ICOT Technical Report: TR-692

TR-692

The Approximate Reasoning in Logic Programming

by
J. Yamaguchi (Kanagawa Univ.)

September, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

The Approximate Reasoning in Logic Programming

Jinsei Yamaguchi

Dept. of Information and Computer Sciences
KANAGAWA UNIVERSITY
2946 Tsuchiya, Hiratsuka
Kanagawa 259-12 JAPAN

ABSTRACT: We study theoretical aspects of the approximate reasoning within the framework of logic programming from a viewpoint of AI. Firstly, several new examples of the approximate reasoning are proposed. Then, three crucial issues, which seem to have never been discussed (seriously) by any conventional theory concerning the approximate reasoning in general, are argued. They would become essential when we embody any approximate reasoning from an angle of our common sense reasoning. The solution, which simultaneously answer all of them, is briefly suggested. It is given by using the methodology of Boolean-valued semantics and the strategy can be reviewed via the paradigm of the cooperatively distributed AI (problem-solving).

KEY WORDS: Logic Programming, Approximate Reasoning, Fuzzy Inference
Analogical Reasoning, Cooperatively Distributed AI

§0. Introduction

In recent years, there are many trials to incorporate a common sense reasoning into the paradigm of logic programming. For example, fuzzy logic programming is the one to combine a fuzzy inference and a logic program. Yet another is the analogical reasoning based on a logic programming technique. Moreover, any logic program + an equational theory may also fall into this category. From an abstract viewpoint, all these programming strategies have a common framework such that

a logic program P + an inference rule based on

the approximate reasoning

$$\frac{A',\ A \to B}{B'} : A' \sim A \text{ and } B' \sim B \qquad \cdots (1)$$

where $\sim$ is a reflexive, symmetric relation over a set of all atoms generated from P.

Remark: Especially, in the case of P + an equational theory (or TRS), $\sim$ becomes an equivalence relation. However, in general, $\sim$ may not satisfy the transitive law. Precise examples are discussed in the next section.

In the following, we investigate the theoretical aspect of this kind of approximate reasoning from a viewpoint of AI.

§1. Several Examples of the Approximate Reasoning

In this section, we present several examples of the approximate reasoning in general. Since the cases of fuzzy logic and the equational theory are rather familiar, we omit examples in these classes.

I. The case that $\sim$ is an equivalence relation

(i) partial identification

Let A(x,…) be a predicate whose arity is more than one. The simplest of the notion of the "partial identification" is, for example,

$A(s,\cdots) \sim A(u,\#)$ iff $s = u$.

Though this kind of partial identification can be defined by means of an equational theory, there are other examples in this category which can't be defined by any equational theory. For example,

$A(s,\cdots) \sim A(u,\#)$ iff both s and u satisfy a property P(x), say, $40 \leqq x \leqq 50$ when the terms represent ages.

Remark: In the abstract examples in this section, we intend that all terms used are ground. In the case that a term contains variables, we employ the technique of lazy evaluation or constraint solving. The novel and remarkable point of using constraint solving in this context is that,

instead of using constraints as parts of programs which are to be solved, we use them to define $\sim$ at the stage of the approximate reasoning. Thus, they might be in a position of the query-driven constrains for the same program. Precise argument concerning this aspect of the approximate reasoning will appear in the other paper. Also, see the discussion of §4.

Practically, the number of the chosen predicates $A_1,\cdots,A_n$ used in this context are finite, and this concept of the partial identification would be useful for, say, a fuzzy deductive database query. (Compare this technique with the usual algebraic method employed by the relational database.) As a little more flexible concept in this direction, we get

(ii) partial pattern matching

In the situation of (i),
$A(s,\cdots)\sim A(u,\#)$ iff s and u have the same pattern.

Here, the notion of the "pattern" can be applied to not only a visual and/or an auditory but also a literal and/or a symbolic pattern including a chemical and/or a biological structure. Note that, in the above, s(u) itself need not be any kind of pattern. It is sufficient that s "points" a pattern. By using this notion,we can connect the approximate reasoning with the conventional realizing algorithm of the mathematical computation.

