ソフトウェア工学 126－4
（2000．3．9）

制約処理機能を持つスプレッドシート

安達 由洋, 須田 祐介
東洋大学

概要

エンドユーザ・プログラミング手法として最も広く使用されているスプレッドシート・パラダイムに制約式の処理機能を付加して機能を拡張する研究を行った．そして，有限領域制約式および線形制約式の制約解消機能を組み込んだスプレッドシート・プロトタイプ Intellisheet を実現した．従来のスプレッドシート・パラダイムでは，セルに算術式を入力すると他のセルの値を参照しながらその算術式を自動的に計算して結果を対応するセルに表示する．Intellisheet は，これらの機能とともに，セルの値に対する有限領域制約式あるいは線形制約式を入力すると自動的にその制約式を満足する解を探索し対応するセルに表示する機能を持っている．制約処理機能の付加により，スプレッドシート・パラダイムをスケジューリング問題，計画問題，あるいは工学的設計でのパラメータ調整問題などを含むより広い分野に応用できるようになる

A spreadsheet paradigm equipped with constraint solvers

Yoshihiro ADACHI, Yuusuke SUDA
Toyo University

Abstract

We describe our research in which the solvers of finite domain constraints and linear constraints are incorporated into the spreadsheet paradigm. The traditional spreadsheet paradigm is based on entering data and arithmetic expressions into individual cells, evaluating the expressions, and then displaying the resulting values on the corresponding cells; our Intellisheet system allows, furthermore, expressions of finite domain constraints and linear constraints to be entered to individual cells that are automatically solved to display the resulting solutions on the corresponding cells. By this incorporation, the spreadsheet paradigm can be used for solving declaratively a wider class of problems including linear optimization and discrete problems.

1 はじめに

スプレッドシート・パラダイムはエンドユーザのためのプログラミング手法としておそらく最も広く使用されているものである．スプレッドシート・パラダイムに基づくシステムとして，市販されているスプレッドシート・システムだけでなく，ビジュアル操作の特徴を生かしてパラダイムを拡張しようとする多くの研究システムが開発されている [1, 2, 3, 4, 5].

スプレッドシート・パラダイムではセルが格子状に配置された2次元の表を用いる．ユーザはデータと関数（算術式）を個々のセルに GUI を用いて入力していく．関数は他のセルの値を入力として用いることができる．データと関数が入力されると，システムが他のセルの値を参照しながら入力された算術式を自動的に計算してその結果を対応するセルに表示する．しかしながら，従来のスプレッドシート・パラダイムではデータと算術式のみが各セルに入力することが許されているだけなので処理能力は限られている（最近のスプレッドシート・パッケージの多くはユーザがマクロを記述することを可能にしているが）．

制約処理（制約伝播）は人工知能の分野において開発された制約充足問題の解決技法である．生成・検査による制約充足問題の解決パラダイムでは，変数の組に値を選んだ後制約を満たしているか否かを検査する．制約処理では，変数の値を選ぶと制約に基づいて伝播させることにより他の変数の値のとる範囲を制限して探索空間を制限する．これにより，制約処理技法は効率よく探索問題を解き，制約を充足する解を求める．制約処理をコンピュータ上で実行するための制約ソルバー（制約処理システム）が数多く開発され [9, 10, 12, 11]，これらのシステムを用いた応用システムの研究も多数報告されている [6]．制約充足問題に制約ソルバーを利用すると，制約 (式) の組を宣言的に表現するだけで制約ソルバーが自動的に解を探索して問題解決できるようになる．

本稿では，スプレッドシート・パラダイムに制約式の処理機能を付加して機能を拡張する研究について報告する．本研究では，市販の有限領域制約式および線形制約式に対する制約ソルバーを組み込んだスプレッドシート・プロトタイプ Intellisheet を実現した．従来のスプレッドシート・パラダイムでは，セルに算術式を入力すると他のセルの値を参照しながらその算術式を自動的に計算して結果を対応するセルに表示する．Intellisheet は，従来のスプレッドシート・パラダイムの機能に加えて，セルの値に対する有限領域制約式あるいは線形制約式を入力すると自動的にその制約式を満足する解を探索し対応するセルに表示する機能を持っている．

