UDC 004.832

TECHNOLOGIES OF INFERENCE IN SOFTWARE SYSTEMS

Ausheva N.M., Doctor of Technical Sciences,
nataauscheva@gmail.com, ORCID: 0000-0003-0816-2971
Shapovalova S.I., PhD (Engin.),
lanashape@gmail.com, ORCID: 0000-0002-3431-5639
National Technical University of Ukraine Igor Sikorsky Kyiv Polytechnic Institute (Kyiv, Ukraine)

The target of research is the implementation technology of inference. The purpose of the article is to determine the main areas of research on automatic inference, software obtained on their basis, and relevant areas of application.

Inference is a common task that is implemented in application software. The efficiency of the software systems, in particular, is determined by the performance of the built-in inference engine. Means of implementing inference shall provide optimal execution time and interaction with other components, as well as meet the requirements of the application task. The article highlights the main areas of improving the software implementation of inference engine and relevant areas of research: (i) expanding the concept and combining several paradigms of logic programming (probabilistic logic programming, defeasible inference, coinductive programming); (ii) reduction of data exchange time between software components and processing time of large knowledge bases (improving pattern matching algorithm for implementing rule-based systems, creating or expanding implementations of inference tools for integration into distributed software systems, developing new implementations of inference engines for specialized programming languages and general-purpose languages); (iii) a combination of conceptually different approaches to inference (a combination of both approaches to inference on logical and productive concepts, the integration of inference paradigms and neural network approach). The combination of software implementations of different concepts, first of all, inference engines and neural network models, gives new opportunities to artificial intelligence,

For each area of research, there are presented software tools, as well as examples of areas of their application in accordance with this concept.

Key words: inference engine, probabilistic logic programming, defeasible reasoning, rule-based systems, PRISM, ProbLog, XSB Prolog, SWI Prolog, DeepProbLog.

Statement of problem. The rapid intellectualization of software determines the need to solve problems of artificial intelligence concurrently with many other complex resource-intensive tasks, which are characterized by their

own computing mechanisms. One of the most popular intellectual tasks is an inference. The efficiency of the software systems, in particular, is determined by the performance of the built-in inference engine. Therefore, it is important to reduce the inference time to values comparable to the time for solving other current problems in software systems. The first step in solving this problem is to determine the software tools for inference, which will provide optimal execution time and interaction, as well as meet the task requirements.

Recent research and publication analysis. Publications, as a rule, provide software tools, selected either by the specifics of inference or by program implementation languages. For example, [1, 2] presents software of defeasible reasoning tools to present and process information of Internet resources and ontologies (Semantic Web, Ontology Based Data). [3] presents an inference engine implemented for Erlang functional programming language. Such analysis is highly specialized. Modern research is conducted in many areas, as well as at their intersection. Therefore, it is advisable to determine the overall impact of research on improving existing and creating new software tools for inference implementation.

Setting article objectives. The objective of the article is to determine the main lines of research on automatic inference, software obtained on their basis, and relevant areas of application.

Main part. The basic paradigms of knowledge representation for step-by-step conclusions based on pre-known and previously proven facts are logical and productive. For each of these paradigms, appropriate software tools have been developed: Prolog family programming languages and rule-based tools for building expert systems. Each software toolkit contains an inference engine and a resource for information presentation. The software is marked in italics in the article.

Modern research on improving inference engines is aimed primarily at expanding the concept and combining several paradigms of logical programming in software implementations. In this area, we can determine:

- Probabilistic logic programming.

The probabilistic hybrid knowledge bases combine the paradigms of Logic Programming languages and Description Logics. The development of this area is carried out both conceptually and by improving the implementation of software tools for probabilistic logic programming. For example, [4] proposes a definition approach with a secure representation of descriptive rules for hybrid knowledge bases with MKNF semantics (Minimal Knowledge with Negation as Failure).

