ALPS: A Logic for Program Synthesis (Motivated by Fuzzy Logic)

Vladik Kreinovich, Daniel E. Cooke

Abstract— **One of the typical problems in engineering and scientific applications is as follows: we know the values $x_1, \ldots, x_n$ of some quantities, we are interested in the values of some other quantities $y_1, \ldots, y_m$, and we know the relationships between x_i, y_j, and, maybe, some auxiliary physical quantities $z_1, \ldots, z_k$. For example, we may know an algorithm to compute y_2 from x_1, x_3 and y_1; we may also know an equation $F(x_1, x_2, y_1) = 0$ that relates these values, etc. The question is: can we compute the values of y_j, and, if we can, how to do it?**

At first glance, this is a problem of numerical mathematics, and, indeed, there exist numerical methods to solve it. These methods use the equations from the very beginning; these equations can be very complicated and therefore, processing them can be very time-consuming. E. Tyugu proposed a two-stage ("lazy computations") approach to solving these problems, and implemented it in a system PRIZ. First, we consider only which quantities are computable from which. Based on this information only (and not using the specific algorithms or relationships of the type $F(x_1, x_2, y_1) = 0$) we find out, whether it is possible to compute y_j, and , if it is possible, what steps should we follow. For example, in the above example we first compute y_1 from x_1 and x_2 by solving the equation $F(x_1, x_2, y_1) = 0$, and then use the resulting value y_1 and the known value x_3 to compute y_2. On the second step we actually perform these computations.

G. Mints showed that the problems of PRIZ's first step can be reformulated in logical terms. He also showed that using the related logical algorithms instead of the original PRIZ's ones can improve the system. However, there are cases when this approach does not work: for example, when we have two equations with two unknowns. No known logic is applicable to this case.

We propose a new logic (motivated by fuzzy logic) that is specially tailored for these problems, show its decidability, analyze its properties, and propose a special resolution rule for it. This new logic allows us to use a two-stage ("lazy computations") approach in the most general case.

The usefulness of a non-classical (fuzzy-type) logic in program design shows that its non-classical features are extremely important not only in the ***soft*** **area of representing human reasoning, but also in the** ***hard*** **parts of computer science.**

Keywords— **program synthesis, fuzzy logic, numerical mathematics, radioastronomy**

V. Kreinovich and D. Cooke are with the Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968, USA. E-mails: {vladik,dcooke}@cs.utep.edu.

This work was supported in part by NASA under cooperative agreement NCCW-0089, by NSF under grants No. DUE-9750858 and EEC-9322370, and by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518.

I. Introduction

A. General Problem

The engineering problem that we are going to solve. One of the typical problems in engineering and scientific applications is as follows:

- we *know the values* $x_1, \ldots, x_n$ of some quantities;
- we *are interested* in the values of some other quantities $y_1, \ldots, y_m$, and
- we *know the relationships* between x_i, y_j, and, maybe, some auxiliary physical quantities $z_1, \ldots, z_k$.

There can be two types of relationships:

- We may know an *algorithm* that allows us to compute some of the unknown values y_j or z_k from x_i and, maybe some other y_l and/or z_m
  For example, we may know how to compute y_2 from x_1, x_3 and y_1.
- We may also know an *equation* that relates some of the values y_j, z_k, and x_i.
  For example, we may know an equation $F(x_1, x_2, y_1) = 0$, where F is some known expression.

The question is: given this knowledge, can we compute the values of y_j, and, if we can, how to do it?

This is a *general* problem of data processing:

- we measure something,
- we know something, and
- we are interested in whether we can extract, from what we know and from what we have measured, the values of some quantities that we could not measure directly.

Let us briefly describe two pedagogical examples from [15] and a real-life example.

First pedagogical example: Kirkhoff laws. In this example, we will consider electric networks. Let us first consider the case of a direct current.

- We *know* the voltages V_i of all the sources and the values of all the resistances R_i. These are our known values (which in the general formulation we denoted by x_i).
- We *are interested* in the values of current I_i. So, in this case, $y_1, \ldots, y_m$ are the values of current.

