TM-0964

制約論理プログラミングシステム CAL第1.4版

相場　亮, 坂井　公, 佐藤　洋祐,
毛受　哲, 川越　恭二（日電）

October, 1990

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

制約論理プログラミングシステム ＣＡＬ

第 1.4 版

(財) 新世代コンピュータ技術開発機構 (ICOT)

内　　容

1. CAL1.4 版のインストール手順

1. CAL 操作手引書 第 1.4 版

1. CAL デバッガ操作手引書 第 1.4 版

1. CAL 言語仕様書 第 1.4 版

1. 集合制約評価系利用手引書 第 1.4 版

1. 線形制約評価系利用手引書 第 1.4 版

1. CAL 処理系 プログラムリスト 第 1.4 版

1. CAL デバッガ プログラムリスト 第 1.4 版

CAL 1. 4版のインストール手順

平成2年9月13日

1. CAL システムを入れるディレクトリを作り、CAL システムのファイルをすべて転送する。
2. CAL デバッガ用のディレクトリを作り、CAL デバッガのファイルをすべて転送する。
3. システムの“cont_inc.esp”とデバッガの“cont_inc.esp”の中のディレクトリパス名を自分用に修正する。
4. CAL システム用のパッケージ“contrainte”を作る。
5. CAL ユーザープログラム用のパッケージ“cal_user”を作り、“contrainte”を継承させる。
6. パッケージ“contrainte”においてシステムの“cont_inc.esp”をカタログし、セーブする。
7. パッケージ“contrainte”においてデバッガの“cont_inc.esp”をカタログし、セーブする。
8. “login.com”のなかの“item_list”に“contrainte‡‡cal_start”と“contrainte‡‡deb_start”を追加する。
9. “cal.init”, “cal_debugger.init”をホームディレクトリにコピーし、中身を自分用にカスタマイズする。詳しくは CAL1.4 版操作手引書を参照のこと。
10. 13個のクラス ‡alg, ‡b_buch, ‡bool, ‡buch, ‡cal, ‡cal_deb,‡cal_debugger, ‡cal_debugger_io, ‡long, ‡pre, ‡setgb, ‡smplx, ‡solver を外部宣言する。

ＣＡＬ操作手引書　第1.4版

昭和63年6月9日
平成元年3月22日
平成元年7月27日
平成2年5月24日
平成2年6月25日

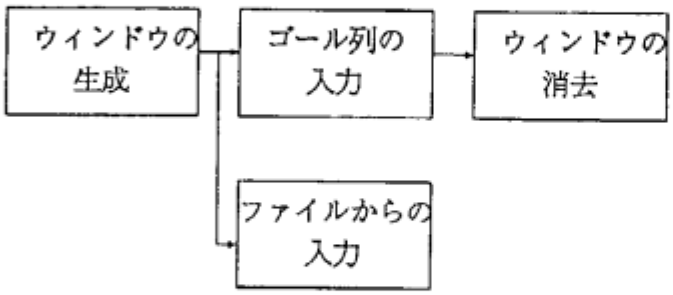
目次

1　はじめに

本手引書は，ＰＳＩ上のＣＡＬバージョン１．０のユーザ・インターフェースを規定するものである．ＣＡＬの言語仕様については，「ＣＡＬ言語仕様書」を参照のこと．

なお，ＰＳＩ，及びＳＩＭＰＯＳ (version 4.2) については，既知のものとする．

ＣＡＬ処理系に対する各処理の相互関係を，以下に図示する．



2　システムの起動と終了

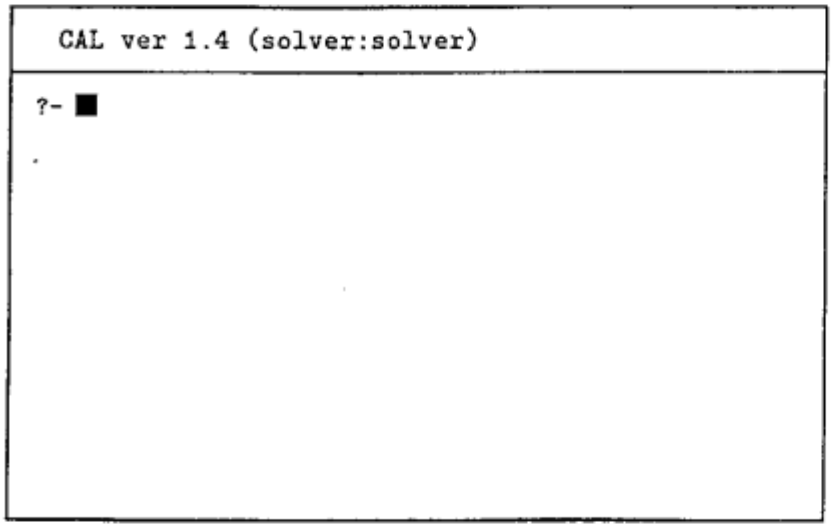
2.1　システムの起動

ＣＡＬ処理系を起動するには以下を行なう．

1. ＰＳＩのシステム・メニュー表示から，項目「ＣＡＬ」を選択（マウス の左ボタンの１回クリックによる）する．
   これを行なうと，画面上の マウス・カーソルがマーク” 「” に変更される．これは，ＣＡＬ処理系 のためのウィンドウの左上隅を示すものであり，適当な場所でマウスの 左ボタンを１回クリックすることにより，確定される．

2. ついで，画面上には，このようにして確定された場所を左上隅とし，現 在のマウス・カーソルの位置を右下隅とする矩形が表マウスの移動に伴 って，その右下隅が移動する．これも同様に，マウスの左ボタンの１回 クリックによって確定させると，ＣＡＬ処理系のためのウィンドウが確 定し，表示される．但し，このようにして確定したウィンドウの大きさ があらかじめ定めたサイズよりも小さい場合には，あらかじめ定められ たサイズのウィンドウが表示される．

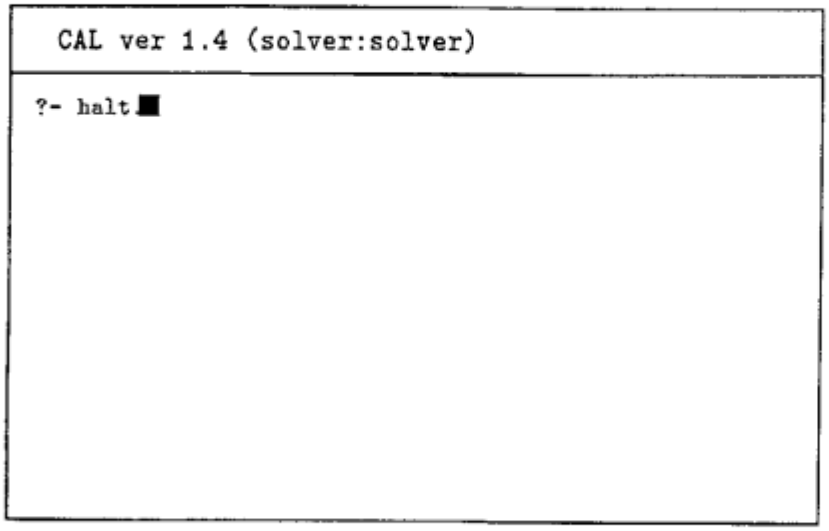
3. カーソルは，ＣＡＬ処理系のためのウィンドウ（以下，ＣＡＬウィンド ウと表記する）の左上隅で点滅しており，これ以降のキーボード入力は， ＣＡＬ処理系に対する入力となる（下図参照）．



ここで，” ?－” はＣＡＬプロンプトであり，■はＣＡＬウィンドウのカーソルである．
ＣＡＬウィンドウはｐｍａｃｓウィンドウであるため，ｐｍａｃｓの各機能を利用することができる．

2.2 システムの終了

ＣＡＬ処理系の実行を終了させるには，ゴール"halt" または"quit" を入力する．これにより，ＣＡＬウィンドウは消滅する．下図は，ＣＡＬウィンドウに"halt" を入力した状態を示す．



3　ユーザの入力

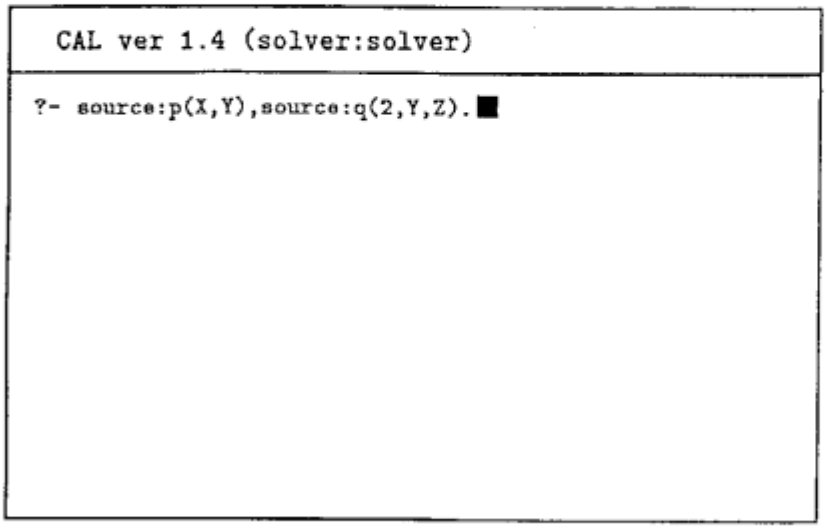
ＣＡＬ処理系への入力には，次の３種類がある．

1. ゴール列
2. ファイル名

以下，これらの入力について述べる．

3.1 ゴール列

入力において，ＣＡＬに対する．プロンプトに引き続くゴール列の入力は，ＣＡＬ処理系に対して，そのゴール列を評価させる機能を持つ．ゴール列の構文に関しては，「ＣＡＬ言語仕様書」を参照のこと．以下に，ゴール列の入力を行なったときのウィンドウの様子を示す．ゴール入力に当たっては，関数子の前に，ソースファイル名を記号":" で区切って付加する．以下は，ゴール"p(X, Y), q(2, Y, Z)" を入力した様子である．但し，ソースファイル名は，"source" とする．



