientific supervisor:
professor Tatiana Kosovskaya

Reviewer:
assistant Petukhova Nina Dmitrievna

Saint-Petersburg
2016

Оглавление

Введение

0.1. Описание задачи

Инструменты и средства математической логики [1, 2] довольно часто используются [3] при решении различных задач искусственного интелекта. В частности, таких как распознавание образов. Примерами подобных задач являются медицинская диагностика и выбор лечения (входными данными в этом случае являются состояние внутренних органов пациента и их текущий уровень и характер взаимодействия друг с другом), анализ рынка (входные данные представлены описанием состояния всех элементов рынка, таких как фирмы, предприятия, банки и т.п., и их относительная кооперация) или распознавание изображений (изображение и различные его характеристики являются входными данными в этом случае).

Рассматриваемые объекты часто представляют из себя множество составляющих его элементов, которые характеризуются своими свойствами и отношениями друг с другом. В этом случае удобно применение языка исчисления предикатов.

В [4] рассмотрены два подхода использующие инструменты математической логики для описания частей объектов и отношений между ними: язык логики высказываний и язык логики предикатов первого порядка. В этой работе было сказано, что язык логики высказываний может быть применен для решения вышеописанной задачи, но, т.к. логика высказываний довольно проста и не обладает достаточной выразительной мощью, то полученное описание классов объектов будет весьма громоздким (число пропозициональных переменных экспоненциально зависит от количества временных этапов, схем действий, объектов и арности произвольной схемы действий), что делает ее неприменимой для практического использования. Вместо этого используется язык предикатов первого порядка. Предикаты используются для описания свойств элементов объекта или отношения между элементами объекта. Их аргументами являются элементы объекта соответственно. Набор постоян-

ных атомарных формул представляет собой описание объекта, точнее описанием класса объектов является дизъюнкция элементарных конъюнкций атомарных формул, истинная для объектов этого класса. При введеном описании можно решать данную задачу с помощью алгоритмов поиска логического вывода в исчислении предикатов, либо, используя алгоритмы, основанные на методе полного перебора.

0.2. Используемый подход и методы решения

В [5] было доказано, что указанная задача относится к классу NP-трудных. Ввиду сложности рассматриваемой задачи для уменьшения числа шагов работы алгоритмов, решающих данную задачу в [6] было предложено многоуровневое описание классов. Процесс решения состоит в проверке выводимости в исчислении предикатов формулы $S(\omega) \Rightarrow \exists \overline{x}_{\neq} A_k(\overline{x})$, где $S(\omega)$ представляет собой множество постоянных литералов, а $A_k(\overline{x})$ — конъюнкция литералов с переменными. В случае, если формула выводима, алгоритмом, решающим данную задачу, предоставляется набор предметных переменных $\overline{x}$ и соответствующая подстановка. Более формально описание данной задачи будет дано в последующих разделах. Многоуровневое описание классов позволяет разбить процесс решения данной задачи на несколько этапов, в каждом из которых будет решаться задача такого же вида, но размерность входных данных будет существенно ниже. Благодаря тому, что количество уровней (и, соответственно, этапов решения) в описании линейно зависит от размера исходных формул, а алгоритмы решающие задачу выводимости имеют экспоненциальные оценки от размера входных данных, достигается существенное снижение числа шагов работы алгоритма, решающего рассматриваемую задачу.

Однако при построении многоуровнего описания решается задача выделения достаточно "простой" подформулы [6] двух формул языка предикатов первого порядка. В частности, в работе было показано, что нахождение данной подформулы может основываться на выделении наибольшей общей подформулы с точностью до имен аргументов, что

является частным случаем задачи частичной выводимости предикатных формул. В [7] приводится утверждение, что данная задача является NP-трудной. Таким образом, несмотря на то, что многоуровневое описание позволяет существенно уменьшить число шагов работы вышеописанных алгоритмов, решение задачи построения самого описания весьма трудоемко.

Идея автоматического построения многоуровневого описания позволила ввести понятие самокорректирующейся (логико-предикатной) сети [8]. Самокорректирующаяся сеть состоит из двух блоков: обучения и распознавания. В блоке обучения решается задача построения и перестроения многоуровнего описания классов. В распознающем блоке осуществляется непосредственно проверка выводимости данной формулы. Таким образом самая трудоемкая часть решения задачи отводится на предварительный этап работы сети. Предполагается, что для решения задач распознавания в определенной области экспертом будет проведена предварительная работа по обучению сети, накоплению знаний о типах классифицируемых объектов, которая в этом случае будет выражена в инициализации многоуровнего описания классов. В таком случае допустимо, что построение описания классов может занимать длительное время. В то время как, время работы разпознающего блока на обученной сети будет существенно меньше.

Ключевым элементом построения многоуровнего описания классов является алгоритм выделения наибольшей общей подформулы. Число шагов его работы оказывает основное влияние на временну́ю сложность работы алгоритма построения многоуровнего описания классов. Ранее [5, 7] были представлены подходы к решению данной задачи.

В рамках данной выпускной квалификационной работы был разработан алгоритм, основанный на методе полного перебора с использованием метода ветвей и границ. Причинами для выбора в качестве основания для разработки алгоритма метода полного перебора являются меньшие показатели для сомножителей в оценке числа шагов работы алгоритма по сравнению с методом резолюций, а также простота его реализации в случае распараллеливания. Представленный ранее алго-

ритм, основанный на методе полного перебора, включал в себя перебор всевозможных подформул входной формулы по множеству предикатов (таким образом перебирались всевозможные подмножества множества предикатов), и для каждого такого набора осуществлялся перебор множества предметных переменных для поиска нужной подстановки. В алгоритме, представленном в данной работе, перебор осуществляется только по множеству предикатов: на каждом шаге фиксируется предикат, который должен графически совпасть с одним из предикатов начальной формулы, тем самым определяя подстановку для некоторого подмножества множества переменных начальной формулы, далее, отбрасывая конфликтующие с данной подстановкой предикаты и запоминая предикаты, которые могли графически совпасть при текущей подстановке, фиксируется следующий предикат. Данные шаги повторяются до тех пора пока не будет получена подстановка для всех переменных, либо размер множества предикатов для дальнейшего перебора не станет слишком мал. Более формально данный алгоритм будет описан в последующих разделах.

