2007年度 計算知能論A
制約充足 (constraint satisfaction)

上田 和紀
早稲田大学理工学部CS学科

有限領域上の制約充足問題

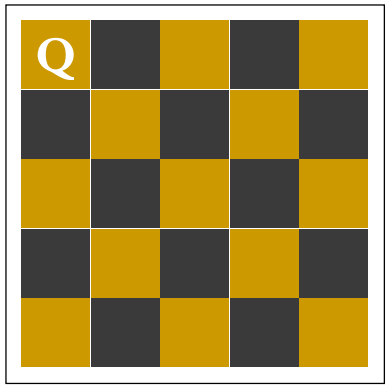
- 有限領域（finite domain）― 各未知数のとりうる値が有限種類であること
- 有限領域上の制約充足問題の例は数多い：
  - 覆面算，*n*-queens
  - 時間割作成，スケジューリング
  - 回路診断
  - 情景理解 (scene analysis)
- 実用上もきわめて重要
  - 制約充足のための処理系は高価でも売れている

有限領域上の制約充足問題の特徴

- 探索木の深さ ＝ 未知数の個数 ＝ 有限
- 制約の能動的な利用が重要
  - 代表テクニック：forward checking

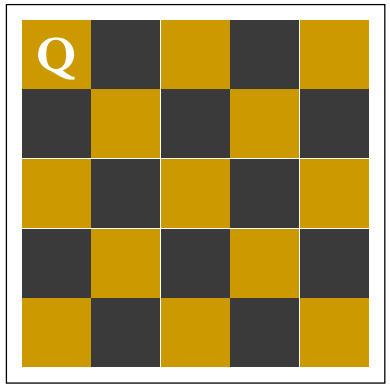
  cf. 制約の受動的な利用（２変数間の場合）：
  制約 $c(x_1, x_2)$ に x_1 と x_2 の両方の値を与えて yes か no かを答えてもらう
- 未知数の値を決定する順序も重要

Five-Queens Problem

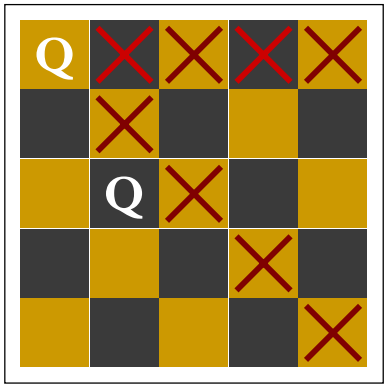


- 未知数：$x_1, \ldots, x_5$
- 領域： $1 \le x_i \le 5$
- 制約： $i \neq j$ ならば
  - $x_j \neq x_i$
  - $x_j \neq x_i + |j - i|$
  - $x_j \neq x_i - |j - i|$