On the other hand, [5] proposed a new data structure to represent possible worlds and a method to convert programs for its application. To prove the effectiveness of the proposed method experimentally, there was implemented a prototype in the logical programming language *XSB Prolog*. For comparison, test problems were also solved in probabilistic logic systems *ProbLog* (Probabilistic extension of ProLog) and *PITA* (Probabilistic Inference with

Tabling and Answer subsumption). Another common language of probabilistic logic programming is *PRISM* (PRogramming In Statistical Modeling). [6] presents the transformation of PRISM into ProbLog and back to highlight their differences.

The implemented PITA algorithm provides probable opportunities for existing inference systems. For example, PITA is implemented in *XSB Prolog*. *XSB Prolog* includes the Coherent Description Framework (*CDF*), which supports the presentation and implementation of complex Semantic Web ontologies that meet the definitions of the World Wide Web Consortium (W3C).

[7] introduces a modification of PITA to extend *SWI Prolog*, which enables to create a web application for probabilistic logic programming.

- Defeasible reasoning.

[8] proves the correctness of defeasible theories compilation, provides the implementation in *Datalog* logical programming language. [1] offers a benchmark and presents the results of testing specialized defeasible reasoning tools: *DeLP, ASPIC +, SPINdle*, *Flora-2, DEFT*. [2] presents a tool to implement the defeasible reasoning *ELDR*, which implements the proposed formalism to present the rules of the program. Computational experiments were performed for: *ELDR, ASPIC +, DEFT, DeLP, DR-DEVICE* and *Flora-2*. Such tools are used to process existing Big Data set, primarily Semantic Web data.

- Coinductive logic programming [9]

Infinite computations are an example to apply coinductive computations. Currently, the corresponding functionality is implemented in the logic programming languages *Logtalk, SWI-Prolog*, the system of formal semi-interactive construction of evidence with machine verification *Coq*. The latter system is designed to formalize complex proofs of mathematical theorems. From a programming point of view, *Coq* can be used to verify programs. The correctness of the *Coq* core has been checked by the *Coq* system [10].

Another area of research is aimed at reducing the processing time of large knowledge bases, both local and distributed. An example is

- Improving pattern matching algorithm for rule-based systems.

The Rete algorithm is a basic method of pattern matching. The known derivatives of Rete are: TREAT and GATOR. However, Rete or its modifications are mostly used to implement inference engines. Improved Rete II, Rete III, Rete NT were developed by the author of Rete. Each subsequent implementation increased performance significantly compared to the previous one.

Research has mainly focused on analyzing the performance of existing inference engines, for example [1, 11, 12], and extending Rete for special applications.

The Rete-OO algorithm [13] enables to extend the application of Rete from first-order inference to defeasible reasoning to be used in the most common specialized knowledge representation engines: 3-valued logic, classical certainty factors, fuzzy, many-valued logic and Bayesian networks. [14]

proposes an algorithm to match RETE-ADH templates, which is designed for composite context-aware service architecture.

[15] offers a Rete-ECA algorithm for device control systems. According to the results of computational experiments conducted using the mouse-tracking environment, the Rete-ECA algorithm in comparison with Rete reduces the required computing resources significantly, including execution time.

Production rule systems are widely used in various industries. For example, the *Drools* production system cover is used to create Business Rules Management Systems (BRMS). [16] introduces the use of *Venus* and *RuleWorks* to ensure Rule Modularity and improve inference for the Java-Based Rule Engine.

- Create or expand implementations of inference tools to integrate into distributed software systems.

Service-oriented architectures have become standard for distributed systems. [17] proposes a service-oriented approach for distributed situated intelligence, according to which inference engines are used as a distributed service. A prototype of this approach for Smart House is presented on *tuProlog*. The service-oriented computing paradigm is widely used, for example, to transfer applications to the cloud environment, as well as the Internet of Things (IoT).

- Creation of new implementations of inference engines.