Также немаловажной частью многоуровнего описания классов является алгоритм проверки равенства двух формул языка предикатов с точностью до имен аргументов. Во время построения многоуровнего описания среди набора общих подформул выделенных для формирования следующего уровня описания весьма часто встречаются формулы равные (с точностью до имен аргументов). Важным моментом здесь является удаление описанных дубликатов, т.к. они не представляют ценность в процессе логического вывода и значительно увеличивают размерность многоуровнего описания. Использование вышеописанных идей предполагает повторный запуск алгоритмов, решающих другую, более трудоемкую задачу, в результате чего заметно увеличивается число шагов работы алгоритма построения рассматриваемого описания. В рамках данной работы был разработан алгоритм позволяющий снизить временные оценки работы алгоритмов при решении описанной задачи. Алгоритм основан на идее приведения формул к специальному виду, в котором проверка равенства формул с точностью до имен аргумен-

тов значительно упрощается. Более формально данный алгоритм будет описан в последующих разделах.

0.3. Актуальность рассматриваемой задачи и другие подходы к её решению

В наши дни задача распознавания образов встает все чаще в различных областях исследований и практических применений [9]. Одной из основных причин является рост вычислительной мощности компьютеров за последние годы. Ввиду NP-трудности рассматриваемой задачи актуальна разработка методов и алгоритмов, позволяющих уменьшить оценку числа шагов решения задачи.

Примерами задач распознавания образов являются:

- оптическое распознавание символов
- распознавание автомобильных номеров
- распознавание изображений (в частности, лица, этот тип очень часто используется в современных фотоаппаратах и смартфонах)
- распознавание речи
- классификация документов основанная на определенной характеристики текстов (эмоциональный настрой автора, отнесение содержания текста к определенной области)
- анализ различных физиологических показаний человека (пульс, температура, длительность физической активности) для составления рекомендаций, связанных с здоровьем пользователя (используется в фитнес-браслетах и других носимых гаджетах)
- различные задачи распознавания сигналов, часто встречаемые в военном деле и информационной безопасности.

Сюда же относятся вышеупомянутые задачи медицинской диагностики пациента и анализа рынка.

Существуют другие подходы к решению данной задачи. Последнее время широкое распространение получили алгоритмы, основанные на модели искусственных нейронных сетей(ИНС) [10]. В основе идеи искусственных нейронных сетей лежит представление о работе человеческого мозга. ИНС представляет собой множество независимых простых вычислительных устройств (может быть реализовано программно или аппаратно), являющихся аналогами нейронов в мозге человека. Искусственные нейроны обмениваются информацией посредством специальных сигналов. Довольно часто находят свое применение ИНС, состоящие из нескольких уровней. В ходе работы указанных ИНС сигнал переходит от уровня к уровню, подвергаясь соотвествующим преобразованиям. Несмотря на довольно простое устройство одного нейрона, совокупность таких нейронов, объединенная в сеть, обладает достаточно большой вычислительной мощностью. В процессе функционирования ИНС выделяют этап обучения и этап работы. На этапе обучения устанавливаются параметры (веса) для сигналов, передаваемых между нейронами, в зависимости от корректности результата работы сети. Данные параметры используются искуственной нейронной сетью на этапе работы для формирования выходного сигнала, который является результатом ее функционирования. ИНС получили довольно широкое распространение и успешно используются при решении множества задач искусственного интелекта, однако ИНС имеет фиксированную структуру и нейроны реализуют признаки, характеризующие объект в целом.

0.4. Реализация

В рамках данной выпускной квалификационной работы была реализована логико-предикатная (самокорректирующаяся) сеть. В частности, были реализованы алгоритм, основанный на методе полного перебора, и модификация этого алгоритма с использованием метода ветвей

и границ. Алгоритм полного перебора был реализован для демонстрации сравнительной эффективности использования метода ветвей и границ. Также был реализован алгоритм проверки равенства предикатных формул для получения более эффективной реализации алгоритма построения многоуровнего описания классов. Реализация была осуществлена на языке программирования C++ [11].

1. Постановка задачи

1.1. Логико-предметный подход к решению задачи

В [5] было дано формальное описание задачи:

Пусть исследуемый объект представлен как множество своих элементов $\omega = \{\omega_1, \ldots, \omega_t\}$. На ω задан набор предикатов $p_1, \ldots, p_n$, характеризующих свойства элементов ω и отношения между ними. Логическим описанием $S(\omega)$ объекта ω называется множество всех атомарных формул или их отрицаний, истинных на ω. Множество всех объектов разбито на классы $\Omega = \cup_{k=1}^{K} \Omega_k$. Логическим описанием класса называется формула $A_k(\overline{x})$ [1], заданная в виде дизъюнкции элементарных конъюнкций, такая что если $A_k(\overline{\omega})$ истинная, то $\omega \in \Omega_k$.

С помощью построенных описаний объектов и классов предлагается решать следующие задачи:

- *Задача идентификации.* Проверить, удовлетворяет ли объект ω или его часть описанию класса $A_k(\overline{x})$ и предъявить эту часть объекта.

- *Задача классификации.* Найти все такие номера k, что верна формула $A_k(\overline{\omega})$.

- *Задача анализа.* Найти и классифицировать все части τ объекта ω, для которых $A_k(\overline{\tau})$.

Решение задач идентификации, классификации и анализа для распознавания сложного объекта сведено к доказательству соответственно

[1] Здесь и далее с помощью $\overline{x}$ обозначается список элементов конечного множества x, соответствующий некоторой перестановке номеров его элементов. Тот факт, что элементами списка $\overline{x}$ являются элементы множества y, будем записывать в виде $x \subseteq y$.

логических следований:[2]

$$S(\omega) \Rightarrow \exists \overline{x}_{\neq} A_k(\overline{x}), \tag{3}$$

$$S(\omega) \Rightarrow \bigvee_{k=1}^{M} A_k(\overline{\omega}), \tag{4}$$

$$S(\omega) \Rightarrow \bigvee_{k=1}^{M} \exists \overline{x}_{\neq}\ A_k(\overline{x}). \tag{5}$$

Строго говоря, вместо (3), (4), (5) следовало бы писать соответственно

$$S(\omega) \Rightarrow (?\overline{x}_{\neq})\ A_k(\overline{x}), \tag{6}$$

$$S(\omega) \Rightarrow (?k_{k=1}^{M})\ A_k(\overline{\omega}), \tag{7}$$

$$S(\omega) \Rightarrow (?k_{k=1}^{M})(?\overline{x}_{\neq})\ A_k(\overline{x}), \tag{8}$$

но рассматриваемые алгоритмы доказательства логических следований не только отвечают на вопрос «*существует ли ...?*», но и предъявляют значения для переменных.

