A New Basis for Spreadsheet Computing: Interval Solver™ for Microsoft® Excel

Eero Hyvönen and Stefano De Pascale

Delisoft Ltd
Urho Kekkosen katu 8 C 30, 00100 Helsinki, FINLAND
Tel. +358 9 6866550, Fax +358 9 68665544
{eero.hyvonen,stefano.depascale}@delisoft.com

Abstract

There is a fundamental mismatch between the computational basis of spreadsheets and our knowledge of the real world. In spreadsheets numerical data is represented as exact numbers and their mutual relations as functions, whose values (output) are computed from given argument values (input). However, in the real world data is often inexact and uncertain in many ways and the relationships, i.e., constraints, between input and output are far more complicated. This paper shows that Interval Constraint Solving, an emerging Artificial Intelligence based technology, provides a more versatile and useful foundation for spreadsheets. The new computational basis is 100% downward compatible with the traditional spreadsheet paradigm. The idea has been successfully integrated with Microsoft Excel as the add-in Interval Solver that seamlessly upgrades the arithmetic core of Excel into interval constraint solving. The product has been downloaded by thousands of end-users in about 70 countries around the world and has been used in various applications on business computing, engineering, education and science. There is an intriguing chance for a major breakthrough of the AI technology on the spreadsheet platform: Tens of millions of Excel users are making important decisions based on spreadsheet calculations.

1. Real world spreadsheet computing

The world is full of uncertainty and complexity. Everyday we are faced with questions like: How can I live within the given budget? Is this technical design possible, given the inaccurate component data? Uncertain data and constraints are extensively used in decision making. But spreadsheets, one of the most commonly used decision making aid of today, force us to use exact numbers for representing inexact data, thus distorting reality.

For example, consider the problem of computing the present value *p* of a future cash flow *c* that will be received after three years. If the annual future interest rates are r_1, r_2, and r_3 then *p* can be computed by using the (discounting) formula:

$$p = c/((1+r_1/100)\cdot(1+r_2/100)\cdot(1+r_3/100)) \qquad (1.1)$$

The problem is that future interest rates are volatile and that the value of *c* can be uncertain, too. The value of *p* is then uncertain as well. The question is: How to represent uncertain numerical values and how to compute them?

Another major limitation of spreadsheets is that the relationships between cell values can only be expressed with functions evaluating output cell values from given input cell values. In the real world things are more complicated. For example, consider the following formula for computing the *y*-coordinate of a projectile trajectory as a function of the *x*-coordinate, firing angle *a* and initial velocity *v*.

$$y = x\cdot\tan(a) + \frac{1}{2}\cdot 9.81\cdot a^2/(v^2\cdot\cos(a)^2) \qquad (1.2)$$

Assume that the target is on a 120m high hill (*y*) at a distance of 3200m (*x*). The initial speed (*v*) of the projectile is between 1250 m/s and 1300m/s. The task is to find out what are the possible angles (*a*) between 0 and 90 degrees for hitting the target. The formula and the given data clearly provide the answer but it is not clear how to back solve *a* from the function.

In a more general setting, the application problem may consist of a set of functions, equations, and inequalities, and the task is to solve any subset of variables involved, not only one variable. For example, what are the solutions to the equations below?

$$\begin{aligned} &\sin(x_1)+\cos(x_2)=\ln(x_3)\\ &\cos(x_1)+2\cdot\ln(x_2)=-\sin(x_3)+3\\ &3\cdot\ln(x_1)=\sin(x_2)-\cos(x_3)+2 \end{aligned} \qquad (1.3)$$

Traditional numerical techniques may find a solution to this problem, but not necessarily. The success depends on the equations and the initial guess values used as the starting point for the iteration. At best one solution is found for one starting point. The average spreadsheet user is not interested in such hidden technical details but simply wants to find a solution or ALL solutions to his/her problem by just pushing a button! Or (s)he wants to be sure that the problem has no solutions at all.

The examples above indicate that the following two key concepts of the current numerical spreadsheet paradigm are not flexible enough:

- **Cell value.** Only exact numbers can be used as values and be evaluated by formulas. Given the widespread and diverse usage of spreadsheets, a simple way for representing uncertainty is needed.
- **Formula.** Only functions can be used as formulas that explicate the relations between cell values. Means for representing arbitrary constraints between variables involved is needed. Especially, equations and inequalities used everywhere in business, engineering, and science should be available.

The case study of this paper shows that the idea of Interval Constraint Solving developed in the fields of Artificial Intelligence and Interval Analysis provides a new practical way to overcome these limitations. By generalizing the two core concepts of the spreadsheet paradigm, "value" and "formula", a new basis for the very idea of using spreadsheets can be laid.

