Application of Mathematical Constraint Resolution to Decision Support System

Feng-Tyan Lin

The Institute of Building and Planning
National Taiwan University
Taipei, Taiwan

Jie-Yong Juang and *Der-Tsai Lee*

The Department of
Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208-3118

ABSTRACT

In this paper, we describe how a MATHematical COnstraint REsolution (MATHCORE) system can be used to design a better decision support system. MATHCORE not only has the flexibility of expressing mathematical equations within logic programming paradigm in a natural way, but also has an ability to deal with systems of nonlinear equations, regression analysis and optimization problems. While most existing constraint logic programming systems try to devise their own constraint solver and are confined to systems of linear equations and simple nonlinear functions, the MATHCORE removes this limitation by directly taking advantages of well-developed numerical methods available in the mathematical libraries. With MATHCORE, complex decision optimization models can be embedded in a rule-based decision support system. Using this methodology, we demonstrate that the interactions among various economic factors in a housing market can be stated in the program body, while various (optimization) goals of social welfare can be expressed as queries.

1. Introduction

The development of decision support systems has recently attracted a lot of interest. A decision support system can be viewed as being composed of three components: *input, inference* and *output*. The purpose of the inference component is to help assess user's (e.g. decision maker's) preferences. The underlined methodology can be algorithmically precise or heuristic[Bui87]. In other words, the inference component should not only have the ability of quantitatively describing relationships among causes and effects, but also have the capability of employing mathematical expressions which model complicated relationships among various economic factors so that future trends can be quantitatively predicted and alternatives can be evaluated in an objective way. Thus, a programming language which can simultaneously express behavior rules and mathematical models in a natural way is desirable.

Conventional imperative programming languages, *e.g.*, FORTRAN, have been widely used to implement mathematical models in a computing environment. However, systems developed using these languages are relatively rigid and not friendly to use. For example, the relationship between input and output in such systems are predefined. Different directions of data flows (input vs. output) imply different program implementations. Programming controls are tedious and usually demand tremendous software engineering efforts. Lack of logic reasoning capability also makes them difficult to be used in developing complex intelligent systems. On the other hand, logic programming languages are more user-friendly and effective in expressing sophisticated ideas. But, they are inefficient in mathematical computation and do not have the ability to combine mathematical models in a natural way. The development of constraint logic programming is motivated by the necessity of combining mathematical solving and logic reasoning together. However, most existing constraint logic programming systems are restricted to solving simple mathematical models of linear equations due to limited capability of their built-in solvers. They are inadequate for developing systems that involve nonlinear equations, optimization and regression analysis which are common techniques used in decision proceses.

MATHCORE takes a different approach. It effectively recognizes and groups related mathematical constraints together into appropriate mathematical models, and calls efficient numerical subroutines in well-developed mathematical libraries to solve the resulting models. In this way, efficient mathematical libraries can be fully exploited, and nonlinear equations, optimization and regression analysis problems can be conveniently expressed in the MATHCORE programming language.

In this article, we shall demonstrate the distinguished features of the MATHCORE for developing decision support systems by giving an example on the analysis of alternative public housing policies. In the next section, we first briefly introduce the MATHCORE programming language. In Section 3, a simplified structure of public housing market is introduced. Based on this basic structure, an implementation of a decision support system for public housing policy in MATHCORE programming language is described in Section 4. The implementation is composed of five segments including (surveyed) data observation, budget allocation, demand and supply curves fitting, equilibrium condition and

0730-3157/89/0000/0685$01.00 © 1989 IEEE

evaluation criteria, respectively. The body of this program expresses the relationships among various events that may occur in the policy making process, and the goals of social welfare are expressed as query statements, some of which are illustrated in Section 5. Conclusions are drawn in Section 6.

2. MATHCORE Programming Language

MATHCORE has a syntactic structure similar to Prolog. A MATHCORE *program* is a collection of *clauses* which are in the form of "$A : -B_1, B_2, \cdots$." A *query* is in the form of "$? - B_1, B_2, \cdots$." In the syntactic structure, A (called *clause name*) denotes a logic atom, B_i (called *clause body*) is either a logic atom or mathematical constraint. The resolution strategy is adapted from SLD-resolution[Lloy84] except that constraints are collected with appropriate variable substitutions[LLJF89].