3.2 ファイルからの入力

ファイル入力には，以下の３つの機能がある．

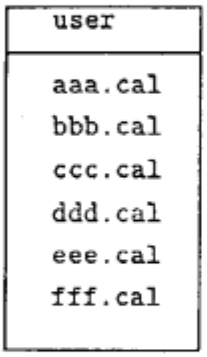
1. ファイルをただ読む．
2. ファイルを修正した上で読む．
3. 新たにプログラムを書き込む．

1 と 2 については，ファイルがあることを前提とする．

ファイルからの入力を行なう方法としては，メニューからのファイル入力と，ＣＡＬウィンドウからのファイル入力の二つの方法がある．

3.2.1　メニューからのファイル入力

マウスカーソルをウィンドウの内部にいれて，左2回ダブルクリックを行なうと，次の形式のメニューウィンドウが開く．このウィンドウの下半分はスクロール可能で，カレントディレクトリ中のＸＸＸ．ｃａｌというすべてのファイルを表示する．



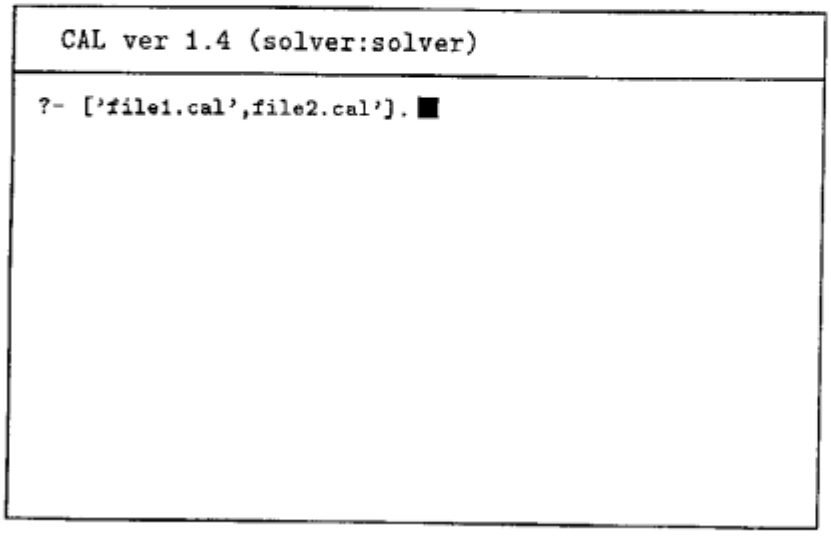
- メニューウィンドウのファイル名に対して左１回クリックを行なうと，そのファイルのｒｅｃｏｎｓｕｌｔを行ない，生成されたオブジェクト セーブする．
- ファイルに対して，修正を行なってからｒｅｃｏｎｓｕｌｔする場合は，メニューウィンドウのファイル名に対して，中１回クリックを行なうと，ｐｍａｃｓウィンドウが開かれ，修正を行なうことができる．ｐｍａｃｓウィンドウに対して，終了操作が行なわれると，編集されたバッファ の内容は”ｕｓｅｒ．ｃａｌ”ファイルに書き込まれて，その内容をｒｅｃｏｎｓｕｌｔし，ｐｍａｃｓウィンドウを消去する．この操作により，メニューウィンドウの下段に”ｕｓｅｒ．ｃａｌ”が登録される．
- メニューウィンドウの項目「ｕｓｅｒ」に対してマウスの中１回クリック するとｐｍａｃｓウィンドウが開き，ファイル”ｕｓｅｒ．ｃａｌ” を新規に作成する．ホームディレクトリに同一のファイルが存在するときは，同一名称で新しいバージョンのファイルが生成される．ｐｍａｃｓウィンドウに対して，終了操作が行なわれた場合には，ファイル”ｕｓｅｒ．ｃａｌ”が登録され，そのファイルの内容をｒｅｃｏｎｓｕｌｔし，対応するオブジェクトをセーブし，ｐｍａｃｓウィンドウを消去 する．この操作により，メニューウィンドウの下段に”ｕｓｅｒ．ｃａｌ”が登録される．
- メニューウィンドウのファイル名に対して，右１回クリックを行なうと，指定されたファイル名に対応するオブジェクトをロードする．
- メニューウィンドウのファイル名に対して，左２回クリックを行なうと，指定されたファイル名に対してｒｅｃｏｎｓｕｌｔのみを行なう．

[注意]
メニューウィンドウに現われるファイルは，現在ホームディレクトリのみである．ユーザが別のディレクトリを指定したい場合はあらかじめホームディレクトリに，"cal.init" というファイルを作成し，その中に対象とするディレクトリを書いておかなければならない．
例

```
current_dir("me:cal2").
```

3.2.2　ＣＡＬウィンドウからのファイル入力

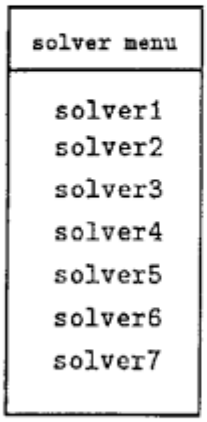
Ｐｒｏｌｏｇ処理系に対する上位互換性を保つために，" […] " 形式のファイル入力を用意している．
ファイル名は，" [" と "] " とで囲み，ファイル名を引用符で囲む．
複数のファイルから入力するには，ファイル名をコンマで区切ることによって
表記する．以下に，ファイル"file1.cal","file2.cal" からの入力を示す．
ファイル名の前に"-" を置いたときには，そのファイルに対してｐｍａｃｓウィンドウを開き修正を行なったあと，
ｒｅｃｏｎｓｕｌｔすることができる．但し，入力可能なファイルの拡張子は，ＣＡＬが望ましい．



3.2.3 制約評価系の変更

制約評価系をメニューにより切替えることができる.

マウスカーソルをウィンドウの内部に入れ, 右1回クリックを行なうと, 次の形式のメニューウィンドウが開く. このウィンドウはカレントディレクトリの制約評価系を記述したXXX.slv というすべてのファイルを表示する. XXX.slv

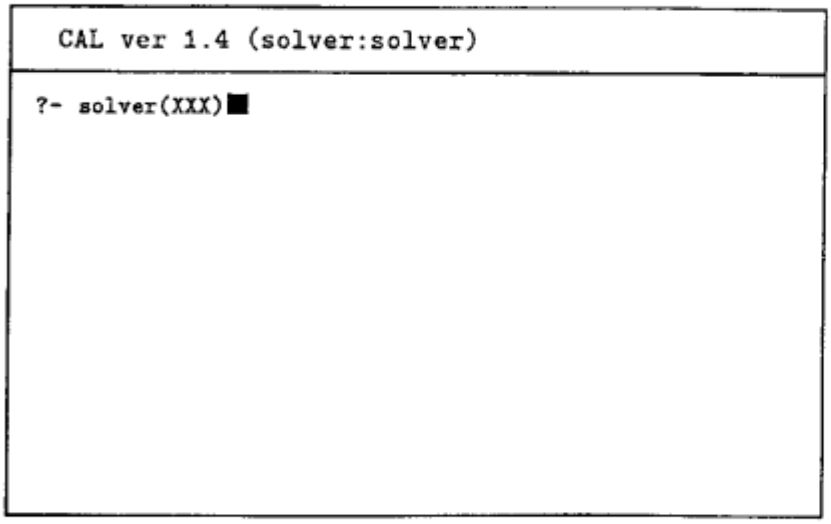


ファイルはあらかじめカタログ, セーブされてクラスオブジェクトをロードできる状態になっていなければならない.

制約評価系が変更されるとCAL ウィンドウのタイトル部分"(solver:XXX)" が指定された制約評価系の名前に変更される.

また, 以下のようにCAL ウィンドウにゴール"solver(XXX)." を入力することにより, 制約評価系をXXX に変更をすることができる.

制約評価系を初期状態のものに戻す時はCAL ウィンドウにゴール"solver." を入力する.