スプレッドシート・パラダイムのもとで制約処理を扱う研究がいくつか報告されている．例えば，Shvetsov 等は制約スプレッドシートに基づいて財務計画問題を解く応用システムを開発した [8]．Renschler は移動電話ネットワークの受信機を配置する Configuration Spreadsheet を報告している [7]．これら特定用途システムの開発研究に対して，Gupta 等 [13] は有限領域の制約式を用いた問題解決プログラムを対話的に開発するためのグラフィカル・スプレッドシート・インタフェースを開発している．しかしながら，従来のスプレッドシート・パラダイムに制約処理機能を付加統合して機能を拡張しようとする研究は見当たらない．

本研究は，有限領域制約式の解を線形制約式や関数の入力に用いることができるなど，従来のスプレッドシート・パラダイムと制約処理機能を統合して機能を真に拡張しようと目指したものである．この機能拡張により，スプレッドシート・パラダイムをスケジューリング問題，計画問題，あるいは工学的設計でのパラメータ調整問題などを含むより広い分野に応用できるようになる．

2 制約処理

制約処理は人工知能の分野において開発された制約充足問題の解決技法で，制約伝播により効率よく探索問題の解を求める手法である．制約充足問題に制約ソルバーを利用すると，制約 (式) の組を宣言的に表現するだけで制約ソルバーが自動的に解を探索して問題解決できるようになる．

本研究では，市販の有限領域制約式および線形制約式に対する制約ソルバー [12] を利用して制約処理機能を持つスプレッドシートのプロトタイプを実現している．使用した制約ソルバーは，宣言的プログラミング言語 Prolog のインタプリタの機能を拡張するパッケージとして提供されている．したがって，Intellisheet システムも Prolog 言語で記述している．

以下に，有限領域制約および線形制約の制約式記述に用いることのできる表現について説明する．

2.1 有限領域制約

有限領域制約は，変数が有限領域の整数に値をとる制約であり，離散組み合わせ問題を解くために用いることができる．有限領域制約で用いることのできる制約表現の一部を以下にあげる．

領域変数：

Variable in *Min*:*Max*
　変数が値としてとることのできる領域を区間として与える．

Variable in [V_1, V_2, V_3, ... V_N]
　変数が値としてとることのできる領域を明示的に値のリストで与える．

Variable in *Min*..*Max*
　変数が値としてとることのできる領域を系列として与える．

有限領域の算術制約：
有限領域の算術制約の構文規則は次のようである：

$$\text{算術制約} ::= \text{領域項} \left\{ \begin{array}{l} ?= \\ ?\backslash = \\ ?< \\ ?=< \\ ?> \\ ?>= \end{array} \right\} \text{領域項}$$

$$\text{領域項} ::= \left\{ \begin{array}{l} \text{変数} \\ \text{領域整数} \\ +\text{領域項} \\ -\text{領域項} \\ \text{領域項} \left\{ \begin{array}{c} + \\ - \\ \star \end{array} \right\} \text{領域項} \\ (\text{領域項}) \end{array} \right\}$$

記号制約：

all_distinct(?*List*)
リスト *List* の要素の任意の対の値が異なる．

atleast(+*Limit*,?*List*,+*Value*)
リスト *List* の少なくとも *Limit* 個の要素が同じ値 *Value* を持つ．

atmost(+*Limit*,?*List*,+*Value*)
リスト *List* の高々 *Limit* 個の要素が同じ値 *Value* を持つ．

最適化：

minimize_bb(+*Goal*,?*Optimum*)
分岐限定法を用いて，+*Goal* の値が最小値 ?*Optimum* となるように探索する．

minimize_maximum(+*Goal*,?*List*)
Goal を満たすリスト *List* の要素の最大値を最小化する．

2.2 線形制約

線形制約は，big integer （約 300,000 桁以下の 10 進数）と有理数からなる線形領域上に値をとる変数に対する制約条件である．線形制約ソルバーは線形領域上の最適化問題に対する解を自動的に探索する．線形領域制約で用いることのできる制約表現の一部を以下にあげる．

線形領域の算術制約：
線形領域の算術制約の構文規則は以下のようである：

$$\text{線形制約} ::= \text{線形項} \left\{ \begin{array}{l} ?= \\ ?\backslash = \\ ?< \\ ?=< \\ ?> \\ ?>= \end{array} \right\} \text{線形項}$$

$$\text{線形項} ::= \left\{ \begin{array}{l} \text{変数} \\ \text{有理制約} \\ +\text{線形項} \\ -\text{線形項} \\ \text{線形項} \left\{ \begin{array}{c} + \\ - \\ \star \end{array} \right\} \text{線形項} \\ \text{線形項} \left\{ \begin{array}{c} / \\ rdiv \end{array} \right\} \text{有理制約} \\ (\text{線形項}) \end{array} \right\}$$

$$\text{有理制約} ::= \left\{ \begin{array}{l} \text{整数} \\ \text{有理数} \\ +\text{有理制約} \\ -\text{有理制約} \\ \text{有理制約} \left\{ \begin{array}{c} + \\ - \\ \star \\ / \\ rdiv \end{array} \right\} \text{有理制約} \\ (\text{有理制約}) \end{array} \right\}$$