Mathematical constraints in MATHCORE programming language can be classified into six categories: *linear, nonlinear, differential equations, optimization objective function, inequality* and *regression equations*. Related constraints which share common variables (abound variables) are grouped together and form a semantic tree. Each node v of a semantic tree contains information such as the category of the constraint system, number of equations, and a list of variables that appear in the constraints stored in the leaves of the subtree rooted at v. After examining the characteristics of constraints, an appropriate numerical subroutine is called for solving the system of constraints when the set of constraints is potentially solvable. Also, some attributes, called *control attributes*, can be accessed and modified by the user to provide necessary information for solving the system. Through this mechanism a *high level control* can be realized. For example, when the system is trying to solve a nonlinear equations, the user can select his or her own preferred package to solve the problem and provide some good guesses of initial values according to his or her professional experience. Examples of MATHCORE programs and constraint processing will be given found in Section 4 of this paper.

Housing market can be modeled by two mathematical curves which represent *demand side* and *supply side* of the market, respectively. Figure 1 depicts an example of such curves. The horizontal axis measures the quantity of houses, while the vertical axis calibrates unit price. A point (h, p) on the demand (resp. supply) curve is interpreted as "If home price per unit is p, then there are h units of houses to be bought (resp. sold)." Normally, a demand curve goes from upper left to lower right which coincides the intuition that the lower the price that a house costs, the higher the demand will be generated from potential buyers. On the other hand, a supply curve goes from lower left to upper right indicating that the higher the price that a house can sell, the more new houses that contractors will be willing to build. The intersection (h^*, p^*) of these two curves represents the *equilibrium point* of the market. In other words, the housing price of this market will eventually reach p^* and the supply will reach h^*, if external conditions remain unchanged.

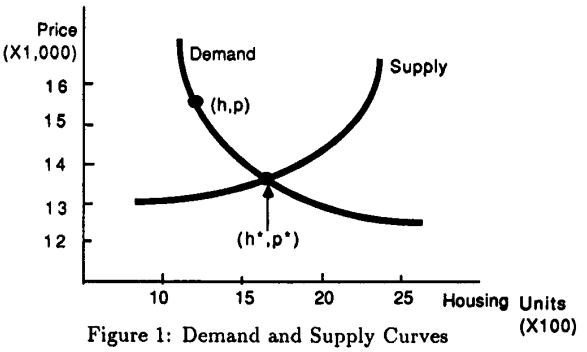

Figure 1: Demand and Supply Curves

Market interaction and public housing policy are two kinds of external forces which will influence a local market. In the following example, we assume that there are four adjacent areas, *area1, area2, area3* and *area4*, in which housing markets dynamically interact with each other. The interactions of these four housing markets can be modeled by integrating them into an overall market. Namely, local demand curves and supply curves are summed up to form an overall demand curve and a supply curve, respectively. The government might establish some policies to pursue its goals of providing "good" social welfare. The government can lever the market via the allocation of budgets. In this example, we assume that the budget might be used in two different ways. The first alternative is to provide financial loan to housing contractors (suppliers) so that they are willing to offer more houses into the market at a reasonable price. Consequently, the supply curve will shift to the right and the new equilibrium point can be achieved. The second alternative is to build or provide more public facilities (such as school, park, road, etc.) so that more potential inhabitants will move into the area and would be willing to pay for a house at a higher price. Consequently, the demand curve will also shift to the right and the new equilibrium point can be achieved with the hope of having more housing units but at a higher price. Note that these two strategies have different effects on the housing market. Although both of them can increase the number of housing units, the former will maintain housing price at the same level while the latter will raise the living standard. The second alternative may generate more revenue through the tax on the increased property value. Furthermore, in the example, we assume that the first strategy will affect the supply curve of the overall market (i.e., all of four local markets at the same time), but the second only influence the demand curves of local markets.