The relationship between x_i and y_j is described by the so-called *Kirkhoff laws*:

- in every node, the algebraic sum of the currents $\sum I_i$ is equal to 0, and
- for every loop, the sum of the terms $I_i \cdot R_i$ equals to the total voltage of all the sources in this loop.

This is a pedagogical example, because in this case, it is known that we can reconstruct y_i, and there exist methods that allow to do that: Kirkhoff equations are linear, so we must solve a system of linear equations.

The same Kirkhoff laws are, however, applicable to more complicated cases, when we are do not know some of the resistances. In this case, we measure *some* of the currents, and we would like to know the unknown resistances. So, the list $x_1, \ldots, x_n$ now includes the sources voltages, known resistances, and the measured values of current; $y_1, \ldots, y_m$ are unknown resistances. We cannot formulate relationship between x_i and y_j without using all the currents, so in this case, the un-measured values of the current are $z_1, \ldots z_k$. In this case, it is not automatically true that we can reconstruct the desired values from the measured ones, and not automatically known how to do it.

The situation of *alternating* current is similar, with the only difference that instead of resistances we have complex *impedances* Z_i, and the current values of the current can also be complex.

Second pedagogical example: triangle. Another pedagogical example is a *triangle.* A triangle is described by its angles A, B, C and side lengths a, b, c, and we know the following relations between them: $A+B+C=\pi$ (the sum of the angles is 180^o, or π radians), $a^2+b^2-2ab\cos C=c^2$ and similar expressions for a and b (cosine theorem), and $a/\sin A = b/\sin B = c/\sin C$ (sine theorem). Now we can ask all kinds of questions: If we know a, b and c, can we determine A? If we know a, b and A, how to compute b? etc.

For example, if we know a, b and c, and we want to determine A, then:

- $x_1 = a, x_2 = b, x_3 = c$;
- $y_1 = A$, and
- z_k's are B and C (namely, $z_1 = B$ and $z_2 = C$) because these quantities are neither known, nor desired, but they are part of the relations that connect x_i and y_j.

It is also a pedagogical example, because there are only finitely many possible problems, and all of them have been solved in elementary geometry.

Real-life example: arc method in radioastronomy. A reader may get a wrong impression that this sort of problems are very simple and are mainly already solved. So we will just mention a real-life problem in solving which one of the authors (V.K.) participated [4], [5], [6]. In Very Long Baseline Interferometry:

- We *measure* the phase shift between the radiosignals that are received by two distant antennas, so x_i are shifts.
- We *are interested* in the coordinates y_j of the radiosources.

The formulas that relate x_i and y_j includes also such unknown variables as the initial clock instability, distance between antennas, atmospheric shifts, etc. (that play the role of z_k).

Initially, it was *believed* that there is *no way* to reconstruct y_j from x_i and these relations. Therefore, the values of z_k were crudely estimated, and the errors of these estimates led to crude estimates for y_j. A mathematical analysis of these relations revealed that we *can* reconstruct the values of y_j from x_i, and, therefore, we became able to reconstruct the coordinates with a hundred times better precision.

B. Methods of Numerical Mathematics

How these problems are solved in numerical mathematics. At first glance, the problems that we describe are the problems of numerical mathematics. And, indeed, there exist numerical methods to solve them. These methods are based on the well-known least squares techniques.

The idea of the least squares method is as follows:

- First, we represent all our knowledge in terms of equations.

  If we have an algorithm that computes a from b and c, then we represent it as an equation $a - f(b,c) = 0$, where by $f(b,c)$ we denote a function that is the result of those computations.

  After this representation, we have p equations $f_e(\vec{a}) \approx 0$, $1 \le e \le p$, to determine the unknown parameters $\vec{a} = (a_1, \ldots, a_n)$.
- Then, we formalize this problem as a mathematical optimization problem $\sum_e f_e^2(\vec{a}) \to \min$.

In case the statistical error is distributed according to the Gaussian law, this expression can be statistically justified.

There exists software packages that use this method.