(iii) qualitative reasoning

Let D be a target domain of data or informations. Divide D into pairwise disjoint subclasses $D_1,\cdots,D_n$ by somehow or other. Let $d_1,\cdots,d_n$ be the representatives (typical examples) or the class(or the category or the concept) names of $D_1,\cdots,D_n$. Write a logic program P by using $d_1,\cdots,d_n$. At the same time, define $\sim$ so that, for any $s,t\in D$,

$s\sim t$ iff s and t belong to the same class, say, D_i.

In this case, we naturally extend $\sim$ so that, for any $s\in D$,

$s\sim d_i$ iff $s\in D_i$.

This kind of qualitative inference is interesting from a viewpoint of cognitive science.

II . The case that ~ is not transitive

(i) disjunctive inference

Let $A(x,y,\cdots)$ be a predicate whose arity is more than two. Define ~ so that

$A(s,t,\cdots)\sim A(u,v,\#)$ iff (s and u are partially identified) *or*
(t and v are partially identified)

Obviously, thus defined relation ~ is not transitive. (For example, $A(c_1,c_2,\cdots)\sim A(c_1,c_3,\cdots)$ and $A(c_1,c_3,\cdots)\sim A(c_4,c_3,\cdots)$, where c_1,c_2,c_3,c_4 are constants. However, $A(c_1,c_2,\cdots)\sim A(c_4,c_3,\cdots)$ does not hold.)
Let call an approximate reasoning defined by the above kind technique "the disjunctive inference". Again, a logic programming based on this kind of approximate reasoning is suitable for a theoretical background of fuzzy deductive database query languages, and/or searching answers in set form. One feature of this kind of approximate reasoning is that, in an inference step (1), the similarity relation ~ of A,A' and that of B,B' can be defined independently as initial conditions (for each query). So, the connection between them is accomplished via → only through message passing, triggered by substitutions which result the pairs (A,A') and (B,B'). This property contrasts with the character discussed in the following (iv).

(ii) generalized partial identification

In the situation of I (i),

$A(s,\cdots)\sim A(u,\#)$ iff an argument of $A(s,\cdots)$ = an argument of $A(u,\#)$.

Obviously, thus defined ~ becomes non-transitive. (Compare with the above disjunctive inference.) Similarly, we can obtain the notion of the generalized partial pattern matching.

(iii) the association based inference

In our daily life, we use, so to speak, the association in many situations. For example,

summer ⇒ hot ⇒ mustard

is a typical case. Here, we usually can't directly associate "summer" with "mustard". Thus the relation ~ based on the association is not transitive. The combination of the association and an inference gives us an interesting way of reasoning like

$$\frac{\text{summer} \rightarrow \text{winter, hot}}{\text{cold}} \quad , \quad \frac{\text{hot} \rightarrow \text{cold , coffee}}{\text{ice cream}}$$

$$\frac{\text{winter} \rightarrow \text{ski , cold}}{\text{snow}} \qquad \text{etc.} \qquad \cdots (2)$$

Of course, the crucial point is how to embody this kind of association based inference. One important tactic would be to use the neural network technique at each association part. In this case, the result becomes the combination of the logic programming paradigm(inference part) and the neural network paradigm. However, any form of neural network is not good at treating drastic change of concepts like summer → hot, owing to the nature (, even if it can represent the notion of summer by somehow or other). Another is to employ symbolic methods. Any example belonging to this latter category is included in the following more abstract class.

(iv) distance and direction strategy

Suppose we can introduce the notion of the " distance " into the set of all atoms (or data or informations) by somehow or other,in accordance with the mathematical structure. For example, analytic or probabilistic measure, topological metric, algebraic norm, geometrical or graphic distance and statistical deviation etc play this role.