The new vision has been materialized as the commercial deployed add-in product Interval Solver for Microsoft Excel, the result of some 16 man-years of research and development in Finland and Japan.

From the mathematical viewpoint, the solving power of the new technology is greater than with any traditional non-interval technique: All solutions (within a given precision level) to equations and other constraints can be found if enough time and memory is available. For example, Interval Solver can actually prove that (1.3) has exactly 5 different solutions. It suffices to push a button.

This paper first explains why and when interval constraint solving and Interval Solver is of use to a spreadsheet user. The architecture of the software is then presented and application of AI techniques is discussed. In conclusion, the significance of interval constraint technology in the development of the current spreadsheet paradigm is discussed.

2. Interval Solver for Microsoft Excel

Interval Solver is an add-in that virtually extends the mathematical basis of Excel into interval constraint solving. From the user's view point this means that (s)he can make better use of imprecise real world data and constraints, and solve new kind of problems that could not be addressed with spreadsheets before. With Interval Solver one can

- ***bound*** worst and best cases satisfying the spreadsheet formulas,
- ***solve back*** argument intervals from given goals,
- ***solve equations and other constraints*** needed in the application, and
- ***find the best solution*** to a problem.

2.1. Bounding worst and best cases

Intervals are perhaps the simplest way of representing uncertain numerical data. An interval [*min*, *max*] is a continuum of values between the bounds *min* and *max*. For example, the interest rate of the next year can be estimated by interval [4.0, 5.0]% meaning any value between 4% and 5%. By using interval analysis, safe bounds for function values with interval arguments can be computed.

Figure 1 depicts the situation of formula (1.1) on an Excel sheet with Interval Solver add-in loaded. The formula for P (seen in the formula bar) has been computed with the given uncertain interest rate and cash flow intervals. All the user has to do is to write the Excel function inside the =I(*expression*) formula of Interval Solver. After this, interval arguments can be used.

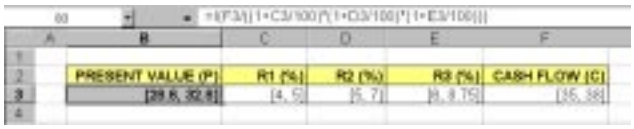

Figure 1. Evaluating an Excel formula with interval arguments.

The function value was initially *free*, i.e., its value was unknown. This equals to interval (-inf, inf). Interval Solver narrowed this value to [28.6, 32.8], the interval that is *guaranteed* to bound all possible values of the function down to user-given precision. The minimum represents the *global* minimum and the maximum the global maximum of the function within the argument interval limits. Notice that a number *x* is actually a collapsed interval [*x*, *x*] having the same lower and upper bound. This means that interval computations generalize traditional spreadsheet computations with exact values.

There are alternative approaches for representing uncertainty of numerical values, too. A simple way is to enumerate scenarios. For example, in figure 1 one could compute the formula with, e.g., all combinations of minimum and maximum values of the arguments. This simple approach is feasible only if the number of scenarios is small. In figure 1 there would be 2^4=16 scenarios if only bounds were considered; in the general case there are infinitely many possibilities. Another problem is that usually it is very difficult to say with what argument values the formula evaluates its global minimum or maximum that is often of greatest interest to the user. In the interval approach, all infinitely many scenarios are bounded within a single interval, and the actual global minimum and maximum can always be found.

A more sophisticated approach for representing numerical uncertainty is to use probability distributions as function arguments and then evaluate the functions using Monte Carlo simulation. This is a widely used approach and there are several software add-in packages available for spreadsheets, such as Crystal Ball (Crystal Ball, 1998) and @Risk (@Risk, 1998). In the probabilistic view, an interval can be seen as a distribution whose form is completely unknown and whose definite integral over the interval is one. All variables are statistically independent from each other.

For the average spreadsheet user, probabilistic modeling may be too difficult to use. Intervals provide a simpler low-end approach for representing numerical uncertainty and

have thus a better chance of being adopted by the spreadsheet users. Furthermore, intervals can be used for constraint solving as will be seen later. This is the key contribution of Interval Solver.

2.2. Solving back argument intervals

Assume that cell A1 contains the formula =A2+A3. Given argument values A2 and A3, the value of A1 is computed. The computational model of spreadsheets is a classical example of forward propagation.

However, in many problems the goal is known and the task is to back solve argument values that lead to feasible solutions. Constraint propagation is a handy classical AI technique for solving such problems. For example, if A1 and one of the arguments, say A2 are known in the above example, then the remaining argument A3 can be computed (A3=A1-A2). This value can then be propagated further to formulas in which variable A3 is used, and so on. Constraint propagation makes it possible to evaluate formulas "backwards" or "symmetrically", not only "forward" from known argument values to function value.

