Minimizing Entropy of Knowledge Representaion

Anatol Slissenko[1]
Dept. of Informatics, University Paris-12, France
and
Institute for Informatics, Academy of Sciences of Russia, St-Petersburg, Russia

Abstract

If to compare the existing general purpose theorem provers, which efficiency is more than modest, and human way of reasoning, which is incomparably more efficient, one can see some principal distinction in knowledge organization in both cases. We give a notion of entropy which permits to set up a thesis that an efficient knowledge organization adheres to the principle of minimizing the entropy. Within the same framework one can define various quantitative characteristics of systems of knowledge. A practical consequence of this point of view is strategy-oriented architecture of theorem provers. We mention an experimental implementation of such an architecture that demonstrated promising results.

0. Introduction

The topic under consideration is motivated by inefficiency of automatic theorem-provers clearly understood already in the 60ies. Since then there were no principal progress in the development of architectures of theorem provers, though the modern ones (one can find many referencies at http://www-formal.stanford.edu/clt/ARS/ars-db.html) are incomparably more powerful then those made by pioneers [AR83a, AR83b]. But the relative power of modern theorem provers is a consequence of development of computers and programming but not of fundamental ideas. The strategy used even in modern theorem provers represent general syntactic observations that practically do play no role in non trivial proofs. They are equally applicable to any domain. On the other hand, to learn to reason in a particular domain takes a lot of time, and moreover, the structure of knowledge of any particular domain develops, as well as methods of search for proofs, as a result of considerable intellectual investments. E. g. compare a two page formalisation of calculus and a handbook on the same subject. Theoretical properties of more or less powerful logics that represent our knowledge, in particular their universality, signify that no general purpose proof search can be efficient. The universality means that any algorithmic difficulty can be modeled in these logics; there are even individual provable propositions that are hard to handle (cf. [Sli75]). These considerations mean that the only way to achieve the efficiency of proof search is to organize the basic knowledge of concrete domains and the knowledge about proof search in these domains in an efficient way. But constructing such a knowledge representation needs adequate tools to restructure this knowledge. We are going to discuss one theoretical point of view and one practical approach to develop such tools.

In the present communication we briefly develop a system of notions that permits to give quantitative characteristics of systems of knowledge representation. The latter are represented as inference systems defined in Section 1. Then in Section 2 we introduce proof complexity that can serve to define various measures of quality of systems of knowledge representation (see [Sli91]). Here we speak only about entropy. The last Section 3 touches a strategy based theorem proving.

[1] *Address:* Dept. of Informatics, University Paris-12, 61 Av. du Gén. de Gaulle, 94010, Créteil, France.
E-mail: slissenko@univ-paris12.fr

We say several words about an experimental computer system that implements a strategy-oriented architecture of theorem proving.

1. Inference Systems

The notion of an infrence system we introduce below, is motivated by knowledge systems where the amount of rules or axioms related to the domain under consideration is much larger than individual proofs we are looking for. As examples of such systems one can take handbooks on particular domains of mathematics or physics, or large expert systems for situation analysis. We look at knowledge representation and processing system as having 2 levels: the first (lower) level contains basic knowledge that represents the problem domain we are interested in, and this knowledge must be precise and reliable, and the second (higher) level consists of tools to describe methods of solving problems under consideration. The lower level knowledge representation will be modeled below as *inference system.*

Let some standard logic syntax be fixed, a syntax of applied predicate logic language with predicate and functional symbols is suitable. To simplify the presentation we will speak only about proving formulas though the approach works also for queries **find** x **such that** $F(x)$ that are of more practical interest.

A *semantics* of the language is defined as a set of inference rules that constitute a knowledge base of the lower level we mentioned above. The main object to manipulate with is a list of formulas each one labeled as *assumption* or *conclusion*, and supplied with an information concerning its origin (an axiom, a result of application of a rule). The latter information will be called *the analysis.* A formula labelled as an assumption and conclusion corresponds to a trivial logical axiom of the form $F \vdash F$. An *inference rule* is an algorithm transforming a list of formulas from its domain into an extension of this list.