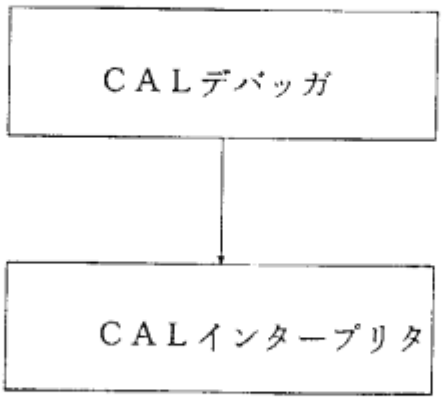
ＣＡＬデバッガ操作手引書 第1.4版

1990年6月25日

目次

1　はじめに

本仕様書は，ＰＳＩ上のＣＡＬデバッガのユーザ・インターフェースを規定 するものである．ＣＡＬの操作方法については，「ＣＡＬ操作手引書」を参照 のこと．
なお，ＰＳＩ，およびSIMPOS(version 6.1)については，既知のものとする．
ＣＡＬデバッガに対する各システムの相互関係を，以下に示す．

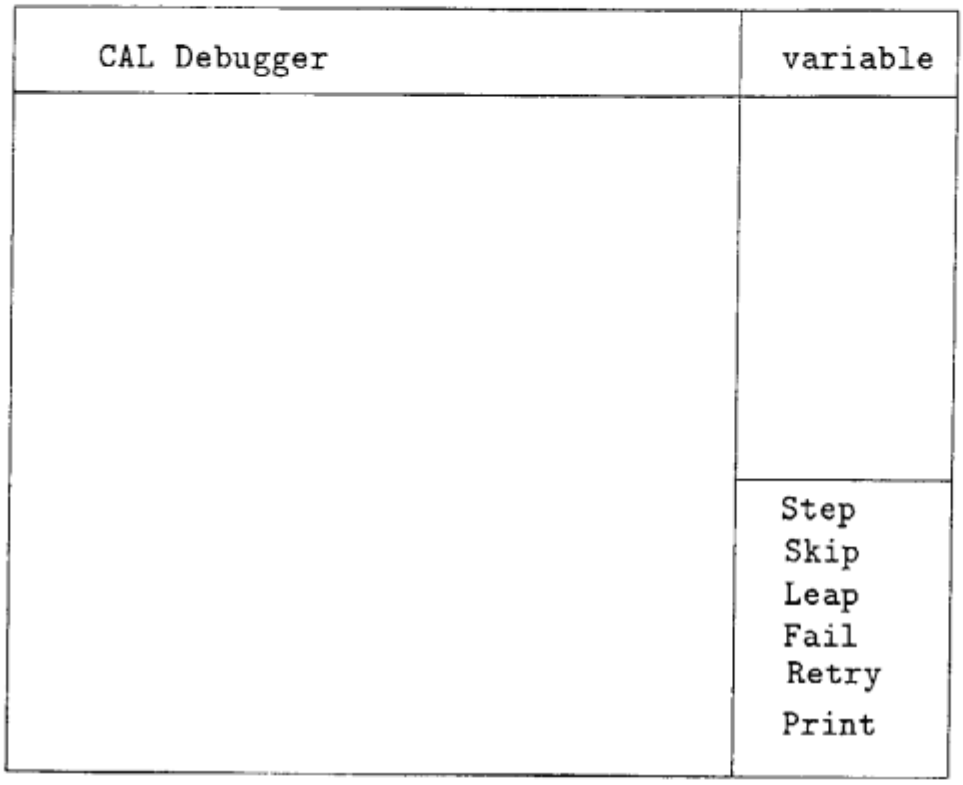


2　システムの起動と終了

2.1　システムの起動

ＣＡＬデバッガは、システムメニューからCAL debuggerを指定することにより ＣＡＬデバッガのウィンドウと，ＣＡＬのウィンドウが開かれ，ＣＡＬデバッ ガシステムが起動される.

以下にＣＡＬデバッガのウィンドウを図示する。



2.2　システムの終了

ＣＡＬデバッガを終了させるには、ＣＡＬインタープリタウィンドウより “halt.” を入力する。これによりＣＡＬインタープリタウィンドとＣＡＬデバッ ガウィンドウは消滅する。

3　ユーザの入力

3.1　モードとモード切替えコマンド

ＣＡＬデバッガには、中断ゲートや対象エントリの範囲を設定するいくつか のモードを指定することができる。中断ゲートとは、ユーザに制御コマンドの 入力を求めて止まるゲートのことである。　これらを設定するには、デバッガメニューのｓｔａｔｅを使用する（３．３．２参照）。

1. 中断ゲートの設定
   ＣＡＬデバッガによるトレースでは、ＣＡＬＬ，ＥＸＩＴ，ＲＥＤＯ，ＦＡＩＬの４種類のゲートがあり、それぞれについて中断ゲートを設定することがで きる。

2. 対象エントリの設定
   ＣＡＬでは、ＣＡＬ述語とＣＡＬシステム述語が存在し、デバッガではＣＡＬ述語のみをトレースするか、ＣＡＬシステム述語を含めてトレースを行なう かを選択する事ができる。　ＣＡＬ述語のみのトレースでは、制約を順に後ろ送りするための補足引数 は隠されているが、ＣＡＬシステム述語を含めたトレースの場合は、この補足 引数も表示される。

3.2 ゴール列の入力とトレース出力

1. ゴール列の入力
   ＣＡＩウィンドウから次のようにゴール列を入力する.
   $? - ham : horse_and_man(3,4,X,Y).$

2. トレース出力
   トレース結果はＣＡＬデバッガウィンドウに出力される.
   上のゴール列のトレース結果を以下に示す.

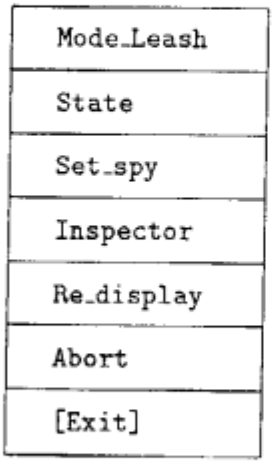
```
(1) 1 CALL method> ham:horse_and_man(3,4,A,B) ?
(2) 2 CALL> horse_and_man(3,4,A,B) ?
(3) 3 CALL method> buch:constr(A=3+4) ?
(3) 3 EXIT method> buch:constr(7=3+4) ?
(4) 3 CALL method> buch:constr(A=2*3+4*4) ?
(4) 3 EXIT method> buch:constr(22=2*3+4*4)
(2) 2 EXIT> horse_and_man(3,4,7,22)
(1) 1 EXIT method> ham:horse_and_man(3,4,7,22)
```

最初の括弧が呼び出し番号（invocation），次の数字が呼び出しの深さ（depth）である．“call”,”exit” と言うのは Pmodel のゲートを表わす.

3.3 コマンド

コマンドにはデバッガウィンドウに表示されている物と、デバッガメニュー から呼び出す物とがある。 デバッガウィンドウに表示されているものは、それをマウスでクリックする ことにより実行される。 デバッガメニューから呼び出すものは、デバッガメニューを開きそこから該 当するコマンドをマウスで選択する。

デバッガメニューを図示する。



デバッガメニューを開くには、デバッガウィンドウの"ｍｅｎｕ"をマウス クリックすれば良い。 "Ａｂｏｒｔ"はコマンドを実行しないで、デバッガウィンドウに戻る。

3.3.1　トレース制御コマンド

1. ｓｔｅｐ
   ステップ実行
   スパイエントリに関係なく、次の制御ゲートで中断する。

2. ｓｋｉｐ
   現在の述語以下のトレースを行なわない。

3. ｌｅａｐ
   次のスパイエントリまでトレース、中断を行なわないで実行する。

4. ｆａｉｌ
   その述語を失敗する。

5. ｒｅｔｒｙ
   その述語を再実行する。

6. ｐｒｉｎｔ
   補足引数の表示を行う（詳細は３．３．５参照）。

7. ｎｏｔｒａｃｅ
   デバッガのトップレベルまで一時的にトレースを中止する。

3.3.2 モードリーシュ

モードリーシュは各種モードの設定や、中断ゲートの指定を行なう。

1. ＣＡＬトレースモード（Ｍｏｄｅ）
   ｃａｌ：ｃａｌコードのみをトレースするモード
   ａｌｌ：ｃａｌ処理系内部もトレースするモード

2. ＣＡＬ表示モード（ｄｉｓｐｌａｙ）
   ｒｅｓｔｒｉｃｔ：補足引数の表示を抑制するモード
   ａｌｌ　　：補足引数も表示するモード

3. ｌｅａｓｈ
   トレースの中断する場所の指定を行なう。

4. ｔａｒｇｅｔ
   トレース表示をする述語の種類を指定する。

5. ｔｒａｃｅ
   トレース出力をするかどうかを指定する。

6. ｌｅａｓｈ
   実行を中断して制御コマンドの入力待をするかどうかを指定する。

```
Mode_Leash
mode    :  cal  all
display :restrict  full
leash   :full  tight  half  loose  no
target  :method local built
trace   :CALL  EXIT  REDO  FAIL
leash   :CALL  EXIT  REDO  FAIL
                                    [EXIT]
```

3.3.3 ステート

ステートはデバッガの各種モードの設定やトレース表示の状態を指定する。

1. 出力抑制（Ｄｅｐｔｈ＆Ｌｅｎｇｔｈ）
   デバッガのトレース表示のレベルを depth と length で抑制する。
   depth：構造の深さ
   length：要素の数
   表示形式についての詳細は「ＳＩＭＰＯＳプログラミング説明書」の「標準入出力（ホロフラスティング）」を参照のこと。

2. マクロ・クラスの設定（ｍａｃｒｏ）
   入力ゴール列をマクロ展開するマクロ・クラスを設定する。初期値はｅｓｐ＿ｍａｃｒｏ＿ｅｘｐａｎｄｅｒである。（ここで指定するマクロ・クラスはａｓ＿ｍａｃｒｏを継承していなくてはならない。）

3. 変数表示
   ゴール列実行時に変数の値を出力する際、出力対象とする変数を設定する。
   all　：　それまでに使った変数全て
   used：　そのゴール列で使った変数全て
   new：　そのゴール列で新しく使った変数
   no：　変数の値を出力しない

4. 変数メニューの表示指定
   デバッガウィンドウにおいて変数メニューを表示するかどうかを指定する。オンであれば表示し、オフであれば表示しない。既定値はオンであるが、オンにすることにより変数メニューのある部分までトレースウィンドウがエキスパンドされ、デバッガウィンドウの前面をトレースウィンドウにすることができる。オンにすることで再度変数メニューを表示することができる。