リストの要素に対する制約：

all_different(+*List*)
リスト *List* のすべての要素が異なる有理数である．

all_negative(+*List*)
リスト *List* のすべての要素が 0 または負である．

all_positive(+*List*)
リスト *List* のすべての要素が 0 または正である．

最適化：

linear_maximize(?*LinearTerm*,*Maximum*)
与えられた線形項の最大値を探索する．

linear_minimize(?*LinearTerm*,*Minimum*)
与えられた線形項の最小値を探索する．

3 Intellisheet システム

本研究で実現したした Intellisheet のユーザ・インタフェース画面を図 1 に示す．伝統的スプレッドシート・パラダイムにおける基本機能である算術計算は，Intellisheet においても同様に実行することができる．

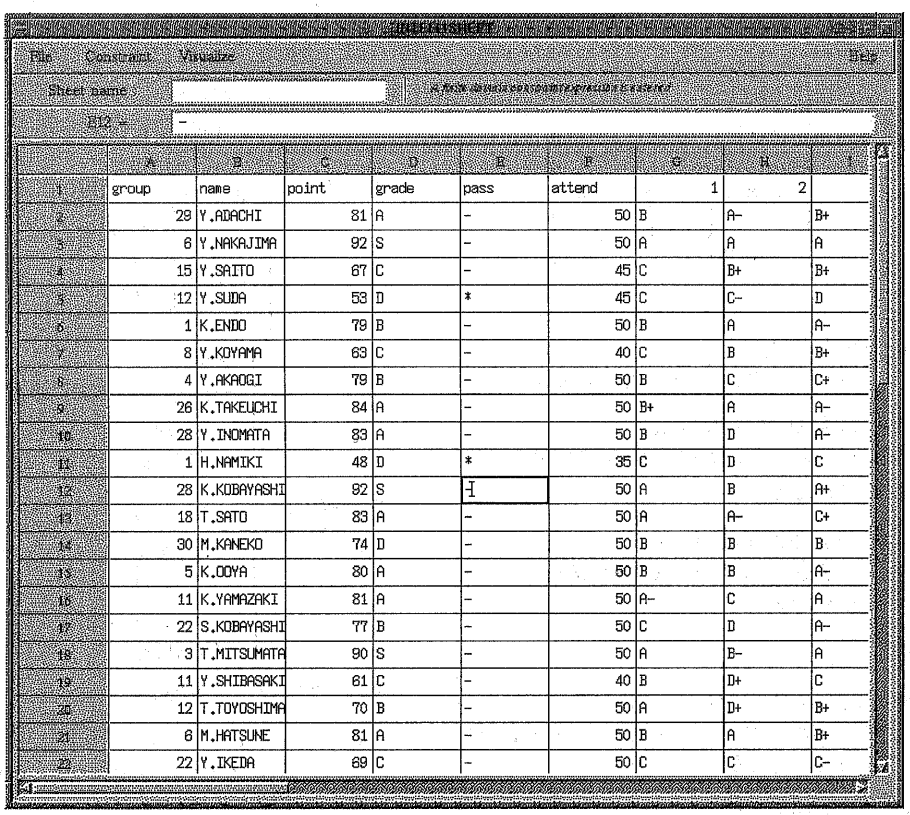

図 1: Intellisheet のユーザインタフェース画面

3.1 3×3 パズル

有限領域制約を用いた問題解決の例として 3×3 パズルを示す．このパズルは 3×3 のマスを考え，それぞれのマスには 1～9 の数字が 1 つずつ入るとする．このとき，全ての縦，横，斜めの列の 3 つの数字の合計が 15 であるという制約条件のもとで各マスに入る数字を求める．

このとき，この 3×3 のマスをスプレッドシートで A1～C3 とすれば，3×3 パズルを解く制約式の集合は次のようになる．

```
[A1,A2,A3,B1,B2,B3,C1,C2,C3] in 0:9,
all_distinct([A1,A2,A3,B1,B2,B3,C1,C2,C3]),
A1 + A2 + A3 ?= 15,
B1 + B2 + B3 ?= 15,
C1 + C2 + C3 ?= 15,
A1 + B1 + C1 ?= 15,
A2 + B2 + C2 ?= 15,
A3 + B3 + C3 ?= 15,
A1 + B2 + C3 ?= 15,
A3 + B2 + C1 ?= 15
```

