# Model-Based Analogue Circuit Diagnosis with CLP($\Re$)*

Igor Mozetič
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria

Christian Holzbaur
Austrian Research Institute for Artificial Intelligence, and
Department of Medical Cybernetics and Artificial Intelligence
University of Vienna
Freyung 6, A-1010 Vienna, Austria

Franc Novak
Jozef Stefan Institute
Jamova 39, 61000 Ljubljana, Slovenia

Marina Santo-Zarnik
Iskra HIPOT
Šentjernej, Slovenia

**Abstract**

Model-based diagnosis is the activity of locating malfunctioning components of a system solely on the basis of its structure and behavior. Diagnostic systems usually rely on qualitative models and reason by local constraint propagation methods. However, there is a large class of applications where ATMS-like systems or pure logic programs are unpractical since they are unable to solve simultaneous equations. In particular, modeling real-valued system parameters with tolerances requires some degree of numerical processing, and feedback loops in general cannot be resolved by

*Appears in *Proc. 4th Intl. GI Congress* (W. Brauer, D. Hernandez, Eds.), pp. 343-353, München, October 23-24, 1991, Springer-Verlag (IFB 291).

local constraint propagation methods. Examples of such systems are analogue circuits, e.g., amplifiers or filters. In the paper we describe the role of Constraint Logic Programs over the domain of reals (CLP($\Re$)) in representing both, qualitative and numerical models. CLP($\Re$) is a logic programming system extended with a solver for systems of linear equations and inequalities over real-valued variables.

# 1 Introduction

Different Computer Aided Engineering tools are available for electronic circuit and systems design. CAE solutions to the *digital* design problem can be regarded as mature, while *analogue* design still lacks sufficient support even in the early design phases. In digital design, schematic capturing process and simulation with timing analysis are tightly coupled to fault simulation and test vector generation. Once a designer has verified the logic scheme of a circuit and has completed logic simulation, a full description of a defect-free version of the circuit together with an initial set of test vectors are available. Fault simulation uses the description and systematically inserts defects (i.e., simulates faults) to check if the given set of test vectors can detect the difference between the operation of the defect-free and the simulated faulty circuit. Usually, the initial set of test vectors has to be upgraded to reach the desirable fault coverage, typically close to 100%.

A similar approach in analogue design would face serious difficulties due to the fact that fault modeling is still a controversial issue (Ohletz 1991). Besides catastrophic (hard) faults which could be to some extend related to the popular digital *stuck-at* fault model, the class of deviation (soft) faults, due to the parameters deviating from the nominal values, must be considered (Duhamel & Rault 1979, Bandler & Salama 1985). As regards fault simulation, in the worst case a complete transient simulation must be performed for each fault. Fault simulation time for a given range of deviation faults may quickly reach unacceptable limits (Ohletz 1991). Hence, fault simulation is relatively uncommon for analogue circuits (Duhamel & Rault 1979).

This situation seems ideal for the application of an AI technique called model-based diagnosis (e.g., Genesereth 1984, Davis 1984, de Kleer & Williams 1987, Reiter 1987). In model-based approach one starts with a model of a real-world system which explicitly represents just the structure and normal behavior of the system components. When the system's actual behavior is different from the expected behavior, the diagnostic problem arises. The model is then used to identify faulty components and their internal states which account for the observed behavior.

However, the applicability of model-based techniques is largely limited to academic problems. In our view one of the major obstacles which prevented a wider application to real-world problems is that models are usually restricted to qualitative (non-numeric) descriptions, and to an ATMS-like local constraint propagation methods. General Diagnostic Engine (GDE, de Kleer & Williams 1987), for example, is unable to solve simultaneous

equations, which makes it unpractical for a large class of applications.

In the paper we describe the role of Constraint Logic Programs over the domain of $\Re$eals (CLP($\Re$), Jaffar *et al.* 1986, Cohen 1990) in representing and diagnosing a larger class of models. CLP($\Re$) is a logic programming system extended with a solver for systems of linear equations and inequalities. It is well suited to model real-valued system parameters with tolerances and feedback loops which in general cannot be resolved by local constraint propagation methods.

In section 2 we give a brief overview of the CLP($\Re$) system. In section 3 we show how models of analogue circuits (operating under the AC conditions) can be concisely specified in CLP($\Re$). The main advantage of our approach, in contrast to standard simulation packages, is that the same model can be used for both, simulation and diagnosis. In section 4 we concentrate on diagnosis of soft faults due to a single parameter value (e.g., a resistor or a capacitor) out of tolerances. The manufacturing technology of a specific device under consideration makes internal probing difficult and undesirable. Preliminary results indicate that CLP($\Re$) has a potential to become a basis for software tools used in the design and testing of analogue circuits.