By allocating budget appropriately, the government wishes to pursue the following goals:

- Maximizing the total housing property value (namely, the number of housing units multiplied by unit price). This goal is very important because the total housing property value is the base of leviable taxes on property, a major revenue of the government.
- Recovering the budget. In other words, the expenses, including principal and interests, should be recovered by the property tax, say 10% of the total housing value. This goal assures that the government is self-supporting.
- Balancing population. Imbalanced developments among residential communities may incur extra social problems. For example, phenomena of traffic congestion and over-loaded public facilities may occur in areas with high population density. In contrast, areas with low population density would have low utilization of facilities, which implies a waste of resources.
- Maintaining housing price within a reasonable range.

Figure 2 illustrates an overall relationship among major public affairs from the viewpoint of the government. Each public affair is represented by a subsystem module in the figure. The housing market subsystem (shaded box) can be influenced by the budget allocation, while some of its elements, such as housing price, population distribution and total property value, will influence global economics subsystem, transportation and public facility subsystem and tax regulation subsystem, respectively. Global economics and tax regulation subsystems will in turn influence the public finance subsystem as a feedback. Therefore, goals pursued by the government can be interpreted as some expected (optimal) behaviors of the system.

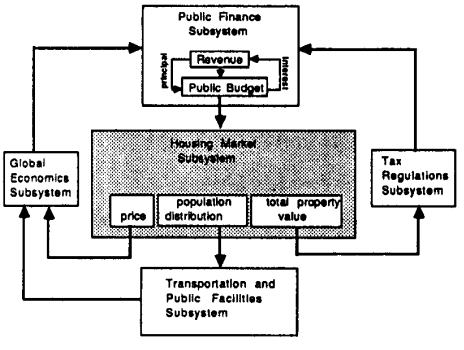

Figure 2: The Overall Public Affairs System

Since the entire system is very complex and need a lengthy program to describe, we are going to focus on the housing market subsystem only in this article. Figure 3 illustrates the internal conceptual structure of the housing market subsystem. As mentioned earlier, there are four areas in the subsystem. Each area is associated with two sets of data, representing demand side and supply side of the housing market, respectively. The data was obtained from surveys conducted on the area. Thus there are eight sets of input data in this example. The government budget is divided into two parts, one for loan to housing contractors and the other for facility development. The loan budget will affect the supply sides of all the four local markets at the same time. The facility budget only affects the demand side of the local market, since each area may be allocated with a different amount. According to the observed data and budget allocation policy, the demand curves and supply curves of four local markets and their overall curves can be estimated by using regression analysis. Consequently, the equilibrium price and housing units of markets can be obtained by finding the intersection points of the corresponding demand and supply curves. Finally, some measurements are made against these predicted results so that achievement of social goals can be evaluated. To do so, we assume the following. The total property value is calculated by multiplying equilibrium price to the number of housing units. The projected revenue from property tax is a function of the total property value. On the other hand, balance of population is measured on the basis of equilibrium housing units in each area. These demand and supply curves are modeled by mathematical equations. Their equilibrium points, total property value, projected revenue and balance of population can be expressed in terms of mathematical constraints in a natural way. In the following sections, we will discuss these mathematical constraints though rather briefly, and show how they can be expressed in the programming language MATHCORE.

4. Implementation in MATHCORE

To implement in MATHCORE the operational structure of public housing policy and market behavior as described above (Figure 3), five segments are composed. These five segments describe observed data, budget allocation, demand and supply curves fitting, equilibrium conditions, and evaluation criteria, respectively. Note that, they only describe relationship among events, whereas the users' objective functions are given as query statements. In doing so, more flexible objective functions can be assessed by the decision support system. The program shown in this article is by no means the best implementation possible using MATHCORE. Its purpose is to show how regression analysis and system of nonlinear equations interact with logic inference. More efficient MATHCORE may be obtained by a more careful design.