Например, при решении задач медицинской диагностики в качестве множеств ω выступают пациенты, рассматриваемые как множество своих частей (органов), а исходные предикаты p_i — свойства этих частей и отношения между ними, обычно называемые симптомами. В этом случае классы объектов — это классы заболеваний или синдромы. Решение задач идентификации, классификации и анализа в этом случае сводится к доказательству тех же самых формул и соответствует таким зада-

[2]Для того, чтобы записать, что значение для переменных списка $\overline{x}$, удовлетворяющие формуле $A(\overline{x})$, различны, вместо формулы

$$\exists x_1 \ldots \exists x_m (\wedge_{i=1}^{m} \wedge_{j=i+1}^{m} (x_i \neq x_j) \wedge A(x_1, \ldots, x_m)) \tag{1}$$

будет использоваться обозначение

$$\exists \overline{x}_{\neq} A(\overline{x}) \tag{2}$$

чам как «обладает ли пациент заданным заболеванием и какие именно симптомы каких органов влекут это заболевание?», «какими заболеваниями страдает заданный орган?», «какие области пациента какими заболеваниями страдают?».

Заметим, что для того, чтобы уметь доказывать $(3), (4), (5)$, достаточно уметь доказывать логическое следование

$$S(\omega) \Rightarrow \exists \overline{x}_{\neq} A(\overline{x}), \tag{9}$$

где $A(\overline{x})$ — элементарная коъюнкция атомарных формул и их отрицаний.

В [5] была доказана NP–полнота задач $(3), (4), (5)$ и, следовательно, NP–трудность задач $(6), (7), (8)$.

1.2. Многоуровневое описание классов

В [6] было предложено иерархическое многоуровневое описание классов распознаваемых объектов. Рассматривается набор описаний классов объектов $A_k(\overline{x})$. Данный набор представляет собой 0-уровень многоуровнего описания. Попарно выделяются общие подформулы $P_i^1(\overline{y})$ описаний $A_k(\overline{x})$. При этом вводится система эквивалентностей вида $p_i^1(\overline{y}^1) \leftrightarrow P_i^1(\overline{y})$, где p_i^1 новые предикаты, предикаты 1-го уровня, а y^1 — новые переменные для списков исходных переменных, переменные 1-го уровня. В полученном наборе описаний исключаются формулы равные с точностью до имен аргументов. Через $A_k^1(\overline{x}^1)$ обозначаются формулы, полученные из $A_k(\overline{x})$ с помощью замены всех вхождений формул вида $P_i^1(\overline{y}_i^1)$ на атомарные формулы $p_i^1(x_i^1),\ y_i^1 \subseteq x$. Здесь $\overline{x}^1$ — список всех переменных формулы $A_k^1(\overline{x}^1)$, состоящий как из некоторых исходных переменных формулы $A_k(\overline{x})$, так и из переменных первого уровня, появившихся в формуле $A_k^1(\overline{x}^1)$. Таким образом формируется новый уровень и процесс продолжается уже относительно данного набора. В результате будет получено $L-$уровневое описание, где на последнем уровне будут содержаться формулы, не имеющие общих подформул, и каждая из данных формул будет являться некоторой подформулой

формулы из начального набора описаний классов, с учетом вышеописанных подстановок.

Таким образом исходная система описаний классов может быть записана с помощью равносильной многоуровневой системы описаний классов вида

$$\begin{cases} A_k^L(\overline{x}^L) \\ p_1^1(x_1^1) \Leftrightarrow P_1^1(\overline{y}_1^1) \\ \vdots \\ p_{n_1}^1(x_{n_1}^1) \Leftrightarrow P_{n_1}^1(\overline{y}_{n_1}^1) \\ \vdots \\ p_i^l(x_i^l) \Leftrightarrow P_i^l(\overline{y}_i^l) \\ \vdots \\ p_{n_L}^L(x_{n_L}^L) \Leftrightarrow P_{n_L}^L(\overline{y}_{n_L}^L) \end{cases}$$

Описанием объекта $S^1(\omega)$ первого уровня называется множество всех атомарных формул вида $p_i^1(\omega_{ij}^1)$, для которых истинна определяющая подформула $P_i^1(\overline{\tau}_{ij}^1),\ \tau_{ij}^1 \subset \omega$, при этом объект первого уровня ω_{ij}^1 представляет из себя список исходных объектов τ_{ij}^1. Аналогично определяется описание объекта для других уровней.

Алгоритм многоуровнего распознавания:

Решение задачи распознавания может быть разбито на $L+1$ этапов, в каждом из которых осуществляется проверка выводимости из $S(\omega) \cup S^1(\omega) \cup \ldots \cup S^{l-1}(\omega)$ формулы $\exists \overline{x}_{j\neq}^l P_j^l(\overline{x}_j^l)$ и нахождение всех таких объектов $l-$го уровня, существование которых утверждается в правой части логического следования.

1.3. Неполная выводимость предикатных формул

В [7] введено понятие (q,r)–фрагмента и неполной выводимости двух предикатных формул, а также с помощью них дано понятие наибольшей общей подформулы.

Пусть $A(\overline{x})$ и $B(\overline{y})$ – две элементарные конъюнкции предикатных

формул со списками предметных переменных $\overline{x}$ и $\overline{y}$ соответственно. Т.е. $A(\overline{x})$ и $B(\overline{y})$ имеют вид

$$L_1 \ \& \ L_2 \ \& \ldots \& \ L_s, \tag{10}$$

где $L_i = p_i(\overline{z_i})$ или $\neg p_i(\overline{z_i})$ Проверка неполной выводимости $A(\overline{x}) \Rightarrow_P \exists \overline{y}_{\neq} \ B(\overline{y})$ заключается в нахождении такой подформулы $Q_{AB}(\overline{z})$ формулы $B(\overline{y})$ и такой подстановки $\lambda_{AQ} = |^z_{y'}$ списка переменных $\overline{y'}$ из $\overline{y}$ вместо переменных списка $\overline{z}$, что $Q_{AB}(\overline{y'})$ является максимальной подформулой формулы $B(\overline{y})$, такой что $A(\overline{x}) \Rightarrow \exists \overline{y'}_{\neq} \ Q_{AB}(\overline{y'})$.