# 2 The CLP($\Re$) system

The Constraint Logic Programming scheme (Jaffar *et al.* 1986) provides a general framework from which extensions of Prolog can be derived. The unification mechanism, as used in Prolog, is replaced by a more general operation — constraint satisfaction over specific domains (Cohen 1990). An instance of the scheme, CLP($\Re$), extends Prolog with interpreted arithmetic functions and a solver for systems of linear equations and inequalities over the domain of $\Re$eals.

We illustrate the CLP($\Re$) language by specifying addition and multiplication of complex numbers. A complex number $Z = Re+j*Im$ is represented by a pair *c(Re,Im)*.

*add( c(Re1,Im1), c(Re2,Im2), c(Re1+Re2, Im1+Im2) ).*

*mult( c(Re1,Im1), c(Re2,Im2), c(Re1*Re2−Im1*Im2, Re1*Im2+Im1*Re2) ).*

The above program allows for queries involving not only addition and multiplication, but subtraction and division of two complex numbers as well. For example:

← *mult( c(1,2), c(3,4), Z ).*
*Z = c(−5,10)*

← *mult( X, c(3,4), c(−5,10) ).*
*X = c(1,2)*

Answering the second query actually requires to solve the following system of equations:

*3*Re1 − 4*Im1 = −5,*
*4*Re1 + 3*Im1 = 10.*

which yields the solution *Re1=1, Im1=2.*

In our implementation of CLP($\Re$) linear equations are kept in *solved form.* Variables appearing in the equations are split into two disjoint sets: *dependent* variables and *independent* variables. Dependent variables are expressed through terms containing independent variables. When a new equation is to be combined with a system of equations in solved form, all its dependent variables are replaced by their definitions which results in an expression over independent variables. An independent variable is selected then, and the expression is solved for it. After the resulting definition has been back-substituted into the equation system, the isolated variable can be added as a new dependent variable, and the equation system is in solved form again. Inequalities are expressed in terms of independent variables.

The satisfiability of a system of linear inequalities is decided by a version of the Shostak's 'Loop Residue' algorithm (Shostak 1981, Kraemer 1989). Each inequality is represented as an edge in the inequality graph $G$. The algorithm only deals with loops in $G$. For each loop, the residual inequality of the loop is computed and entered as a new edge into $G$. The loop residue computation is iterated until no more loops can be created, or one of these new edges is determined to correspond to an unsatisfiable inequality. Each loop residue computation essentially eliminates one variable — therefore an unsatisfiable inequality will eventually result in a ground inequality $k < 0$, where $k$ is a positive constant. The basic algorithm was extended to strict and nonstrict inequalities.

The ability to solve systems of linear inequalities enables simple computation with intervals. Each interval is implemented as a conjunction of two inequalities which associate an upper and a lower bound with a variable. Having the graph $G$ to encode the set of inequalities is particularly useful when one is interested in implied inequalities, i.e., in current upper and lower bounds of a variable. Asking for implied inequalities is nothing but the residual inequality computation along a path in $G$. In many cases one is interested in a relation between a specific variable and zero, i.e., in interval bounds for the variable. The algorithm computes the set of all paths from the variable to zero, which is a distinguished node in $G$. Among many possible paths the one yielding the tightest bound is selected.

In order to account for tolerances in model parameters we allow constants in linear expressions to be specified by a pair *i(Min, Max)* which denotes a lower and an upper bound. Take the following specification of the behavior of a resistor:

*resistor( R, V1, V2, I )* $\leftarrow$ *V1−V2 = R*I.*

Now consider two resistors in a series, with voltages of 12.5 and 10 Volts applied at the ends (an example from McKeon & Wakeling 1990). Both resistances are within the range *i(1000, 2000)* $\Omega$. The question is: What is the voltage range at the node between the two resistors? The query returns the following set of constraints:

$\leftarrow$ *resistor( i(1000,2000), 12.5, V, I ), resistor( i(1000,2000), V, 10, I ).*

*12.5* $\geq$ *1000*I + V,*
*2000*I + V* $\geq$ *12.5,*

$V \geq 10 + 1000*I,$
$10 + 2000*I \geq V,$
$I > 0$

from which the interval bounds for the voltage $11.6667 \geq V,\ V \geq 10.8333$ are deduced. In contrast to our, symbolic approach, McKeon & Wakeling use an iterative, numeric approach to compute the interval bounds.