In our case, we can use a similar method: namely, if we have p equations $F_e(x_i, y_j, \ldots, z_k) = 0$ that relate x_i, y_j, and z_k, then we determine the values y_j from the condition that $E \to \min$, where by E we denoted the sum

$$E = \sum_{e=1}^{p} F_e^2(x_i, y_j, \ldots, z_k).$$

So the method is as follows:

- form a function E, and apply some numerical optimization techniques to find the values y_j, for which this function E attains its minimum;
- if the minimal value $E_{\min}$ is positive, this means that the conditions are inconsistent;
- if $E_{\min} = 0$, and minimum is attained for several different values of y_j, this means that y_j cannot be uniquely determined from x_i and the known relations;
- if $E_{\min} = 0$, and minimum is attained for only one value of $\vec{y} = (y_1, \ldots, y_m)$, then this value y_j is the one that is uniquely determined from x_i and a given knowledge.

These conclusions can be easily justified: indeed, the condition that $F_e = 0$ for all e is equivalent to the condition that $\sum_e F_e^2 = 0$. Therefore, if $E = 0$, this means that all the equations are satisfied. If $E_{\min} > 0$, this means that the equations cannot be satisfied, so our knowledge is inconsistent with the measurement results x_i.

This numeric method is rather successful. This method was actually (and rather successfully) implemented in a system MARS (see, e.g., [8], [9]).

This implementation uses a library of powerful optimization techniques, and therefore, it works reasonably fast even when we have dozens of different variables and dozens of relations.

The main drawback of the numerical method. The main drawback is that it is a *brute-force* method, aimed at most complicated problems, and it is not flexible. In many cases we humans know that we do not need to use all the equations, and thus we can essentially simplify the problem.

For example, if we apply this method to a triangle problem, we end up with a non-linear functional E that is equal to the sum of squares of all the equations that represent cosine law, sine law, etc. To minimize this function of six variables, we need a lot of computation time. But in high school geometrical problems we never do that: if we know A and B and we want to know C, then we immediately see that one equation $A+B+C=\pi$ will be sufficient, and determine C as $\pi-A-B$. If we know a, b, and A, and we want to determine C, then we determine B from the sine theorem, and then compute C from A and B.

C. PRIZ: A Case When Logic Helps

Tyugu's approach: general idea. E. Tyugu proposed a two-stage ("lazy computations") approach of solving these problems, and implemented it in a system PRIZ [7], [10], [11], [12], [13], [15], [16]:

- On the first stage, we analyze which quantities are computable from which:
  - If we are given an algorithm that computes some quantity C from A and B (where $A, B, \ldots$ can mean any of the values x_i, y_j, z_k), then we write it down as $A, B \rightarrow C$. This arrow means "implies", and should be read as "if we already know A and B, then we are able to compute C".
  - Suppose now that we have a relation $F(A,B,C)=0$. If we know all of these values but one (for example, A and B), then we have an equation with one unknown, from which in general we can compute C. Similarly, if we know A and C, then we can compute B, and from B and C we can compute A. So each equation leads to as many computability relations as there are unknowns in it. In our case we get three computability relations: $A, B \rightarrow C$; $A, C \rightarrow B$; and $B, C \rightarrow A$.
- Based on this information only (and not using the specific form of the algorithms or relationships of the type $F(A,B,C)=0$) we find out, whether it is possible to compute y_j, and, if it is possible, what steps should we follow.
- Finally, we follow the steps and compute y_j.

Tyugu's approach: example. In the triangle case, the relations turn into the following formulas: $A, B \rightarrow C$; $B, C \rightarrow A$; $A, C \rightarrow B$; (these three stem from the equation $A+B+C=\pi$) $A, a, b \rightarrow B$; $A, a, B \rightarrow B$; ...(from sine theorem), and $a, b, C \rightarrow c$; $a, b, c \rightarrow C$; $a, c, C \rightarrow b$; $b, c, C \rightarrow a$; ...(from cosine theorem).