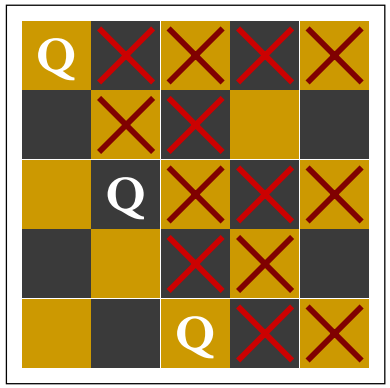
Five-Queens Problem, forward checking



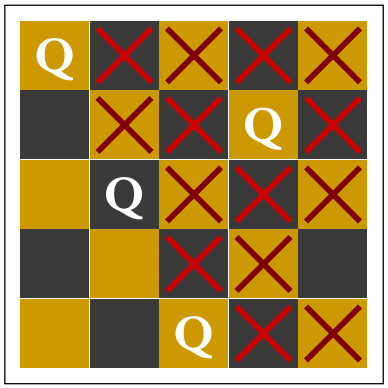
Five-Queens Problem, forward checking



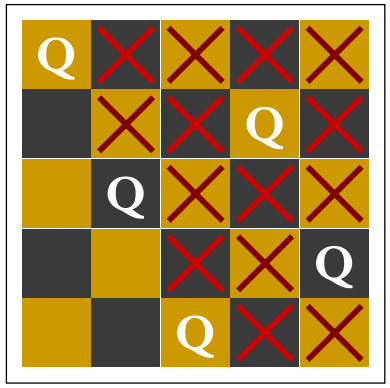
Five-Queens Problem, forward checking



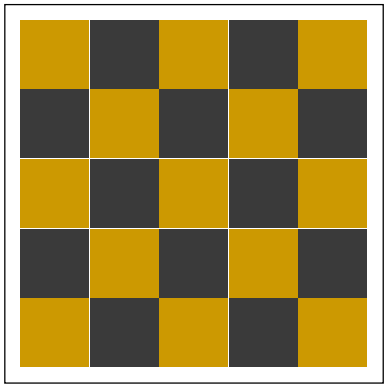
Five-Queens Problem, forward checking



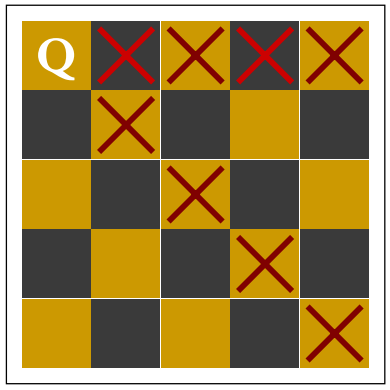
Five-Queens Problem, forward checking



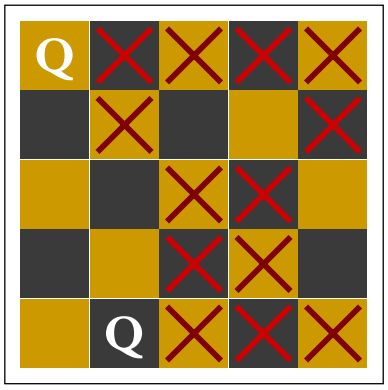
Five-Queens (2)