Our implementation of CLP($\Re$) is preferred over existing versions (Heintze *et al.* 1987a, Jaffar 1990) since it allows for the simultaneous use of solvers for different domains in a consistent framework. This suits well the computational demands that arise in the context of hierarchical abstractions (Mozetic & Holzbaur 1991a). The numerical level of the model can be formulated with CLP($\Re$) for example, and successive abstractions thereof typically utilize constraint propagation over finite domains. The implementation of the specialized solvers is based on user-definable extended unification. As the solvers are written in Prolog, they can easily be customized to specific demands. The choice of Prolog as an implementation language for the equation solver for CLP($\Re$) led to a reduction in code size by an order of magnitude.

Beside the principal (software engineering) issues that motivated our implementation of CLP($\Re$), the availability and the quality of Sicstus Prolog (Carlsson & Widen 1990) somehow aposteriori justified the selection of Prolog as an implementation language. Sicstus Prolog has a compiler which can produce native machine code and a garbage collector. The basic mechanisms provided for the implementation of *freeze/2* and *dif/2* are very useful for the implementation of extended unification, the basis of our approach.

Our first CLP($\Re$) implementation was based on the C-Prolog interpreter (Holzbaur 1990). For the performance comparison against the C implementations of CLP($\Re$) this was disadvantageous, as the unification extensions, i.e., the CLP($\Re$) solver, were interpreted only. However, given the Sicstus compiler, the performance of our current Prolog CLP($\Re$) implementation is somewhere in-between the IBM (Jaffar 1990) and the Monash (Heintze *et al.* 1987a) implementations. A further improvement of our version of CLP($\Re$), which did not require any extra effort from our side, accrues from the increased numerical precision in floating point operations in Sicstus (double precision). Since Sicstus also provides infinite precision integer arithmetics, the implementation of CLP($\mathcal{Q}$) ($\mathcal{Q}$ = rationals) is easy and reasonably efficient.

# 3 Modeling analogue circuits

Model-based reasoning about a system requires an explicit representation (a model) of the system's components and their interconnections. Reasoning is typically based on theorem proving if a model is represented by first-order logic (Genesereth 1984, Reiter 1987), or on constraint propagation coupled with an ATMS (de Kleer & Williams 1987). Dague *et al.* (1990) use an ATMS-like system, augmented with the ability to compute with intervals,

but unable to solve simultaneous equations, for the diagnosis of analogue circuits.

We represent models by logic programs, by CLP($\mathcal{B}$) ($\mathcal{B}$ = booleans) (Mozetic & Holzbaur 1991b), or by CLP($\Re$), depending on the domain of application. The first application of CLP($\Re$) to the analysis of analogue circuits was reported by (Heintze *et al.* 1987b).

**Definition**. A *model* of a system is a triple $\langle SD,\ COMPS,\ OBS\rangle$ where

1. *SD*, the system description, is a logic program with a distinguished top-level binary predicate *m(COMPS, OBS)* which relates states of the system components to observations.

2. *COMPS*, states of the system components, is an $n$-tuple $\langle S_1, \ldots, S_n\rangle$ where $n$ is the number of components, and variables $S_i$ denote states (e.g., normal or abnormal) of components.

3. *OBS*, observations, is an $m$-tuple $\langle P_1, \ldots, P_i, In_{i+1}, \ldots, In_j, Out_{j+1}, \ldots, Out_m\rangle$ where $P$ are the model parameters, and *In* and *Out* denote inputs and outputs of the model, respectively.

In a logic program, $n$-tuples are represented by terms of arity $n$. Variables start with capitals and are implicitly universally quantified in front of a clause, and constants start with lower-case letters. In *SD* we refer to a distinguished constant *ok* to denote that the state $S_i$ of the component $i$ is normal.

We illustrate design and modeling of analogue circuits on a filter example. Using Micro-cap III (Spectrum), a standard electronic circuit simulation package, an active 5th order low pass RC filter has been designed (Figure 1). The filter is actually composed of two FDNR stages realized in thick film hybrid technology connected on a printed circuit board. In order to simplify the example, we concentrate on a single filter stage and ignore parameter tolerances. In what follows we also omit the operational amplifier model which was taken from the Micro-cap III manual and instantiated with the data provided by the manufacturer.

