UDC 519.688

The Algorithms of the Mulri-Threaded Relevant LP-inference

S. Yu. Bolotova, S. D. Makhortov

Voronezh State University
1, Universitetskaya pl., Voronezh, Russia, 394006

The relevant LP-inference, which is based on the solution of logical equations, is an effective tool that can be used for research and optimization of production-logical systems. It significantly reduces the number of executed queries to an external source of information (either to a database or an interactive user). The preference is given to testing the facts that are really needed in the inference. However, experiments have shown that the process of using the relevant LP-inference may require an excessive amount of computational resources of the computer. So, the relevant LP-inference method was modified to use parallel computing algorithms. This paper describes the implementation of a multi-threaded algorithms for relevant LP-inference and provides the pseudocodes of these algorithms. Multi-threading is a fundamentally new element in the implementation, which allows speeding up the process of constructing sets of facts that are required in the inference, and their further processing.

Key words and phrases: backward inference, relevant inference, logic equations, parallelism, multi-threading.

1. Introduction

Production systems is an important area of the theoretical and applied research in the field of artificial intelligence. They are used in the theory of learning and problem solving systems, the development expert systems, covering a large variety of applications in several areas, such as medicine, engineering, exploration, and others [1]. Production systems have very rich history and the fact that they are still of interest is confirmed by numerous modern applied and theoretical researches in this area [2–5].

However, production system programs are very computationally intensive and run quite slowly. In particular, they often require an exponential number of computation and intensive exchange with external memory. This fact reduces their effectiveness, and therefore, many experts try to avoid working with programs that require extreme amounts of resources [1]. Consequently, a significant increase of production systems performance is the important part in theoretical and applied research.

There are several possible ways for speeding up the execution of production systems. This paper focuses on one of the methods based on using parallel computing in the production systems models [6] and presents the architecture of an object-oriented class LPStructure, encapsulating the most important properties and methods described in [6], including searching logical reductions and solutions of the production logical equations. Multi-threading is a fundamentally new element in the implementation. This class was developed in MS Visual Studio, using the C++ language and STL library. For the purpose of reusing the class in other software systems, its interface is designed as a dynamic C-library, called LPStructure.dll.

2. A General Description of the Algorithm

The considered software system implements the relevant parallel backward chaining based on the solution of the production logical equations and for this purpose it uses mathematically-based algorithms for finding the truth preimage and accelerating backward inference [6]. Hereafter we use the notations and definitions used in the mentioned article.

LP-structure (lattice-production structure) — is the lattice with an additional binary relation set on it, which has a number of production-logical properties [6].

Received 30$^{\text{th}}$ December, 2013.

The relevant backward inference strategy is aimed at minimization of a query number to the external information source (either to a database or an interactive user). The inference based on equation solving starts with the creation of all minimal initial preimages in the LP-structure for the atoms that correspond to the values of an examined object. Using this constructed set it is enough to find the preimage that contains only true facts. If founded it makes possible to have a conclusion about the corresponding value of the object. The effective way to achieve that, is to prioritize the viewing of the preimages containing the values of the most relevant objects [6]. These are first of all the objects, whose values are present in a maximal number of the constructed preimages. A negative answer to a unique query eliminates all subsequent queries about the elements of a facts subset. Along with a significantly reduced number of queries, when using LP-inference, the preference is given to testing the sets of facts of a minimal cardinality.

However, experiments have shown that the process of simultaneous construction of the minimal initial preimages for large knowledge bases and their "deep" structure may require an excessive amount of computational resources of the computer. Considering this relevant LP-inference method was modified to use parallel computing algorithms. Parallel relevant LP-inference allows speeding up the process of constructing sets of facts that are required in the inference, and their further processing [7].

As an illustration we shall give simplified descriptions of algorithms of a relevant LP-inference and a parallel LP-inference.

Input:
initial facts – true (set T), false (set F);
set of rules R;
hypothesis b.

Output:
true initial preimage $X^0 = \{x_1^0, ..., x_n^0\}$ for b
(or $X^0 = null$).

// Relevant LP-inference
$X^0 = null$
{X} = *getPreImages*(b)
while $X^0 = null$ **and** $\{X\} \neq \emptyset$ **do**
$k = getRelevantIndex(\{X\}, T)$
$Ask\ (x_k)$
foreach $X_j \in \{X\}$ **do**
if $X_j \subseteq T$ **then** $X^0 = X_j$
break
end
if $X_j \cap F \neq \emptyset$ **then** $\{X\} = \{X\} \backslash X_j$
end
end

The function $Ask(x)$ asks a user (or an external database) about the truth of an atomic fact x and according to the answer modifies sets T and F. Function $getPreImages(b)$ solves the equation with a right-hand side b, i.e. it constructs set $\{X\}$ of all minimal initial preimages for atom b.

The equation solving is carried out by splitting relation R into "layers" $R_t, t \in T$, each containing no more than one solution [6]. The layer has the maximum possible set of relationship pairs with the unique right-hand sides, and two layers differ in one pair at least. In each separate layer the process of the equation solving is reduced to the problem of finding the set of initial nodes of the graph $G_{R_t,b}$, corresponding to a layer.