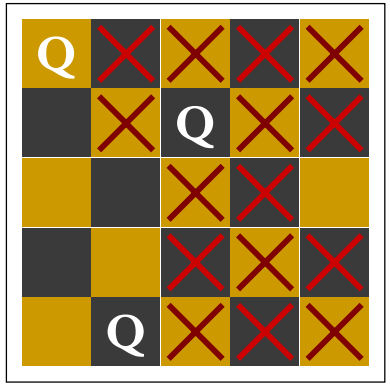
Five-Queens (2), forward checking



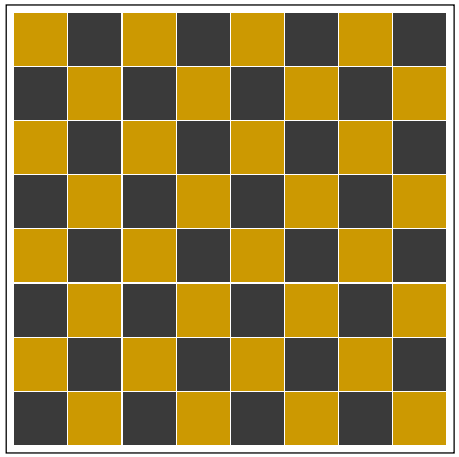
Five-Queens (2), forward checking



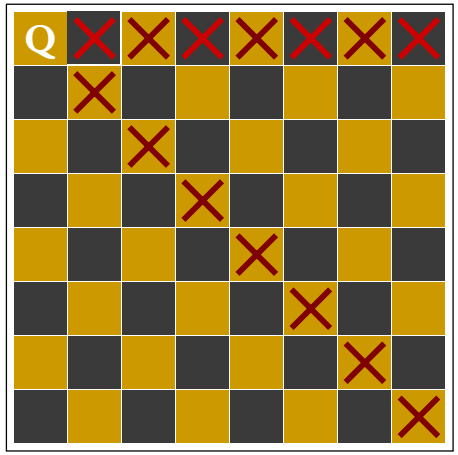
Five-Queens (2), forward checking



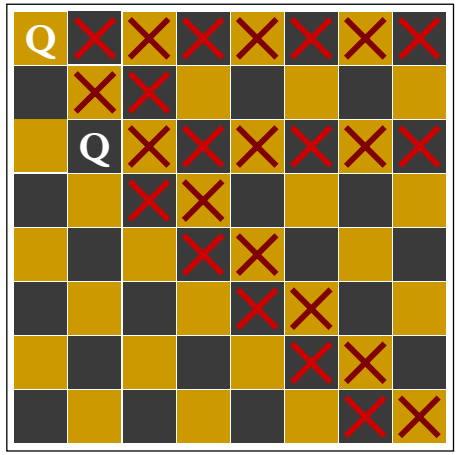
Eight-Queens, forward checking



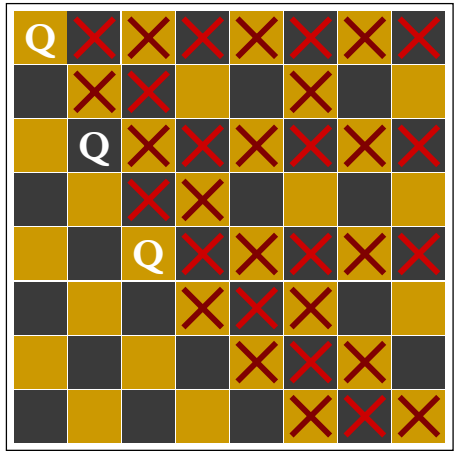
Eight-Queens, forward checking



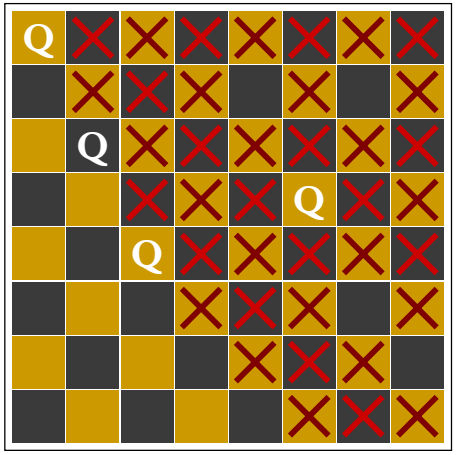
Eight-Queens, forward checking



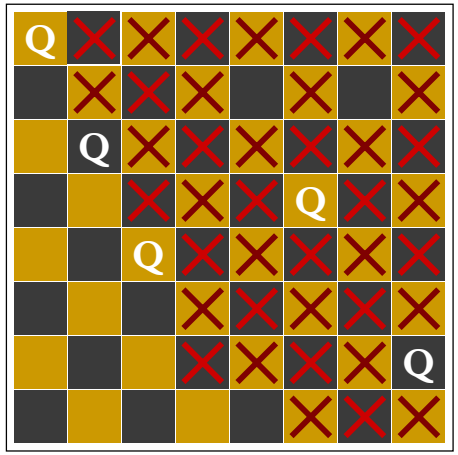
Eight-Queens, forward checking



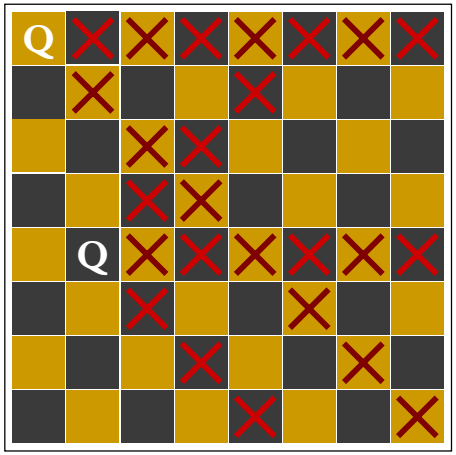
Eight-Queens, forward checking