The idea of constraint propagation is not new in spreadsheet computing. It was actually adopted already by the early developers of the first major spreadsheet program, VisiCalc, in the late 70's. The best known result of this branch of development is TK!Solver (TK!Solver, 1998), a tool for mathematical modeling.

In interval constraint solving, the classical numerical value propagation is generalized into a still more versatile computational model: Intervals are propagated instead of exact numbers. The idea is to narrow initial variable intervals by using local consistency filtering techniques developed originally for solving discrete constraint satisfaction problems. The result of the narrowing procedure is a set of intervals that definitely bound all exact solutions to the constraints, i.e., the *solution set*.

For example, reconsider figure 1. The problem now is to determine the needed cash flow C and interest rates that would match a desired present value P. In figure 2, the user has set present value goal P=29 in the situation of figure 1. In ordinary Excel, one cannot assign to a cell a formula and a value simultaneously, but with Interval Solver this can be done by double-clicking the cell. In response, Excel has refined two interest rates and the cash flow accordingly. Modified values are shown in bold font for user's convenience. Interval Solver has bounded all possible scenarios that may lead to the goal within the given initial intervals – a useful piece of information for the decision-maker.

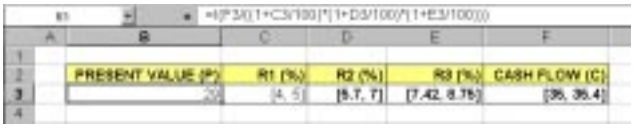

Figure 2. Interval goal seeking in Interval Solver.

Interval Solver is capable of determining the global value interval of function formulas, but in back solving only locally consistent (Hyvönen, 1992) bounds may be obtained. A problem of locally consistent bounds is that they may have excess width in certain situations, i.e., local narrowing does not necessarily result in the narrowest intervals bounding all solutions. Various consistency criteria can be used for filtering, such as arc-consistency, box consistency, 2B-consistency, and 3B(w)-consistency (Lebbah, Lhomme, 1998), but the problem of excess width remains in the general case. In spite of this limitation, narrowed bounds provide insight to the user regarding the safe space of possibilities available. If needed, the methods discussed below can be used for verifying the feasibility of an arbitrary point within a cell interval.

2.3. Solving equations, inequalities, and other constraints

The goal-seeking example above illustrated the idea that a spreadsheet formula is, from the mathematical viewpoint, actually a constraint *equation*. It tells how the function value and its arguments relate to each other, i.e., what value combinations are mathematically possible. In the same spirit the whole spreadsheet can be interpreted as a set of equations, inequalities, and other constraints whose variables have initial interval ranges. A spreadsheet thus formulates an Interval Constraint Satisfaction problem. The natural task then is either to

- **bound** all solutions of the constraint system or
- **find** its individual solutions.

This interpretation extends the usability of spreadsheet computing tremendously. Equations and inequalities can now be used on the sheet in addition to the traditional functions. In Interval Solver the expression inside the =I(*expression*) formula can be not only a function, but also an equation or an inequality, such as:

```
=I(A1^2*B2=SIN(C1)+3)
=I(LN(A1)^B1>=TAN(C1)+C1^3)
```

One can also use logical and numerical constraints mixed. For example:

```
=I(IMPLIES(A1>B1^2, AND(D2=0, A1<SIN(B1))))
```

Solutions to the equations and logical constraints can be generated automatically by pushing a button. The spreadsheet has become an expert for equation solving and Boolean logic.

For example, figure 3 depicts a typical problem encountered in electrical designing. The circuit to be analyzed consists of batteries and resistors whose voltages and resistances are known. The task is to solve the nine currents C13:C21 shown on the sheet. This can be done in a standard way by using Kirchoff's laws (equations) that relate resistances, voltages and currents with each other. The equations are written and shown in cells B25:B30 (for the currents) and B32:B34 (for the voltages).