Аналогично определяется (q, r)-фрагмент $Q_{BA}(\overline{x'})$в случае $B(\overline{y}) \Rightarrow_P \exists \overline{x}_{\neq} \ A(\overline{x})$. Ясно, что $Q_{AB}(\overline{y'})$ и $Q_{BA}(\overline{x'})$ совпадают с точностью до имён переменных. Таким образом, в качестве максимальной можно брать любую из этих формул. Говоря неформально: наибольшая общая подформула с точностью до имён переменных, это формула, являющаяся максимальной по числу литералов, полученная при помощи некоторой подстановки переменных формул $A(\overline{x})$ и $B(\overline{y})$ при которой графически совпало некоторое количество литералов данных формул.

1.4. Самокорректирующаяся сеть

Самокорректирующаяся сеть состоит из двух блоков: обучающего и распознающего. В ходе работы обучающего блока формируется рассмотренное выше описание классов. Предполагается, что на вход сети в процессе обучения будет дана выборка, содержащая описания требуемых классов объектов. Таким образом, на данном этапе будет проведена самая трудоемкая часть процесса решения поставленной задачи.

В процессе распознавания используется только распознающий блок. Его работа заключается в выполнении алгоритма многоуровнего распознавания. В случае обнаружения неправильного распознавания (в данной сети не содержалось полной информации о классе объектов) возможно дообучение сети, которое заключается в перестроении обучающего блока посредством добавления новой формулы к $0-$уровню многоуровнего описания классов и запуска процесса выделения наи-

больших общих подформул в соответствии с алгоритмом построения многоуровнего описания классов. Схема работы логико-предикатной сети представлена на рис. 1.

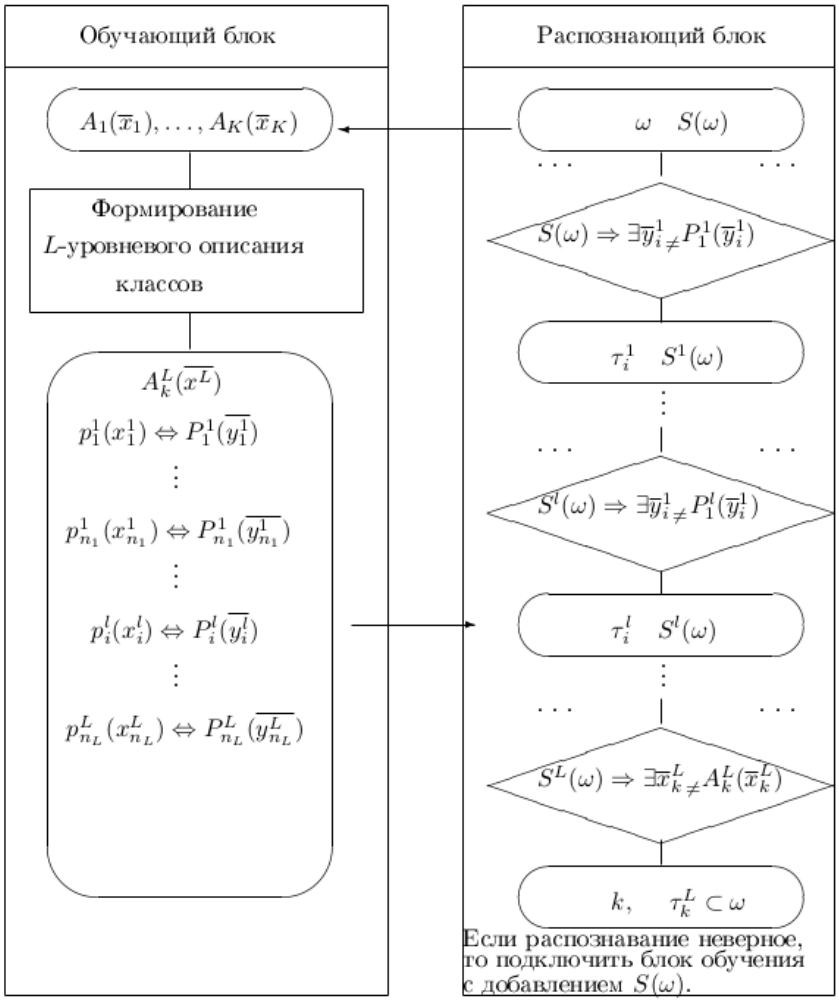

Рис. 1. Логико-предикатная сеть.

2. Описание решения поставленной задачи

2.1. Алгоритм полного перебора литералов и переменных

В [7] был подробно описан алгоритм полного перебора. Ниже приведено описание данного алгоритма, незначительно измененного. Изменения обусловлены различием в постановке задачи поиска наибольшей общей подформулы и задачи распознавания объектов с неполной информацией. Не уменьшая общности рассмотрим случай $A(\overline{x}) \Rightarrow_P \exists \overline{y}_{\neq}\ B(\overline{y})$:

1. Исключим из рассмотрения те литералы формул $A(\overline{x})$ и $B(\overline{y})$, которые встречаются только в одной из формул с учетом возможных подстановок(например, $p_i(\overline{z_i})$ является частью формулы $B(\overline{y})$, но в $A(\overline{x})$ нет предиката с номером i) Получим формулы $A'(\overline{x'})$ и $B'(\overline{y'})$;

2. В качестве формулы $C(\overline{z})$ возьмём формулу $B'(\overline{y'})$. Обозначим через k количество литералов в формуле $C(\overline{z})$;

3. Для каждого из C_n^k наборов литералов, где n равно числу литералов в формуле $B'(\overline{y'})$, сформируем соответствующую формулу $C(\overline{z})$ с помощью конъюнкции текущего набора литералов.

4. Для каждого из A_t^m наборов переменных, где $m = |y'|$, $t = |x'|$, подставив этот набор переменных в формулу $C(\overline{z})$, проверим наличие каждого литерала из $C(\overline{z})$ в $A'(\overline{x'})$;

5. Если для некоторого набора переменных каждый из литералов $C(\overline{z})$ содержится в $A'(\overline{x'})$, то подформула и подстановка найдены и алгоритм заканчивает свою работу.