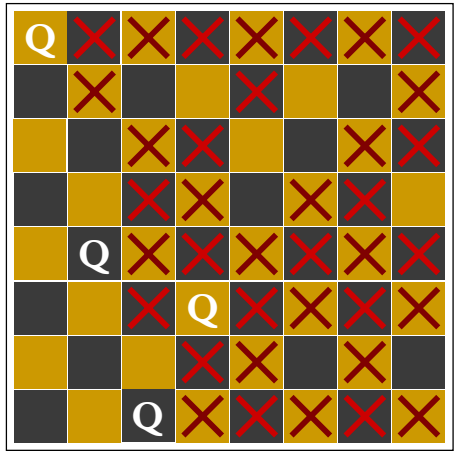
Eight-Queens, forward checking



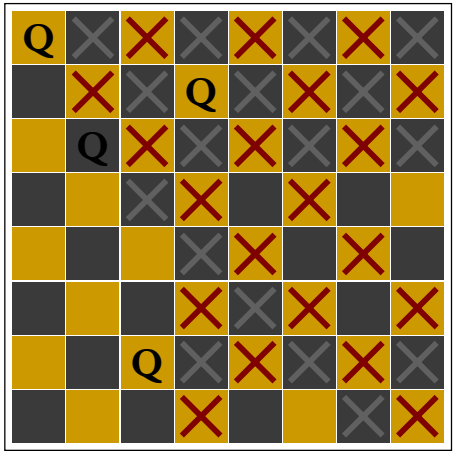
Eight-Queens, forward checking



Eight-Queens, forward checking



Eight-Queens, forward checking

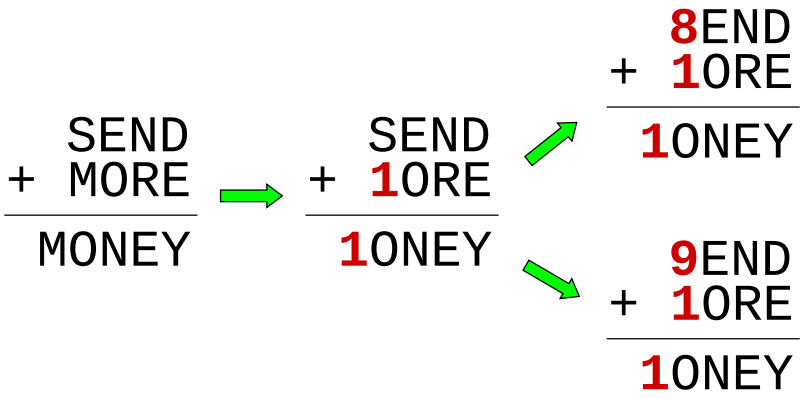


覆面算 (Cryptarithmetic)

$$\begin{array}{r} \text{SEND} \\ +\ \text{MORE} \\ \hline \text{MONEY} \end{array}$$

- 未知数と領域
  - $0 \le s, e, n, d, m, o, r, y \le 9$
  - $0 \le c_1, c_2, c_3, c_4 \le 1$
- 制約：
  - s, e, n, d, m, o, r, y は相異なる
  - $s \neq 0,\ m \neq 0$
  - $d + e = 10 \times c_1 + y$
  - $c_1 + n + r = 10 \times c_2 + e$
  - ...