この有限領域制約式を制約ソルバーで解くことにより，図 2に示す解が得られる．

3.2 覆面算

覆面算の例として有名な問題に，アルファベットを用いた

$$\mathrm{SEND} + \mathrm{MORE} = \mathrm{MONEY}$$

がある．

この式の中で個々のアルファベットは 0～9 までの整数のいずれかをとる変数である．(図 3) このような覆面算も，有限領域制約を用いたスプレッドシートで解くことができる．

H2,H3,H4,H5,H6,H7,H8,H9 のセルは各々 I2,I3, I4,I5,I6,I7,I8,I9 のセルに対応している．したがって，例えば文字 S にあたる数字は I2 に表示される．そして

```
[I2,I3,I4,I5,I6,I7,I8,I9] in 0..9,
[E1,D1,C1,B1] in 0..1,
I9 + 10 * E1 ?= I3 + I5,
I3 + 10 * D1 ?= I8 + I4 + E1,
I4 + 10 * C1 ?= I7 + I3 + D1,
I7 + 10 * B1 ?= I6 + I2 + C1,
I6 ?= B1,
I6 ?>= 1,
I2 ?>= 1,
all_distinct([I2,I3,I4,I5,I6,I7,I8,I9])
```

が実際の計算部分にあたる制約である．

3.3 線形制約による農地管理問題の解決

畑に小麦かとうもろこしを植える問題を考える．小麦ととうもろこしでは 1 ヘクタール当りの収穫量が異なり，また，栽培にかかる時間も異なるとする．このとき，労働時間４０時間，農地１００ヘクタールの場合の最大収穫量を計算する．

```
all_positive([C3,C4]),
C5 $>= C3 + C4,
40 $>= E3 * C3 + E4 * C4,
F5 $= C3 * D3 + C4 * D4,
linear_maximize(F5,MaxOutput)
```

3.4 投資計画問題

会計係が投資計画に基づいて今後５年間に必要となるお金を予測する．最初の２年間は年初に１０，０００ドルのお金がいる．また，その後の３年間は年初に２０，０００ドルのお金がいる．会計係は次の投資を考える：中期債は２年後に各１．００ドルの投資に対して１．４７ドルの報酬がある，長期債は３年後に各１．００ドルの投資に対して１．７８ドルの報酬がある，また短期債は１年後に各１．００ドルの投資に対して１．２０ドルの報酬がある．会社は現在１００，０００ドルのお金があり，会計係は計画終了時にお金の総額を最大にする投資計画を立案したときに得られる総額を求める．

```
all_positive([C3, C4, D4, E4, F4, G11,
    C6, D6, E6, F6, G6,
    C10, D10, E10, F10, G10,
    C7, C8, C9,
    D7, D8, D9,
    E7, E8, E9,
    F7, F8,
    G7
]),
C4 + C5 + C10 $= C3 + C6,
D4 + D5 + D10 $= C4 + D6,
E4 + E5 + E10 $= D4 + E6,
F4 + F5 + F10 $= E4 + F6,
G11 + G5 + G10 $= F4 + G6,
C10 $= C7 + C8 + C9,
C10 + C5 $=< C3,
C6 $= C7 * 0r120/100,
D10 $= D7 + D8 + D9,
D10 + D5 $=< C4,
D6 $= D7 * 0r120/100 + C8 * 0r147/100,
E10 $= E7 + E8 + E9,
E10 + E5 $=< D4,
E6 $= E7 * 0r120/100 + D8 * 0r147/100
+ C9 * 0r178/100,
F10 $= F7 + F8,
F10 + F5 $=< E4,
F6 $= F7 * 0r120/100 + E8 * 0r147/100
+ D9 * 0r178/100,
G10 $= G7,
G10 + G5 $=< F4,
G6 $= G7 * 0r120/100 + F8 * 0r147/100
+ E9 * 0r178/100,
linear_maximize(G11,Max)
```