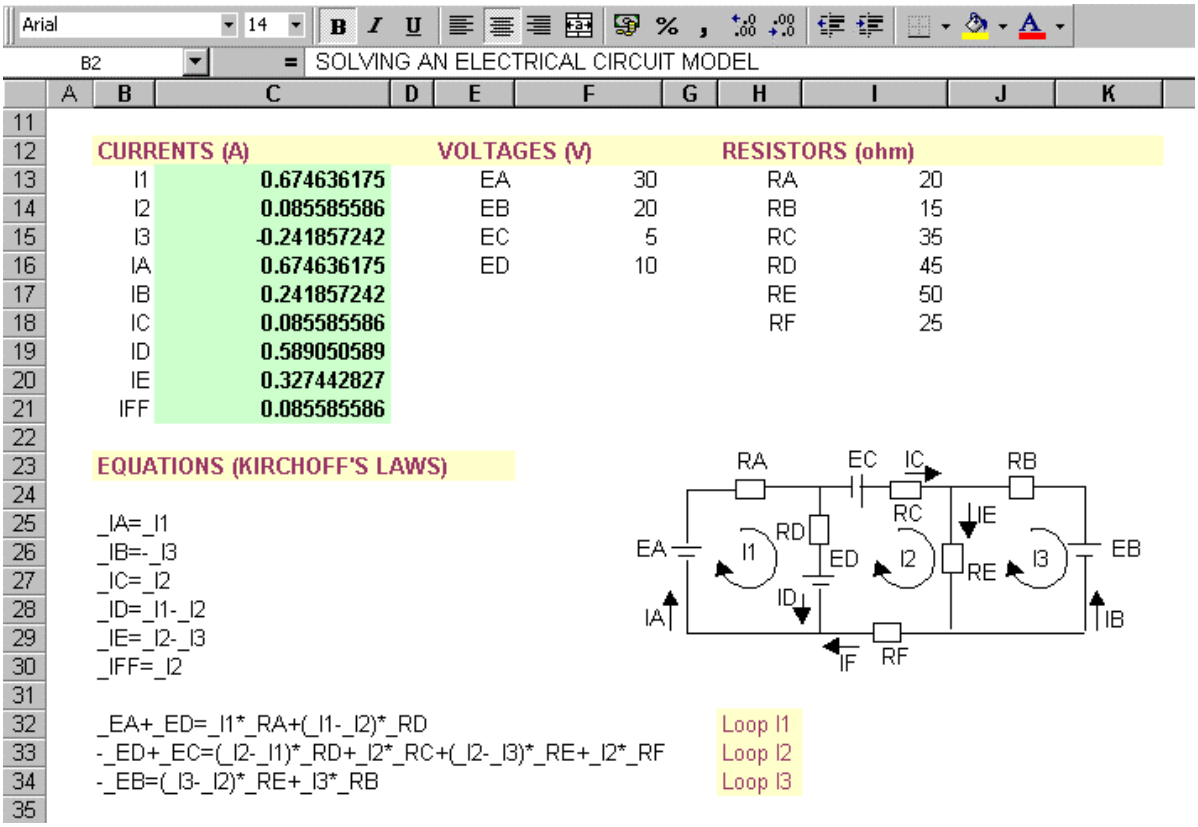

Figure 3. Solving the nine unknown currents of an electrical circuit. Cells have been given mnemonic names by using Excel's Name command on the Insert menu.