5. 使用ヒープエリアの計測
   入力されたゴール列の実行において、使用したヒープエリアの計測を行なうかどうかを指定する。既定値はオフであるがオンにすることにより入力ゴール列の実行において使用するヒープエリアの量を計測することができる。単位はワードである。

6. 実行時間の計測
   入力されたゴール列の実行において、実行時間の計測を行なうかどうかを指定する。既定値はオフであるが、オンにすることにより入力ゴール列の実行において実行時間の計測をすることができる。単位は msec である。

以下にステートウィンドウを図示する。

```
CAL stat
Depth>
Length>
macro class>
display vars :  all used new no
variable menu :  on off
heap counter :  on off
time :  on off
                                    [EXIT]
```

3.3.4 スパイ

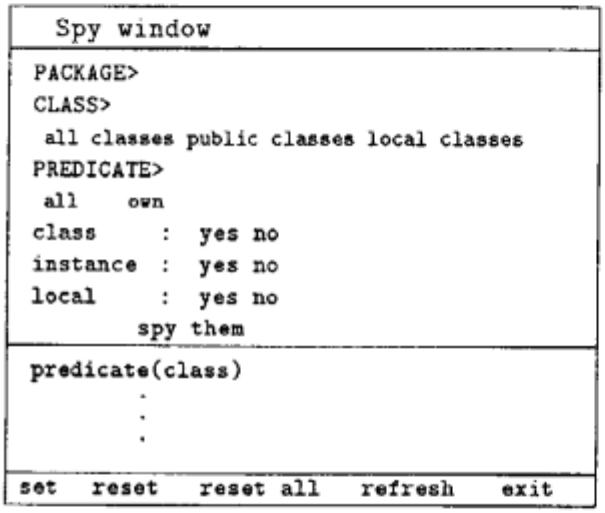
中断点の指示の方法として、全述語を対象として中断する場合と特定の述語のみを対象とする場合の２通りがある。後者の場合、対象とする特定の述語のことをスパイエントリと呼ぶ。つまり、スパイエントリでのみ中断してユーザに制御の指示を求めることができる。デバッガではスパイエントリの登録をスパイウィンドウにより行なうことができる。 デバッガメニューの"Spy" を選ぶとスパイウィンドウが表示される。デバッガウィンドウに戻るには、スパイウィンドウの"exit" を選ぶ。

スパイエントリの操作方法

1. 登録
   スパイエントリとして登録を行なう述語のクラス名をウィンドウのｃｌａｓｓに、述語名をウィンドウのｐｒｅｄｉｃａｔｅの欄にカーソルをマウスで移動させてから入力し、スパイウィンドウの"Set" を選ぶ。

2. 削除
   スパイウィンドウに表示されている述語をクリックして"reset" を選ぶ。

以下にスパイウィンドウを図示する。



スパイウィンドウの下半分のウィンドウには、現在登録されている述語が表示されている。

3.3.5　インスペクタ

インスペクタは、現在実行中の述語の状態を調べるためのものである。　詳しくは、「ＳＩＭＰＯＳ操作説明書」の「インスペクタ」を参照のこと。

3.3.6　補足引数の表示

補足引数とは、多項式の移行を行なった節について、制約を順に後ろ送りするための物である。　補足引数は２つある。１つは前から渡されてきた制約とユニファイするＧＢｉｎ、もう１つは次の述語に制約を渡すためのＧＢｏｕｔである。　補足引数の表示はこれらの補足引数の値を見やすい形にして表示する機能である。

4 デバッガのカスタマイズ

デフォルトディレクトリ（ディレクトリ名”ｍｅ：”）に“cal_debugger.ini”ファイルを作ることにより、デバッガの各種初期設定をカスタマイズ出来る。以下のようなタームの形式で必要なもののみ記述する。このとき、タームは任意の順序で指定できる。

```
debugger_size(X,Y).      % X,Y = 整数  デバッガウィンドウのサイズ
cal_sw(Mode).            % Mode = cal,all
display(Mode).           % Mode = restrict,all
leash(Leash).            % Leash = full,tight,half,loose,no
print_depth(N).          % N = 整数
print_length(N).         % N = 整数
font(String).            % String = "文字列"
package(Package).        % Package = パッケージ名アトム
macro(Class).            % Class = クラス名アトム
control_menu(SW).        % SW = on,off
```

また、リーシュ指定のfull,tight,half,loose,noで実際にどのような設定をするかということも以下のようにユーザが定義できる。

```
full([target(on,on,on),gate(leash,leash,leash,leash)]).
tight([target(off,on,on),gate(leash,trace,trace,leash)]).
half([target(off,off,on),gate(leash,off,off,leash)]).
loose([target(off,on,off),gate(leash,off,off,trace)]).
no([target(off,off,off),gate(off,off,off,off)]).
```

(注)
target,gateの各引数の意味は以下のようになっている。

```
target(Method,Local,predicate,Builtin).
gate(CALL,EXIT,REDO,FAIL).
```

CAL 言語仕様書　第 1.4 版

(財) 新世代コンピュータ技術開発機構 (ICOT)

昭和 63 年 7 月 5 日
平成元年 2 月 16 日
平成元年 3 月 22 日
平成元年 12 月 4 日
平成 2 年 2 月 5 日
平成 2 年 8 月 29 日
平成 2 年 9 月 28 日

目次

第 1 章

はじめに

1.1 言語概要

本仕様書は、PSI 上の CAL バージョン 1.4 の言語仕様について述べたものである。

1.2 構文規則

以下に，本仕様書において用いる構文規則について述べる。これは、BNF 記法に、若干の拡張を加えたものである。

1) {X} は、X の 1 回以上の繰り返しを示す。

2) 漢字、あるいはカタカナ以外で始まる構文要素は、非終端記号を示す。

3))、(、{、} などは、終端記号を示す。但し、1) との冗長性に関しては、終端記号である場合に、その都度その旨明記する。

第 2 章

項

2.1 変数

変数は、英大文字、または下線 "_" で始まる名前文字か、下線からなる文字列で表わされる。

変数の有効範囲は、その変数の出現した節内に限られる。同一の節においては、同じ名前を持った変数は、同一の変数として扱われ、結合された値を共有する。逆に、同じ名前を持った変数でも、出現する節が異なれば、別の変数として扱われる。

下線一文字のみからなる変数は、無名変数と呼ばれ、同一の節内に複数箇所現われても、すべて異なる変数として扱われる。

[例]　X　_　_var　V1

構文規則

変数 ::= 変数接頭辞|変数接頭辞 { 名前文字 }
変数接頭辞 ::= _ |英大文字

2.2 データオブジェクト

2.2.1 文字

すべての文字は、JIS コードで表現する。CAL で使用できる文字を以下に示す。これらの文字は、大文字小文字の区別も含めて、すべて異なる文字として扱われる。

構文規則

```
文字   ::=  名前文字|記号文字|区切り文字

名前文字   ::=  数字|英小文字|英大文字|漢字|_
数字   ::=  0|1|2|3|4|5|6|7|8|9
英小文字   ::=  a|b|c|...|x|y|z
英大文字   ::=  A|B|C|...|X|Y|Z
記号文字   ::=  @|#|$|~|&|+|-|=|^|?|:|>|<|*|/...
区切り文字   ::=  ()|[]|{}|,|!|;
```

2.2.2 アトム

アトムは、名前により一意的に識別される基本的なデータ・オブジェクトである。CAL における述語名などはアトムである。CAL においては、以下に示すようにアトムを表記する。

1) 英小文字、漢字から始まる名前文字、及び下線の並び。
[例] abc 漢字 atomic_symbol

2) 記号文字の列
[例] :- ?- + - * /

3) 引用符 (') で囲まれた任意の文字列
[例] 'koreha アトムです' '235ef'

4) 以下に示す 4 種類の文字
! | , ;

5) 空リスト
[]

2.2.3 数値

CAL においては、整数、有理数、複素数の 3 種類の数値を用いることができる。

構文規則

数値 ::= 有理数|複素数
有理数 ::= 整数|分数

整数

整数は、一定範囲の整数値を表わすデータの表現である。整数の記法としては、通常の 10 進表記法を用いる。

[例] 123 −123 0

構文規則

整数 ::= 符号 { 数字 }|{ 数字 }
符号 ::= + | −

分数

分数は、ある分数値を表わすデータの表現である。

[例] 1/2 −2/32 3/1

構文規則

分数 ::= 符号 分数絶対値|分数絶対値

分数絶対値 ::= 分子/分母
分子 ::= { 数字 }
分母 ::= { 数字 }

複素数

複素数は、ある複素数値を表わすデータの表現である。

[例] c(2,3) c(−1/5,4) c(2/3,−4/5)

構文規則

複素数 ::= c(実部, 虚部)
実部 ::= 有理数
虚部 ::= 有理数

2.2.4 リスト

CAL においては、通常の Prolog と同様、リストを用いることができる。リストは、2 進木を表現するデータの表現であり、その第 1 要素を頭部、第 2 要素を尾部と呼ぶ。リストに対しては、リスト記法と呼ばれる特別な略記法を用意する。

$$\begin{aligned} [X] &\rightarrow [X\|[\,]] \\ [X,\ldots Y] &\rightarrow [X\|[\ldots\|[Y]\ldots]] \\ [X,\ldots Y|Z] &\rightarrow [X\|[\ldots[Y|Z]\ldots]] \end{aligned}$$