Tyugu's approach: an algorithm. There exists a natural algorithm to decide whether y_j is computable: a so-called *wave algorithm*. According to the wave algorithm, we first mark the variables that we know; then we look at all the rules, find those, for which all the conditions are marked and the conclusion is not, and mark the conclusion. Then we repeat the same procedure.

After each iteration:

- either did not add anything, which means that we are done (nothing else can be computed),
- or we add at least one marked variable.

Since there are finitely many variables, this process will eventually stop. If the desired y_j are marked, then we *can* compute them, else we *cannot*.

For example, suppose that in the triangle example, we know A and B, and we want to compute C and a. Then, according to the algorithm, we first mark A and B. There is only one rule whose conditions are marked: the rule $A, B \rightarrow C$. So, we mark C. On the second iteration, we find three rules whose conditions are marked: $A, B \rightarrow C$; $B, C \rightarrow A$; and $B, C \rightarrow A$, but their conclusion have already been marked. So, we stop.

As a result, C is marked, which means that we can compute C. Moreover, we know how to compute C: C was obtained from a rule $A, B \rightarrow C$ that stems from $A+B+C=\pi$, so we must solve an equation $A+B+C=\pi$, in which A and B are known, and C is the only unknown. The PRIZ system includes an embedded equation solver (based on a version of Newton's method) that solves equations with one unknown.

As for a, it is not marked, and therefore, cannot be computed.

Actually, the wave algorithm is the simplest algorithm, and the PRIZ system implements a more complicated but faster method (for the fastest possible methods, see [2]).

This approach can be reformulated in logical terms. G. Mints showed (see, e.g., [12], [13]), that the first step of PRIZ can be reformulated in *logical* terms. Namely, we can interpret the rules $A, B \rightarrow C$ that stem from the relations as the propositional formulas $A \& B \rightarrow C$ with variables $A, B, \ldots$ that can take the values "true" or "false": "true" means that we can compute the corresponding variable, and "false" means that we cannot. So our knowledge can be represented as a set of propositional formulas that include all the rules and all the atoms A that represent the known variables x_i.

We want to know whether the values y_j are computable, or, in the propositional terms, whether the variables that correspond to y_j are true or not. So, in logical terms, we want to know whether these variables are deducible from the knowledge base.

In the triangle case, we have a knowledge base $A \& B \rightarrow C$; $B \& A \rightarrow C$; ...; A; B, and we want to know whether C and a follow from these formulas.

In this example, the application of logic is (somewhat) trivial, but in many complicated cases it really helps.

In many cases, but not always: there exist case when this logical approach does not work.

D. Cases When Traditional Logic Does not Help

Alas, Tyugu's ideas are not always applicable. Let us consider the case when we want to know the values of two unknowns y_1 and y_2, and we know the two relations between them: $y_1+y_2-1=0$ and $y_1-y_2-2=0$. Evidently, in this case we can determine both y_1 and y_2, because we have a system of two linear equations with two unknowns. However, Tyugu's approach will not work:

Indeed, the first equation will translate into two rules $Y_1 \to Y_2$, $Y_2 \to Y_1$, where propositional variables Y_i correspond to y_i. The second equation will lead to these same rules. From these two formulas we cannot logically conclude that Y_1 is true (because if Y_i are both false, still both rules are true), and therefore, we cannot conclude that y_i are computable.

This is not a specific feature of this weird example: the same situation occurred in the above-described radioastronomical example.

PRIZ has some means of handling such cases. In PRIZ, there are some means of handling these situations, but they are rather ad hoc: they are based on trying to determine whether there is a system of two equations with two unknowns, or a system of three equations with three unknowns, etc. These heuristic are often helpful, but they do *not* give a *general* solution. (And we do not want to use any general-case monster system inspired by numerical mathematics, if we can avoid it.)

Alternative approaches. There exist several other approaches that attempt to incorporate equations into the rule-based knowledge (see, e.g., [1], [3], [17]), but none of them gives a general solution to our problem.