6. Если $k = 0$, то формулы абсолютно различны и алгоритм заканчивает свою работу. В противном случае уменьшаем k на единицу. Переходим к пункту 3.

2.2. Алгоритм полного перебора литералов методом ветвей и границ

Вначале введём некоторые обозначения:

- Не уменьшая общности рассмотрим случай $A(\overline{x}) \Rightarrow_P \exists \overline{y}_{\neq}\ B(\overline{y})$. С целью уменьшить число ненужных шагов, будем считать, что эти формулы получены после выполнения пункта 1 предыдущего алгоритма. Для краткости эти формулы будем обозначать A и B соответственно;
- n — число литералов в формуле A;
- k — число литералов в формуле B;
- $C(\overline{z})$ - максимальная по числу литералов найденная подформула, на текущем этапе работы алгоритма, для краткости обозначим через C;
- r — число литералов в C;
- S — текущая частичная подстановка переменных из формулы A вместро переменных формулы B, т.е. $S = |_{x'}^{y'},\ x' \subset x,\ y' \subset y$;
- P — множество литералов, являющихся литералами формулы A, которые не выкинуты из рассмотрения на текущем шаге работы алгоритма;
- L — множество литералов, являющихся литералами формулы B, каждый из которых графически совпадает с каким-либо литералом формулы A при текущей частичной подстановке;
- L' — множество литералов, являющихся литералами формулы B, которые не выкинуты из рассмотрения на текущем шаге работы алгоритма, и, которые могут быть включены в L.

Алгоритм:

1. Формула C не инициализированна, подстановка S пуста, множество P совпадает с множеством литералов A, множество L' совпадает с множеством литералов B, множество L пусто;

2. При текущей подстановке S проверяем, какие литералы из L' заведомо не могут графически совпасть ни с одним литералом A (например, $p_i(y_1, y_2, y_3)\ |S = p_i(x_1, y_2, y_3)$, но в A нет ни одного литерала с предикатом p_i без отрицания и первым аргументом которого является переменная x_1). Такие литералы отбрасываются из множества L'. Также проверяем, какие литералы графически совпали с одним из литералов формулы A. Их перемещаем из множества L' в L;

3. При текущей подстановке S проверяем, какие литералы из P не могут быть выполнены (например, $p_i(x_1, x_2) \in P$, $S = |_{x_2}^{y_3}$, но в L' нет литералов с предикатом p_i без отрицания, где вторым аргументом не было ещё ничего подставлено или стояла x_2. Т.е. заведомо никакой литерал из L' не может графически совпасть с данным) и удаляем такие литералы из P;

4. Если количество литералов в L больше количества литералов в C, то конъюнкцию литералов из L записываем в C;

5. Если подстановка S полная, т.е. $S = |_{\overline{x}}^{\overline{y}}$, то переходим к п.8 (предыдущий шаг рекурсии);

6. Если $|L| + |L'| < r$, то переходим к п. 8 (предыдущий шаг рекурсии);

7. Каждый литерал из L' пытаемся последовательно унифицировать каждым литералом из P. Если унификация возможна, то делаем рекурсионный шаг — переходим к п.2, при этом удаляя выбранный литерал из P, добавляя к S переменные опредяемые данной подстановкой, также перемещаем литерал из L' в L если он графически совпал с одним из литералов из P;

8. Проверка аналогична п.6. Если все возможные подстановки перебраны, то переходим к п.8 (предыдущий шаг рекурсии; если его не было, алгоритм заканчивает работу). В противном случае продолжаем перебор (переходим к п.7).

В любой момент работы алгоритма глобальными являются только переменные A, B и C. Все остальные переменные являются локальными копиями текущего шага и влияют только на результат последующих шагов. Все действия в п.7 осуществляются над данными для следующего шага рекурсии. Выбор пары в п.7 непротиворечив для подстановки S.

2.3. Алгоритм проверки равенства предикатных формул с точностью до имен аргументов

Здесь и далее под равенством и неравенством предикатных формул будет подразумеваться равенство и неравенство предикатных формул с точностью до имен аргументов соответственно.

***Определение* 1.** *Характеристикой переменной v в формуле A для литерала L_i,* $\chi(v)^A_{L_i}$, называется упорядоченный набор $(e_1, e_2, ..., e_q)$, где q — количество переменных литерала L_i, где e_j обозначает количество атомарных формул с тем же предикатом, что и в литерале L_i, в которых переменная v является j-ым аргументом.

***Определение* 2.** *Полной характеристикой переменной v в формуле A,* $\chi(v)^A$ называется упорядоченный набор $(\chi(v)^A_{L_1}, ..., \chi(v)^A_{L_i}, ..., \chi(v)^A_{L_n})$, где L_i пробегает по всем литералам формулы A.

***Замечание* 1.** *Стоит заметить, что равенство полных характеристик переменных* $\chi_1(v)^A = (\chi_1(v)^A_{L_1}, ..., \chi_1(v)^A_{L_i}, ..., \chi_1(v)^A_{L_n})$ *и* $\chi_2(u)^A = (\chi_2(u)^A_{L'_1}, \ ... \ , \chi_2(u)^A_{L'_i}, ..., \chi_2(u)^A_{L'_n})$ *влечет за собой равенство множеств литеральных символов* $\{L_1, ..., L_n\}$ *и* $\{L'_1, ..., L'_{n'}\}$.

***Определение* 3.** *Множеством характеристик формулы A,* $\chi(A)$ называется мультимножество $\{\ \chi(v_k)^A\ \}$, где v_k пробегает множество всех переменных формулы A.

Обозначения:

- исходные формулы $A(\overline{x})$ и $B(\overline{y})$ соответственно;
- k_A и k_B — количество различных предметных переменных в формулах соответственно;
- n_A и n_B — количество литералов в формулах соответственно;
- l_A и l_B — количество различных предикатных символов в формулах соответственно.

Алгоритм:

1. Если $k_A \neq k_B$, или $n_A \neq n_B$, или $l_A \neq l_B$, то алгоритм возвращает отрицательный ответ, иначе п.2;
2. Если $\chi(A) \neq \chi(B)$, то алгоритм возвращает отрицательный ответ, в противном случае алгоритм возвращает положительный ответ.