*SD* of the filter stage model (Figure 1) consists of the following CLP($\Re$) program. *COMPS* is a seven-tuple *comps(R1,...,R5,C1,C2)*, where $R_i$ and $C_i$ denote states of resistors and capacitors, respectively — we assume that the amplifiers do not fail. *OBS* is a triple *obs(F,V1,V2)*, where $F$ is a given frequency, and *V1*, *V2* are input and output voltages of the stage, respectively.

```
stage( comps(R1,R2,R3,R4,R5,C1,C2), obs(F,V1,V2) )  ←
        W = 2*3.14159*F, Vgnd = c(0,0),
        resistor( R1, 5513, V1, V2, Ir1 ), Ir1 = Ir2,
        resistor( R2, 727, V2, V3, Ir2 ),
        add( Ic1, Ia1, Ir2 ),
        capacitor( C1, 10.0e-9, W, V3, V4, Ic1 ),
        add( Ic1, Io2, Ir3 ),
```

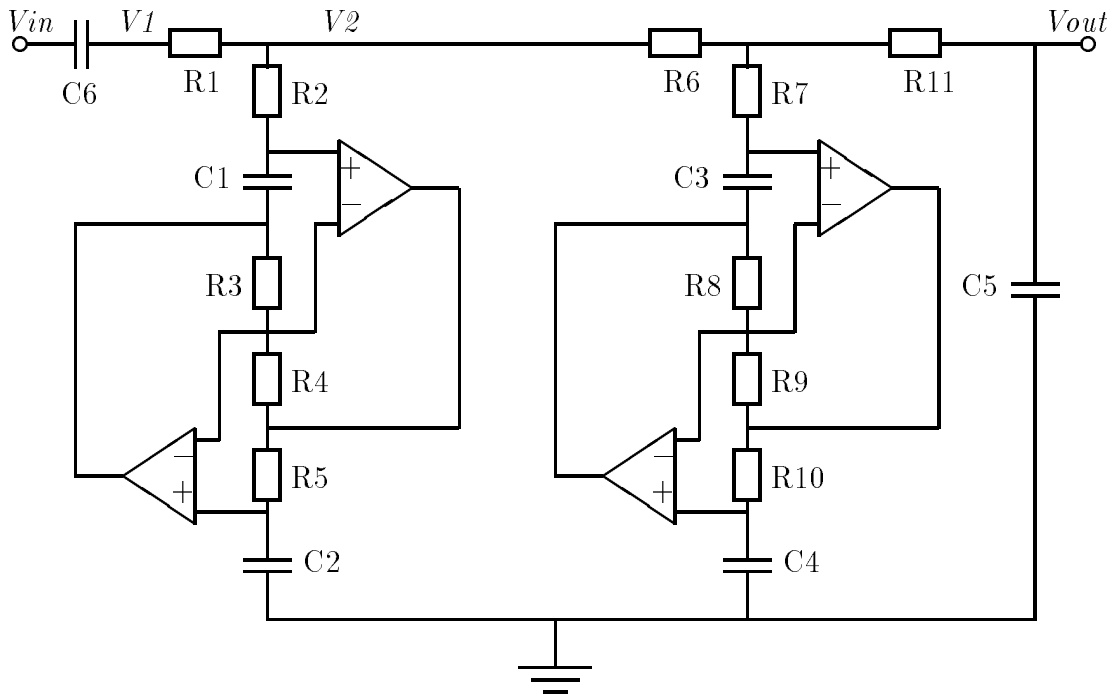


Figure 1: A low pass filter consisting of two structurally equivalent stages, and additional resistor *R11* and capacitors *C5* and *C6*.

*resistor( R3, 10000, V4, V5, Ir3 ),*
*add( Ir4, Ib, Ir3 ), add( Ib1, Ib2, Ib ),*
*resistor( R4, 10000, V5, V6, Ir4 ),*
*add( Ir4, Io1, Ir5 ),*
*resistor( R5, 5693, V6, V7, Ir5 ),*
*add( Ic2, Ia2, Ir5 ),*
*capacitor( C2, 10.0e-9, W, V7, Vgnd, Ic2 ),*
*amplifier( W, V3, V5, V6, Ia1, Ib1, Io1 ),*
*amplifier( W, V7, V5, V4, Ia2, Ib2, Io2 ).*

The model relates states of resistors and capacitors to the frequency and measurable voltages. Since the filter operates under the AC conditions, all the voltages and currents are represented by complex numbers. Literals in the body of the clause represent model components which enforce local constraints between voltages and currents (e.g., Ohm's law). In nodes, Kirchoff's current law is enforced by the *add/3* predicate. Shared variables represent connections between the components and enforce global constraints, e.g., Kirchoff's law for voltages.