Implementation of inference engines directly in the language of the software systems provides rapid data exchange. Therefore, such implementations exist for the most common programming languages. New implementations are being developed primarily for specialized programming languages. Historically, the first programming languages, for which inference engines were developed, were C, Java, Perl, Pyton, Erlang. [3] presents the implementations of inference engines in the functional programming language Erlang. These implementations correspond to both concepts of inference: logical - *Erlog* (ProLOG interpreter in ERlang) and rule-based: *ERESYE* (ERlang Expert SYstem Engine), *SERESYE* (Swarm oriented ERlang Expert SYstem Engine) and *RUNES II*. Also, [3] proposes the method to implement production model for Erlang on the basis of its built-in engine of pattern matching.

Studies on the combination of conceptually different approaches to inference are the most promising.

- Integration of both inference paradigms: logical and rule-based.

[18] presents *SWISH* (SWI-Prolog for SHaring or SWIProlog SHell), which is a web interface for Prolog that consists of a web server (implementation on *SWI Prolog*) and a client web application (implementation on JavaScript). In addition to the SWISH architecture, two examples of Prolog extensions are provided: probabilistic logical programming and Logic Production System.

In addition, modern artificial intelligence is impossible without neural networks.

■ Integration of inference paradigms and neural network approach.

[19] presents a structure that combines inference engines (reasoning), probabilistic logic programming, and general-purpose neural networks, and offers a programming language *DeepProbLog* that enables to use these engines.

It is the combination of inference engines and neural network models that provides new possibilities for artificial intelligence, such as explaining the conclusions of neural networks or reasoning based on facts obtained from the recognition of external signals.

Conclusions. There have been defined modern researches on inference implementation which have led to development of new or improvement of existing inference engines. Relevant software tools have been presented and examples of areas of their application according to their concept have been given.

Literature

1. Hecham A., Croitoru M., Bisquert P. A First Order Logic Benchmark for Defeasible Reasoning Tool Profiling. *RuleML+RR*, 2018, Luxembourg, P.81-97. DOI:10.1007/978-3-319-99906-7_6.
2. Hecham, A., Croitoru, M., Bisquert P. On a flexible representation for defeasible reasoning variants. *Proc. of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'18)*, 2018, P. 1123–1131.
3. Шаповалова С.І. Формалізація представлення продукційних правил в Erlang. *Математичне та комп'ютерне моделювання. Серія: технічні науки.* 2020. Вип. 21. С.125-139. DOI:10.32626/2308-5916.2020-21.125-139
4. Alberti M, Lamma E, Riguzzi1 F., Zese R. A Distribution Semantics for non-DL-Safe Probabilistic Hybrid Knowledge Bases. *Proc. of the Workshop on Probabilistic Logic Programming (PLP 2017)*. 2017. CEUR. Vol. 1916, pp. 40-50.
5. Nampally A., Zhang T., Ramakrishnan C. Constraint-Based Inference in Probabilistic Logic Programs. *Theory and Practice of Logic Programming*. 2018. 18(3-4), pp. 638-655. doi:10.1017/S1471068418000273
6. Vandenbroucke A., Schrijvers T. From PRISM to ProbLog and Back Again. *Proc. of the Workshop on Probabilistic Logic Programming (PLP 2017)*. 2017. CEUR. Vol. 1916, pp. 26-40.
7. Riguzzi F., Wielemaker J., Zese R. Probabilistic Inference in SWI-Prolog. *Proc. of the Workshop on Probabilistic Logic Programming (PLP 2018)*. 2018. CEUR. Vol. 2219, pp. 15-27.
8. Maher, M. Defeasible Reasoning via Datalog. *Theory and Practice of Logic Programming*. 2021. pp. 1-43. DOI:10.1017/S1471068421000387
9. Komendantskaya, E., & LI, Y. Productive corecursion in logic

programming. *Theory and Practice of Logic Programming*, 2017. 17 (5-6), pp. 906-923. DOI:10.1017/S147106841700028X