Main problem and what we are going to do. We would like to have a sort of logical approach that would be applicable also to the case when we have several equations with the same unknowns. Since traditional propositional logic does not help, *we need a new logic.*

II. Informal Discussion of the New Logic

Simplest case: getting ready for resolutions techniques. Let us start with the simplest equation $F(A,B)=0$. As we have already argued, this equation means that if we know A, then we can compute B, and vice versa. So this equation will give way to two rules: $A \to B$ and $B \to A$. The most wide-spread deduction techniques for propositional formulas is the *resolution method* (and it is also one of the basic techniques of PRIZ). In order to apply it, we need to reformulate the propositional formulas in terms of disjunctions, i.e., rewrite $A \to B$ as $\neg A \vee B$, and rewrite $B \to A$ as $A \vee \neg B$.

In the desired logic of rules, $A \to B$ implies $B \to A$. Suppose that we have a rule $A \to B$. This means that we are able to compute B from A. Let us denote by $f(A)$ the result of applying these computations. Then we have a relation between A and B: $B=f(A)$, or, in terms that we got used, $B-f(A)=0$. But this means that, in general, we can reconstruct A from B as well, i.e., that we have a rule $B \to A$.

From the viewpoint of classical logic, this conclusion may sound weird, but from the physical viewpoint, it is quite natural. Indeed, if we know that a variable B is uniquely determined by the value of the variable A, then it is natural to expect that we can invert this relation and use B to determine A. For example, if the temperature T determines a density ρ of a substance, and we know the dependency, then from this dependency we can reconstruct a temperature if know ρ.

It looks like we do not need negation. In terms of disjunctions, our conclusion is that if $\neg A \vee B$, then $A \vee \neg B$. Similarly, if we consider a relation with three unknowns, we come to a conclusion that if $\neg A \vee \neg B \vee C$ is true, then both $A \vee \neg B \vee \neg C$ and $\neg A \vee B \vee \neg C$ are true. It looks like the truth of the disjunction does not depend on which variables we negate. In other words, it looks like the negation symbol $\neg$ does not influence on the truth of the formula, and can therefore be omitted.

Omitting negations will simplify computations. Indeed, if we have a relation $F(A,B,C)=0$, then with negation we will have *three* rules $A,B \to C$; $B,C \to A$; and $A,C \to B$, that in disjunctive form are $\neg A \vee \neg B \vee C$, $A \vee \neg B \vee \neg C$, and $\neg A \vee B \vee \neg C$. If we delete negations, then all three disjunctions will turn into *one* and the same rule: $A \vee B \vee C$. Similarly, any relation leads to only one rule.

So, we decrease the total number of rules, and, therefore, the amount of computations.

Within classical logic, the above idea is impossible, but fuzzy logic makes this idea implementable. What we really want is to be able to use a logic in which some statements A are equivalent to their negations $\neg A$. In classical (two-valued) logic, this is clearly impossible. But luckily, there is another logic: a fuzzy logic, in which the equivalence between A and $\neg A$ is quite possible. This made us think that logic *can* be useful in non-traditional program synthesis situations.

In principle, we could have probably used the general fuzzy logic, but since we only needed one feature of it (and enlarging logic would make computations more complicated), we decided to restrict ourselves to a specially tailored logic, which can be thus viewed as a *intermediate* logic between classical and fuzzy, a logic that incorporates only *some* features of fuzzy logic in its definitions.

III. The New logic: Definitions and Properties

A. Description of the New Logic

Formulas. In accordance with the above informal description, in this logic, we start with the list of *variables* $A_1, \ldots, A_n$. These variables can be combined into *disjunctions*, i.e., into formulas D of the type $A \vee B$, $A \vee B \vee C$, etc.

Deduction. A typical problem in this logic is as follows: we know that several disjunctions $D_1, \ldots, D_k$ are true, and we must check whether some other disjunction D follows from these ones. This *implication* will be described as $D_1 \& \ldots \& D_k \to D$ or as a *deduction*

$$\frac{D_1,\ \ldots,\ D_k}{D}.$$

To complete the description of the logic, we must specify when a deduction is true and when it is not true.