In addition to the structure of the model, behavior of its components must be defined. Normal behavior of a component specifies a relation between voltages and currents when

the component is in a state *ok* (e.g., for a capacitor, $V1 - V2 = -\frac{j}{\omega C} * I$):

```
resistor( ok, R, V1, V2, I )  ←
        add( DV, V2, V1 ),
        mult( c(R,0), I, DV ). capacitor( ok, C, W, V1, V2, I )  ←
        add( DV, V2, V1 ),
        mult( c(0,W*C), DV, I ).
```

For analogue components, it is relatively easy to specify the behavior in the case of hard faults (e.g., open or shorted). However, there is an infinite number of soft faults in between, due to the possible shifts in parameter values. In order to capture them all, no constraints between voltages and currents should be imposed by the fault model, i.e., a *weak* fault model must be used:

```
resistor( ab(R), _, V1, V2, I )  ←
        add( DV, V2, V1 ),
        mult( c(R,_), I, DV ).

capacitor( ab(C), _, W, V1, V2, I )  ←
        add( DV, V2, V1 ),
        mult( c(_,W*C), DV, I ).
```

We denote an abnormal state of a component by a term *ab(X)* instead of a constant *ab*. The idea is that a faulty resistor exhibits some unknown resistance $R$, and a faulty capacitor some unknown capacitance $C$ which can be computed from voltages and currents. This can help in estimating relative likelihood of an individual component beeing faulty during diagnosis, as described in the next section.

# 4 Analogue diagnosis

Assume that the design of the filter stage has been proven to meet the desired specification and that a prototype series has been manufactured. The gain-frequency characteristics of the designed stage is depicted in Figure 2 (top). The graph was obtained by simulating the CLP($\Re$) model, and results closely match the simulation results of the Micro-cap III package. However, it should be noted that in practice the simulation results might be quite different from reality, and that construction of good simulation models of analogue circuits cannot be taken for granted. Nevertheless, we will assume that for our purposes the simulation model does match the reality.

During the fabrication process, thick film resistors *R1, R2, R3,* and *R4* are adjusted to nominal values within the expected tolerances. Resistor *R5* is used for active laser trimming to compensate for variations in other components values. In the process of adjusting *R5*, possible hard faults are detected, hence only deviation faults in *R1, R2, R3, R4* and *C1, C2* may pass undetected. The filter stage is set to a chosen pole frequency by adjusting *R5*. Now, let us assume that due to a drift or possible failures in laser trimming process, the