Важно отметить, что алгоритм не является абсолютно точным. В большинстве случаев положительный результат работы алгоритма означает равенство формул. В следующем разделе будут приведены статистические оценки этого факта. В случае возврата отрицательного ответа результат работы алгоритма абсолютно верен.

***Теорема* 1.** *Если алгоритм проверки равенства предикатных формул возвращает отрицательный ответ, то формулы A и B не равны.*

Доказательство. Очевидно, что формулы не равны, в случае возврата отрицательного ответа при выполнении п.1 алгоритма. Рассмотрим возврат из п.2.

Предположим противное — формулы A и B равны. Неравенство множеств $\chi(A)$ и $\chi(B)$ означает, что в множестве $\chi(A)$ есть по крайней мере одна полная характеристика переменной (не уменьшая общности, обозначим ее через $\chi(x)^A$), которой нет в $\chi(B)$. С другой стороны, равенство формул A и B означает, что существует такая подстановка

переменных $S = |\frac{\overline{x}}{\overline{y}}$ и такая перестановка литералов в записи конъюнкции формулы A, что в результате формула A графически совпадет с формулой B. Следовательно, с помощью описанных операций была изменена полная характеристика $\chi(x)^A$. Однако данные операции не могут изменить число вхождений переменной в предикаты или символы предикатов, одним из аргументов которых является переменная?! $\square$

Однако из равенства множеств характеристик двух предикатных формул, не следует равенство самих формул.

***Пример* 1.**

$$A(\overline{x}) = p_1(x_3, x_0, x_2)\ \&\ p_1(x_1, x_3, x_0)\ \&\ p_1(x_2, x_1, x_3)$$
$$B(\overline{y}) = p_1(y_3, y_0, y_2)\ \&\ p_1(y_1, y_2, y_0)\ \&\ p_1(y_2, y_1, y_3)$$

Нетрудно видеть, что формулы не равны, однако их множества характеристик

$$\chi(A) = \{\chi(x_0)^A, \chi(x_1)^A, \chi(x_2)^A, \chi(x_3)^A\} =$$
$$= \{\ (\chi(x_0)^A_{p_1}), (\chi(x_1)^A_{p_1}), (\chi(x_2)^A_{p_1}), (\chi(x_3)^A_{p_1})\ \} =$$
$$= \{\ ((0,1,1)), ((1,1,0)), ((1,0,1)), ((1,1,1))\ \};$$

$$\chi(B) = \{\chi(y_0)^B, \chi(y_1)^B, \chi(y_2)^B, \chi(y_3)^B\} =$$
$$= \{\ (\chi(y_0)^B_{p_1}), (\chi(y_1)^B_{p_1}), (\chi(y_2)^B_{p_1}), (\chi(y_3)^B_{p_1})\ \} =$$
$$= \{\ ((0,1,1)), ((1,1,0)), ((1,1,1)), ((1,0,1))\ \};$$

совпадают.

2.4. Оценки числа шагов работы алгоритмов

Алгоритм полного перебора литералов и переменных:

В [7] была доказана оценка числа шагов работы алгоритма схожего с данным. Она верна и для этого алгоритма. Действительно, в самом худшем случае k пробегает значения от n до 0. Таким образом, число

шагов ограничего сверху порядком

$$O(\ \sum_{i=0}^{n} C_n^i \cdot A_t^m\ ) = O(\ 2^n \cdot A_t^m\ ), \tag{11}$$

что согласуется с вышеупомянутой оценкой.

Алгоритм полного перебора литералов методом ветвей и границ:

Алгоритм заканчивает работу за конечное число шагов, т.к. число литералов в формулах A и B конечно, и каждый шаг рекурсии удаляет хотя бы один элемент из каждого из множеств L' и P.

На асимптотику алгоритма помимо размера входных данных сильное влияние оказывает их структура. Под структурой, в данном случае, подразумевается размерность предикатов и характер распределения переменных в формуле.

В худшем случае, каждый шаг рекурсии удаляет фиксированное количество a элементов из множеств L' и P. Это возможно, например, в случае, когда каждая переменная встречается только в двух предикатах[3], каждый предикат двуместный и все литералы равны с точностью до имён аргументов. В таком случае число шагов работы алгоритма имеет порядок

$$\underbrace{O((n\cdot k)\cdot((n-a)\cdot(k-a))\cdot\ldots\cdot((n-\left\lfloor\frac{n}{a}\right\rfloor\cdot a)\cdot(k-\left\lfloor\frac{k}{a}\right\rfloor\cdot a)))}_{min\{\left\lfloor\frac{n}{a}\right\rfloor,\ \left\lfloor\frac{k}{a}\right\rfloor\}} \approx O(n^n\cdot k^k) \tag{12}$$

В лучшем случае глубина рекурсии равна единице. Это возможно, например, в случае, когда $|\overline{x}| = |\overline{y}| = t$ и все предикаты t–местные. Тогда выбор единственной пары графически совпадающих литералов определяет подстановку всех переменных. В таком случае число шагов

[3]Если переменная встречается только в одном предикате, то она не зависит от подстановок остальных переменных, и для нее сразу можно определить единственно верную подстановку, не учитывая при переборе

работы алгоритма имеет порядок

$$O(n \cdot k \cdot (n \cdot k)) = O(n^2 \cdot k^2) \tag{13}$$

В среднем после каждого шага рекурсии количество элементов в множествах L' и P будет уменьшаться в α раз ($\alpha > 1$). В таком случае число шагов работы алгоритма имеет порядок

$$O(\underbrace{n \cdot k \cdot \frac{n \cdot k}{\alpha} \cdot \frac{n \cdot k}{\alpha^2} \cdot \ldots \cdot 1}_{d = log_\alpha(n \cdot k)}) \approx O\left((n \cdot k)^{1/2 \ log_\alpha(n \cdot k)}\right) \tag{14}$$

Алгоритм проверки равенства предикатных формул с точностью до имен аргументов:

Основной этап алгоритма заключается в построении множества характеристик формул и проверки их на равенство. Для построения данных множеств нужно пройтись по всем литералам соответствующих формул и посчитать число вхождений переменных в них. Этот этап занимает порядка $O(n \cdot d)$ шагов, где n — число литералов в формуле, $d = \max_{i=1..n}\{|L_i|\}$, L_i - литерал с предикатным символом p_i. Далее следует проверка равенства полученных множеств характеристик. Эта операция занимает порядка $O(k^2)$ шагов, где k - число различных переменных в формулах. Таким образом, число шагов работы всего алгоритма имеет порядок

$$O(n \cdot d + k^2). \tag{15}$$

Для получения статистических оценок равенства самих формул при равенстве множеств характеристик двух формул были проведены численные расчеты. Производился запуск данного алгоритма и проверка корректности его работы с помощью алгоритма выделения наибольшей общей подформулы с точностью до имен аргументов. Для этого использовались пары формул с числом переменных от 3 до 7, с числом литералов от 3 до 10 и с размерностью предикатов от 3 до 5 (в обеих формулах эти числа совпадали). Число переменных всегда было больше размерности предикатов. Указанные числа выбирались случайным образом,

также распределение переменных в формулах было случайным и было близко к равномерному. После проверки 10^6 данных пар формул, было получено, что только в 0.038% случаях имеет место быть ложноположительный результат работы алгоритма. Таким образом, точность алгоритма составляет $\approx 99.95\%$.

В реализации самокорректирующейся сети, разработанной в рамках данной работы, во время построения многоуровнего описания классов при проверке равенства формул вначале используется данный алгоритм; в случае возврата положительного ответа производится проверка с помощью алгоритма выделения наибольшей общей подформулы.

3. Сравнение и анализ результатов

Была написана реализация вышеописанных алгоритмов и самокорректирующейся сети на языке программирования C++. Все данные о времени работы алгоритмов относятся к запуску без распараллеливания на компьютере с процессором Intel B950 с частотой 2.1 ГГц. Все данные получены в результате сбора статистики после 50 запусков соответствующих алгоритмов. Время работы приведено в секундах.

3.1. Сравнение скорости работы алгоритмов выделения наибольшей общей подформулы

Для получения сравнительных результатов скорости работы алгоритмов выделения наибольшей общей (с точностью до имен аргументов) подформулы двух формул было проведено несколько запусков на наборах пар формул. Распределение переменных в формулах было случайным и было близко к равномерному, т.е. все переменные встречались в записи формулы примерно одинаковое число раз, во всех формулах был использован один и тот же предикатный символ (без отрицания), если не оговорено противное.

Алгоритм полного перебора литералов и переменных. 10 литералов, 6 переменных, размерность предиката — 3.

- Минимальное: **0.008**
- Среднее: **0.0194**
- Максимальное: **0.026**

Алгоритм полного перебора литералов и переменных. 12 литералов, 8 переменных, размерность предиката — 3.

- Минимальное: **4.652**

- Среднее: **6.022**
- Максимальное: **7.43**

Полученные результаты согласуются с экспоненциальными оценками времени работы, приведенными выше. Нижеследующие результаты относятся к алгоритму полного перебора литералов методом ветвей и границ.

10 литералов, 6 переменных, размерность предиката — 3.

- Минимальное: **0.0013**
- Среднее: **0.0022**
- Максимальное: **0.0242**

12 литералов, 8 переменных, размерность предиката — 3.

- Минимальное: **0.0048**
- Среднее: **0.0083**
- Максимальное: **0.017**

20 литералов, 10 переменных, размерность предиката — 3.

- Минимальное: **0.425**
- Среднее: **0.6286**
- Максимальное: **0.851**

25 литералов, 12 переменных, размерность предиката — 3.

- Минимальное: **7.914**
- Среднее: **12.078**

- Максимальное: **18.321**

Для данного алгоритма также прослеживается экспоненциальная зависимость времени работы от размера входных данных, что согласуется с вышеприведенными оценками.

3.2. Анализ зависимости от структуры входных данных алгоритма перебора литералов методом ветвей и границ

Выше утверждалось, что на число шагов работы данного алгоритма оказывает серьезное влияние структура входных данных. Следует заметить, что предыдущие результаты, в этом смысле, приведены для наихудшего случае. Существует зависимость числа шагов работы алгоритма, как уже было сказано ранее, от размерности предикатов, являющихся частью исходных формул.

25 литералов, 12 переменных, размерность предиката — 4.

- Минимальное: **0.146**
- Среднее: **0.217**
- Максимальное: **0.269**

25 литералов, 12 переменных, размерность предиката — 5.

- Минимальное: **0.0478**
- Среднее: **0.0518**
- Максимальное: **0.0655**

Результаты показывают существенное снижение числа шагов работы алгоритма при увеличении размерности предиката, это является

следствием уменьшения глубины рекурсии в процессе перебора множеств литералов во время работы алгоритма. Также существенное влияние оказывает количество различных предикатных символов, участвующих в записи формулы.

25 литералов, 12 переменных, 2 различных предикатных символа, размерность предикатов — 3.

- Минимальное: **0.452**
- Среднее: **0.723**
- Максимальное: **1.131**

25 литералов, 12 переменных, 3 различных предикатных символа, размерность предикатов — 3.

- Минимальное: **0.083**
- Среднее: **0.164**
- Максимальное: **0.352**

Выбор предикатных символов во время запуска случаен, все символы встречались примерно равное количество раз в записи формулы. Результаты показывают существенное снижение числа шагов работы алгоритма при увеличении количества различных предикатных символов в формулах. Данный факт является следствием увеличения эффективности отброса "ветвей" во время процесса перебора множеств литералов, т.к. наличие нескольких предикатных символов, в определенном смысле, разбивает процесс поиска на подзадачи меньшей размерности. Также немаловажным фактором для времени работы алгоритма является распределение переменных в записи формулы. Все вышеприведенные результаты получены с учетом равномерного распределения переменных, т.е. с равным в среднем числом вхождений переменных в формулу.

25 литералов, 12 переменных, 1 предикатный символ, размерность предиката — 4. Равномерное распределение переменных.

- Минимальное: **0.156**
- Среднее: **0.213**
- Максимальное: **0.261**

25 литералов, 12 переменных, 1 предикатный символ, размерность предиката — 4. Нормальное распределение с параметрами (6, 2).