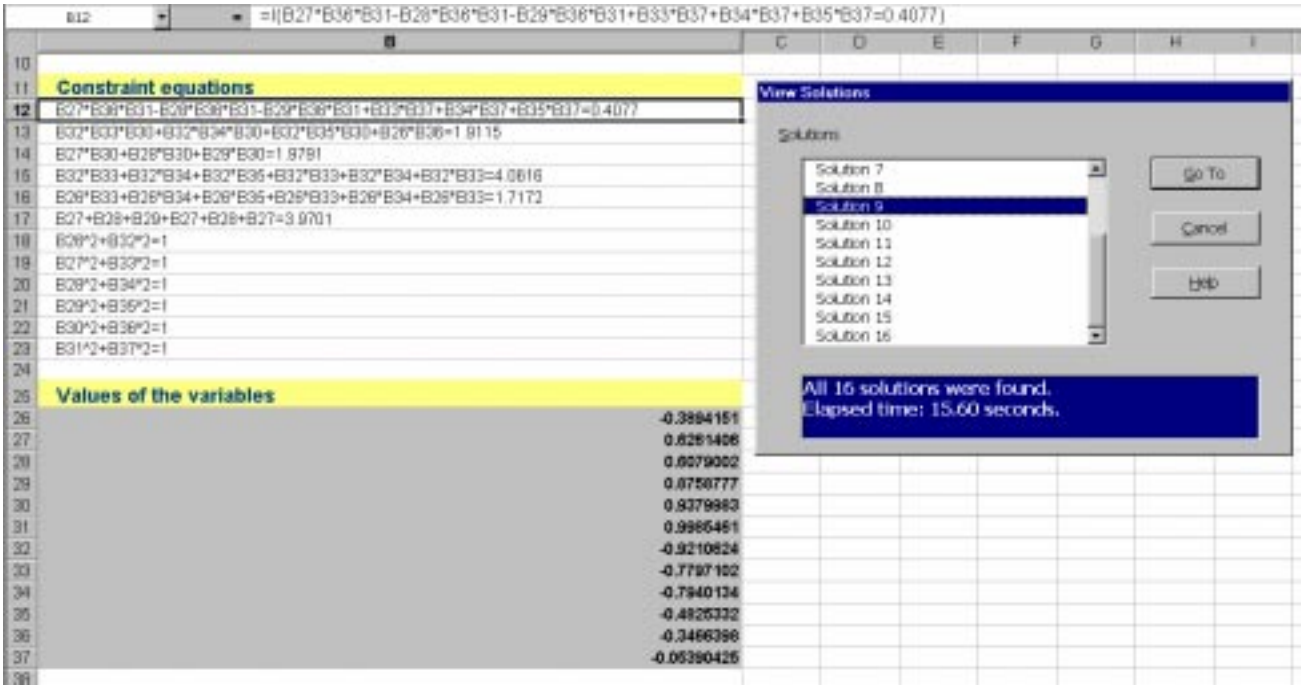

Figure 4. Finding all 16 solutions to kinematics equations describing a robot arm. The ninth solution is viewed.

Since an equation does not have a numerical value, the equation itself is shown as the value of the corresponding =I(*equation*) formula. Initially, values for the voltages and resistances were given and values for currents were unknown, i.e., they have very large interval values. The unique solution is found immediately and is shown in the figure.

Interval constraint solving techniques differ from other numerical techniques in one important way. Possible solutions are never accidentally lost. As a result, *all* solutions can always be found if enough time and memory is available. Furthermore, if a situation is found infeasible, then the problem has no solutions for sure. This guarantee holds even when rounding errors are present. In interval computations outward rounding interval arithmetic is used, and imprecise floating-point numbers are represented by tiny safe intervals bounding the actual value.

For the spreadsheet user this theoretical robustness is of great importance. Traditional numerical methods cannot in general guarantee that a solution will be found even if there were one. Convergence of iteration depends, e.g., on the gradients of the equations, initial guess values for the variables etc. It is not feasible to assume that a non-expert spreadsheet user understands the restrictions, conditions and limitations related to traditional numerical equation solving techniques. In interval solving *all* solutions can in principle *always* be found.

For example, figure 4 depicts the problem of finding the solutions to a difficult non-linear set of 12 kinematics equations. After evaluation, the View Solutions dialog box of Interval Solver has popped up and all 16 solutions to the equations can be viewed on the sheet. The user can be sure that this equation system has precisely these 16 solutions within the precision criteria used.

2.4. Finding the best solution to a problem

In figures 1 and 2, intervals were used for bounding the solution set. After the system has narrowed the intervals, the user can constrain the problem further by inserting new constraints or by modifying the intervals. For example, in figure 2 the target, present value P, was modified. After any modification, Interval Solver may be able to narrow related intervals further. The user and Interval Solver can work together in a mixed-initiative mode and the problem can be solved in a top-down fashion by refining stepwise constraints for the solutions. This is not possible in traditional spreadsheet computing.

This approach can be used for finding the best solution to the problem at hand, i.e., for solving optimization problems. The user sets desired goal values, Interval Solver narrows related cell values, the user modifies them again according to his preferences, and so on. If the situation is found at some point infeasible, special relaxation (Hyvönen, 1991) commands of Interval Solver can be applied in order to enlarge intervals and to make the bounds feasible again.

Interval Solver also contains a tool for solving traditional optimization problems directly with the help the of "Solver" add-in that comes with each Microsoft Excel copy. An interval CSP in Interval Solver consists of the interval bounds set for the cell values, their value types (real/integer), equations, inequalities, and logical constraints written on the sheet. These constructs can be transformed into a classical Excel Solver model and be solved using Excel Solver. By this way individual solutions can be found by which a given target function (cost function) gets its minimum, maximum or a preset specific value, given a set of constraints. By bounding solutions first with the interval model, the initial guess values can be selected within a reasonable range, and the optimization problem can be solved more easily. Interval Solver provides a natural way for expressing optimization problems. Any variable involved, not only the target cell value can be optimized dynamically based on the interval model.

However, since Excel Solver is a classical optimization tool, interval techniques are not used. This means that the solution found might be only a local optimum and that only at most one solution corresponding to the set of initial guess values can be found. There are in the general case no guarantees that a solution will be found even if there were one. Constrained interval optimization (Hansen, 1992) provides a remedy to this problem and will be available in future releases of Interval Solver as an alternative optimization tool.

3. Uses of AI technology

The mathematical basis of Interval Solver lays in the three InC++ interval libraries for C++, developed originally at VTT Technical Research Centre of Finland (Hyvönen, De Pascale, 1995; InC++, 1998):

Library	**Purpose**
LIA InC++	Overloads C++ arithmetic into extended interval arithmetic.
GIA InC++	Library for evaluating the global value range of a function with interval arguments.
ICE InC+	Interval Constraint Solving library based on LIA and GIA InC++.