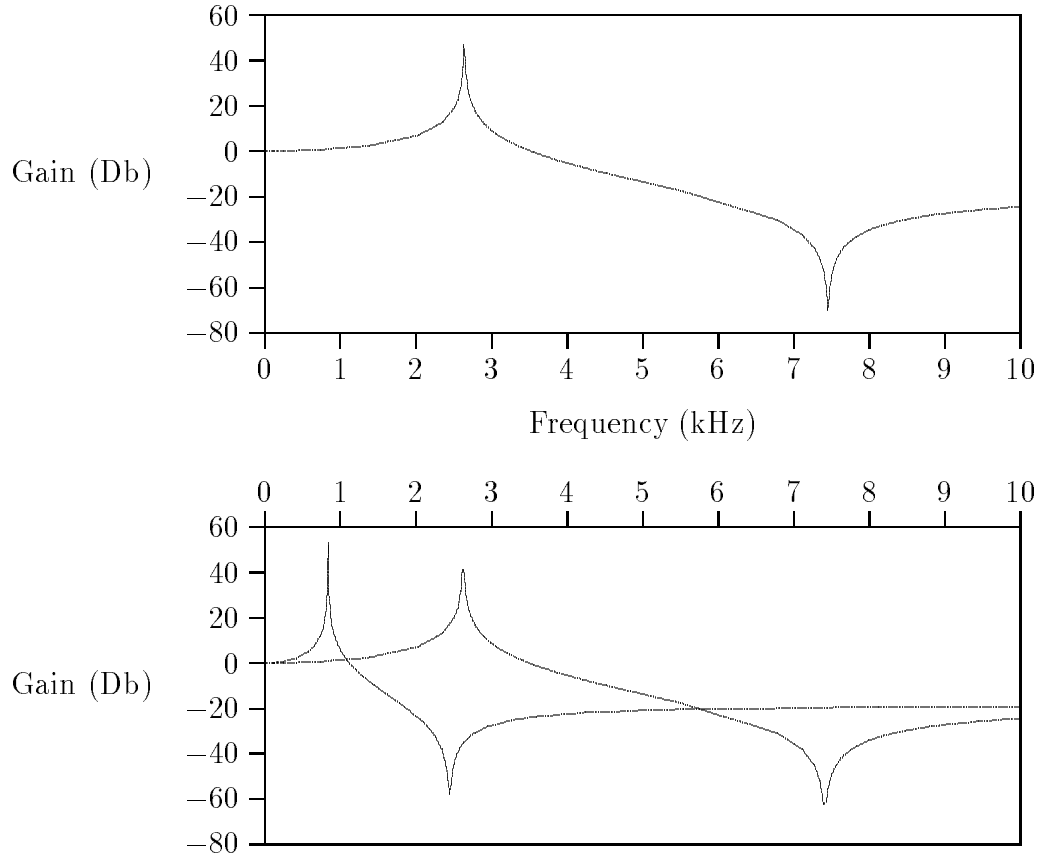


Figure 2: The gain-frequency graph of a normal filter stage (top), and two examples of faults (bottom). Pole and zero shifted to the left are due to a wrong capacitor ($C1 = 100$ $nF$ instead of $10$ $nF$), and cut off peaks are due to a resistor out of tolerances ($R5 = 5760$ $\Omega$ instead of $5693$ $\Omega$).

resistor $R5$ has achieved a value that deviates from the correct one. Let faulty $R5$ value be 5760 $\Omega$ instead of the correct 5693 $\Omega$; assume also that other components have correct nominal values (Figure 2, bottom).

Analogue testing can be regarded as a process of checking whether a given product operates within acceptable margins for critical parameters (GO, NO-GO test). If a specification is not met we try to localize faults. Fault location is motivated either by the cost of rejecting and the possibility of repairing a board containing discrete analogue components, or by identification of a possible cause in order to prevent malfunctioning of the subsequent series of products. In our particular case, imprecise laser trimming process may cause malfunctioning of the whole series of products. Hence, it is neccessary to locate possible faults in the very first few manufactured stages.

The stage production process makes internal probing difficult and potentially destructive, therefore we rather avoid it. Instead, measurements are taken under different testing conditions, and a consistent hypothesis which explains most of the differences between

the expected and measured values is sought after. For diagnosis one can use a standard simulation package. Based on the expert knowledge, a value is assigned to a suspected component, the circuit operation is simulated, and results are compared to the measured values. The process is repeated until the simulated values are close to the measured ones.

On the other hand, we can use the same CLP($\Re$) model for both, simulation and diagnosis. The simulation proceeds under the assumption that all model components are *ok*. For a given frequency $F$, an input voltage *V1* is applied to the model, and the amplitude *Vmax* and *Phase* delay of the output voltage *V2* are computed:

```
simulate( F, Vmax, Phase ) ←
        V1 = c(1,0),
        stage( comps(ok,ok,ok,ok,ok,ok,ok), obs(F,V1,V2) ),
        polar_coord( V2, Vmax, Phase ).
```

Suppose that we measured *Vmax* and *Phase* delay at 7400 and 2800 Hz. The simulation yields the following predicted values which differ from the measured ones:

```
← simulate( 7400, Vmax, Phase ).     % Measured Vmax = 0.00054, Phase = −145
Vmax = 0.00440081, Phase = −3.75498

← simulate( 2800, Vmax, Phase ).     % Measured Vmax = 6.27, Phase = −180
Vmax = 4.8306, Phase = −179.894
```

For diagnosis, the same model is used just the other way around. Input and output voltages, *V1* and *V2*, are given and we are asking for the states of the model components (presumably some will be abnormal) such that the consistency between the input and output is restored. This effectively means that some local constraints, governing the behavior of abnormal model components, are suspended. Here we also make a single fault assumption, i.e., all but one component are *ok*:

```
diagnose( F, Vmax, Phase, Diag ) ←
        V1 = c(1,0),
        complex_coord( Vmax, Phase, V2 ),
        single_fault( Diag ),
        stage( Diag, obs(F,V1,V2) ).
```