ただし、[ ] は空リストを表わすアトムであり、ニルと呼ばれる。

構文規則

リスト ::= 空リスト|非空リスト

空リスト ::= [ ]
非空リスト ::= [列]|[列|項][1]
列 ::= 項|項, 列

2.2.5 文字列

2 重引用符で囲まれた文字の列を文字列と呼ぶ。但し、2 重引用符自身は、連続する 2 つの 2 重引用符で表わす。

[例] " 文字列 " " " " " "abd21"

構文規則

文字列 ::= "{ 文字列文字 }"

文字列文字 ::= 文字 (ただし 2 重引用符を除く)|" "

[1] [列|項] の "|" は終端記号である。

2.3 項

2.3.1 項

項の構文は、以下の通りである。

構文規則

項 ::= アトム|複合項

2.3.2 複合項

複合項は、以下のように表記される。

構文規則

複合項 ::= 関数子 (引数列)
関数子 ::= アトム
引数列 ::= 引数|引数, 引数列
引数 ::= 項|制約式[2]

[2]制約式に関しては、各制約領域にしたがって構文も異なるため、本仕様書では述べない。

第 3 章

プログラム構造

プログラムは、文の並びからなる。文は、コマンドまたは節から構成され、これらは終止記号によって区切られる。終止記号は、ピリオドと、それに続く空白、または改行コードで示される。　節の基本構成要素をゴールと呼ぶ。ゴールは、アトム、複合項、制約、カット、公開宣言、型宣言などからなる。

CAL プログラムにおいて、そのトップ・レベルの述語は、公開宣言がなされている必要がある。また、CAL においては、各述語に対して型宣言を行う。型宣言においては、型名列に含まれる型名の個数はその関数子の位数に一致していなければならない。型宣言中の _ は、その位置にある引数が節の単一化の際に通常の単一化によってあつかわれることを示し、型名がアトムの場合は、それぞれのアトムによって示される制約評価系における制約となることを示す。

構文規則

```
プログラム    ::=  { 文 }
文            ::=  節 終止記号|コマンド 終止記号
終止記号      ::=  . { 空白|改行コード }
節            ::=  頭部|
                   頭部 :- 本体部|
                   公開宣言|
                   型宣言
公開宣言      ::=  :- public 公開宣言列
公開宣言列    ::=  公開宣言要素|公開宣言要素 {, 公開宣言要素 }
公開宣言要素  ::=  関数子/{ 数字 }
型宣言        ::=  :- type 型宣言列.
型宣言列      ::=  型宣言要素|型宣言要素 {, 型宣言要素 }
型宣言要素    ::=  関数子 (型名列)
型名列        ::=  型名|型名 {, 型名 }
型名          ::=  アトム|_
頭部          ::=  アトム|複合項
本体部        ::=  ゴール|ゴール {, ゴール }
ゴール        ::=  通常ゴール|特殊ゴール
通常ゴール    ::=  アトム|複合項|
```

ESP ゴール|外部ゴール
特殊ゴール ::= リスト|カット
カット ::= !
外部ゴール ::= クラス：アトム|クラス：複合項

制約についての詳細は、制約仕様書に記述されていなければならない。

第 4 章

コメント

文字 "%" から改行コードまでをコメントとし、システムはこれを無視する。

第 5 章

制約

CAL における制約とは、一般には、変数および定数の間において成立する関係をいう。しかし、原則として制約処理系は CAL システムと独立であり、制約の具体的な構文や意味については、個々の制約処理系ごとに制約仕様書によって規定されなければならない。ここでは、CAL が標準で提供する代数制約とブール代数制約について述べる。

制約処理系仕様書に従って制約処理系を作成した者 (制約処理系提供者) は、ここの記述を参考にして自分の制約処理系についての仕様書を作成されたい。

5.1 制約とは

CAL は、標準として、代数制約とブール代数制約を提供する。これらの制約は、能動的なものであり、これらの関係は制約評価系によって評価され、関係が十分な情報を含んでいる場合には、制約中に出現する変数の値が確定する。

1) 代数制約
   - 1.1) 多項式からなる等式
   - 1.2) 多項式を記号 "==" で結んだもの
   - 1.3) `write`
   - 1.4) 多項式を記号 "\==" で結んだもの
   - 1.5) 多項式を記号 "\=" で結んだもの

2) ブール代数制約
   - 2.1) ブール式からなる等式
   - 2.2) ブール式を記号 "==" で結んだもの
   - 2.3) `write`
   - 2.4) ブール式を記号 "\==" で結んだもの

上記中、1) は代数制約を、2) はブール代数制約を記述するためのものである。1.1)、2.1) はともに通常の等式制約を表し、1.2)、2.2) はともにその時点までに集められている制約において、 == の右辺と左辺とが等しくなるかどうかをチェックするものである。また、1.3)、2.3) はその時点までに集められている制約において、write の引数として与えられている多項式、あるいはブール式の値を求め、それを表示するものである。1.4)、および 2.4) は、その時点までに集められている制約において、\== の両辺が等しくないかどうかを判定する制約である。1.5) は、\= の両辺が等しくないという制約である。

```
制約 ::= 代数制約|ブール制約
代数制約 ::= 多項制約|特殊代数制約
多項制約 ::= 多項等式
多項等式 ::= alg: 多項式 = 多項式
多項式 ::= 代数項|加減作用素 代数項|
           多項式 加減作用素 代数項
代数項 ::= 因子|代数項 乗除作用素 因子
因子 ::= 1 次子|1 次子 ^{ 数字 }
1 次子 ::= 定数|変数|(多項式)|
           dif(多項式, 変数)
定数 ::= 有理数|複素数
加減作用素 ::= +|-
乗除作用素 ::= *|/
特殊代数制約 ::= alg: 多項式 == 多項式|alg:write(多項式)|
                 alg: 多項式 \== 多項式|alg: 多項式 \= 多項式
ブール制約 ::= ブール等式|特殊ブール制約
ブール等式 ::= bool: ブール多項式 = ブール多項式
ブール多項式 ::= ブール和|
                 ブール和< - >ブール和|
                 ブール和- >ブール和|
                 ブール和< -ブール和|
                 ブール和++ブール和|
                 ブール和<>ブール和
ブール和 ::= ブール積|ブール和 論理和作用素 ブール積
ブール積 ::= リテラル|リテラル 論理積作用素 ブール 1 次子
リテラル ::= ブール 1 次子|否定作用素 ブール 1 次子
ブール 1 次子 ::= ブール定数|変数|(ブール多項式)
否定作用素 ::= ~|\
論理和作用素 ::= \/
論理積作用素 ::= /\|&
特殊ブール制約 ::= bool: ブール多項式 == ブール多項式|
                   bool:write(ブール多項式)|bool: ブール多項式 \== ブール多項式
ブール定数 ::= 0|1
```

[注記]

数を表現するデータ表現において、整数、分数、および複素数は、同一の数を表現している場合には、その表現法が異なる場合にも、同一の数と認識される。たとえば、3 (整数)、3/1 (分数)、6/2 (分数)、c(3,0)

(複素数) などは、いずれも同一の数であると認識される。ブール多項式において、++ および <> は、ともに排他的論理和を表す。

5.2 制約評価系

制約を能動的に解くために必要な、CAL の重要な構成要素のひとつが制約評価系である。システムに組み込まれている制約評価系は、以下の 2 種類である。

1. 代数的等式のための制約評価系 (ブフバーガ・アルゴリズム)

   各変数のとりうる範囲は複素数である。最上位の関係演算子は、等号 (=)、二重等号 (==)、write、二重非等号 (\==)、および非等号 (\=) である。式中の係数としては、有理数、および複素数を許す。また、演算子としては、通常の加減乗除およびべき乗を許すが、この制約評価系で扱えるのは多項式の範囲のみである。

2. ブール式のための制約評価系 (ブーリアン・ブフバーガ・アルゴリズム)

   各変数のとりうる範囲はブール定数である。最上位の関係演算子は、等号 (=)、二重等号 (==)、write、および二重非等号 (\==) である。演算子としては、論理和、論理積、否定、含意、同値、排他的論理和を許す。

5.3 コマンド

CAL 処理系に対し、ゴール入力時に以下のコマンドを入力することができる。(実際には、CAL プログラムの節の本体に書いても良いが、制約処理系の実行環境を変える 1 種のメタコマンドであるから、注意深い扱いが必要であり、なるべくゴール入力時のみの限定使用が望ましい。)

代数制約の評価に際しては、単項間の順序が、評価結果に大きな影響を与える。優先度関連のコマンドは、以下の構文に従う。

構文規則

コマンド	::=	優先度宣言コマンド|優先度参照コマンド|優先度初期化コマンド
優先度宣言コマンド	::=	評価系:pre(被宣言子, 優先度).
優先度参照コマンド	::=	評価系:pre(変数名 (ground term),ESP 変数).
優先度初期化コマンド	::=	評価系:clear_pre.
被宣言子	::=	変数名 (ground term)|[変数名 (ground term) 列]
優先度	::=	整数
評価系	::=	alg|bool

優先度宣言コマンドは，被宣言子の優先度を指定された優先度に変更する。優先度参照コマンドは，指定された変数名の現在の優先度と指定された ESP 変数とを unify する。優先度初期化コマンドは，すべての変数の優先度を暗黙値 (0) に戻す。優先度の取り得る範囲は、-2147483648 から 2147483647 の間の整数である。

集合制約評価系利用手引書　第 1.4 版

(財) 新世代コンピュータ技術開発機構 (ICOT)

