Proceedings of the Conference on Mathematical Foundations of Informatics
MFOI2016, July 25-29, 2016, Chisinau, Republic of Moldova

On computer aided knowledge discovery in logic and related areas

Andrei Rusu, Elena Rusu

Abstract

The present paper describes the architecture of a software for computer aided knowledge discovery dealing with the problems of functional expressibility of formulas in a nonstandard propositional logic. The achieved system rests on a distributed agent-based platform. Some software agents have been designed in order to solve some particular problems, the user interact with them via a protocol provided by an interface software agent. The methods used by some agents to solve some particular problems are based on technics inspired from nature: genetic programming. The agent framework JADE used for these objectives is a FIPA-complained open-source agent platform.

Keywords: non-classic propositional logic, expressibility of formulas, software agents, genetic programming, symbolic regression.

1 Introduction

The well-known problem of expressibility of a formula via a system of formulas in a given logic is considered [1, 2]. E. Post have investigated this problem and the related ones in the case of classical propositional logic [3].

The purpose of this paper is to present an agent-based system that can be used to asist the researcher in investigation of the above mentioned problem of expressibility and related to it ones in an arbitrary propositional calculus. As a software platform for this task we consider the open-source distributed software agent platform JADE [4].

©2016 by Andrei Rusu, Elena Rusu

2 Problems related to expressibility of formulas

Consider a propositional logical calculus L. It is defined usually by an alphabet of used symbols (propositional variables, i.e. $p, q, r, \ldots$, and a set of logical connectives, i.e. in the case of classical propositional logic $\&, \vee, \supset, \neg$), by formulas based on the given alphabet, by axioms (which are some formulas of the calculus L) and by some rules of inference (one well-known rule in classical propositional logic is the Modus Ponens rule, which allows passing from two formulas A and $A \supset B$ to the formula B) [5].

They say formula F is a consequence of the formulas $G_1, \ldots, G_n$ (formulas $G_1, \ldots, G_n$ are said to be hypotheses for F) in the logic L, denoted by

$$G_1, \ldots, G_n \vdash F,$$

if there is a sequence of formulas $F_1, \ldots, F_k$ with the following properties:

- F_k is actually formula F,
- for any index i, $i = 1, \ldots, k$ at least one of the following takes place:
  - F_i is an axiom of L,
  - $F_i \in \{G_1, \ldots, G_n\}$,
  - F_i is obtained from $F_{i_1}, \ldots, F_{i_j}$ by some rule of inference of L, where $\{i_1, \ldots, i_j\} \subseteq \{1, \ldots, i-1\}$

In the case $\{G_1, \ldots, G_n\} = \emptyset$ they say formula F is a theorem of L, denoted by

$$\vdash F.$$

Formulas F and G are said to be equivalent in L, denoted by $F \sim G$ if the following relations are valid in L:

$$F \vdash G,$$
$$G \vdash F.$$

Usually the equivalence of the formulas F and G in L is an equivalence relation over formulas of L [6].

They say [1, 2, 7] the formula F is expressible via a system of formulas Σ in the calculus L if there exists a finite sequence of formulas $F_1, \ldots, F_n$ (called expression of F in L) with the following properties:

- F_n is actually formula F,
- for any $i = 1, \ldots, n$ at least one of the following holds:
  - F_i is a variable;
  - F_i belongs to Σ;
  - F_i is obtained from some previous formulas in the sequence using:
    - the week substitution rule, which allows passing from two formulas A and B to the result of substitution of B in A instead of every occurence of a given variable p of A (denoted by $A[p/B]$, or $A[B]$);
    - the rule of replacement by an equivalent formula in L, which permits passing from formula A to formula B if formulas A and B are equivalent in L.

A system of formulas Σ is said to be complete with respect to expressibility of formulas in the calculus L if any formula of L is expressible via formulas of Σ. The system Σ is known to be pre-complete (relative to expressibility of formulas) in L if Σ is not complete in L, but for any formula F which is not expressible in L via Σ the system $\Sigma \cup \{F\}$ is already complete in L [1].

Now there are some general problems related to expressibility of formulas [2]:

- The problem of expressibility of a formula F via a system of formulas Σ in L is one mentioned for boolean funtions by Emil Post [3].

- The problem of completeness relative to expressibility in L of system of formulas Σ.
- different variations of the above problems regarding formula bases.

In order to approach the above problems there is usually a need to perform multiple various calculations.

3 Tools and main ideas

We use Genetic Programming (GP) to discover new knowledge about expressibility problems in logic L and combine this technique with an agent system based on Jade framework. Genetic Programming is a computational method inspired by biological evolution, which discovers computer programs tailored to a particular task [8]. GP maintains a population of individual programs. Computational analogs of biological mutation and crossover produce program variants. Each variant's suitability is evaluated using a user-defined fitness function depending on the concrete problem related to expressibility of formulas in the given logic L.