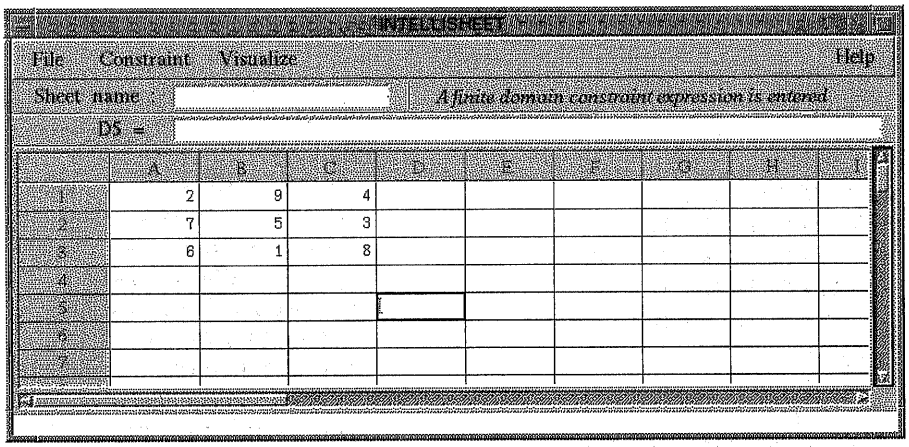

図 2: 3 × 3 パズルの一解答例

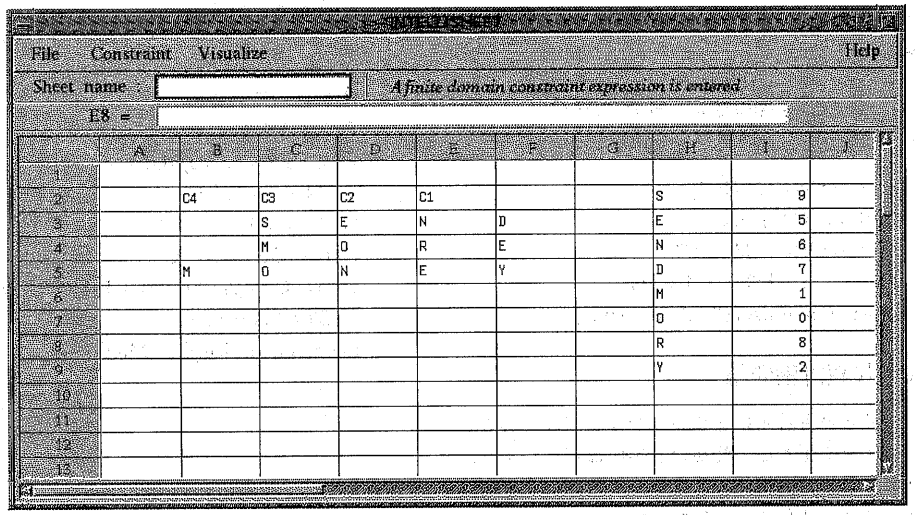

図 3: 覆面算

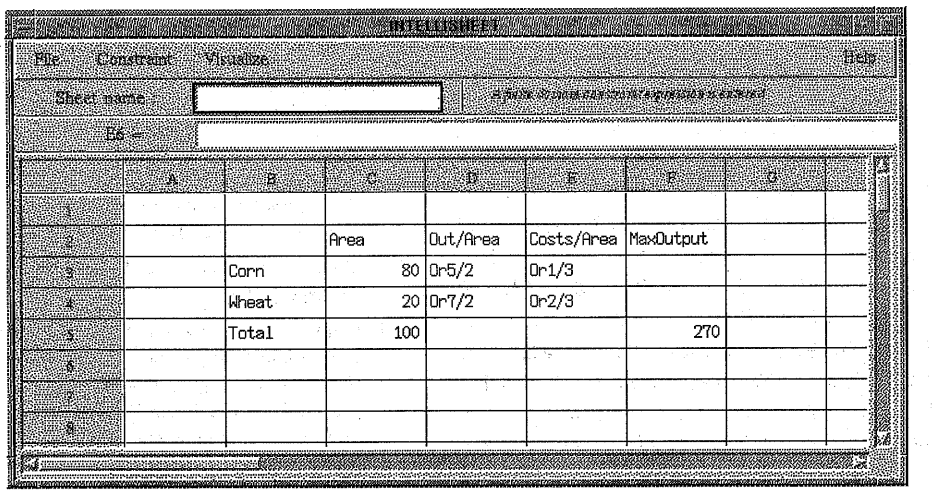

図 4: 農地管理問題

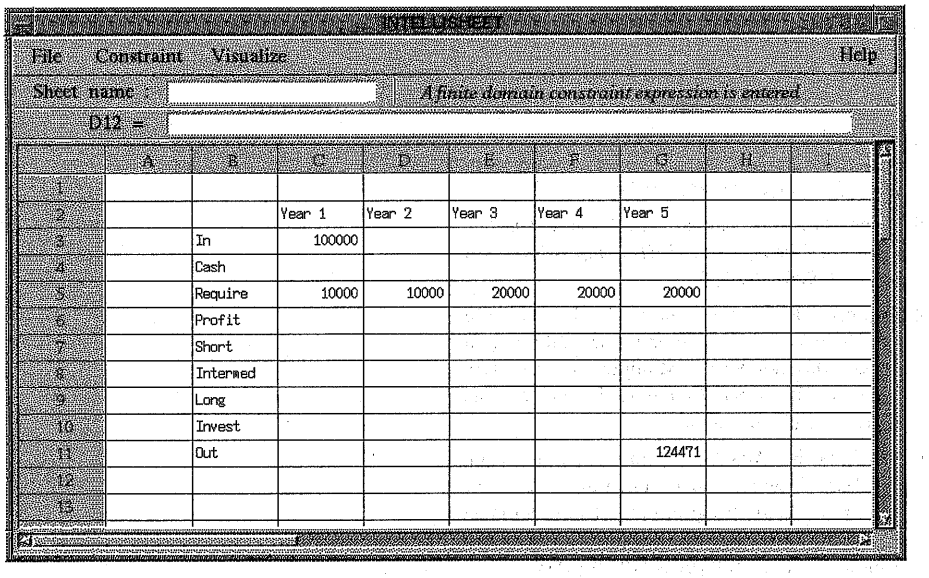

図 5: 投資計画問題

4 まとめ

スプレッドシート・パラダイムに制約式の処理機能を付加して機能を拡張する研究を行い，プロトタイプシステム Intellisheet を実現した．有限領域制約式の解を線形制約式や関数の入力に用いることができるなど，従来のスプレッドシート・パラダイムと制約処理機能を真に統合した機能を持つことが Intellisheet システムの大きな特徴である．今後，本システムをより実用的な規模の問題に対して適用し，その有用性を検証することが必要である．また，有限領域制約式，線形制約式および関数などを定義する際に，問題が複雑になるとセル間の依存関係を把握することが難しくなる．したがって，セル間の依存関係の分かり易い可視化，デバッギング・ツールなど，ユーザのスプレッドシート・プログラミングを支援する研究も重要な課題である．

参考文献

[1] Wilde, N., and Lewis, C., Spreadsheet–Based Interactive Graphics: From Prototype to Tool, *ACM Conf. Human Factors in Computing Systems*, (1990), 153–159.

[2] Myers, B., Graphical Techniques in a Spreadsheet for Specifying User Interfaces, *ACM Conf. Human Factors in Computing Systems*, (1991) 243–249.

[3] Yoder, A., and Cohn, D., Real Spreadsheets for Real Programmars, *Proc. IEEE Int. Conf. Computer Languages*, (1994), 20–30.

[4] Wang, G., and Ambler, A., Solving Display–Based Problems, *Proc. IEEE Symp. Visual Languages* (1996) 122-129.