1990 年 1 月 30 日
1990 年 6 月 11 日
1990 年 9 月 3 日

目 次

1　はじめに

本手引書は、ＰＳＩ上のＣＡＬシステムにおいて、「集合方程式」のための制約評価系（以降、ＳＥＴＣＡＬと略す）を操作するための必要条項をまとめたものである。ＰＳＩ、ＳＩＭＰＯＳ、ＣＡＬについては、既知のものとする。

2　ＳＥＴＣＡＬが扱える集合について

2.1　集合

ＳＥＴＣＡＬで扱える対象は、要素、要素の集まりである集合、のどちらかである。集合は、有限個の要素を列挙した有限集合、およびその補集合である。ＳＥＴＣＡＬでは集合と要素を区別して扱っているので、集合と要素の両方に同じアトムを使用しても、違うものとして処理される（２．４節参照）。

特殊な集合として、空集合と普遍集合がある。ＳＥＴＣＡＬでは、空集合は“０”、普遍集合は“１”と表す。

要素１，要素２，…，要素ｎから成る集合は、

｛要素１，要素２，…，要素ｎ｝

と書く。また、要素１，要素２，…，要素ｎ以外から成る集合は、

｛要素１，要素２，…，要素ｎ｝^{c}

と書く。ここで、“^{c}”は補集合を意味する。普遍集合は無限集合なので、この２番目の集合は無限集合である。

2.2 集合の演算

ＳＥＴＣＡＬで扱える集合演算は、集合和、集合積、集合差、対称差、補集合である。

集合和	＼／ 及び ∪
集合積	／＼，∩ 及び &
集合差	−
対象差	⊕，＋＋ 及び <>
補集合 （前置）	～ 及び ＼
補集合 （後置）	ᶜ

また、次の関係が扱える。

集合の同値	=，== 及び ＼==
集合同士の包含	⊆ 及び ⊇
集合と要素の包含	∈ 及び ∋
集合と要素の包含の否定	∉ 及び ∌

上の演算及び関係に対し、次の関係が成り立つ。ここで、英大文字は集合を表す式、英小文字は要素を表す。

X = Y	≡	X ⊕ Y = 0
X ⊆ Y	≡	(X ∩ Y) ⊕ X = 0
X ⊇ Y	≡	(X ∩ Y) ⊕ Y = 0
x ∈ Y	≡	({x} ∩ Y) ⊕ {x} = 0
X ∋ y	≡	(X ∩ {y}) ⊕ {y} = 0
x ∉ Y	≡	{x} ∩ Y = 0
X ∌ y	≡	X ∩ {y} = 0
X ∪ Y	≡	(X ∩ Y) ⊕ X ⊕ Y
X − Y	≡	(X ∩ Y) ⊕ X
Xᶜ	≡	X ⊕ 1

評価系内でも上の関係を使って、

（対象差演算と集合積演算のみからなる式） = 0

の形に変換して処理している。

2.3　グレブナ・ベース

ＳＥＴＣＡＬは、与えられた式の列に対するグレブナ・ベースを求める。グレブナ・ベースは、直感的には、与えられた式の列と同じ解空間を表す、より簡単な式の列である。解が一意であれば、その値を示す式のみになり、解が複数あれば一般には入力より分かり易い式となるが、余分な式が付け加わる場合もある。

例

入力

$$a \in x \cup y \cup z^c$$
$$a \notin x \cap y \cap z$$
$$a \notin x^c \cap y$$
$$a \notin y^c \cap z$$
$$a \notin z^c \cap x$$

出力

$$z \not\ni a$$
$$y \not\ni a$$
$$x \not\ni a$$

例

入力

$$x \cap y = z$$

出力

$$y \supseteq z$$
$$x \supseteq z$$
$$y \cap x = z$$

2.4 集合と要素の型

ＳＥＴＣＡＬでは、集合を表すアトムと要素を表すアトムを別に処理している。従って同じアトムを集合と要素の両方に使用しても、違うものとみなされる。そこで、次の２式は矛盾しない。

$$x \in y,\ y \in x$$

同じアトムが集合と要素の両方に使用されているかをチェックする機構をつけることは可能だが、現在のところサポートしていない。

2.5 普遍集合について

ＳＥＴＣＡＬでは、

$$\{要素1,\ \cdots,\ 要素n\} = 1$$

という関係式が導かれると、矛盾しているとみなす。これは、ＳＥＴＣＡＬにおける１が、要素となり得るアトムすべてからなる無限集合を意味するためである。従って、ある有限集合を全体集合としたいときは、

$$u = \{要素1,\ \cdots,\ 要素n\}$$

のように、ユーザが定義しなければならない。そして、補集合を取るときは、必ず上のようにして定義したｕとの集合積を取るようにしなければならない。

例

入力	u＝｛a，b｝
	x＝｛a｝ᶜ ∩ u
出力	u＝｛a，b｝
	x＝｛b｝

2.6 要素の存在について

ＳＥＴＣＡＬでは、関係“≠”を表現することができない。従って、「ある集合ｘが空集合ではない」というような表現はできない。これは、「ある集合ｘは要素を持つ」という制約によって解の範囲を狭めることができないことを意味する。ただし、「ある集合ｘが空集合になり得るか」を知ることはできる。同じことだが、集合に含まれる要素の数を制約とすることもできない。

3　ＳＥＴＣＡＬ操作手引

ＳＥＴＣＡＬは、ＣＡＬシステムの制約評価系の１つである。従って他の制約評価系と同様に、ＳＥＴＣＡＬを使う場合はＣＡＬシステムの中から評価系を呼び出すことになる。ＣＡＬシステムに関する詳細はＣＡＬ操作手引書を参照のこと。

3.1　ＳＥＴＣＡＬの準備

ＳＥＴＣＡＬを使用するには、ＣＡＬシステムの他に次のＥＳＰプログラムがカタログされていなければならない。

setgb.slv

ＳＥＴＣＡＬでは集合用の特殊記号を使用している。ほとんどの記号はＪＩＳにあり標準サポートしているが、⊕、^{c}、$\not\in$、$\not\ni$の４つはＪＩＳにないので、これらも使用する場合には専用のフォント

kanji_cal.font

をディレクトリ“>sys>font”に用意しなければならない。このフォントはＰＳＩが標準に持つ“kanji_16.font”の４文字を以下のように置き換えたものである。

ＪＩＳコード 2267	$\propto$	⇒	$\not\in$
ＪＩＳコード 2268	$\because$	⇒	$\not\ni$
ＪＩＳコード 2269	$\int$	⇒	^{c}
ＪＩＳコード 226A	$\iint$	⇒	⊕

3.2　ＣＡＬシステムの起動

ＣＡＬシステムの起動の要約は次のようになる。

起動：ＰＳＩのシステムメニューから、集合制約評価系を含むＣＡＬシステムをマウスにより選択する。次に、ウィンドウの左上隅と右下隅の位置をマウスにより選択すると、ＣＡＬシステムのウィンドウが開かれ、システムが起動される。

3.3 ユーザの入力

ＣＡＬシステムが起動されると、ウィンドウ内にプロンプト“？－ ”と、システムへの入力開始位置を示すマーク“「”が表示される。このマークからの入力を、トップレベルからの入力と呼ぶ。入力開始マーク以降で終止点“． ”を打って改行すると、開始マークから終止点までをシステムへの入力とみなす。終止点を打たずに改行すると、入力が続いているとみなされるので、数行にわたって１つの入力をすることもできる。ＳＥＴＣＡＬにおいて最初に入力しなければならないのは、ウィンドウの初期化を行うコマンド“setgb:init”である。コマンドについては、下でまとめて述べる。

ＣＡＬウィンドウはｐｍａｃｓウィンドウなので、ｐｍａｃｓの機能を使うことができる。従って、編集をしたり、日本語入力モードにして漢字をアトムとして入力することもできる。

入力には、ゴール列、ファイル名、コマンドの３種類がある。言語仕様については、４章およびＣＡＬ言語仕様書を参照のこと。

ゴール列 ： トップレベルから直接、制約式を入力することも、ファイルに書かれているゴールを呼び出すこともできる。ファイルに書かれているゴールを呼び出す場合は、関数子の前にソースファイル名を記号“：”で区切って付加する。ただし、この場合はあらかじめ次の『ファイル名の入力』を行っていなければならない。

(注) 集合制約をトップレベルから入力する場合、変数名もすべて小文字で入力しなければならない。大文字で入力した場合、結果は内部表現の形そのままで出力されるか、表示されないかのどちらかであるが、無矛盾性のチェックを行うことはできる。また、要素名として大文字を使用する場合には、その式が処理される前に何らかの値に具体化していなければならない。詳しくは３．６節を参照のこと。

ファイル名 ： ＣＡＬプログラムのファイルを作成したら、オブジェクトファイルに変換し、カタログしなければならない。また、プログラムのファイルを修正したり新たに作成する場合にも、この機能を使うことができる。この入力方法は、ＣＡＬ操作手引書を参照のこと。

コマンド ： ＳＥＴＣＡＬを使うときは、まずウィンドウのセットアップをしなければならない。また、集合制約を評価する上で変数の優先度を指定したり、ゴール列実行後にグレブナ・ベースを表示するのを止めるなどの指定をするのに、以下のコマンドが用意されている。

setgb:init

集合の特殊記号を入出力するための、ウィンドウの設定を行う。ウィンドウのフォントは、ディレクトリ“>sys>font”に“kanji_cal”があればそれに、なければ“kanji_16”に設定される。

setgb:gb(型名なしの集合式のリスト)