Now, in the form of (1) , suppose that A → B and a distance ρ over the initially intended domain (or space) S_A including A is given. Let d(x,y) be the distance function over S_A. Our aim (or the program) is to try to find A' in S_A such that $d(A,A') \leqq \rho$. (Or, given $A' \in S_A$ and ρ , try to find A → B such that $A \in S_A$ and $d(A,A') \leqq \rho$.) In this case, the candidate may not be unique. Suppose a candidate A' is found. Then, we define that A~A' holds. Next, depending on B, the new distance function $\mu(B)(d)$ and a distance $\mu(B)(d(A,A'))$ with respect to the domain S_B including B is obtained by using the "state-changing meta-function" μ . Using this latter distance, we (the program) try to find $B' \in S_B$ such that $\mu(B)(d)(B,B') \leqq \mu(B)(d(A,A'))$. If such a B' is found, then

we define that B ~ B' holds.

Remark: In the above, we have argued about the case of the forward reasoning in the form of (1). The similar argument can also be applied to the case of the backward reasoning.

Here, there might happen cases that not only the distance but also the notion of the "direction" had better be employed to search both A' and B'. In these cases, we use the "direction fan or (more generally) direction corn" condition (constraint) $\delta(\rho, A, y)$ over S_A, which the candidate $A' \in S_A$ should satisfy, in addition to the distance function d(A,y). (In some cases, d(A,y) might be absorbed into $\delta(\rho, A, y)$. Of course, in general, two conditions $\delta(\rho, A, A')$ and $\delta(\rho, A', A)$ are different.)
As the consequence, the state-changing meta-function μ ought to treat $\delta(\rho, A, y)$ to obtain the new condition

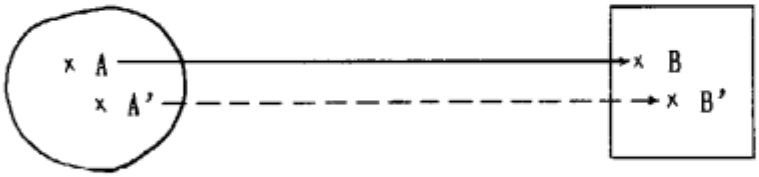
$$\mu(B)(\delta)(\mu(B)(\rho), B, z')$$

with respect to S_B.

Note the fact that not only the symbolic or numerical approach but a kind of analogue approach (like fuzzy inference) and even some neural network technique can be classified to this general category, by suitably defining d(x,y), μ and $\delta(\rho, x, y)$.

§2. The Non-transitive Inference

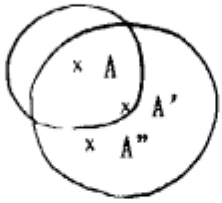
From now on, we study a few interesting and important issues concerning the approximate reasoning. We begin our argument by considering the case of an analogical reasoning as a typical example. (In order to embody this kind of reasoning, we can employ, say, the distance and the direction strategy in the above example.) Suppose a rule A → B is given, and A ~ A'. The first step is to obtain B' from A' based on A → B by somehow or other, under the name of an analogy.



… (3)

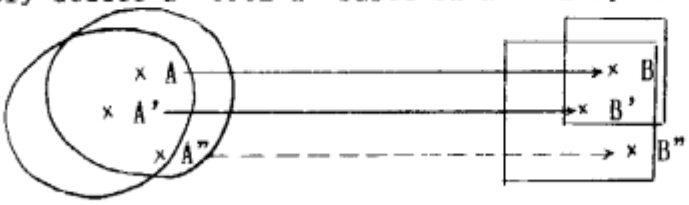
Now, suppose the desired A' → B' is admitted (by passing the check of the plausibility and the legitimacy, if the program equips the testing part). This means that the relation B ~ B' is recognized. Next, suppose also that A' ~ A" is given. Since we already have the plausible rule A' → B', the program may deduce a consequence B" from A" by somehow or

other. Here, the crucial point is that, from A ~ A' and A' ~ A", the program may not always deduce A ~ A". Theoretically speaking, this is so because the similarity relation ~ is not transitive in general.



… (4)

However,even if the relation A ~ A" holds, there often happens the case that we (our brain and AI program) can't, or more strongly, should not directly deduce B" from A" based on A → B by an analogy.