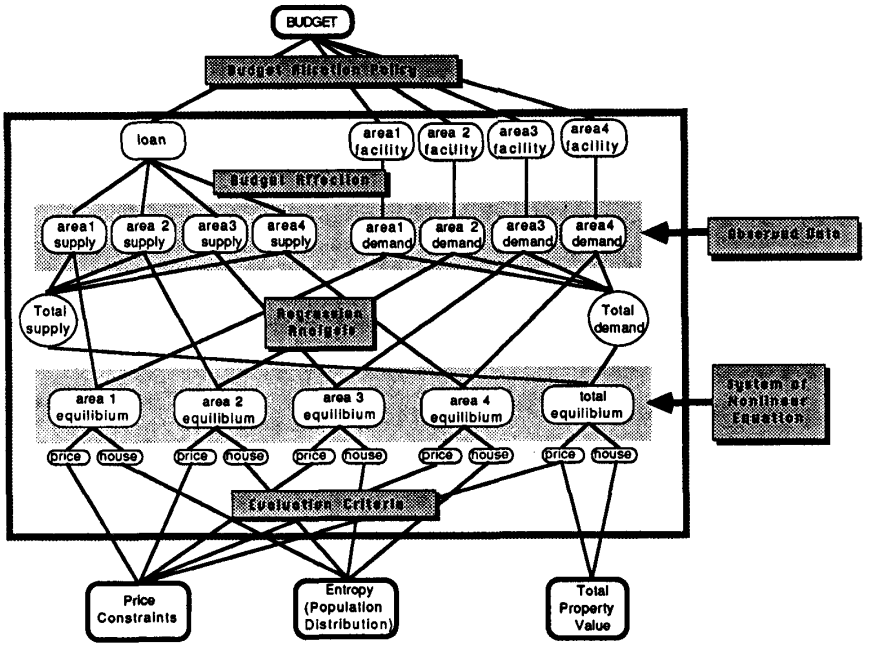

Figure 3: Structure of a Housing Market Subsystem

The first segment contains observed data. They can be expressed as *facts*, i.e., positive unit clauses, in MATHCORE. The formats of facts of demand and supply data have the same types of arguments. For example, `demand(area1,[[dp1, dh1], [dp2, dh2], ···])` denotes the demand side data surveyed from area 1, where `[dp, dh]` is a pair of observed data representing housing demand `dh` at a certain price `dp`. Similarly, supply data of area 1 has the format of `supply(area1, [[sp1,sh1], [sp2,sh2], ···])`. This data segment is also called the *base of domain knowledge* in some expert systems and explicitly separated from the *inference engine*, which corresponds to our other four segments discussed below [Radig 86].

The second segment describes budget allocation. Rule `budget` has three arguments. The first one represents the total amount of budget. The second one is loan budget. The third one is a list of local facility budgets. In the example, there are four elements in the list corresponding to four areas. Presumably, the first argument should be equal to the sum of the second one and elements of the third argument. In particular, rule `budget(T,L,[FT|FT])` recursively defines their relationship where variable `T` denotes the total budget, `F` the loan budget, `FH` the first element of the list of local budget, `FT` the rest elements of the list, `TFT` the sum of elements in `FT`. On the other hand, rule `budget(T,L,[ ])` indicates that all the total budget is allocated for loaning (i.e. the local budget is denoted by an empty list). This rule is also used as a terminal condition when the recursive rule `budget(T,L,[FT|FT])` is executed.

```
budget(T,L,[FH|FT]) :-
        T = L + FH + TFT, L >= 0, FH >= 0,
        budget(TFT,0,FT).
budget(T,L,[ ]) :- T = L, L >= 0.
```

The third segment performs regression analyses of demand and supply curves. The predicate `d_para` denotes 'parameters of demand curve', while `s_para` is that for supply curve. These two analyses are similar except that different source data and fitting models are used. There are many linear and nonlinear mathematical models which can be used to estimate a regression curve[WW79]. In this example, we introduce four of them with two variables, where `X` is the independent one ,`Y` is the dependent one, and `A`, `B` and `C` are coefficients to be estimated in the regression analysis.

- linear: $Y = A + B * X$.
- parabola: $Y = A + B * X + C * X^2$.
- Cobb-Douglas function: $Y = A + X^B$.
- exponential: $Y = A + e^{B*X}$.