覆面算の探索木

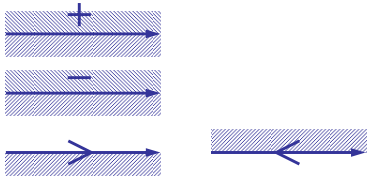


制約の能動的な利用

1. 未知数のうちの一つを選び，その値を仮に決める（guess）
2. 仮に決めた値を用いて制約の伝播（constraint propagation）を行い，他の未知数のとりうる値を限定する
   - ある未知数の取りうる値が一つもなくなったら 1. に後戻り（バックトラック）
   - さもなければ残った未知数について 1. から作業を行う

線画の解釈

- ２次元情報から３次元情報を復元する (!)
- 求めるもの：線画の各辺の役割（ラベル）
  - 凸辺
  - 凹辺
  - 輪郭辺（２種類）

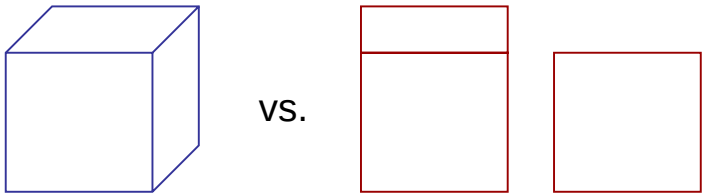


- 制約条件：複数の線が会する箇所での，可能なラベルの組合せ（プリントの図）
- 辺が n 本あると，場合の数は 4^n？
  - cf. 人間は非常に効率良く線画を認識できる

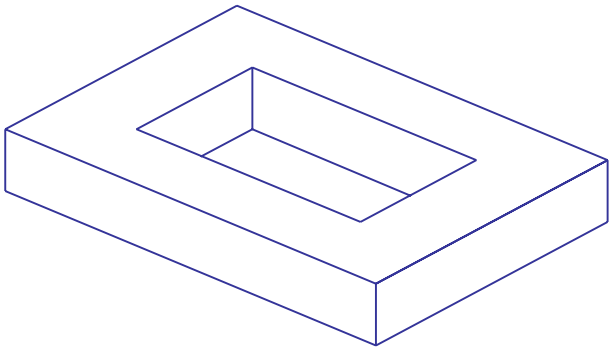
線画の解釈

- 失った情報を推定するためには仮定が必要
  - 正則多面体，かつ三面頂点多面体
  - 多面体の全体が描かれている
  - 影やひびは描かれていない
  - 一般の視点

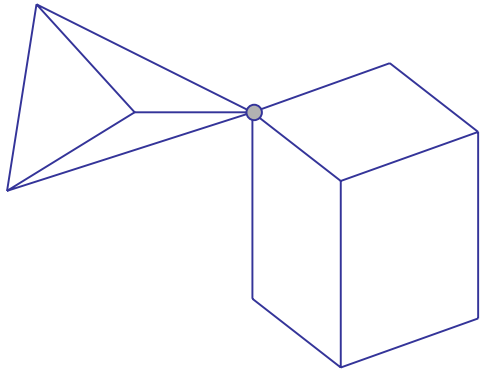
（一般でない視点）



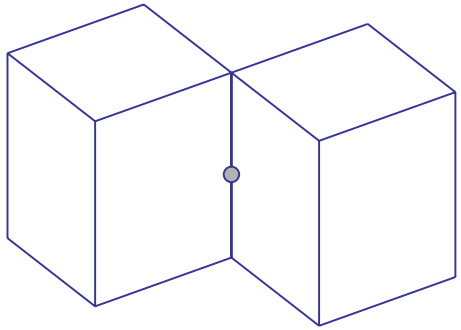
正則多面体



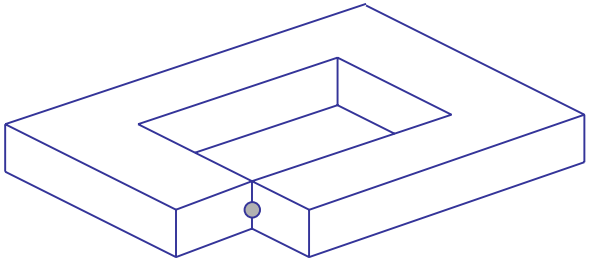
正則でない多面体 (1)