A list L_1 is a *result of an application of a rule* r to a list L if L_1 can be constructed due to the following prescription. Take any sublist K of L belonging to $dom(r)$. The formulas of $r(K)$ consist of all the formulas of K, maybe with changed labels. Denote these formulas corresponding to K by K_1, and the rest by $K_2 = r(K) \setminus K_1$. Those formulas of K_2 that are marked as assumptions are catenated with L as its prefix, and those marked as conclusions are catenated as suffix of the L. The appropriate information about the origin of the formulas of $r(K)$ is ascribed to them.

A system consisting of a syntax and of a semantics is called an *inference system.* Elsewhere below S is an inference system.

Proof in S *or* S*-proof* is a list of formulas of the form described above. I. e. every formula of the list is marked as assumption or conclusion, and is provided with its analysis when marked as conclusion. The proper ancestors of a conclusion are situated to the left of the conclusion itself. Without loss of generality we consider only proofs not containing repetitions of formulas.

The next step to develop the setting needs a notion of complexity. We start with fixing conventional notations.

The length of a string W is denoted by $|W|$. For a finite set E of strings $|E|$ denotes the sum of the lengths of the elements of E. If S is a set of algorithms then we mean that its elements are represented as strings, and we treat $|S|$ according to this representation.

By $ASSUMP(L)$ and $CONCL(L)$ we denote respectively the set of the formulas of a list L marked as assumptions and the set of the formulas of L marked as conclusions. And as a measure of size we will use $lc(L) = |CONCL(L) \backslash ASSUMP(L)|$.

Let Φ and Γ be finite sets of formulas. Denote by $PRF_S(\Phi \mid \Gamma)$ the set of all S-proofs D such that $\Phi \subseteq CONCL(D)$ and $ASSUMP(D) \subseteq \Gamma$

A set of formulas or a formula Φ is *provable in* S or S*-provable* from formulas (assumptions) Γ

if $PRS_S(\Phi \,|\, \Gamma) \neq \emptyset$, and Φ is *S-provable* if it is provable with empty Γ.

2. Measuring Quality of Knowledge Representation

Proof Complexity

The principal notion for the setting under consideration is *proof complexity* of a set of formulas Φ under assumptions Γ:

$d_S(\Phi \,|\, \Gamma) = \min\{lc(D) \colon D \in PRF_S(\Phi \mid \Gamma)\}, \quad d_S\,(\Phi) = d_S(\Phi \mid \emptyset)$.

Here we assume that $\min \emptyset = \infty$.

The function d_S can be treated as a one-directional metrics; a simple symmetrization like $d_S(\Phi \,|\, \Psi) + d_S(\Psi \,|\, \Phi)$ gives a genuine *metrics* on the set of finite sets of formulas. One can see that the measure d_S is a generalisation of known measures, and that an appropriate choice of S can lead as to measures like time complexity as well as to measures of the type of Kolmogorov complexity.

One can prove the following properties of d_S:

$d_S(\Phi \,|\, \Gamma) = 0, \quad d_S(\Phi \,|\, \Phi) = 0,$

$\Phi \subseteq \Psi \Rightarrow d_S(\Phi \,|\, \Gamma) \leq d_S(\Psi \,|\, \Gamma), \quad \Gamma \subseteq \Delta \Rightarrow d_S(\Phi \,|\, \Gamma) \geq d_S(\Phi \,|\, \Delta),$

$\max\{d_S(\Phi \,|\, \Gamma),\, d_S(\Psi \,|\, \Gamma)\} \leq d_S(\Phi \cup \Psi \,|\, \Gamma) \leq d_S(\Phi \,|\, \Gamma) + d_S(\Psi \,|\, \Gamma).$

Entropy of a Set of Formulas

We assume that a reasonable notion of *inconsistency* of an inference system is introduced. Two inference systems are *consistent* if the same is valid for their union. Since any formula can be treated as an inference rule, i.e. as an axiom, these notions can be extended to sets of formulas.

Let Φ be a finite set of formulas, S and V be inference systems, and ξ be a natural number. The notion of entropy introduced below has a flavour similar to the notion of ε-entropy of compacts in metric spaces:

$entr_S(\Phi \,|\, V, \xi) = \min\{|U| \colon U \text{ is consistent with } S \text{ and } \forall\, F \in \Phi\, (d_{V \cup U}\,(F) \leq \xi)\},$

$entr_S(\Phi, \xi) = entr_S(\Phi \,|\, \emptyset,\, \xi).$

One can prove that the entropy is not descreasing in Φ, not increasing in V and semi-additive in Φ.