… (5)

That is, we can't always obtain B ~ B" from B ~ B' and B' ~ B". Here, note that the non-transitivity of ~ in the sense of the approximate reasoning is also applicable to some kinds of fuzzy inferences.

Since the above kind of limitations are often encountered in our daily common sense reasoning, it surely need to be represented as an AI inference. However,neither the conventional analogical reasoning in the field of machine learning nor fuzzy inference seems to be conscious of this non-transitivity.

§3. Two More Issues Concerning LP + AR

So far, we study the notion of the approximate reasoning (AR for short) alone. The similarity relation ~ among atoms (,or more generally, among formulas) can be defined as you like, without considering the logical structure of the logic programming (LP for short) part as a whole. However, any inference is a rule on which a logic program runs, and a LP + an AR determine the total program as a theory. In this sense, the AR part and the LP part work together cooperatively, and the former can't escape from the influence of the latter and vice versa. In this situation :

1. From (1), we obtain A' → B' as the plausible consequence. Since the above is only one step inference, the validity of the clause A'→ B' itself (not the simple B') might be well preserved to a certain extent,

because of the status of B'. (Remember A'∼A and B'∼B) However, after successive n-step inferences based on, say, $A_1 \rightarrow A_2$, $A_2' \rightarrow A_3$, ⋯, $A_n' \rightarrow A_{n+1}$ and $A_1 \sim A_1'$, $A_2 \sim A_2'$, ⋯, $A_{n+1} \sim A_{n+1}'$, can we admit $A_1' \rightarrow A_{n+1}'$ as a legitimate clause without hesitation ? If the relations $A_1 \sim A_1'$, $A_2 \sim A_2'$, ⋯, $A_{n+1} \sim A_{n+1}'$ are defined independently of the given clauses $A_1 \rightarrow A_2$, $A_2' \rightarrow A_3$, ⋯, $A_n' \rightarrow A_{n+1}$, there seems to be no assurance which claims the logical validity of $A_1' \rightarrow A_{n+1}'$.

Remark: Even if → does not represent the logical implication in its rigid sense, we still hope that the resulting $A_1' \rightarrow A_{n+1}'$ has some reasonable meaning by itself which we can interprete and understand (without asking for the formal relationship between the intermediate $A_1 \rightarrow A_2$, $A_2' \rightarrow A_3$, ⋯, $A_n' \rightarrow A_{n+1}$ and $A_1 \sim A_1'$, $A_2 \sim A_2'$, ⋯, $A_{n+1} \sim A_{n+1}'$). This expectation is nothing but the backbone idea of the possibility of machine learning.
Here, note the fact that the above vagueness has nothing to do with the condition of ∼ being transitive or not.

To tell the truth, this phenominon is the starting point where the scheme of LP + AR (which include the fuzzy logic programming in its real sense, not the mere logic programming with the calculation of certainty factors,)is perplexed. Nevertheless, owing to the nature, the similarity relation ∼ should be independent of the logical structure of LP part, because we ought to expect to utilize the same relation ∼ for different logic programs. How should we solve this dilemma ?

2. Moreover, especially in the case that ∼ is not transitive, the next problem arises, too. Even if we obtain the suitable procedural semantics of LP + AR, what amounts to the corresponding declarative semantics ? Obviously, the logical consequence in the classical sense is not worth being called the answer. Is there any good candidate for this concept ? Surprisingly, there seems to exist no conventional theory which covers and explains the topics of the model theoretic aspect of LP + AR. Since the matter is so, theoretically important notion of the "completeness" (in what sense ?) of LP + AR has never been discussed.