10. Sozeau M, Boulier S., Forster Ya., Tabareau N., Winterhalter T. Coq Coq Correct! Verification of Type Checking and Erasure for Coq, in Coq. *Proc. ACM Program. Lang*. 4, POPL, Article 8 (January 2020), 28 pages. DOI:10.1145/3371076
11. Шаповалова С. І., Мажара О. О. Визначення ефективності механізмів логічного виведення. Системи управління, навігації та зв'язку. 2020. Вип. 4 (62). С.81- 87. DOI:10.26906/SUNZ.2020.4.081
12. Shapovalova S. Generation of test bases of rules for the analysis of productivity of logical inference engine. *Innovative Technologies and Scientific Solutions for Industries*. 2020. No. 3(13). P. 88–96. DOI:10.30837/ITSSI.2020.13.077
13. Sottara D., Mello, P. Proctor, M. A Configurable Rete-OO Engine for Reasoning with Different Types of Imperfect Information. *IEEE Trans. on Knowledge and Data Engineering*. 2010. Vol. 22, No. 11, pp. 1535–1548
14. Kim M., Lee K., Kim Y., Kim T., Lee Y., Cho S., Lee C.-G. RETE-ADH: An improvement to RETE for composite context-aware service. *Int. Journal of Distributed Sensor Networks*, 2014, Vol. 2014, Article ID 507160, p. 11. DOI:10.1155/2014/507160
15. Lee R. and Cho S.. Rete-ECA: A Rule-based System for Device Control. *Proceedings of the International Workshop on Artificial Neural Networks and Intelligent Information Processing*. 2014, pp. 95-102. DOI: 10.5220/0005140500950102
16. Proctor M., Fusco M., Vacchi E., Sottara, D. (2019). Rule modularity and execution control enhancements for a Java-based rule engine. *Proc. of IEEE 2nd International Conference on Artificial Intelligence and Knowledge Engineering, AIKE-2019*, pp. 89-96. DOI: 10.1109/AIKE.2019.00023
17. Calegari R., Denti E., Mariani S., Omicini A. Logic programming as a service. *Theory and Practice of Logic Programming*, 2018. 18(5-6), pp. 846-873. DOI:10.1017/S1471068418000364
18. Wielemaker J., Riguzzi F., Kowalski R., Lager T., Sadri F., Calejo M. Using SWISH to realise interactive web based tutorials for logic based languages. Theory and Practice of Logic Programming. 2019. 19(2), pp. 229-261. DOI: 10.1017/S1471068418000522
19. Manhaeve R., Dumančić S., Kimmig A., Demeester T, De Raedt L. Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence. 2021*. Vol. 298, 103504. DOI:10.1016/j.artint.2021.103504

ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ ЛОГІЧНОГО ВИВЕДЕННЯ В ПРОГРАМНИХ КОМПЛЕКСАХ

Аушева Н.М., Шаповалова С.І.

Об'єктом дослідження є технології реалізації логічного виведення. Метою статті є визначення основних напрямів досліджень з автоматичного логічного виведення, отриманих на їх основі програмних засобів та відповідних областей застосування.

*Логічне виведення є розповсюдженою задачею штучного інтелекту, яка реалізується в прикладному програмному забезпеченні. Оперативність роботи програмного комплексу, зокрема, визначається ефективністю вбудованого механізму логічного виведення. Засоби реалізації логічного виведення мають забезпечувати оптимальний час виконання і взаємодії з іншими компонентами, а також задовольняти вимогам прикладної задачі. В статті виокремлено основні галузі вдосконалення програмних реалізацій механізмів логічного виведення та відповідні напрями досліджень: (*i*) розширення концепції та поєднання декількох парадигм логічного програмування (ймовірнісне логічне програмування, нездатне логічне виведення, коіндуктивне програмування); (*ii*) скорочення часу обміну даними між компонентами програмного комплексу та часу обробки великих баз знань (вдосконалення моделей співставлення зі зразком в системах, що базуються на правилах, створення або розширення реалізацій засобів логічного виведення для інтеграції в розподілені програмні системи, розробку нових реалізацій механізмів логічного виведення для спеціалізованих мов програмування та мов загального призначення); (*iii*) поєднання концептуально різних підходів до логічного виведення (поєднання обох підходів до виведення за логічною та продукційною концепціями, інтеграцію парадигм логічного виведення та нейромережевого підходу). Саме поєднання програмних реалізацій різних концепцій, насамперед, механізмів формування логічних висновків та нейромережевих моделей, надає нові можливості штучному інтелекту.*