For the first measurement, we get the following seven alternative diagnoses:

```
← diagnose( 7400, 0.00054, −145, comps(R1,R2,R3,R4,R5,C1,C2) ).

R1 = ab(−46503.5), R2=ok, R3=ok, R4=ok, R5=ok, C1=ok, C2=ok ;
R2 = ab(700.32), R1=ok, R3=ok, R4=ok, R5=ok, C1=ok, C2=ok ;
R3 = ab(9647.06), R1=ok, R2=ok, R4=ok, R5=ok, C1=ok, C2=ok ;
R4 = ab(10365.8), R1=ok, R2=ok, R3=ok, R5=ok, C1=ok, C2=ok ;
R5 = ab(5759.79), R1=ok, R2=ok, R3=ok, R4=ok, C1=ok, C2=ok ;
C1 = ab(9.6334e-09), R1=ok, R2=ok, R3=ok, R4=ok, R5=ok, C2=ok ;
C2 = ab(9.65935e-09), R1=ok, R2=ok, R3=ok, R4=ok, R5=ok, C1=ok
```

A diagnosis consists of an assignment of states *ok* or *ab(X)* to all the components. In

addition, for an abnormal component, its predicted value $X$ (resistance or capacitance) is computed. The measured output voltage can be accounted for only if the faulty resistor or capacitor actually assumes the predicted value. From the above diagnoses we can immediately rule out *R1* as a candidate of beeing faulty, since a resistor cannot have negative resistance. In order to decide between the remaining alternatives we take another measurement.

The seconds measurement yields the following diagnoses:

$\leftarrow$ *diagnose( 2800, 6.27, −180, comps(R1,R2,R3,R4,R5,C1,C2) ).*

*R1 = ab(5296.07), R2=ok, R3=ok, R4=ok, R5=ok, C1=ok, C2=ok ;*
*R2 = ab(539.949), R1=ok, R3=ok, R4=ok, R5=ok, C1=ok, C2=ok ;*
*R3 = ab(9658.83), R1=ok, R2=ok, R4=ok, R5=ok, C1=ok, C2=ok ;*
*R4 = ab(10353.3), R1=ok, R2=ok, R3=ok, R5=ok, C1=ok, C2=ok ;*
*R5 = ab(5759.62), R1=ok, R2=ok, R3=ok, R4=ok, C1=ok, C2=ok ;*
*C1 = ab(9.65867e-09), R1=ok, R2=ok, R3=ok, R4=ok, R5=ok, C2=ok ;*
*C2 = ab(9.65894e-09), R1=ok, R2=ok, R3=ok, R4=ok, R5=ok, C1=ok*

Now we make an assumption that faults are non-intermittent. A faulty component assumes some value different than nominal, but this value does not change during testing. This means that predicted values should remain stable across several measurements. Consequently, we can rule out *R2* as a possible diagnosis since its predicted values considerably vary.

| Faulty component | Relative standard deviation (%) | Predicted (mean) value ($\Omega$,$nF$) | Nominal value ($\Omega$,$nF$) |
|---|---|---|---|
| R5 | 0.002 | 5760 | 5693 |
| C2 | 0.003 | 9.66 | 10 |
| R4 | 0.085 | 10360 | 10000 |
| R3 | 0.086 | 9653 | 10000 |
| C1 | 0.19 | 9.65 | 10 |
| R2 | 18 | 620 | 727 |
| R1 | 180 | −20600 | 5513 |

Table 1: Relative ordering of components in decreasing likelihood of faults after two measurements.

Due to the imprecision of measurements, even the predicted value of the component known to be faulty slightly varies. We assume the standard Gaussian distribution of predicted values of individual components, and calculate the mean and standard deviation across several measurements. Table 1 gives a list of faulty components, ranked by the relative standard deviation of their predicted values. *R5* seems the most probable cause of malfunctioning, but additional test measurements have to be taken to single it out. *R2*

and *R1* can already be eliminated as possible causes with a high degree of confidence.

In general, predicted values of faulty components are not constants but intervals. We have to take into account the accuracy of measurements and tolerances of fault-free components which leads to internal voltages and currents beeing within some interval ranges. Their tightest bounds can be extracted only at the end of the model simulation, and from them the predicted interval values of faulty components can be computed. This increases the computational demands of the CLP($\Re$) model interpreter, but the basic diagnostic strategy remains the same.