Note that the capability of analyzing regression problem is one of distinct features in MATHCORE which cannot be found in other constraint logic programming systems. The mathematical model for fitting a curve is expressed as a constraint in a rule. The predicate `regression` is used to trigger the curve-fitting processing. Due to its semantics, a rule of `regression` should be fully instantiated before an appropriate subroutine can be called to estimate the coefficients for

the desired mathematical model. To illustrate the capability of handling nonlinear equations, we arbitrarily choose to use the Cobb-Douglas function to estimate demand curve and exponential function for supply function.

```
d_para([ ],_).
d_para([AH|AT], [[X,Y]|XYT]) :-
       demand(AH, DL),
       regression(DL, [X,Y], cobb_douglas),
       d_para(AT, XYT).

s_para([ ],_).
s_para([AH|AT], [[U,V]|UVT]) :-
       demand(AH, SL),
       regression(SL, [U,V], exponential),
       s_para(AT, UVT).

regression([[P,H]|T], [A,B], linear) :-
      A + B * H ≈ P,
      regression(T, [A,B], linear).
regression([[P,H,|T], [A,B,C], parabola) :-
      A + B * H + C * H² ≈ P,
      regression(T, [A,B,C], parabola).
regression([[P,H]|T], [A,B], cobb_douglas) :-
      ln(P) ≈ ln(A) + B * ln(H),
      regression(T, [A,B], cobb_douglas).
regression([[P,H|T], [A,B], exponential) :-
      ln(P) ≈ ln(A) + B * H,
      regression(T, [A,B], exponential).
```

In the fourth segment, the equilibrium condition `equi` is defined as the intersection of the demand and supply curves. In terms of mathematical expressions, the equilibrium condition is the solution of a system of two corresponding nonlinear functions. In the body of rule `equi`, parameters of demand and supply functions are calculated and stored in lists `Dpara` and `Spara`. Based on them, two equations are constructed for demand and supply curves by the rule "`formula`". The equations are respectively in the form of $P = \sum_{i=1}^{n} X_i(H+F_i/P)^Y$ and $P = \sum_{i=1}^{n} U_i e^{(H+L/P)V_i}$, where P is the overall housing price, H the number of the overall housing units, L the loan budget, X_i, Y_i, U_i and V_i, parameters, and F_i the facility budget for area i.

```
equi(A, L, F, P, H) :-
       d_para(A, Dpara),
       s_para(A, Spara),
       formula(Dpara, Spara, L, F, PD, PS, P, H),
       P = PD,
       P = PS.


formula([ ], [ ], _, [ ], 0, 0, _, _).
formula([[X,Y]|DT], [[U,V]|ST], L, [F|FT], PD1,
        PS1, P, H) :-
       formula(DT, ST, L, FT, PD2, PS2, P, H),
       PD1 = X * (H + F/P)^Y + PD2,
       PS1 = U * e^(H+L/P)*V + PS2.
```

Most existing constraint logic programming systems solve mainly linear equations, and use delay mechanism to handle nonlinear ones. The strategy of delay mechanism is to wait for nonlinear equations to be reduced to a simple solvable function after substitutions of known results for certain variables. However, there is no clear definition as to when a nonlinear function is considered 'simple'[DSH88, LMY88]. Practically, a simple nonlinear function has the characteristic that only primitive nonlinear functions (e.g., *sin, cos, exp,* $\cdots$) are called for. However, a system of nonlinear equations usually can hardly be reduced to simple. Consequently, solvers in most of existing constraint logic programming systems will quit constraint evaluation or return the system of nonlinear equations intact. On the other hand, MATHCORE takes the strategy of invoking subroutines in mathematical library in which many efficient subroutines have been developed for tackling problems represented in systems of nonlinear equations. Thus, MATHCORE can obtain the solution of the system of nonlinear equations without hopelessly waiting for the mathematical system to become simple.

