

題 名	制約論理プログラミング言語実験システム
目 的	<ul style="list-style-type: none"> (1) より書きやすく、読みやすいプログラム (2) より抽象度の高いプログラミング (3) より効率の良い問題解決技法の研究
概要 及び 特徴	<ul style="list-style-type: none"> (1) 論理型言語の制約パラダイムの融合 (2) 複数制約評価系の共存 (3) 非線形代数方程式、およびブール方程式の求解機能 (4) Prolog の自然な拡張
構 成	<pre> graph TD User((ユーザ)) -- "プログラム / 問い合わせ / コマンド" --> PreProc[前処理] PreProc -- "内部表現" --> Inference[推論エンジン] Inference -- "回答" --> User Inference <--> "標準形" Constraints[制約評価] Constraints -- "制約" --> Inference </pre> <p>The diagram illustrates the system's architecture. It starts with a 'ユーザ' (User) circle on the left. An arrow points from the user to a '前処理' (Pre-processor) box, labeled 'プログラム / 問い合わせ / コマンド'. From the pre-processor, an arrow labeled '内部表現' (Internal representation) points to the '推論エンジン' (Inference engine) box. A feedback arrow labeled '回答' (Answer) points from the inference engine back to the user. Below the inference engine is the '制約評価' (Constraint evaluation) box, which is represented as a stack of three boxes. A double-headed arrow labeled '標準形' (Standard form) connects the inference engine and the constraint evaluation box. A single arrow labeled '制約' (Constraint) points from the constraint evaluation box up to the inference engine.</p>

代数 CAL の例

CAL ver0.1

```
?- ['ham.cal'].
"ham.obj" is generated.
```

yes

```
?- ham:horse_and_man(m,h,4,10).
m = 3 .
h = 1 .
```

yes

?- 「

(1) 鶴亀算

CAL ver0.1

```
?- ['lag.cal'].
"lag.obj" is generated.
```

yes

```
?- lag:ex(x^2+y^2,[x+y=a],[x,y]).
y = 1/2*a .
x = 1/2*a .
```

yes

?- 「

(2) 条件付き極値問題

```
:- public horseandman/4.
```

```
horseandman(Men, Horses, Heads, Legs) :-
    Heads = Men + Horses,
    Legs = 2*Men + 4*Horses.
```

```
:- public ex/3.
```

```
:- public lag/2.
```

```
ex(F, Constraint, Vars) :-
    lag(Constraint, Lag),
    difs(Vars, F, Lag).
```

```
lag([], 0) :- !.
```

```
lag([L=R|Cs], Mult*(L-R)+Lag) :-
    L=R:alg, !,
    lag(Cs, Lag).
```

```
difs([], _, _) :- !.
```

```
difs([Var|Vars], F, Lag) :- !,
    dif(F, Var)=dif(Lag, Var):alg, !,
    difs(Vars, F, Lag).
```

ブール CAL の例

```

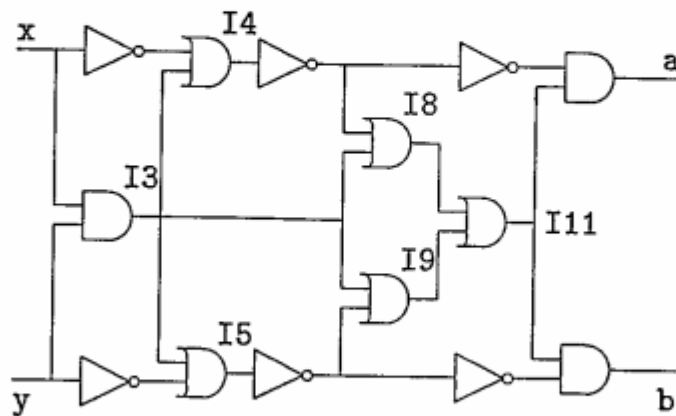
CAL ver0.1
┌─ [-'cross.cal'].
│ "cross.obj" is generated.
│
│ yes
│
│ ?- cross:cross(a,b,x,y).
│ y = a .
│ x = b .
│
│ yes
│ ?- ─
└─
    
```

(3) 交差回路

`:- public cross/4.`

`cross(X, Y, A, B) :-`

`I3 = X & Y:bool, I4 = ~X ∨ I3:bool, I5 = ~Y ∨ I3:bool,`
`I8 = ~I4 ∨ I3:bool, I9 = ~I5 ∨ I3:bool,`
`A = I4 & I11:bool, I11 = I8 ∨ I9:bool, B = I5 & I11:bool.`



```

CAL ver0.1
?- ['-count.cal'].
"count.obj" is generated.

yes
?- count:circuit(1,0,1,1,0,y1,y2,y3).
y1 = 0 .
y2 = 1 .
y3 = 1 .

yes
?- count:circuit(x1,x2,x3,x4,x5,1,0,1).
x1 = 1 .
x2 = 1 .
x3 = 1 .
x4 = 1 .
x5 = 1 .

yes
?- 「

```

(4) 入力の1の個数を数える回路

```
:- public circuit/8.
```

```

circuit(X1, X2, X3, X4, X5, Y1, Y2, Y3) :-
  I1=X1&X2:bool,   I2=X1\X2:bool,   I3=X3&X4:bool,
  I4=X3\X4:bool,  I5=~I1:bool,    I6=~I2:bool,
  I7=~I3:bool,    I8=~I4:bool,    I9=I1\I3:bool,
  I10=I1&I3:bool, I11=I6\I8:bool,  I12=I6&I8:bool,
  I13=~X5:bool,   I14=I5&I2:bool,  I15=I7&I4:bool,
  I16=~I14:bool,  I17=~I15:bool,   I18=I15\I16:bool,
  I19=I14\I17:bool, I20=I14\I15:bool, I21=I16\I17:bool,
  I22=I9&I4&I2&X5:bool,
  I23=I11&I7&I5&I13:bool,
  I24=X5&I18&I19:bool,
  I25=I13&I20&I21:bool,
  I26=I22\I10:bool, I27=I26\I23\I2:bool,
  Y1=I26:bool,     Y2=~I27:bool,     Y3=I24\I25:bool.

```