# 5 Conclusion

In the paper we outlined first experiences with the use of CLP($\Re$) for analogue circuits testing. For a non-trivial circuit we re-created simulation results as produced by a dedicated simulation package. The advantage of CLP($\Re$) is that it is a general purpose programming language, and that it tightly integrates numeric and symbolic computation. Models of circuits can be specified concisely, and used for both, simulation and diagnosis.

For a designer of analogue circuits, closing the gap between the simulation model and the reality is important. More realistic models may require more sophisticated numerical processing that current implementations of CLP($\Re$), restricted to linear systems, provide. However, our implementation of CLP($\Re$) makes extensions easier, and allows for the integration of several specialized solvers within the same framework. We envision CLP($\Re$) as a potential basis for software tools which support rapid model specification, experimentation in the design process, and testing during the early manufacturing phases.

# Acknowledgements

The first two authors are supported by the Austrian Federal Ministry of Science and Research. They wish to thank Robert Trappl for making some of this work possible. The last two authors acknowledge the support of the Slovene Research Council.

# References

Bandler, J.W., Salama, A.E. (1985). Fault diagnosis of analog circuits. *Proc. IEEE 73 (8)*, pp. 1279-1826.

Carlsson, M., Widen, J. (1990). Sicstus Prolog user's manual, SICS/R-88/88007C, Swedish Institute of Computer Science, Kista, Sweden.

Cohen, J. (1990). Constraint logic programming languages. *Communications of the ACM 33 (7)*, pp. 52-68.

Dague, P., Deves, P., Luciani, P., Taillibert, P. (1990). Analog systems diagnosis. *Proc. 9th ECAI*, pp. 173-178, Stockholm.

Davis, R. (1984). Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence 24*, pp. 347-410.

de Kleer, J., Williams, B.C. (1987). Diagnosing multiple faults. *Artificial Intelligence 32*, pp. 97-130.

Duhamel, P., Rault, J.C. (1979). Automatic test generation techniques for analog circuits and systems: a review. *IEEE Trans. on Circuits and Systems CAS-26 (7)*, pp. 411-440.

Genesereth, M.R. (1984). The use of design descriptions in automated diagnosis. *Artificial Intelligence 24*, pp. 411-436.

Heintze, N., Jaffar, J., Michaylov, S., Stuckey, P., Yap, R. (1987a). The CLP($\Re$) programmer's manual. Dept. of Computer Science, Monash University, Australia.

Heintze, N., Michaylov, S., Stuckey, P. (1987b). CLP($\Re$) and some electrical engineering problems. *Proc. 4th Intl. Conference on Logic Programming*, pp. 675-703, Melbourne, Australia, The MIT Press.

Holzbaur, C. (1990). Specification of constraint based inference mechanisms through extended unification. Ph.D. Thesis, Vienna University of Technology, Austria.

Jaffar, J. (1990). CLP($\Re$) version 1.0 reference manual. IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY.

Jaffar, J., Lassez, J.-L., Mahler, J. (1986). A logic programming language scheme. In D. de Groot, G. Linstrom (eds.), *Logic Programming: Functions, Relations, and Equations*, Prentice-Hall, Englewood Cliffs, NJ.

Kraemer, F.-J. (1989). A decision procedure for Presburger arithmetic with functions and equality. SEKI working paper SWP-89-4, FB Informatik, University of Kaiserslautern, Germany.

McKeon A., Wakeling, A. (1990). Model-based analogue circuit fault diagnosis. *Proc. TEST'90*, pp. 1-14, London.

Mozetic, I., Holzbaur, C. (1991a). Integrating qualitative and numerical models within Constraint Logic Programming. Proc. *1991 Intl. Logic Programming Symposium, ILPS-91*, San Diego, MIT Press.

Mozetic, I., Holzbaur, C. (1991b). Controlling the complexity in model-based diagnosis.

Report TR-91-3, Austrian Research Institute for Artificial Intelligence, Vienna, Austria.

Ohletz, M.J. (1991). Hybrid built-in self test for mixed analogue/digital integrated circuits. *Proc. 2nd European Test Conf. TEST'91*, pp. 307-316, Munich.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence 32*, pp. 57-95.

Shostak, R. (1981). Deciding linear inequalities by computing loop residues. *Journal of the ACM 28 (4)*, pp. 769-779.

Spectrum. Micro-cap III electronic circuit analysis program instruction manual. Spectrum Software, 1021 S. Wolfe Road, Sunnyvale, CA 94086.