Informal description of what deduction means. We interpret each variable as a physical quantity, and each disjunction as a relationship between physical quantities. For example, a disjunction $A \vee B$ means that there is a relationship $F(A, B) = 0$ between the values of the variables A and B.

In general, this relationship can be *non-linear*. However, we usually know the *approximate* values $\tilde{A}$ and $\tilde{B}$ of the measured quantities, i.e., we know that A belongs to a small neighborhood of $\tilde{A}$, and that B belongs to a small neighborhood of $\tilde{B}$. In these small neighborhoods, the function $F(A, B)$ can be, within a reasonable accuracy, replaced by the first order terms of its Taylor expansion, i.e., by a *linear* relation of the type $c_1 A + c_2 B = c_3$. Therefore, in the following text, we will interpret each formula as the existence of such a linear relation.

At first glance, it looks like we are ready for a definition: we may proclaim the deduction $D_1 \& \ldots \& D_k \to D$ as true if for all possible values of the coefficients describing relations D_i, there exists some non-trivial relation corresponding to D. However, this is not exactly what we want. Let us give a simple example why.

If we have *two* identical disjunctions $A \vee B$ and $A \vee B$, this means that we have *two* relationships between the same variables. Of course, if the corresponding two linear equations simply coincide, then we cannot find the value of A from these two equations. However, from a physical viewpoint, it is highly unprobable that two different relations would lead to exactly the same equations, and if these two equations are different, we can indeed get A.

So, it is reasonable to interpret deduction as meaning not "for all values of the coefficients", but "for almost all values of the coefficients", where "almost all" is understood in the standard mathematical sense (everywhere except for a set of measure 0).

Thus, we arrive at the following formal definition:

Formal definition of deduction. Let $D_1, \ldots, D_k, D$ be disjunctions. To check whether the deduction $D_1 \& \ldots \& D_k \to D$ is true, we do the following:

- We represent the first disjunction $D_1 = A \vee \ldots \vee B$ as a linear equation $c_1 \cdot A + \ldots + c_k \cdot B = c_{k+1}$, the second disjunction $D_2 = C \vee \ldots \vee D$ as an equation $c_{k+2} \cdot C + \ldots + c_l \cdot D = c_{l+1}$, etc., until we represent the last disjunction D_k by an equation $\ldots = c_N$.
- Then, we say that a disjunction is *true* if for almost all values of the coefficient vector $\vec{c} = (c_1, \ldots, c_N)$, from the equations that represent D_i, we can conclude that there is a non-trivial linear relation between the variables that represent D.

We will also consider deduction of the type $D_1 \& \ldots \& D_k \to \perp$, where $\perp$ stands for "false"; such a deduction would mean that for almost all values of the coefficient vector $\vec{c}$, the corresponding linear equations are inconsistent.

B. Examples

To illustrate this definition, let us give examples of formulas that are true according to this definition:

$$\frac{A,\ A}{\perp}.$$

Indeed, if we have *two* different equations that describe the same value, then in almost all cases, these two equations are inconsistent.

$$\frac{A \vee B,\ A \vee B}{A}.$$

If we have two equations with two unknowns, then, in general, we can reconstruct A (and, similarly, B). This example can be generalized to n equations with n unknowns:

$$\frac{A_1 \vee \ldots \vee A_p,\ \ldots,\ A_1 \vee \ldots \vee A_p\ (p \text{ times})}{A_i}.$$

C. Relation to Resolution Method: Similarity

These examples illustrate a natural derivation idea: If we have a relationship that relates $A, \ldots, B$, and some variable C, and some other relationship that relates C with other variables $D, \ldots, E$, then, we can use the first equation to express C in terms of $A, \ldots, B$, and substitute the resulting expression into the second equation. As a result, we get a new equation that contains $A, \ldots, B, D, \ldots, E$, and does not contain C any more. So, we have the following derivation rule:

$$\frac{A \vee \ldots \vee B \vee C,\ C \vee D \vee \ldots \vee E}{A \vee \ldots \vee B \vee D \vee \ldots \vee E}.$$

This rule is very similar to the above-mentioned *resolution method*, one of the main methods of automated reasoning.