Among these libraries, GIA and ICE are interesting from the AI viewpoint. Reconsider figure 1. The problem of determining the actual global minimum and maximum of P is easy in this case because the formula function happens to be monotonic. However, in the general non-monotonic case (e.g., formula (1.2)) the problem is very difficult both from the algorithmic and computational viewpoints. The global min/max is then not obtained by a combination of argument interval limits.

There is only one class of numerical techniques that is guaranteed to always find the global minimum/maximum, global interval optimization techniques (Hansen, 1992). These algorithms vary in detail but the underlying idea in

all of them is to perform an exhaustive branch-and-prune search in which the initial argument intervals are split into tighter and tighter subintervals until precision conditions for the solution are satisfied. The search is accelerated by various pruning heuristics based on the best min/max candidate found thus far, on the mathematical properties of the function (such as first and second gradients), and on special narrowing operators for the arguments (such as the interval Newton operator). An indication of the implementational complexities involved is that GIA InC++ library employed in Interval Solver consists of over 50.000 lines of C++ code.

```
# include <ice.h>
// Include ICE InC++ library header
main () {
  // Construct the equation set object "I" (of class Ice)
  Ice I;
  I.SetDefaultUnknown("[-1e8,1e8]");
  // Default interval bounds for variable values
  I.InsertConstraint("x1^2+x2^2+x3^2-1=0");
  I.InsertConstraint("x1^2+x2^2+x3^2-2*x1=0");
  I.InsertConstraint("x1^2+x2^2-1=0");

  // Solve (evaluate) equation constraints
  I.SetPropagationMode(PMGlobal);
  // Set mode for finding individual solutions
  I.SetMaxSolutionNumber(1000);
  // Search for up to 1000 solutions
  ICEConsistencyType c;
  // Consistency type after evaluation
  I.Evaluate(&c);
  // Analyze and display the result
  if (c!=CTInfeasible)
      I.DisplayGlobalSolutions();
  return 0;
};
```

Figure 5. An example of programming with ICE InC++. All solutions to a set of tree equations in three variables are solved and displayed.

The key technology underlying Interval Solver is Interval Constraint Satisfaction (Davis, 1987; Cleary, 1987, Hyvönen, 1989) developed in the fields of artificial intelligence, (constraint) logic programming, and interval analysis (Interval, 1998). The interval constraint satisfaction problem (ICSP) corresponding to a sheet is represented as a C++ object of class Ice included in the ICE InC++ library. This class has a simple dynamic string-based interface by which constraints can be inserted and removed from the ICSP, interval domains set for the variables, precision criteria set for solutions etc. For an example of the programming interface, the C++ code in Figure 5 shows how to solve three equations in three variables with ICE InC++.

User operations on a sheet, such as inserting a formula, setting a cell value, etc. are mapped into sequences of member function calls of the underlying Ice object. This mapping is written in the macro language of Excel, Visual Basic. 117 different member functions of ICE InC++ library are used for the interface.

ICE InC++ library does all mathematical constraint solving regarding formulas written inside the =I(*expression*). Excel itself maintains the algebraic formulas on the sheet mutually consistent. This makes Interval Solver invisible to the user and the integration seamless. For example, changing cell names in formulas when copying, pasting or moving cells is automatic as usual. User-defined cell names, as well as the different alias function names used in the various country versions of Excel are available with Interval Solver, too. Figure 6 illustrates the general integration architecture.

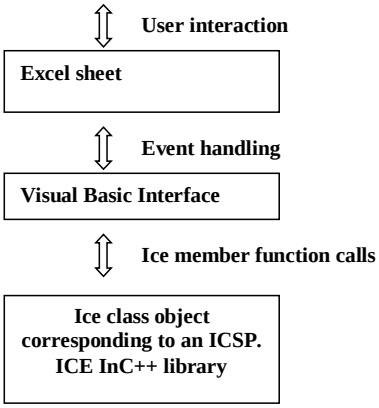

Figure 6. The interaction model of Interval Solver. Visual Basic interface layer catches user interactions (inserting data, moving cells, etc.) and commands. The Ice object, i.e., the ICSP corresponding to the sheet, is updated accordingly, or solved depending on the user interaction.

ICE InC++ library uses a large variety of interval constraint solving techniques. The constraint set is manipulated and simplified by algebraic manipulation routines in order to make it easier to solve numerically. In numerical evaluation, the tolerance propagation approach (Hyvönen, 1992) is enhanced with global interval optimization algorithms (Hansen, 1992), narrow operators (Benhamou et al., 1994; Van Hentenryck et al., 1997), conditioning matrices (Kearfott, 1996), and structure sharing techniques (Hyvönen, De Pascale, 1996).