リストに含まれる集合制約式からグレブナ・ベースを求める。集合制約式をプログラム内やトップレベルに直接書いた場合には、制約式ごとにこのコマンドを実行していると考えていい。

setgb:cprint

ゴール列内におけるその時点でのグレブナ・ベースを表示する。

setgb:write(集合制約式)

ゴール列内におけるその時点でのグレブナ・ベースによって、与えられた集合制約式を簡単にする。

setgb:pmode(スイッチ)

ゴール列処理終了後に求まったグレブナ・ベースを表示するか指定する。スイッチがｏｎなら表示し、ｏｆｆなら表示しない。初期設定はｏｎである。

setgb:pre(被宣言子, 優先度)

優先度が与えられれば被宣言子の優先度をその値に設定し、優先度が変数であればシステムにおける被宣言子の優先度の値を表示する。被宣言子は ground term でなければならない。優先度は -2147483648 から 2147483647 の間の整数であり、値が大きいほど優先されるので、結果の欲しい ground term には大きい数を優先度として指定するといい。優先度の初期値は 0 である。このコマンドにより優先度を指定すると、次に同じ被宣言子に対して指定し直すまで優先度は保存される。

setgb:clear_pre

すべての優先度を初期値に戻す。

例　第1引数が集合のリスト、第2引数がその和、である関係"union"をファイル"union.cal"に定義し、オブジェクトファイルに変換、カタログして実行する。

まず、エディタを使って次のプログラムを、"union.cal"という名前でカレントディレクトリにセーブする。

```
:- public union/2.

union([], 0).
union([X|Y], U) :- union(Y, V), setgb:V∪X=U.
```

次に、CALシステムにおいてマウス左ダブルクリックを行うとCALプログラムのメニューが現れるので、"union.cal"を選択してマウス左クリックを行う。

```
?-
"union.cal" is generated.

 yes

?- union:union([{a}, {c, d}, x], {a, b, c, d}).


{d, a, c} ^c∩x = {b}


 yes

?-
```

例

```
?- setgb:x=y.

y = x

 yes

?- setgb:pre(y, Y).
```

```
Y = 0

  yes

?- setgb:pre(x, 100), setgb:x=y.

x = y

  yes

?- setgb:clear_prc.

  yes

?- setgb:x=y.

y = x

  yes

?-
```

3.4　集合の特殊記号の入力について

１０個の特殊記号、⊆、⊇、∈、∋、∉、∌、∪、∩、⊕、ᶜは、端末から入力するのが大変である。次の入力方法があるので、ユーザはそれらを自由に使い分けることができる。

3.4.1　別の記号による入力

図１のように、キーボードにある記号を使って入力することもできる。

x ∪ y	⇒	x ／＼ y
x ∩ y	⇒	x ＼／ y
x ⊕ y	⇒	x ＋＋ y
xᶜ	⇒	～x
x ⊆ y	⇒	～y ／＼ x ＝ 0
x ⊇ y	⇒	～x ／＼ y ＝ 0
x ∈ y	⇒	～y ／＼ {x} ＝ 0
x ∋ y	⇒	～x ／＼ {y} ＝ 0
x ∉ y	⇒	y ／＼ {x} ＝ 0
x ∌ y	⇒	x ／＼ {y} ＝ 0

3.4.2　ＪＩＳコードによる入力

ＪＩＳコードを覚えていれば、それを使って入力することもできる。この方法の利点は、マウスを使わずに特殊記号を入力できることである。

入力するときは、マーカをカーソル位置へ移し（Ｃｔｒｌ－ｓｐａｃｅ）ＪＩＳコードを１６進で入力する。そしてファンクション・キーＰＦ７を押すと、特殊記号に変換される。

ＪＩＳコード表にない記号 ∉、∌、⊕、ᶜ は、∝（16#"2267"）、∵（16#"2268"）、∫（16#"2269"）、∬（16#"226A"）のところにそれぞれ定義してある。

3.4.3　単漢字変換による入力

ＣＡＬウィンドウをｗｏｒｐｒｏモードにし（Ｃｔｒｌ－ｘ，ｍ）、“がくじゅつ”と入力してから単漢字変換を行う（Ｍｅｔａ－ｊ）。このとき、マウス中２回クリックをすると、学術記号のメニューが現れる。

3.4.4　カナ漢字変換による入力

ＣＡＬウィンドウをｗｏｒｐｒｏモードにし（Ｃｔｒｌ－ｘ，ｍ）、特殊記号を前述の方法でウィンドウに表示し、簡易ユーザ辞書登録（Ｃｔｒｌ－ｘ，Ｃｔｒｌ－ｕ）を行う。次からは登録した“読み”を入力し、カナ漢字変換をすれば入力できる。

3.5 ＳＥＴＣＡＬの出力

ＳＥＴＣＡＬは、入力した式に対するグレブナ・ベースを求めるので、基本的には集合等式の列を出力する。しかし、以下のように明らかに包含関係に直せる式は、そのように変換して出力する。

ここで、α，βは集合アトムの積、a，b，…は要素アトム、A＝｛a，b，…｝とする。

(1) $A^c \cap \alpha = 0$	$\Rightarrow$	$A \supseteq \alpha$
(2) $A \cap \alpha = 0$	$\Rightarrow$	$\alpha \not\ni a, b, \cdots$
(3) $A \cap \alpha = A$	$\Rightarrow$	$\alpha \ni a, b, \cdots$
(4) $A \cap \alpha \cap \beta = A \cap \alpha$	$\Rightarrow$	$\beta \supseteq A \cap \alpha$
(5) $A \cap \alpha \cap \beta = \alpha$	$\Rightarrow$	$A \cap \beta \supseteq \alpha$

3.6 ＣＡＬシステムの終了

終了 ： ＣＡＬウィンドウのトップレベルから、“ halt.”を入力する。これによりＣＡＬウィンドウは消滅し、システムは終了する。

3.7 同一化

集合制約評価系内では同一化を行わない。ただし集合に関しては、同値関係“＝”によって同じ結果を得ることができる。

集合制約評価系に集合名として変数を使用すると、その集合名に対する関係式は、内部表現のままで出力されるか出力されないかのどちらかである。そこで、計算結果が欲しい集合名はアトムで指定しなければならない。

要素名としては、変数を使用することはできない。ただし、集合制約評価系を呼び出す時点で具体化されていれば、使用しても構わない。

例

```
?- setgb:x={a}.

x = {a}

 yes

?- setgb:X={a}.

X = (a(0, a)+0)*1+0

 yes

?- setgb:x=x∩y.

y ⊇ x

 yes

?- setgb:X=X∩Y.

"X"
"Y"

 yes

?-
```

例

```
?- {X=a}, setgb:x={X}.

X = a
x = {a}

 yes

?- setgb:x={X}, {X=a}.

 no

?- setgb:x={a}, setgb:y=x.
```

```
y = {a}
x = {a}

  yes

?-
```

4　ＳＥＴＣＡＬ言語仕様

構文規則、項は、ほぼＣＡＬ言語仕様書に準じる。記号文字として、以下の集合特有の記号が加わる。

$\subseteq$、$\supseteq$、$\in$、$\ni$、$\cap$、$\cup$、$\notin$、$\not\ni$、^{c}、$\oplus$

ＳＥＴＣＡＬが扱う集合制約式は、以下のような構文規則に従う。アトム、変数は、ＣＡＬシステムに準ずる。ただし、'$ALL'と$\propto$、$\because$、$\int$、$\iint$の４つの演算子は、内部で特別に使用しているので、ユーザーは使用することができない。

集合制約	::=	集合式 | 特殊集合制約
集合式	::=	setgb: 集合多項式 = 集合多項式 |
		setgb: 集合多項式 $\subseteq$ 集合多項式 |
		setgb: 集合多項式 $\supseteq$ 集合多項式 |
		setgb: アトム $\in$ 集合多項式 |
		setgb: 集合多項式 $\ni$ アトム |
		setgb: アトム $\notin$ 集合多項式 |
		setgb: 集合多項式 $\not\ni$ アトム
集合多項式	::=	集合１次子 |
		集合多項式 集合和作用素 集合多項式 |
		集合多項式 集合積作用素 集合多項式 |
		補集合作用素１ 集合多項式 |
		集合多項式 補集合作用素２ |
		集合多項式 対称差作用素 集合多項式
集合１次子	::=	既知集合 | 未知集合 | 特殊集合
既知集合	::=	“｛”アトム {, アトム} “｝” |
		“｛”アトム {, アトム} “｝” 補集合作用素２
未知集合	::=	変数 | 変数 補集合作用素２
特殊集合	::=	普遍集合 | 空集合
普遍集合	::=	1
空集合	::=	0 | ϕ
集合和作用素	::=	$\vee$ | $\cup$
集合積作用素	::=	$\wedge$ | $\cap$ | &
補集合作用素１	::=	$\sim$ | $\backslash$
補集合作用素２	::=	^{c}
対称差作用素	::=	++ | <> | $\oplus$
特殊集合制約	::=	setgb: 集合多項式 == 集合多項式
		setgb: 集合多項式 $\backslash$== 集合多項式

線形制約評価系利用手引書　第 1.4 版

(財) 新世代コンピュータ技術開発機構 (ICOT)

1990 年 4 月 4 日
1990 年 6 月 20 日
1990 年 9 月 12 日

目 次

1　はじめに

本手引書は、ＰＳＩ上のＣＡＬシステムにおいて、単体法による「線形不等式」のための制約評価系（以降、ＬＰＣＡＬと略す）を操作するための必要条項をまとめたものである。ＰＳＩ、ＳＩＭＰＯＳ、ＣＡＬについては、既知のものとする。