Thus, hopefully, we can still use modern automated reasoning techniques to check implication in the new logic, and thus, to solve our program synthesis problems.

D. Relation to Resolution Method: Differences

There are differences. The above useful analogy does not mean that we can immediately apply the resolution techniques from classical logic; these techniques must be changed.

First difference: absence of negation. In the traditional resolution method, we have a slightly different resolution rule: it is indeed similar to the above one, but with C in one of the disjunctions, and its negation $\neg C$ in another disjunction. Since we are identifying each variable with its negation, we get this rule in its above form.

Second difference: problems with chain reasoning. In the traditional resolution rule, we use *multi-step* (chain) reasoning, and for that, we need deductions in which the conlusion is not only a *single* disjunction, but *several* of them. In classical logic, we simply say that a formula $\mathcal{D} = D_1\&\ldots\&D_k$ implies a formula $\mathcal{D}' = D'_1\&\ldots\&D'_l$ if the first formula implies *all* the disjunctions D'_j from $\mathcal{D}'$.

If we simply repeat a similar definition for our new logic, then, for thus defined implication $\to$, we lose the ability to perform chain reasoning, i.e., to conclude, from $\mathcal{D} \to \mathcal{D}'$ and $\mathcal{D}' \to \mathcal{D}''$, that $\mathcal{D} \to \mathcal{D}''$. Indeed, from $\mathcal{D} = A$, we can deduce each of the disjunctions of $\mathcal{D}' = A\&A$, but from $\mathcal{D}'$, we can deduce the contradiction $\mathcal{D}'' = \perp$, while from the original formula $\mathcal{D}$, we cannot deduce the contradiction.

How to perform chain reasoning? Thus, if we want to be able to make meaningful chain deductions in the new logic, we cannot use $\to$, we must use a more complicated implication operation $\mathcal{D} \Rightarrow \mathcal{D}'$ meaning that for every other formula $\mathcal{D}''$, if $\mathcal{D}' \to \mathcal{D}''$, then $\mathcal{D} \to \mathcal{D}''$.

Examples: $A\&A \Rightarrow \perp$; $(A \vee B)\&(A \vee B) \Rightarrow A$; $(A\vee B)\&(A\vee B) \Leftrightarrow A\&B$ (meaning that $(A\vee B)\&(A\vee B) \Rightarrow A\&B$ and $A\&B \Rightarrow (A \vee B)\&(A \vee B)$).

The new implication *implies* the old one, but not the other way around: e.g., $A \to A\&A$, but $A \not\Rightarrow A\&A$.

One can easily check that thus defined new implication is already *transitive*: if $\mathcal{A} \Rightarrow \mathcal{B}$ and $\mathcal{B} \Rightarrow \mathcal{C}$, then $\mathcal{A} \Rightarrow \mathcal{C}$.

This new modification is also in line with fuzzy logic. The fact that A is not equivalent to $A\&A$ also reminds of fuzzy logic, in which, unless we use min as a t-norm, $A\&A$ is not equal to A.

E. How Can We Actually Check Deducibility in the New Logic?

The existing algorithm. One possibility is to use a *Monte-Carlo* method that is based on the following idea: When the values of the coefficients c_i are fixed, we can use linear algebra packages to check whether the variables from D are really linearly related. So, we can:

- use random number generators to generate random values c_i and
- check whether for these values, we get the desired conclusion.

If the desired conclusion is true for almost all $\vec{c}$, then it should be true for random coefficients with probability 1, i.e., practically always.

On the other hand, if this conclusion is not true with probability 1, then, as one can see, it is true with probability 0, i.e., practically never.

The ideal algorithm. Ideally, however, we would like to have a purely *logical* algorithm.

Acknowledgments

The authors are greatly thankful to A. Gates and M. Gelfond (El Paso, Texas), A. Dikovskii and M. Kanovich (Moscow), G. Mints (Stanford), G. Pospelov (Moscow), and E. Tyugu (Tallinn) for valuable discussions.