Interval constraint solving was originally proposed as a new computational basis for spreadsheet programs in (Hyvönen, 1991; Hyvönen, De Pascale, 1996).

4. Application Use and Payoff

Interval Constraint Solving technology has recently gained more and more attention not only in AI research but in business as well. There has been a lot of development activity in the logic programming community resulting in several interval constraint extensions of Prolog, such as BNR Prolog (BNR Prolog, 1998) and Prolog IA (PrologIA, 1998). First stand-alone mathematical solvers based on the new scheme, such as Numerica (Numerica, 1998) and UniCalc (UniCalc, 1998), have been introduced in the

market. These systems and tools are intended mainly for expert usage and programmers.

In contrast, Interval Solver targets (also) non-expert users of spreadsheets. With the help of the new computational basis, the usage of spreadsheets has been extended to new classes of applications that deal with uncertain data and/or involve problem solving under user-given constraints, a typical situation in business planning, technical design, science, and in many other fields.

Interval Solver is a generic tool for end-users spreadsheets. It virtually generalizes the arithmetical basis of Excel and the range of applications is therefore as wide as that of spreadsheets. Most users indicate business computations as their main area of interest. Typical applications include cash flow analysis, budgeting, and risk analysis involving uncertain future data. In engineering applications, Interval Solver provides the user with a simple tool for performing design and other calculations involving e.g. component data with tolerances, and for solving equations. The pay-off comes from getting more realistic and better results to support decision making. Non-expert users can now solve -- with little training in their customary spreadsheet environment -- new classes of mathematically complicated problems.

Together with word processing, database and Internet tools, spreadsheets have been one of the most influential software applications of information technology. According to a Gartner Group some 30 million spreadsheet programs were shipped in 1997, most of which (90%) were copies of Excel, and the market is growing rapidly. There is the intriguing possibility that interval constraint technology will eventually lead to a paradigm shift in utilizing spreadsheets.

5. Development and Deployment

In the late 80's and early 90's, research on using interval arithmetic as the basis for interval constraint satisfaction was carried at VTT Technical Research Centre of Finland, and in a joint project with Electrotechnical Laboratory, Japan. A result of this work was an interval constraint solver and an interval spreadsheet demo system implemented in Lisp. Based on the first results, it was decided to implement the technology for industrial applications in C++ and to apply it to a major commercial spreadsheet program, Excel.

It turned out, however, that the 16-bit address space provided by Excel at that time was too small for handling problems of reasonable size. Also Excel's macro language turned out to be too limited for a commercial level implementation of the new interval vision. With the new 32-bit Windows versions and the new Visual Basic macro language, the situation changed rapidly in 1996. The first implementation of interval constraints for Excel called "Range Solver" was exhibited by VTT at CeBIT 96 fair in Hannover, Germany. Interval Solver is its direct descendant commercialized by Delisoft Ltd, a spin-off of VTT.

The first version of Interval Solver 97 was finished during autumn 1997 and was released internationally in April 1998 at COMDEX Japan, Tokyo. The software evaluation kit has been available through Delisoft Web site (www.delisoft.com), Ziff-Davis Libraries, Download.com, etc., and recently also through representatives in various countries (USA, Canada, Australia, Hong Kong, Korea,…). There are currently several thousand downloaders in about 70 countries.

The software consists of over 100,000 lines of code 80% of which is mathematical routines in C++. Several people were involved with the research on interval computations, but the actual code of Interval Solver as well as the manual, setup program, and electronic tutorial were written by the authors of this paper.

The development and especially the commercialization phases of Interval Solver were far more demanding than was initially expected. In order to meet the high efficiency requirements of spreadsheet users, the software had to be geared and tuned very carefully. Tiny modifications in the algorithms easily resulted in order of magnitude differences in performance. Computational efficiency of interval constraint solving techniques is very sensitive not only to the algebraic form of the ICSP but also to the initial interval values used. Various heuristics can be used to speed up convergence, but there is no single optimal strategy that works always fine.

Besides the technical difficulties in implementing and tuning the mathematical constraint engine, lots of difficulties were encountered with the interface to Excel. A key problem there was the enormous versatility of ways in which the user may interact with Excel and potentially confuse the system by making the sheet and the underlying ICSP model mutually incoherent. Most operations in Excel such as inserting a formula can be made in several alternative ways. All of them have to be caught. An additional practical problem was that a new fundamentally different version of macro language provided by Microsoft for Excel 97 was released in the middle of the development process causing redesign needs for the interface. Fortunately, the new version was more versatile from Interval Solver viewpoint. Last but not least, several deficiencies were encountered in different Excel versions. They had to be circumscribed by special programming tricks.