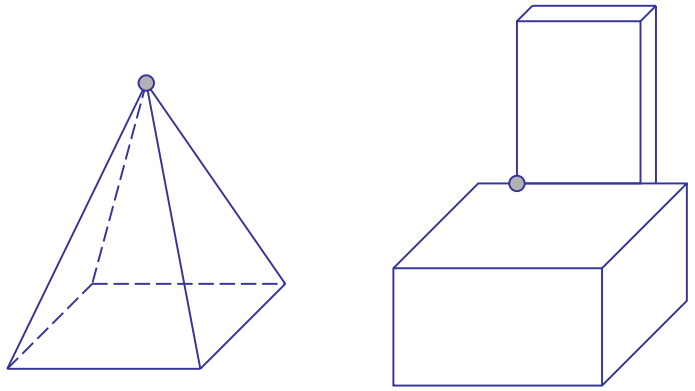
正則でない多面体 (2)



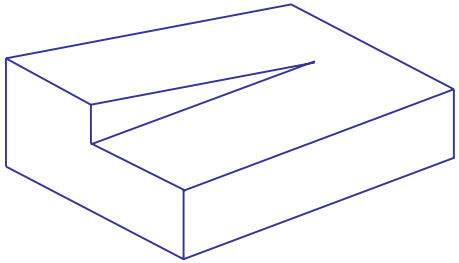
正則でない多面体 (3)