The main steps of a genetic programming algorithm are:

1. **Preparatory steps** to specify:
   (a) the set ot terminals, usually these are variables;
   (b) the set of primitive functions, i.e. initial formulas;
   (c) the fitness measure (for explicitly or implicitly measuring the fitness of individuals in the population);
   (d) certain other parameters for controlling the execution of the algorithm;
   (e) the termination criterion
2. **Execution steps** of the algorithm are:

(a) create in random fashion an initial population, i.e the 0-generation, of formulas composed of the available formulas;

(b) iterate the following steps on the population until termination criterion is fulfilled:

(a) Evaluate each formula in the population according to the fitness of the desired problem;

(b) Select necessary amount of individuals with a probability based on fitness to participate in the genetic operations at the next step;

(c) Create new individuals for the population by applying the following genetic operations with specified probabilities:

i. *Reproduction:* Copy the selected individual to the new population;

ii. *Crossover:* Create new individuals by random recombination of the randomly chosen parts of the selected individuals;

iii. *Mutation:* Create one new offspring formula for the new population by randomly mutating a randomly chosen part of one selected formula;

iv. *Architecture-altering operations:* Choose an architecture-altering operation from the available ones and create one new offspring formula for the new population by applying the chosen architecture-altering operation to one selected formula.

3. When termination criterion is fulfilled, the single best formula in the population produced during the execution is considered the result of the algorithm. If the execution is successful, the result formula may be a solution to the problem.

4 The agent system based on Jade

The idea of the built software is to provide intelligent software agents which can assist researcher in investigating problems related to problems of expressibility of formulas mentioned in previous section. A short introduction into the domain of agent-oriented software engineering can be consulted in [9].

Thus we consider agents for various particular tasks related to problems of expressibility of formulas in a propositional logic a researcher comes in touch with, such as:

- What models has the given propositional calculus?
- Does a given formula F of L conserves a given relation R on the given model?
- Which unary formulas are in a given class of formulas defined by a predicate on an algebra?
- Is the system of formulas Σ complete with respect to expressibility in the given logic L?
- Is the formula F expressible in the given logic L via the given system of formulas Σ?
- etc.

In order to answer the above mentioned questions we design various agents, some of them implement intelligent search based on genetic programming algorithm to do symbolic regression.

The designed agents respects the standards of The Foundation for Intelligent Physical Agents (FIPA), which is a body for developing and setting computer software standards for heterogeneous and interacting agents and agent-based systems [10]. As a software platform for the designed agents we consider the open-source Java Agent DEvelopment Framework, or JADE, which is a software framework for the development of intelligent agent, implemented in Java [4]. JADE provides:

- An environment where JADE agents are executed.
- Class Libraries to create agents using heritage and redefinition of behaviors.
- A graphical toolkit to monitoring and managing the platform of Intelligent Agent agents.

and is a distributed agents platform, which has a container for each host where agents are running. Additionally the platform has various debugging tools, mobility of code and content agents, the possibility of parallel execution of the behavior of agents, as well as support for the definition of languages and ontologies.

Depending on the concrete problem related to the expressibility of formulas in the given logic L the agents cooperate to solve it or to assist the researcher to solve it.

5 Conclusion

The developed multi-agent system for support for knowledge discovery in logic is robust, extensible, relatively simple to implement and new features can be easily added to it. Each agent can be implemented to support different intelligent behaviour depending on various situations. Since the system is based on Java it can run on almost all platforms.

The present research can also be used in other fields of interest, for example, for measuring the impact of science research for different actors in Moldova, using the established National Bibliometric Tool (`http://ibn.idsi.md/`).

Acknowledgments. Information Society Development Institute have supported part of the research for this paper in SCIFORM project Ref. Nr. 15.817.06.13A.

References

[1] A. V. Kuznetsov, *Über funktionelle Ausdrückbarkeit in superintuitionistischen Logiken.* Mat. Issled., vol. 6, no. 4 (1971), pp. 75–122.

[2] M. F. Ratsa, *Expressibility in propositional calculi.* Chişinău: Ştiinţa, 1991, 204 pp, ISBN: 5-376-00961-0.

[3] E. L. Post, *Two-valued iterative systems of mathematical logic.* Annals of Mathematics Studies, no. 5, Princeton University Press, Princeton, N.J., 1941, 122 p.

[4] JAVA Agent DEvelopment Framework, `http://jade.tilab.com/`.

[5] Mendelson, E., *Introduction to Mathematical Logic.* Chapman and Hall/CRC, 5th ed, 2010.

[6] W. J. Blok, D. Pigozzi, *Algebraizable Logics.* Memoirs of the American Mathematical Society, vol.77, no. 396 (1989), 89 p.

[7] A. V. Kuznetsov, *Analogs of the Sheffer stroke in constructive logic.* Sov. Math., Dokl., vol. 6 (1965), pp. 70–74.

[8] Koza, J. R. *Genetic programming: on the programming of computers by means of natural selection.* MIT Press, 1992.

[9] Wooldridge, M., Ciancarini, P. *Agent-oriented Software Engineering: The State of the Art.* in: First International Workshop, AOSE 2000 on Agent-oriented Software Engineering, Springer-Verlag New York, Inc., 2001, pp. 1-28.

[10] IEEE Foundation for Intelligent Physical Agents, `http://www.fipa.org/`.

Andrei RUSU[1,2], Elena RUSU[2]

[1]Information Society Development Institute
Email: andrei.rusu@idsi.md

[2]Ovidius University of Constanta
Email: agrusu@univ-ovidius.ro

[3]Technical University of Moldova
E–mail: elenarusu2006@yahoo.com