2　ＬＰＣＡＬについて

いくつかの変数に対する１次不等式と１次方程式が制約式として与えられたとき、その制約の下でそれらの変数に関する１次式（評価式）を最小化したり最大化したりする問題を、線形計画問題という。ＬＰＣＡＬは線形計画問題を解く評価系で、制約式が１つ入力されるとその式の冗長性、無矛盾性をチェックする。冗長であればその式を無視し、矛盾すればそこで失敗する。その他の場合には新しい制約として保持する。そして評価式が入力されれば、そのとき保持している制約のもとで単体法を実行する。ＬＰＣＡＬでは変数が負の値でも構わないので、逆に非負の場合は制約式の１つとして入力しなければならない。また、現在のＬＰＣＡＬでは、新しい制約式によって冗長になった古い制約式は制約を出力するときに除くようになっており、制約が入力される度に取り除くようにはなっていない。

3　ＬＰＣＡＬ操作手引

ＬＰＣＡＬは、ＣＡＬシステムの制約評価系の１つである。従って他の制約評価系と同様に、ＬＰＣＡＬを使う場合はＣＡＬシステムの中から評価系を呼び出すことになる。ＣＡＬシステムに関する詳細はＣＡＬ操作手引書を参照のこと。

3.1　ＬＰＣＡＬの準備

ＬＰＣＡＬを使用するには、ＣＡＬシステムの他に次のＥＳＰプログラムがカタログされていなければならない。

smplx.slv

3.2　ＣＡＬシステムの起動

ＣＡＬシステムの起動の要約は次のようになる。

起動：ＰＳＩのシステムメニューから、集合制約評価系を含むＣＡＬシステムをマウスにより選択する。次に、ウィンドウの左上隅と右下隅の位置をマウスにより選択すると、ＣＡＬシステムのウィンドウが開かれ、システムが起動される。

3.3 ユーザの入力

ＣＡＬシステムが起動されると、ウィンドウ内にプロンプト“？－ ”と、システムへの入力開始位置を示すマーク“「”が表示される。このマークからの入力を、トップレベルからの入力と呼ぶ。入力開始マーク以降で終止点“. ”を打って改行すると、開始マークから終止点までをシステムへの入力とみなす。終止点を打たずに改行すると、入力が続いているとみなされるので、数行にわたって１つの入力をすることもできる。

ＬＰＣＡＬにおいては、新しい問題を評価する前には“smplx:clear”を入力しなければならない。ＬＰＣＡＬは１つの入力の処理が終わっても制約を保存するので、制約を初期化するためにこのコマンドを入力することが必要である。しかしこの機能により、ある制約に対して幾つかの式の評価を続けて行うことが可能になっている。

ＣＡＬウィンドウはｐｍａｃｓウィンドウなので、ｐｍａｃｓの機能を使うことができる。従って、編集をしたり、日本語入力モードにして漢字をアトムとして入力することもできる。

入力には、ゴール列、ファイル名、コマンドの３種類がある。言語仕様については、４章およびＣＡＬ言語仕様書を参照のこと。

ゴール列 ： トップレベルから直接、制約式を入力することも、ファイルに書かれているゴールを呼び出すこともできる。ファイルに書かれているゴールを呼び出す場合は、関数子の前にソースファイル名を記号“：”で区切って付加する。ただし、この場合はあらかじめ次の『ファイル名の入力』を行っていなければならない。

(注) 制約式をトップレベルから入力する場合、変数名にアルファベット大文字で始まる文字列を使うには注意が必要である。例えば、制約式や評価式を幾つかに分けて入力する場合には、そのような変数名を異なる入力にまたがって使用しても、違うものとして処理される。また、そのような変数を含む式は出力されないので、結果の出力が欲しい変数名は小文字で始めなければならない。

ファイル名 ： ＣＡＬプログラムのファイルを作成したら、オブジェクトファイルに変換し、カタログしなければならない。また、プログラムのファイルを修正したり新たに作成する場合にも、この機能を使うことができる。この入力方法は、ＣＡＬ操作手引書を参照のこと。

コマンド ： 新しい問題を入力するときは、ＬＰＣＡＬが保持する制約を初期化しなければならない。また、評価式の入力や冗長な式の除去、保持している制約の出力をするために、以下のコマンドが用意されている。

smplx:clear

ＬＰＣＡＬが保持する制約を初期化する。ＬＰＣＡＬでは制約をクラス・スロットに保持しているため、ＣＡＬシステムを起動してＬＰＣＡＬを使用するとき、前の制約が消えずに残っている場合がある。そこで、システムを起動したときもこのコマンドを実行するほうがいい。

smplx:max(X)

smplx:min(X)

その時点でＬＰＣＡＬが保持する制約に対し、１次式 X の最大値または最小値を求め、その値およびその値を与える変数値を表示する。保持する制約は変更しない。

smplx:max(X, Xmax)

smplx:min(X, Xmin)

smplx:interval(X, [Xmin, Xmax])

ここで、X は目的関数、Xmax, Xmin は次のようなリストである。

Xmin, Xmax : [X=Value,Conditions]

Conditions は、目的関数 X が最大値 (最小値) をとるときの変数の値。ただし、Conditions としてはＬＰＣＡＬが保持する全ての線形等式制約を出力しているので、目的関数 X に含まれない変数の値やその他の線形等式制約が含まれている可能性があることに注意。保持する制約は変更しない。

smplx:cprint

その時点でＬＰＣＡＬが保持する制約から冗長な式を取り除き、表示する。

smplx:normalize

その時点でＬＰＣＡＬが保持する制約から冗長な式を取り除く。表示はしない。

smplx:positive(Variables)

smplx:negative(Variables)

線形計画問題では変数に非負条件が付いていることが多い。この２つのコマンドは、そのような非負条件、非正条件のついた変数のリストを Variables として、まとめて入力するためのものである。これらのコマンドは、他の制約式より後に入力した方が計算時間が短くなる。

smplx:constr(Constraints)

以下の線形制約、コマンドのリスト Constraints を、先頭から順に実行する。

X=Y, X=<Y, X>=Y, positive(X), negative(X), max(X,Xmax), min(X,Xmin), interval(X,[Xmin,Xmax]), max(X), min(X), normalize

smplx:get_result(ResultSet)

ＬＰＣＡＬが保持する制約集合を ResultSet とユニファイする。保持する制約は変更しない。ここで、ResultSet は以下の形式である。

ResultSet : [[変数 = 定数], [線形等式], [線形不等式]]

(注) 上のコマンドでは、ウィンドウに出力するときを除いて、値は整数 (32 bit 符号付き) または以下のような有理数の形をしている。

real(符号, 分子, 分母)

符号は + または −、分子、分母は 10 進整数を 4 桁ずつ区切って逆順に並べたリストである。

3.4 ＬＰＣＡＬの出力

ＬＰＣＡＬは単体法を用いているので、計算中にスラック変数を導入する。従って、スラック変数が出力に現れることがあり、ＬＰＣＡＬでは '$SLACK'(n)（nはスラック変数の番号を表す自然数）と表示する。スラック変数は非負であるが、特にその旨を表示はしない。

3.5 ＣＡＬシステムの終了

終了：ＣＡＬウィンドウのトップレベルから、" halt. "を入力する。これによりＣＡＬウィンドウは消滅し、システムは終了する。

4 ＬＰＣＡＬ言語仕様

ＬＰＣＡＬが扱う制約式は、以下のような構文規則に従う。アトム、変数は、ＣＡＬシステムに準ずる。

制約式	:: =	smplx: 線形多項式 = 線形多項式 |
		smplx: 線形多項式 =< 線形多項式 |
		smplx: 線形多項式 >= 線形多項式
線形多項式	:: =	単項 | 加減作用素 単項 |
		単項 加減作用素 線形多項式
単項	:: =	有理数 | 変数 |
		有理数 乗算作用素 単項 |
		単項 除算作用素 有理数
加減作用素	:: =	＋ | －
乗算作用素	:: =	*
除算作用素	:: =	／

5 実行例

例

```
?- smplx:clear,  smplx:y=< -x+5,  smplx:9>=x+3*y.

   18msec

 yes

?- smplx:max(x+2*y).

max = 7

 y = 2
 x = 3

   197msec

 yes

?- smplx:max(2*x+y).

max = +inf

   54msec

 yes

?- smplx:min(x+y).

min = -inf

   55msec

 yes

?- smplx:max(x+y).

max = 5

 y = -1/2*'$SLACK'(2)+2
 x = 1/2*'$SLACK'(2)+3

   170msec

 yes

?-
```

例

```
?- smplx:clear, smplx:y>=1, smplx:y=<1,
   smplx:cprint.

 y=1

  51msec

 yes

?- smplx:clear, smplx:y>=2, smplx:y=<1.

  9msec

 no

?- smplx:clear, smplx:x>=1, smplx:x>=2,
   smplx:cprint.

 x = '$SLACK'(2)+2

  69msec

 yes

?- smplx:clear, smplx:y=<x, smplx:y=<2-x,
   smplx:y>=0, smplx:interval(2*y+x, X).

X = [[2*y+x=0, [x=0, y=0]], [2*y+x=3, [x=1, y=1]]]

  201msec

 yes

?- smplx:clear, smplx:x=2876341/736413,
   smplx:y>=0, smplx:s=t, smplx:get_result(X).

X = [[x=real(+, [6341, 287], [6413, 73])], [s=t], [y>=0]]

  149msec

 yes

?-
```