§4. Cooperatively Distributed Logic Programming: CDLP

In [14],[15], we already give the (possibly not unique, but highly reasonable and convincing) answer to the above three inevitable issues

concerning LP + AR, not case-by-casely but theoretically. Our solution is to use a map ⟦ ⟧ from the set of all ground atoms generated by the LP part P to a complete Boolean algebra 𝔹 which reflect the logical structure of P. The map ⟦ ⟧ bridge the side of LP and the side of AR to charge new constraints on both sides. As the consequence, we gain new and legitimate concepts of both the declarative and the procedural semantics and we can show that the crucial and interesting notion of the " relativized completeness" of LP + AR is preserved , though we totally omit the argument concerning this aspect of LP+AR here in this paper.

Solving the above stated problems by giving the universal and general answer and, at the same time, creating new and legitimate concepts for the solution is the merit of our investigating such an abstract entity like LP+AR theoretically. Our methodology is to fill up a gap between LP and AR by inserting new constraints, while preserving the autonomy of both LP part and AR part. Talking with one rank higher level perspective, we may obtain the following obsevation from a viewpoint of AI or cognitive science.

Through the methodology of studying the abstract,we can get an insight that the above kind of scheme might be qualified to be a member of the programming paradigm of the cooperatively distributed AI (or problem solving). That is, two different and distributed problem solving categories of LP part and AR part work cooperatively to become a compound programming technique. This viewpoint can be easily gained by means of the abstraction of each particular case. We believe that this idea of the separation and the combination becomes useful at the time of constructing a prototype program which is rather big and practical. For example, instead of a usual logic program, say,

$$\{\ A\leftarrow,\ Q\leftarrow,\ B\leftarrow A,\ P\leftarrow Q\ \},$$

we can have a LP + AR such that

$$\{\ Q\leftarrow,\ P\leftarrow Q\ \}\ +\ \{\ A\sim Q,\ B\sim P\ \},$$

if there are any similarity $\sim$ between A and Q, as well as B and P, and the similarity really happens among practical informations.

Furthermore, once we stand on this status of cooperatively distributed logic programming (CDLP for short), we ought to have an inclination to expect that other strategies might be able to be combined with LP + AR. For example,in a fuzzy inference, we calculate truth-values or certainty factors or priorities at the time of AR, and the category of the computation of truth-values (CT for short) is confusingly mixed with that of

AR in many cases. (This confusion is the one reason why the resulting value becomes inappropriate to the supposedly expected semantics. Many fuzzy theorists seem to forget about the fact that CT, in addition to AR, should reflect the logical structure of LP.) However, there are a lot of ways to compute a value consistently for each AR step, which are different from the analogue technique employed by the fuzzy inference. Moreover, there is no necessity for us to restrict the truth-value to a number. On the ground of this observation, we notice that the category of CT is distinguishable from that of AR.Once we stand on this viewpoint, there is no wondering that we try to connect an example of CT with other AR examples like an equational theory or even an analogical reasoning or a hypothetical reasoning. Thus, we can obtain the resulting scheme LP + AR + CT as a member of CDLP. In addition, if we hope, we might further introduce the strategy of constraint solving to obtain CLP + AR + CT. Yet other combinations are, of course, possible. In [14] and [15], we already offer examples of LP + AR + CT + α, where α is closely related to the notion of the "situation" and that of the "hypothesis or assumption".

§5. Conclusion

In this paper, we abstractly discuss the crucial property of the logic programming (LP) based on the approximate reasoning (AR) from a viewpoint of AI. The key points are;

1. The similarity relation $\sim$ used at AR need not be transitive.
2. Several new examples of AR are proposed.
3. Three issues concerning AR are argued.
4. By separating the AR part from LP part, the total program can be seen to be a member of the cooperatively distributed AI.

Moreover,we suggest a promising and systematic solution based on our proposing Boolean-valued methodology, which answers the above 3, though we omit the precise investigation of our methodology here in this paper. (See [14],[15].) The important point is that our Boolean-valued technique is strong and general enough to not only be available for the solution of the three issues but be helpful for other topics in AI like hypothetical reasoning (including ATMS), situation theory,semantic negation, hierarchical inheritance, machine learning etc. (See [11],[17].) Further arguments concerning these aspects of our Boolean-valued methodology will appear in future.

