Алгоритм поиска наибольшей общей подформулы в задаче построения логико-предикатной сети

Петров Д.А.

студент кафедры информатики СПбГУ
imsohay@gmail.com

Аннотация

В работе рассматривается алгоритм выделения наибольшей общей подформулы двух формул специального вида заданных на языке предикатов. Данная задача возникла в ходе построения многоуровнего описания классов. Описание строится с целью уменьшения числа шагов работы алгоритмов, решающих задачи распознавания образов, анализа рынка, медицинской диагностики, основанных на поиске вывода в исчислении предикатов.

В работе описан алгоритм и даны асимптотические оценки числа шагов его работы

Введение

При решении задач искусственного интеллекта, включающих в себя такие задачи как распознавание образов, медицинская диагностика и выбор лечения, анализ рынка и выбор действий в рыночных условиях, всё чаще [5] используются язык и методы доказательства математической логики. Подход, описанный в данной работе, основан на логическом выводе из множества аксиом в исчислении предикатов, которые представляют знания об имеющемся состоянии частей рассматриваемого объекта (например, состояние органов пациента), различных формул, которые характеризуют состояние объекта в целом (например, постановка диагноза больному).

Внимание данной работы сфокусировано на алгоритме выделения общей подформулы формул, описывающих свойства частей объекта и отношения между ними. Ранее [2] было показано, что в общем случае это задача является NP–трудной, также были предложены два алгоритма: алгоритм основанный на методе полного перебора и алгоритм основанный на методе логического вывода в исчислении предикатов. В данной работе представлен алгоритм, также основанный на методе полного перебора, но с использованием метода ветвей и границ.

Постановка задачи

В [3] было представлено формальное описание задачи распознавания образов, а также, для уменьшения числа шагов работы алгоритмов решающих данную задачу, было предложено иерархическое многоуровневое описание классов распознаваемых объектов. В алгоритме построения многоуровнего описания классов главную вычислительную сложность представляет этап выделения наибольших общих подформул из формул, составляющих описание классов. В [2] доказана NP-трудность данных задач. В [1] введено понятие $(q,r)-$фрагмента и наибольшей подформулы.

***Определение* 1.** *Фрагментом* формулы $A(x')$ называется любая ее подформула $\tilde{A}(y')$, где список переменных y' является подсписком списка переменных x'.

Пусть a и a' — количество атомарных формул в $A(x')$ и $\tilde{A}(y')$ соответственно, m и $\tilde{m}$ — количество предметных переменных в $A(x')$ и $\tilde{A}(y')$ соответственно

Числа q и r вычисляются по формулам $q = \frac{\tilde{a}}{a}, r = \frac{\tilde{m}}{m}$ и характеризуют степень совпадения формул $A(x')$ и $\tilde{A}(y')$. При этом $0 < q \leq 1,\ 0 < r \leq 1$. Кроме того, $q = r = 1$ тогда и только тогда, когда $\tilde{A}(y')$ совпадает с $A(x')$.

***Определение* 2.** При таким обозначениях формулу $\tilde{A}(y')$ будем называть $(q,r)-$ *фрагментом формулы* $A(x')$.

Пусть $A(\overline{x})$ и $B(\overline{y})$ – две элементарные конъюнкции предикатных формул со списками предметных переменных $\overline{x}$ и $\overline{y}$ соответственно. 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 **наибольшей** можно брать любую из этих формул.

Описание алгоритма

Проверка неполной выводимости для $A(x') \Rightarrow_P \exists y'_{\neq} B(y')$. Введём некоторые обозначения: для краткости исходные 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).

Оценка числа шагов работы алгоритма

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, каждый предикат двуместный и все литералы равны с точностью до имён аргументов. В таком случае число шагов работы алгоритма имеет порядок

$$\underbrace{O((n \cdot k) \cdot ((n-a) \cdot (k-a)) \cdot \ldots \cdot ((n - \left\lfloor \frac{n}{a} \right\rfloor \cdot a) \cdot (k - \left\lfloor \frac{k}{a} \right\rfloor \cdot a)))}_{min\{\left\lfloor \frac{n}{a} \right\rfloor, \left\lfloor \frac{k}{a} \right\rfloor\}} \approx O(n^n \cdot k^k) \tag{1}$$

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 работы алгоритма имеет порядок

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

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

$$\underbrace{O(n \cdot k \cdot \frac{n \cdot k}{\alpha} \cdot \frac{n \cdot k}{\alpha^2} \cdot \ldots \cdot 1)}_{d = log_\alpha(n \cdot k)} \approx O\left((n \cdot k)^{1/2 \; log_\alpha(n \cdot k)}\right) \tag{3}$$

Заключение

Исходя из оценок шагов работы алгоритма, можно сделать вывод, что для решения вышеописанной задачи может быть успешно применён данный алгоритм. Требует дальнейшего исследования зависимость числа шагов работы алгоритма от входных данных. По видимому, оказывает сильное влияние распределение переменных в качестве аргументов предикатов. В соответствии с этим, можно сказать, что возможно усовершенствование алгоритма на этапе выбора литералов из множеств P и L'. Также данный алгоритм не вводит порядка перебора возможных подстановок: это не оказывает сильного влияния на более "равномерных" данных, но, скорее всего, будет играть значительную роль в случае, когда некоторые переменные будут на порядок чаще встречаться в предикатах, чем другие. Также возможен эвристический вариант алгоритма, ограничивающий глубину рекурсии или множество выбираемых литералов для следующего шага рекурсии.

Существует возможность при построении логико-предикатной сети [4] для конкретной предметной области учесть характерные структурные свойства предикатов и формул, описывающих свойства частей объекта и отношения между ними, тем самым уменьшив время работы алгоритма.

Литература

[1] Косовская Т. М. Частичная выводимость предикатных формул как средство распознавания объектов с неполной информацией // Вестн. С.-Петербург. университета. Сер. 10. 2009. Вып. 1. С. 74 – 84.

[2] Косовская Т.М. Некоторые задачи искусственного интеллекта, допускающие формализацию на языке исчисления предикатов, и оценки числа шагов их решения // Труды СПИИРАН, 2010. Вып. 14. С. 58 – 75.

[3] Косовская Т.М. Подход к решению задачи построения многоуровневого описания классов на языке исчисления предикатов // Труды СПИИ-РАН - 2014. - №3 (34). - С. 204-217.

[4] Косовская Т.М. Логико-предикатные сети // Труды СПИИРАН, 2015. (в печати)

[5] Рассел С., Норвиг П. Искусственный интеллект: современный подход / Пер. с англ. и ред. К. А. Птицына. М.: Изд. дом Вильямс, 2006. 1408 с.