The fifth segment calculates the population distribution among four local areas. Given an overall housing price and facility budgets, rule `house` calculates the number of local housing units. Then, rule `sum` determines the actually number of total housing units, and rule `ratio` calculates the population ratio of each area. Finally, the value of entropy is used to measure the degree of population distribution. The equation of entropy is $E = -P_i \sum_{i=1}^{n} P_i$, where E is the value of entropy, P_i is the ratio of the population in area i to the overall population.

```
popu_dist(A, F, P, E) :-
       house(A, P, F, HL),
       sum(HL, HT),
       ratio(HL, HT, HW),
       entropy(HW, E).

house([ ], _, [ ], [ ]).
house([AT|AT], P, [F|FT], [HH|HT]) :-
       d_para(AH, [X,Y]),
       P = X * (HH + F/P)^Y,
       house(AT, P, FT, HT).

sum([ ], 0).
sum([H|T], N) :-
       sum(T, S),
       N = H + S.

ratio([ ], _, [ ]).
ratio([H|T], Total, [HWH|HWT]) :-
       HWH = H / Total,
       ratio(T, Total, HWT).

entropy([ ], 0).
entropy([LH|LT], E) :-
       entropy(LT, E1),
       E = - LH * ln(LH) + E1.
```

5. Examples and Queries

As mentioned above, the program body of the decision support system describes relationships among essential factors within a housing market, while the evaluation of different goals that the decision makers want to achieve can be independently queried in the goal statement. To illustrate how the decision support system works, an example and its possible queries are presented in this section. A collection of observed data of demand and supply for the example are shown in Table 1. The fitted demand and supply curves of each area and overall market based on these data are illustrated in Figure 4.

Price	Housing Units (× 100)							
	Demand				Supply			
(× 1000)	area1	area2	area3	area4	area1	area2	area3	area4
13.0	7.2	-	8.0	-	-	0.4	2.6	0.4
13.5	5.8	10.0	5.6	8.6	0.4	3.0	3.6	4.0
14.0	5.0	7.0	3.6	5.0	3.2	5.4	4.4	5.4
14.5	4.4	5.4	2.6	2.6	5.0	7.2	5.0	6.2
15.0	4.0	4.2	1.8	1.2	6.4	8.4	5.8	7.0
15.5	3.6	3.2	1.4	0.6	7.8	9.4	6.6	7.4
16.0	3.2	2.5	1.0	-	8.6	-	7.2	7.8
16.5	2.8	2.0	0.8	-	9.0	-	7.8	8.2
17.0	2.5	1.6	0.8	-	10.0	-	8.4	8.4
17.5	2.2	1.2	-	-	-	-	9.0	8.6

Table 1 Demand and Supply Data Observed from Surveys.

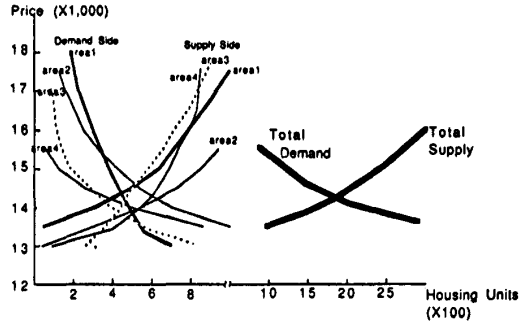

Figure 4: Fitted Demand and Supply Curves

Based on these data and curves, two typical queries are illustrated below. The first query calculates overall equilibrium and population distribution in the case of having no budget allocation involved. From the viewpoint of computation, regression analysis of demand and supply curves and finding the intersection of these two nonlinear curves are the new features which other constraint logic programming systems do not provide.