A working with different layers is organized independently and in parallel.

The primary application thread creates secondary threads (which are limited by the $MaxThread$ — the maximum number of threads), passing them a block of data. Hereinafter the "threads" are threads of execution [7]. Created thread searches the solution of production logic equation in a separate layer. Then the LP-inference program

module immediately checks its "truth", sending queries to the external information source, if necessary. If one of the preimages is not true, program stores the information about false facts (it reduces possible number of rules, required for next equation solution) and calculates the next preimage which also significantly speeds up the process and after that threads exit.

In implementing a thread pool mechanism is used [7]. The main thread takes thread from the pool and passes the necessary data for processing. If the number of active threads is less than the maximum, a new thread is created. When the number of active threads is maximized, request queued and waiting for the release of one of the threads. For threads synchronizing the critical sections, which are initiated in the activation process, are used.

Hereafter is an algorithm of $getPreImages(b)$ function, where R' is a set of layers $\{R_i\}$. As already mentioned, the solution in each layer is calculated in a separate thread. The maximum number of threads in the pool is limited by $MaxThreads$ parameter. The number of active threads is stored in the $countUsedThreads$ variable. If the pool has a free thread, then it schedules this thread for execution by passing it pointer to a $FindEquationDesicion(R_i)$ function, that finds a solution in the next layer R_i.

// Finding the solutions of the equation for each layer in a separate thread
foreach $R_i \in R'$ **do in parallel**
 if $countUsedThreads < MaxThreads$ **then**
  $BeginThread()$; // Start the thread
  $countUsedThreads++$;
  $FindEquationDesicion(R_i)$;
  $ExitThread()$; // End the thread
  $countUsedThreads--$;
 else
  $Waiting()$; // Wait until the thread is freed
 end;
end.

Function $getRelevantIndex(\{X\}, T)$ finds index k of any of the most relevant and not checked for truth facts contained in the current $\{X\}$. The function uses two above indices of relevance to minimize the number of calls of the function $Ask(x_k)$. The process of identifying relevant objects is very expensive, so it is also parallelized. For the each number of facts the system calculates the number of preimages, containing these facts using multiple threads (number of threads is limited by $MaxThreads$ parameter). The fact, that belongs to the maximum number of preimages, increments its relevance score by 1. When accessing shared resources threads are synchronized using the critical sections mechanism.

The benefits of parallel LP-inference are confirmed experimentally. When processing the large knowledge bases the use of parallel algorithms speeds up LP-inference by up to 30%.

3. Conclusions

Often the processing of large knowledge bases requires a lot of computational resources. Thus it is necessary to find ways to improve the efficiency of such systems. The presented library demonstrates the methods of using the theory of LP-structures in order to speed up the backward inference production systems, as well as the program based on the use of multi-threading.

References

1. *Gupta A.* Parallelism in Production Systems. — London: Pitman, 1987.
2. *Liberatore P.* Redundancy in Logic II: 2CNF and Horn Propositional Formulae // Artificial Intelligence. — 2008. — Vol. 172, No 2–3. — Pp. 265–299.

3. *Maciol A.* An Application of Rule-Based Tool in Attributive Logic for Business Rules Modeling // Expert Systems with Applications. — 2008. — Vol. 34, No 3. — Pp. 1825–1836.
4. *Poli R., Langdon W. B.* Backward-Chaining Evolutionary Algorithms // Artificial Intelligence. — 2006. — Vol. 170, No 11. — Pp. 953–982.
5. *Katerinenko R. S., Bessmertnyi I. A.* A Method for Acceleration of Logical Inference in the Production Knowledge Model // Programming and Computer Software. — 2009. — Vol. 37, No 4. — Pp. 197–199.
6. *Makhortov S. D.* Mathematical Foundations of Artificial Intelligence: Theory LP-structures for the Construction and Studying of Knowledge Models of Production Type / Ed. by V. A. Vasenin. — Moscow: MCCME, 2009.
7. *Richter J.* Programming applications for Microsoft Windows. — Grove City: Microsoft Press, 1999.

УДК 519.688

Параллельные алгоритмы релевантного LP-вывода

С. Ю. Болотова, С. Д. Махортов

Воронежский государственный университет
Университетская площадь, д. 1, Воронеж, Россия, 394006

Релевантный LP-вывод, который основывается на решении логических уравнений, является эффективным средством для исследования и оптимизации продукционно-логических систем. Он позволяет существенно сократить количество выполняемых запросов к внешнему источнику информации (к базе данных или интерактивному пользователю). Предпочтение отдаётся исследованию только тех фактов, которые действительно необходимы при выводе. Однако эксперименты показали, что процесс использования релевантного LP-вывода может потребовать большого количества вычислительных ресурсов компьютера. В связи с этим метод релевантного LP-вывода был модифицирован путём использования параллельных вычислений. В этой статье описывается реализация параллельных алгоритмов релевантного LP-вывода и приводятся псевдокоды этих алгоритмов. Многопоточность является абсолютно новым элементом в реализации, который позволяет ускорить процесс построения множеств фактов, которые необходимы при выводе, и их дальнейшего исследования.

Ключевые слова: обратный вывод, релевантный вывод, логические уравнения, многопоточность.