Для кожного напряму досліджень представлено програмні засоби, також наведено приклади галузей їх застосування відповідно зазначеній концепції.

Ключові слова: механізми логічного виведення, ймовірнісне логічне програмування, нездатне логічне виведення, системи, що базуються на правилах, PRISM, ProbLog, XSB Prolog, SWI Prolog, DeepProbLog.

ТЕХНОЛОГИИ РЕАЛИЗАЦИИ ЛОГИЧЕСКОГО ВЫВОДА В ПРОГРАММНЫХ КОМПЛЕКСАХ

Аушева Н.Н., Шаповалова С.І.

Объект исследования - технологии реализации логического вывода. Цель статьи - определение основных направлений исследований автоматического логического вывода, полученных на их основе программных средств и соответствующих областей применения.
Логический вывод является распространенной задачей искусственного интеллекта, реализуемой в прикладном программном обеспечении. Оперативность работы программного комплекса определяется эффективностью встроенного механизма логического вывода. Средства реализации логического вывода должны обеспечивать оптимальное время выполнения и взаимодействия с другими компонентами, а также удовлетворять требованиям прикладной задачи. в статье выделены основные области усовершенствования программных реализаций механизмов логического вывода и соответствующие направления исследований: (i) расширение концепции и сочетание нескольких парадигм логического программирования (вероятностное логическое программирование, оспоримый вывод - defeasible inference, коиндуктивное программирование); (ii) сокращение времени обмена данными между компонентами программного комплекса и времени обработки больших баз знаний (усовершенствование моделей сопоставления с образцом в системах, базирующихся на правилах, создание или расширение реализаций средств логического вывода для интеграции в распределенные программные системы, разработка новых реализаций механизмов логического вывода для специализированных языков программирования и языков общего назначения); (iii) сочетание концептуально разных подходов к логическому выводу (сочетание выводов по логической и продукционной концепциям, интеграция парадигм логического вывода и нейросетевого подхода). Само сочетание программных реализаций различных концепций, прежде всего механизмов формирования логических выводов и нейросетевых моделей, предоставляет новые возможности искусственному интеллекту.

Для каждого направления исследований представлены программные средства, также приведены примеры отраслей их применения согласно указанной концепции.

Ключевые слова: механизмы логического вывода, вероятностное логическое программирование, оспоримый логический вывод, системы, базирующиеся на правилах, PRISM, ProbLog, XSB Prolog, SWI Prolog, DeepProbLog.