A system S is said to be *optimal* for Φ and a given ξ if $entr_S(\Phi \mid S, \xi) = 0$ and $entr_S(\Phi, \xi) = len(S)$.

The value of the entropy is, apparently, very sentitive to the class of admissible systems U. But on the whole it is clear that we improve quality of the inference system if we use effective rules with wide domain of applications. This consideration and observations concerning developments of theories permits us to forth

Thesis. *The quality of an inference system to solve a set of problems increases with diminishing of the entropy of this set with respect to the inference system.*

The thesis implicitely presumes some value of the parameter ξ that is usally small. Surely, one can consider more adequate versions of entropy with some averaging and so on.

3. Strategy Based Inference Search

Structuring of knowledge to improve the efficiency of its representation may involve highly creative activity such as inventing new approaches, profound new theories etc. But there is more modest activity that corresponds, say, to the way an undergraduate student learns to solve some problems. For example, the problems of finding symbolic limits. The general problem is undecidable. However, for sets of problems containing rather non trivial ones, the student manages to develop strategies of solving them. Such a strategy gives a rule that considerably ameliorates the efficiency of the corresponding knowledge representation. To support the development of strategies one can try to make a computer tool with a language to represent strategies and with possibilities to easily modify them, preferably on-line. A system of this kind had been implemented by my PhD student V. V. Tarasov [Tar96] on the ideas mentioned in [Sli91] and considerably developed by us during this work. It deals with symbolic limits, but can be tuned to similar domains based on mathematical terms manipulations.

The system gives tools to represent user's semantical considerations by looking for particular patterns in formulas and proofs. In fact, the common basis for strategy representation language is a language of occurrences. For example, for the problem of symbolic limits, we try first of all to classify the term under treatment, say, as polynomial, rational function, term containing trigonometric functions and so on. We look for subformulas with knowm limits. We try to commute the lim-operator with functions etc. In terms of such a language we can describe some features of formulas to analyze. And in addition we need algorithms to detect the desired patterns. To facilitate the construction of strategies we use meta-rules of inference search that represent common local strategies to try to apply.

The system [Tar96] contains tools to define a language to describe the basic rules, a language to describe patterns, control predicates, control functions and control rules. There are also additional means to organize, say, backtracking. In a concrete knowledge base the basic rule $\lim_{x\to\infty} \sin \frac{c}{x} = 0$ is written as `TSUBST(SL_sincdivx_inf;lim(arg(u,inf),sin(div(c,u)));0;)`. Among control predicates one can find "The expression of the limit operator occurring in a formula F contains tg$x$, ctg$x$, sec$x$ or cosec$x$" or "The expression of the limit operator occurring in a formula F is a difference or a sum of fractions". All this machinery permit to write a one page strategy that finds many limits among them $\lim_{n\to\infty} ((n+1)^\alpha - n^\alpha)$, where $\alpha \in \mathbf{R}$ and $0 < \alpha < 1$. The latter is not trivial.

References

[AR83a] *The Automation of Reasoning I. Classical Papers on Computational Logic 1957–1966.* Springer Verlag, 1983.

[AR83b] *The Automation of Reasoning II. Classical Papers on Computational Logic 1967–1970.* Springer Verlag, 1983.

[Sli75] A. Slissenko. Finite approach to the problem of optimizing theorem-proving algorithms. *Zapiski Nauchnykh Seminarov LOMI*, 49:123–130 (Russian), 1975. J. of Soviet Mathematics, 10:4(1978):597–603 (English).

[Sli91] A. Slissenko. On measures of information quality of knowledge processing systems. *Information Sciences: An International Journal*, 57–58:389–402, 1991.

[Tar96] V. V. Tarasov. *Inference search control in expert systems based on explicit meta-rules of inference search.* PhD thesis, St. Petersgurg Institute for Informatics and automation, Russian Academy of Sciences, 1996. (Russian.).