References

[1] W. F. Clocksin, "A technique for translating clausal specifications of numerical methods into efficient programs", *The Journal of Logic Programming,* 1987, Vol. 4, pp. 231–242.

[2] A. Dikovskii and M. Kanovich, "Computational models with separable problems", *Cybernetics (Technicheskaya Kibernetika),* 1985, No. 5, pp. 36–59 (in Russian).

[3] M. Dincbas and P. van Henenryck, "Extended unification algorithms for the integration of functional programming into logic programming", *The Journal of Logic Programming,* 1987, Vol. 4, pp. 199–227.

[4] A. Dravskikh, A. M. Finkelstein, and V. Kreinovich, "Astrometric and geodetic applications of VLBI 'arc method' ". *Modern Astrometry, Proceedings of the IAU Colloquium No. 48, Vienna, 1978*, pp. 143–153.

[5] A. F. Dravskikh, O. M. Kosheleva, V. Kreinovich, and A. M. Finkelstein, "Optimization of the procedure for measuring arcs by radiointerferometry", *Pisma v Astronomichesky Zhurnal*, 1979, Vol. 5, No. 8, pp. 422–425 (in Russian); English translation: *Soviet Astronomy Letters*, 1979, Vol. 5, No. 4, pp. 227–228.

[6] A. F. Dravskikh, O. M. Kosheleva, V. Kreinovich, and A. M. Finkelstein, "The method of arcs and differential astrometry", *Pisma v Astronomichesky Zhurnal*, 1979, Vol. 5, No. 6, pp. 300–303 (in Russian); English translation: *Soviet Astronomy Letters*, 1979, Vol. 5, No. 3, pp. 160–162.

[7] M. Kahro, A. Kalja, and E. Tyugu, *Instrumental programming system ES EVM (PRIZ).* Finansy i Statistika, Moscow, 1981 (in Russian).

[8] V. Kotov, "Concurrency + Modularity + Programmability = MARS", *Communications of the ACM*, 1991, Vol. 34, No. 6, pp. 32–45.

[9] G. I. Marchuk and V. E. Kotov,*Modular Asynchronous Reconfigurable System,* Technical Reports No. 86, 87. Academy of Sciences, Novosibirsk, Computing Center, 1978 (in Russian).

[10] M. Meriste and J. Penjam, "Toward knowledge-bases specifications of languages". In: J. Barzdins and D. Bjoerner (editors), *Baltic Computer Science*, Springer Lecture Notes in Computer Science, Vol. 502, Springer–Verlag, Berlin, Heidelberg, 1991, pp. 65–76.

[11] G. Mints, J. M. Smith, and E. Tyugu, "Type-theoretical semantics of some declarative languages", In: J. Barzdins and D. Bjoerner (editors), *Baltic Computer Science*, Springer Lecture Notes in Computer Science, Vol. 502, Springer–Verlag, Berlin, Heidelberg, 1991, pp. 18–32.

[12] G. Mints and E. Tyugu, "The programming system PRIZ", *Journal of Symbolic Computations,* 1988, Vol. 5, pp. 359–375; reprinted in: J. Barzdins and D. Bjoerner (editors), *Baltic Computer Science*, Springer Lecture Notes in Computer Science, Vol. 502, Springer–Verlag, Berlin, Heidelberg, 1991, pp. 1–17.

[13] G. E. Mints and E. H. Tyugu, "Propositional logic programming and the PRIZ system", *Journal of Logic Programming*, 1990, Vol. 9, pp. 179–193.

[14] A. Togashi and S. Nogushi, "A program transformation from equational programs into logic programs", *The Journal of Logic Programming,* 1987, Vol. 4, pp. 85–103.

[15] E. Tyugu, *Knowledge-based programming,* Addison-Wesley, Wokingham, England, 1988.

[16] E. Tyugu, "Three new-generation software environments", *Communications of the ACM*, 1991, Vol. 34, No. 6, pp. 46–59.

[17] M. H. Van Emden and K. Yukawa, "Logic programming with equations", *The Journal of Logic Programming,* 1987, Vol. 4, pp. 265–288.