1. What are the equilibrium price, the number housing units and population distribution index (entropy) if the market is not influenced by any any budget allocation?

```
?- equi([area1,area2,area3,area4], 0,
[0,0,0,0], Price, House),
popu_dist([area1,area2,area3,area4], [0,0,0,0],
Price, Entropy).
```

Answer ==> `Price` = 14,160, `House` = 190, `Entropy` = 1.981

Figure 5 illustrates the diagram of the logic derivations, constraint solving and data flow when the query of example 1 is executed in the MATHCORE. Bold lines indicate derivation steps, while dotted lines show the directions of data flow. Since the complete steps are too tedious, some details are omitted when they are understandable. The query consists of two parts leading by predicates `equi` and `popu_dist` The `equi` part in turn invokes three subgoals and two constraints. Among them, subgoals `d_para` and `s_para` match observed data `demand` and `supply`, then call for regression analyses by fitting these two sets of data against Cobb-Douglas and exponential functions, respectively. All coefficients, denoted by $[x_i, y_i]$ and $[u_i, v_i]$, are estimated and propagated to `formula`. Rule `formula` constructs predicted overall demand and supply functions. As a result, equilibrium housing price and units can be obtained by solving the system of four nonlinear equations.

Propagating the equilibrium price from rule `formula`, rule `popu_dist` calculates the degree of population distribution. Subgoal `d_para` in rule `house` is recalled to obtain demand parameters, then local housing units of each local area are calculated by solving demand function. The local housing units are summed up by `sum`. Theoretically, the total number of housing units should be equal to the overall equilibrium housing units. However, we omit this checking to simplify the discussion of the program. Subgoal `ratio` calculates the population ratio of each area. Finally, the function of entropy is called to obtain the degree of population distribution.

The second query calculates local equilibriums affected by budget allocation. The result can be compared to that of query 1 so that the budget effect can be realized by the decision maker.

2. Assume a loan budget of $500,000, facility budgets for area1, 2, 3 and 4 are $200000, 250000, 180000, 230000, respectively. What are the equilibrium price, and the number of housing units in each area?

```
?- equi([area1, area2, area3, area4], 500000,
[200000, 250000, 280000, 230000], Price,
House), house([area1, area2, area3, area4],
Price, [200000, 250000, 280000, 2230000],
[House1, House2, House3, House4]).
```