- Минимальное: **0.047**
- Среднее: **0.056**
- Максимальное: **0.065**

Нормальное распределение переменных следует воспринимать с учетом нумерации переменных вида x_i, в данном случае i от 0 до 11, т.е. наибольшее число вхождений в среднем у переменной x_6, меньше у x_5, x_7 и т.д. Результаты показывают существенное снижение числа шагов работы алгоритма при неравномерном распределении переменных. Данный факт также является следствием более эффективного избавления от "ветвей" в процессе перебора множеств литералов.

Суммируя все замечания касательно структуры входных данных, можно добиться более эффективного использования данного алгоритма.

25 литералов, 12 переменных, 1 предикатный символ, размерность предиката — 3. Равномерное распределение переменных.

- Минимальное: **7.914**
- Среднее: **12.078**
- Максимальное: **18.321**

100 литералов, 40 переменных, 6 предикатных символов, размерность предикатов — 5. Нормальное распределение с параметрами (20, 6).

- Минимальное: **11.04**
- Среднее: **16.85**
- Максимальное: **34.53**

3.3. Анализ результатов времени работы самокорректирующейся сети

С целью получения результата времени работы алгоритма построения многоуровнего описания (блок обучения) была произведена серия запусков на исходных наборах (описания классов объектов) различного размера. Исходные формулы имели следующие характеристики: 12 переменных, 30 литералов, 3 различных предикатных символа, размерность предикатов — 4, случайное равномерное распределение переменных.

Набор из 5 классов:

- Минимальное: **0.267**
- Среднее: **0.306**
- Максимальное: **0.359**

Набор из 7 классов:

- Минимальное: **0.662**
- Среднее: **0.743**
- Максимальное: **0.842**

Набор из 10 классов:

- Минимальное: **2.109**
- Среднее: **2.52**
- Максимальное: **3.017**

Набор из 15 классов:

- Минимальное: **13.96**
- Среднее: **16.21**
- Максимальное: **18.98**

Результаты показывает сильную зависимость числа шагов работы блока обучения от размера набора исходных формул. Тем не менее, данная зависимость близка к линейной, что согласуется с тем фактом, что основная вычислительная нагрузка ложится на алгоритм выделения наибольшей общей подформулы.

Для оценки эффективности алгоритма распознавания (блок распознавания) была проведена серия запусков с использованием уже построенного многоуровнего описания. Для получения сравнительного результата, время работы данного алгоритма сопоставлялось с временем, затраченным на проверку выводимости без использования указанного описания (осуществлялась проверка выводимости формулы, представляющей собой описание объекта, последовательно для каждой из формул входящей в исходный набор описаний классов объектов). Все формулы имели следующие характеристики: 1 предикатный символ, размерность предиката — 3, равномерное распределение переменных. Число классов в исходном наборе — 7.

5 различных переменных, 13 литералов.

Время	Многоуровневое описание	Непосредственно
Минимальное	0.0072	0.005
Среднее	0.0016	0.015
Максимальное	0.0035	0.038

8 различных переменных, 20 литералов.

Время	Многоуровневое описание	Непосредственно
Минимальное	0.0014	0.182
Среднее	0.0034	0.611
Максимальное	0.0076	1.438

10 различных переменных, 25 литералов.

Время	Многоуровневое описание	Непосредственно
Минимальное	0.0025	2.92
Среднее	0.008	10.48
Максимальное	0.018	20.29

Заключение

Исходя из сравнительных результатов, можно сделать вывод, что для решения задач (3), (4) и (5) может быть успешна применена модель самокорректирующейся (логико-предикатной) сети. При создании описания для требуемых классов объектов эксперту в данной области стоит особое внимание обратить на структуру исходных данных. Как видно из результатов работы алгоритма полного перебора литералов методом ветвей и границ, структура данных может оказать серьезное влияние на временные характеристики работы всей сети. Так сто́ит, по возможности, использовать предикаты большей размерности при создании описаний, вводить большее количество различных предикатных символов. Также при выборе объектов в качестве аргументов для предикатов стоит учесть желательную неравномерность такого набора, например, в случае анализа рынка вполне естественна ситуация, когда некоторая часть предприятий фигурирует в гораздо большем количестве связей, по сравнению с другими.

Анализ времении работы блока распознавания показывает высокую эффективность для использования на практике, с учетом уже имеющегося многоуровнего описания классов. Тем не менее, время работы алгоритма выделения наибольшей общей подформулы имеет высокие асимптотические оценки. В дальнейшем возможно использование эвристического варианта рассматриваемого алгоритма, с вероятностной оценкой выводимости формул. Также требует дальнейшего исследования алгоритм проверки равенства формул с точностью до имен аргументов. Возможна классификация случаев ложноположительного результата работы и доработка алгоритма, исходя из полученных данных.

Список литературы

[1] Герасимов А.С. Курс математической логики и теории вычислимости // Санкт-Питербург: Лема. — 2011. — Vol. 27.

[2] Клини С. Математическая логика: Пер. с англ. — Мир, 1973.

[3] Нильсон Н. Искусственный интеллект. — Рипол Классик, 1973.

[4] Рассел Стюарт, Норвиг Питер. Искусственный интеллект: современный подход. — Вильямс, 2006.

[5] Косовская Татьяна Матвеевна. Некоторые задачи искусственного интеллекта, допускающие формализацию на языке исчисления предикатов, и оценки числа шагов их решения // Труды СПИИРАН. — 2010. — Vol. 3, no. 14. — P. 58–75.

[6] Косовская Т.М. Многоуровневые описания классов для уменьшения числа шагов решения задач распознавания образов, описываемых формулами исчисления предикатов // Вестн. С.-Петербург. ун-та. Сер. — 2008. — Vol. 10. — P. 64–72.

[7] Косовская Татьяна Матвеевна. Частичная выводимость предикатных формул как средство распознавания объектов с неполной информацией // Вестн. С.-Петерб. ун-та. Сер. — 2009. — Vol. 10. — P. 74–84.

[8] Тимофеев Адиль Васильевич, Косовская Татьяна Матвеевна. Нейросетевые методы логического описания и распознавания сложных образов // Труды СПИИРАН. — 2013. — Vol. 4, no. 27. — P. 144–155.

[9] Дуда Р., Харт П. Распознавание образов и анализ сцен. — Рипол Классик, 1976.

[10] Хайкин Саймон. Нейронные сети: полный курс, 2-е издание. — Издательский дом Вильямс, 2008.

[11] Stroustrup Bjarne. The C++ programming language. — Pearson Education, 2013.