References

1. Hecham, A., Croitoru. M., & Bisquert, P. (2018). A First Order Logic Benchmark for Defeasible Reasoning Tool Profiling. *RuleML+RR*, Luxembourg, 81-97. DOI:10.1007/978-3-319-99906-7_6.
2. Hecham, A., Croitoru, M., & Bisquert, P. (2018). On a flexible representation for defeasible reasoning variants. *Proc. of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'18)*, 1123–1131.
3. Shapovalova, S. (2020). Formalization of the Rules of Inference in Erlang. *Mathematical and computer modelling. Series: Technical sciences*, 21, 125-139. DOI:10.32626/2308-5916.2020-21.125-139 [in Ukrainian]
4. Alberti, M, Lamma, E, Riguzzi, F., & Zese, R. (2017). A Distribution Semantics for non-DL-Safe Probabilistic Hybrid Knowledge Bases. *Proc. of the Workshop on Probabilistic Logic Programming (PLP 2017),* CEUR, Vol. 1916, 40-50.
5. Nampally, A., Zhang, T., & Ramakrishnan C. (2018). Constraint-Based Inference in Probabilistic Logic Programs. *Theory and Practice of Logic Programming*, 18(3-4), 638-655. doi:10.1017/S1471068418000273
6. Vandenbroucke, A., & Schrijvers, T. (2017). From PRISM to ProbLog and Back Again. *Proc. of the Workshop on Probabilistic Logic Programming (PLP 2017),* CEUR, Vol. 1916, 26-40.
7. Riguzzi, F., Wielemaker, J., & Zese R. (2018). Probabilistic Inference in SWI-Prolog. *Proc. of the Workshop on Probabilistic Logic Programming (PLP 2018),* CEUR, Vol. 2219, 15-27.
8. Maher, M. (2021). Defeasible Reasoning via Datalog¬. *Theory and Practice of Logic Programming*, 1-43. DOI:10.1017/S1471068421000387
9. Komendantskaya, E., & LI, Y. (2017). Productive corecursion in logic programming. *Theory and Practice of Logic Programming*, 17(5-6), 906-923. DOI:10.1017/S147106841700028X
10. Sozeau, M, Boulier, S., Forster, Ya., Tabareau, N., & Winterhalter, T. (2020). Coq Coq Correct! Verification of Type Checking and Erasure for Coq, in Coq. *Proc. ACM Program. Lang*. 4, POPL, Article 8 (January 2020), 28 pages. DOI:10.1145/3371076
11. Shapovalova, S., Mazhara, O. (2020). Measuring efficiency of inference engines. *Control, Navigation and Communication Systems,* 4 (62), 81-87 [in Ukrainian]
12. Shapovalova, S. (2020). Generation of test bases of rules for the analysis of productivity of logical inference engine. *Innovative Technologies and Scientific Solutions for Industries,* 3(13), 88–96. DOI:10.30837/ITSSI.2020.13.077

13. Sottara, D., Mello, P. & Proctor, M. (2010). A Configurable Rete-OO Engine for Reasoning with Different Types of Imperfect Information. *IEEE Trans. on Knowledge and Data Engineering*. *22*, 11, 1535–1548
14. Kim, M., Lee, K., Kim, Y., Kim, T., Lee, Y., Cho, S., & Lee, C.-G. (2014). RETE-ADH: An improvement to RETE for composite context-aware service. *Int. Journal of Distributed Sensor Networks*, 2014, Article ID 507160, p.11. DOI:10.1155/2014/507160
15. Lee, R. & Cho, S. (2014). Rete-ECA: A Rule-based System for Device Control. *Proceedings of the International Workshop on Artificial Neural Networks and Intelligent Information Processing*, pp. 95-102. DOI: 10.5220/0005140500950102
16. Proctor, M., Fusco, M., Vacchi, E., & Sottara, D. (2019). Rule modularity and execution control enhancements for a Java-based rule engine. *Proceedings of IEEE 2nd International Conference on Artificial Intelligence and Knowledge Engineering (AIKE-2019)*, 89-96. DOI: 10.1109/AIKE.2019.00023
17. Calegari, R., Denti, E., Mariani, S., & Omicini, (2018). A. Logic programming as a service. *Theory and Practice of Logic Programming*, 18(5-6), 846-873. DOI:10.1017/S1471068418000364
18. Wielemaker, J., Riguzzi, F., Kowalski, R., Lager, T., Sadri, F., & Calejo, M. (2019). Using SWISH to realise interactive web based tutorials for logic based languages. *Theory and Practice of Logic Programming*. 19(2), 229-261. DOI: 10.1017/S1471068418000522
19. Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T, & De Raedt, L. (*2021)*. Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence*. 298, 103504. DOI:10.1016/j.artint.2021.103504