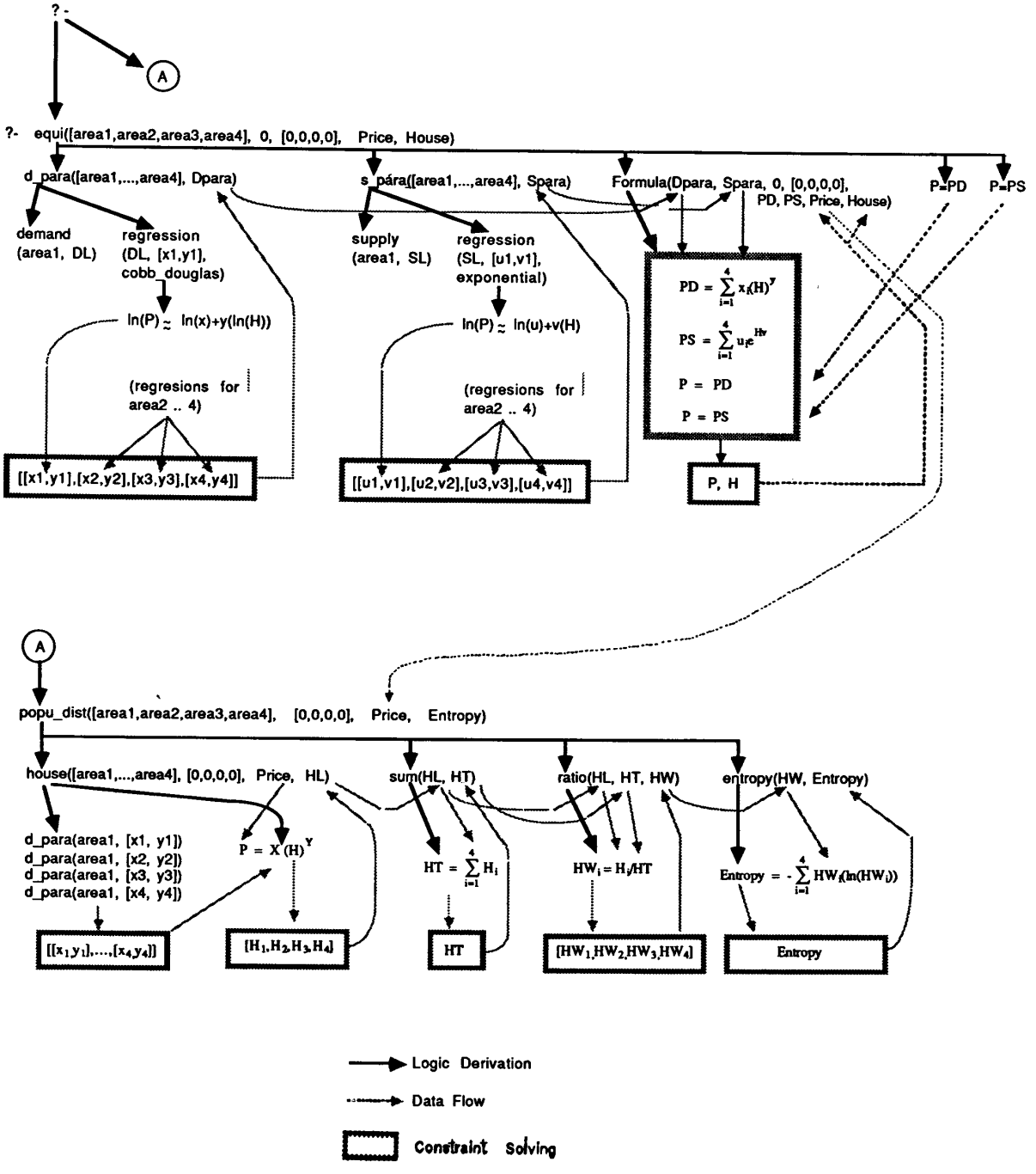

Figure 5: Logic Derivations, Constraint Solving and Data Flow of Example 1

6. Conclusion

A powerful decision support system requires the capability of combining sophisticated mathematical models within the framework of logic programming. Due to the distinguished features of MATHCORE, regression analysis, systems of linear or nonlinear equations, optimization problems and other mathematical constraints can be naturally expressed in a MATHCORE program. While most existing constraint logic programming techniques are confined within the territory of solving linear equations and simple nonlinear functions, MATHCORE goes beyond these limitations by taking advantages of existing well-developed mathematical libraries. In this paper, the use of MATHCORE for implementing a simple decision support system on public housing policy is demonstrated. Through this demonstration, we show how more sophisticated decision support systems can efficiently be created using the proposed MATHCORE constraint logic programming system.

Acknowledgement

This research was supported in part by the National Science Foundation under grant IRI-8709072 and by the Ministry of Education, R.O.C.

REFERENCES

[Bui87] T. X. Bui, "Co-op: A Group Decision Support System for Cooperative Multiple Criteria Group Decision Making", *Lecture Notes in Computer Science*, Springer-Verlag, NY, 1987.

[DSH88] M. Dincbas, H. Simonis and P. Van Hentenryck, "Solving a Cutting-Stock Problem in Constraint Logic Programming", in *Logic Programming, Proc. of The 5th Int. Conf. and Symp.* Ed. R. A. Kowalski and K. Bowen, MIT press, 1988.

[LLJF89] F. T. Lin, D. T. Lee, J. Y. Juang, and E. H. Freeman, "Issues on Solving Many-mathematical Models for Constraint Logic Programming", the Annual Conference of the Rocky Mountain Society for Artificial Intelligence (RMSAI), June, 1989.

[Lloy84] J. W. Lloyd, "Foundations of Logic Programming", Springer-Verlag, NY, 1984.

[LMY87] C. Lassez, K. McAloon, and R. Yap, "Constraint Logic programming and Option Trading", *Fourth IEEE Symp. on Logic Programming*, San Francisco, Aug. 1987.

[Radig86] B. Radig, "Design and Applications of Expert Systems", in *Artificial Intelligence and Man-machine Systems*, Ed. H. Winter, *Lecture Notes in Control and Information Sciences*, Springer-Verlag, NY, 1986.

[WW79] R. J. Wonnacott and T. H. Wonnacott, "Econometrics", 2nd Ed.