References

[1] Arikawa,S., Program Synthesis by Inductive Inference and Analogical Reasoning, *J. of JSAI.* 2: 299-306 (1987).

[2] Burstein,M.H., Concept Formation by Incremental Analogical Reasoning and Debugging, in:R.S.Michalski, J.G.Carbonell and T.M.Mitchell (eds.) *Machine Learning: An Artificial Intelligence Approach Vol. II*, Morgan Kaufmann, 1986.

[3] Carbonell,J.G., Analogy in Problem Solving, in:R.S.Michalski, J.G. Carbonell and T.M.Mitchell (eds.) *Machine Learning: An Artificial Intelligence Approach Vol. II*, Morgan Kaufmann, 1986.

[4] van Emden,M.H. and Yukawa,K., Logic Programming with Equations, *J. Logic Programming.* 4: 265-288 (1987).

[5] Haraguchi,M., Computational Analogy and Inference of Incomplete Information, *JSAI SIG-FAI-8804:* 31-39 (1989).

[6] Jaffar,J., Lassez,J.L. and Maher,M.J., A Theory of Complete Logic Programs with Equality, *Proc. Int. Conf. of FGCS:* 175-184 (1984).

[7] Leung,K.S. and Lam,W., A Fuzzy Expert System Shell Using Both Exact and Inexact Reasoning, *J. of Automated Reasoning* 5: 207-233 (1989).

[8] Li,D. and Liu,D., *A Fuzzy PROLOG Database System*, Research Studies Press, 1990.

[9] Yamaguchi,J., Boolean-valued Logic Programming Language Scheme: LIFE-Ⅲ —A Theoretical Background —, to appear.

[10] Yamaguchi,J., Universal Unification from a Viewpoint of LIFE-Ⅱ,NEC LR-5347,1987, revised version to appear.

[11] Yamaguchi,J.,Boolean-valued Logic Programming Language Scheme:LIFE-Ⅲ [1]Inferentially Transforming Logic Programs,(in Japanese) *5th Conference Proceedings Japan Soc.Software Sci.&Tech.*,237-240 (1988), revised version to appear.

[12] Yamaguchi,J.,Boolean-valued Logic Programming Language Scheme:LIFE-Ⅲ [2]A Technique to Execute a Logic Program Efficiently,(in Japanese) *5th Conference Proceedings Japan Soc.Software Sci.&Tech.*, 241-244 (1988), revised version to appear.

[13] Yamaguchi,J., Classification and Boundary Problem, (in Japanese) *Proceedings of LPC'89*,49-58, English version to appear in Lecture Notes in AI series, Springer.

[14] Yamaguchi,J., LIFE-Ⅲ [3]Relativized Completeness, (in Japanese) *Proc. of the 3rd Annual Conference of JSAI*,33-36 (1989), revised version to appear.

[15] Yamaguchi,J., LIFE-Ⅲ [4]A Fuzzy Inferential System,(in Japanese) *6th Conference Proceedings Japan Soc.Software Sci.&Tech.*, 121-124 (1989), revised version to appear.

[16] Yamaguchi,J., LIFE-Ⅲ[5]From a Viewpoint of Situation Theory (Ⅰ) (in Japanese) *Proc. of the 4th Annual Conference of JSAI*,93-96(1990), revised version to appear.

[17] Yamaguchi,J., LIFE-Ⅲ[6]From a Viewpoint of Situation Theory (Ⅱ) (in Japanese) *7th Conference Proceedings Japan Soc.Software Sci.& Tech.*,369-372 (1990), revised version to appear.

[18] Yamaguchi,J., Phases,Informations and the Permitting Relation— a Boolean-valued Representation of Knowledge in AI—, accepted in *9th International Congress of LMPS*, 1991.

[19] Yamaguchi,J., Boolean-valued Logic Programming Language Scheme LIFE -Ⅳ—The Beginning of a New Era of the Logic Programming—, to appear.

[20] Zadeh,L.A., Fuzzy Logic and Approximate Reasoning, *Synthese* 30: 407-428 (1975).