[5] Burnett, M., and Gottfried, H., Graphical Definitions: Expanding Spreadsheet Languages through Direct Manipulation and Gestures, *ACM Trans. Computer–Human Interactions*, 5(1) (1998), 1-33.

[6] Tsuchida, K., Adachi, Y., Imaki, T., and Yaku, T., Tree Drawing Using Constraint Logic Programming, Logic Programming, Proc. the Fourteenth International Conference on Logic Programming, The MIT Press (1997) p.414.

[7] Rencshler, M., Configuration Spreadsheet for Interactive Constraint Problem Solving, Proc. Practical Applications of Constraint Tech., (1998).

[8] Shvetsov, I., Kornienko, V., and Preis, S., Interval Spreadsheet for problems of financial Planning, Proc. PACT97, (1997) 373–385.

[9] Van Hentenryck, P., Constraint Satisfaction in Logic Programming, The MIT Press (1989).

[10] Meier, M., ECLiPSe User's Manual, IC-PARC Tech. Rep., (1997).

[11] Constraints research group at University of Washington, UW Constraint-Based Systems, http://www.cs.washington.edu/research/constraints/index.html, (2000).

[12] Siemens AG Austria, IF/Prolog V5.1 Constraint Package (1998).

[13] Gupta, G., and Akhter, S. F., Knowledgesheet: A Graphical Spreadsheet Interface for Interactively Developing A Class of Constraint Programs, Proc. the Tenth Workshop on Logic Programming Environments, (1999) 65-79.