6. Conclusions

Interval constraint solving and Interval Solver provide a more versatile basis for spreadsheet computing. Two key concepts of spreadsheets have been generalized:

- The idea of *cell value* is generalized from exact numbers to intervals. An exact value is a special case of the notion of interval.
- The idea of (function) *formula* is generalized into equations, inequalities, and logical constraints. A function is a special case of an equation.

From the computational viewpoint, the idea of *forward propagation* of exact values is generalized into interval constraint propagation. Again, forward propagation is a special case of the new model, interval constraint propagation.

Interval constraint solving is a conceptually simple, robust scheme for representing and solving difficult mathematical problems under uncertainty. Solutions are never lost as when using traditional numerical techniques. A price to be paid for the robustness and the ease of usage is increased computational complexity. However, results indicate that in many cases interval constraint solving methods can successfully compete with or even outperform the best traditional numerical techniques (Van Hentenryck et al., 1997).

Spreadsheet programs are among the most widely used applications of information technology. However, after the pioneering days of VisiCalc in 1979, their underlying computational idea has not changed much. Interval Solver demonstrates that artificial intelligence techniques can make a substantial contribution in the development of the spreadsheet paradigm.

Acknowledgements

Thanks to Dr. Eero Peltola, Technology Development Centre of Finland, Sitra, and VTT Information Technology for fruitful co-operation.

References

@Risk (1998) Product information of @Risk available at http://www.palisade.com.

Benhamou, F., McAllester, D., Van Hentenryck, P. (1994) CLP(Intervals) revisited. Proceedings of the International Symposium on Logic Programming (ILPS-94), Ithaca, New York, 124-138.

BNR Prolog (1998) Home page of BNR Prolog: http://www.als.com/als/clpbnr/clp_info.html.

Cleary, J. (1987) Logical Arithmetic. *Future Computing Systems* 2 (2), 1987.

Crystal Ball (1998). Product information of Crystal Ball available at http://www.decisioneering.com.

Davis, E. (1987) Constraint propagation with interval labels. *Artificial Intelligence* 8, 99-118.

Hansen, E. (1992) *Global Optimization Using Interval Analysis*, Marcel Dekker, New York.

Van Hentenryck, P., MacAllister, D., Kapur, D. (1997) Solving polynomial systems using a branch-and-prune approach. *SIAM Journal of Numerical Analysis*, 34 (2).

Van Hentenryck, P., Michel, L., Deville, Y. (1997) *Numerica. A Modeling Language for Global Optimization*. MIT Press, Cambridge.

Hyvönen, E. (1989) Constraint reasoning based on interval arithmetic. *Proceedings of IJCAI-89*, Morgan Kaufmann, Los Altos, Calif., 1193-1198..

Hyvönen, E. (1991) Interval constraint spreadsheets for financial planning. *Proceedings of the 1st International Conference on Artificial Intelligence Applications on Wall Street*. IEEE Press, New York.

Hyvönen, E. (1992) Constraint reasoning based on interval arithmetic: the tolerance propagation approach. *Artificial Intelligence* 58, 71-112.

Hyvönen, E., De Pascale, S. (1995) InC++ library family for interval computations. *Reliable Computing*, supplement, Proceedings of Applications of Interval Computations, El Paso, Texas, 1995.

Hyvönen, E., De Pascale, S. (1996) Interval Computations on the Spreadsheet. In: (Kearfott and Kreinovich, 1996), 169-210.

Hyvönen E., De Pascale, S.: Shared Computations for Efficient Interval Function Evaluation. In: G. Alefeld, E. Frommer (eds.): *Scientific Computing, Computer Arithmetic and Validated Numerics*, Akademie-Verlag, Berlin, Germany, 1996.

Interval (1998) Home page of interval computations research: http://cs.utep.edu/interval-comp/main.html.

InC++ (1998) Home page of InC++ libraries: http://www.delisoft.com/InCLibraries.

Interval Solver (1998) Interval Solver home page: http://www.delisoft.com/ExcelProducts/IntervalSolver.

Kearfott, B. (1996) *Rigorous Global Search: Continuous Problems*. Kluwer, New York.

Kearfott, B., Kreinovich, V. (eds.) (1996) *Applications of Interval Computations.* Kluwer, New York.

Lebbah, Y., Lhomme O. (1998) Acceleration methods for numeric CSPs. *Proceedings of AAAI-98*, AAAI Press, Menlo Park, Calif., 19-24.

Moore R. (1966) *Interval Analysis*. Prentice-Hall, Englewood Cliffs, N.J.

Numerica (1998) Home page of Numerica: http://www.ilog.com/html/products/optimization/numerica.htm.

Prolog IA (1998) Home page of Prolog IA software: http://prologianet.univ-mrs.fr/Us.

TK!Solver (1998) TK!Solver home page: http://www.uts.com/.