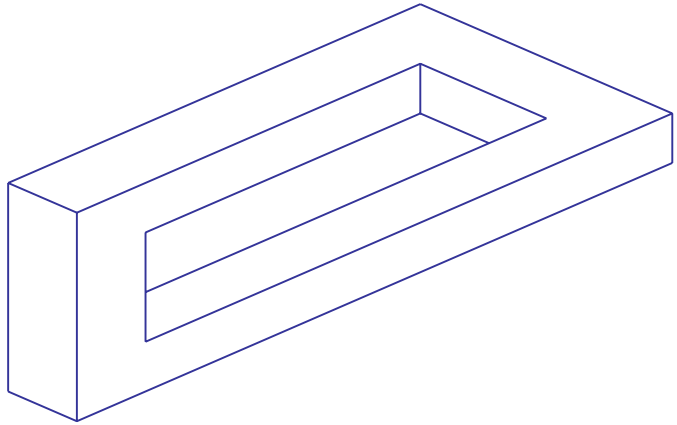
三面頂点多面体でない正則多面体



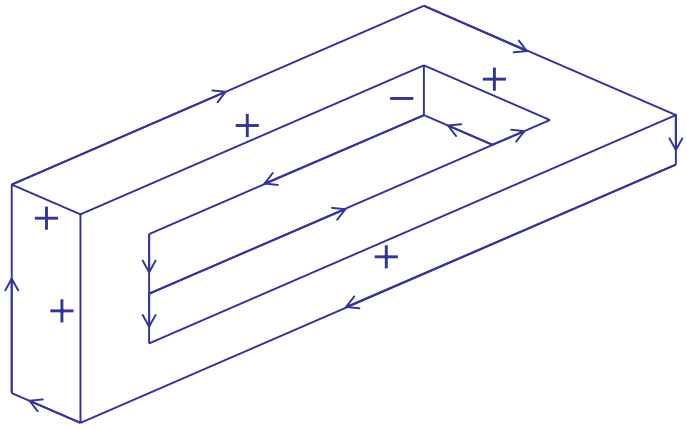
“不可能物体”タイプ１



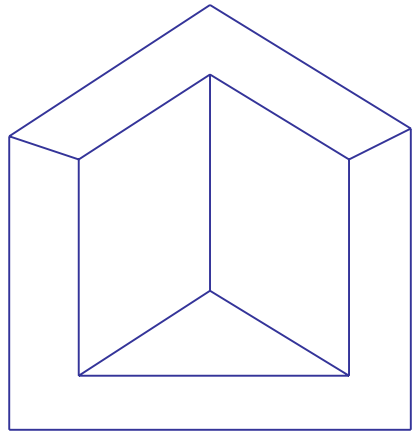
“不可能物体”タイプ２



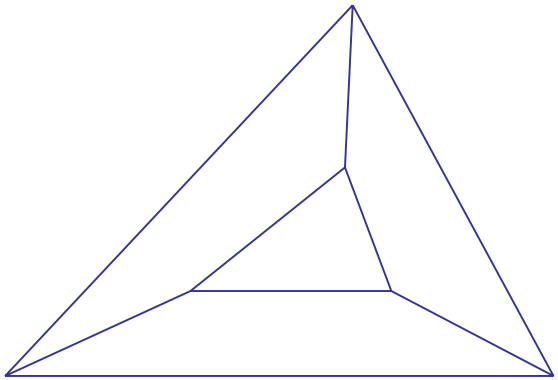
“不可能物体”タイプ２



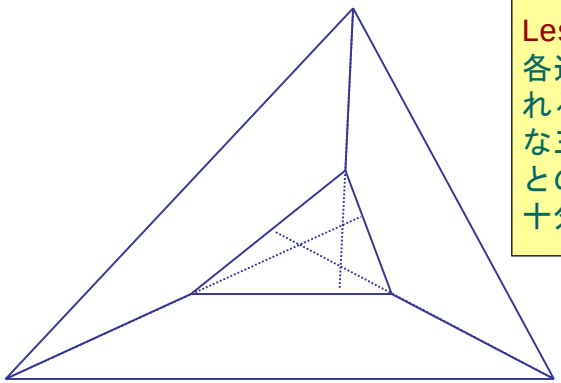
“不可能物体”タイプ２



“不可能物体”タイプ２



“不可能物体”タイプ２



Lesson:
各辺にラベルがつけられることは，実現可能な三次元物体であることの必要条件であるが十分条件ではない。

制約プログラミング用ライブラリ

- SWI Prolog, SICStus Prolog, GNU Prolog などに付属
- 下記の形で問題を記述
  - 各未知数のとりうる値の領域を宣言 (次頁の `in`)
  - 各未知数が満たすべき制約をポスト (cf. 実行) (次頁の `#` つき制約と `all_different`)
    - 制約の検査と伝播のためのネットワークが張られる
  - 可能な値の組合せを，guess と制約伝播とバックトラックで探索 (次頁の `label`)

覆面算 (SWI Prolog with ‘clp/bounds’ library)

```
:- use_module(library('clp/bounds')).
send([[S,E,N,D], [M,O,R,E], [M,O,N,E,Y]]) :-
      Digits   =  [S,E,N,D,M,O,R,Y],
      Carries  =  [C1,C2,C3,C4],
      Digits  in  0..9,
      Carries in  0..1,
      M           #=         C4,
      O + 10 * C4 #= M + S + C3,
      N + 10 * C3 #= O + E + C2,
      E + 10 * C2 #= R + N + C1,
      Y + 10 * C1 #= E + D,
      M #>= 1,
      S #>= 1,
      all_different(Digits),
      label(Digits).
```

領域宣言

制約のポスト

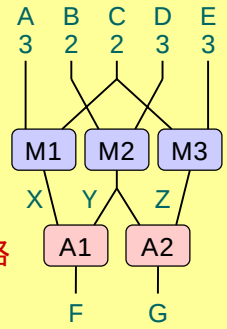
← 探索開始

回路故障診断 (SWI Prolog with 'clp/bounds' library)

```
:- use_module(library('clp/bounds')).

add(1,A,B,C) :-  C #= A+B.
add(0,A,B,C) :-  C #¥= A+B.
mul(1,A,B,C) :-  C #= A*B.
mul(0,A,B,C) :-  C #¥= A*B.

circuit([M1,M2,M3,A1,A2],F,G):-
  A=3, B=2, C=2, D=3, E=3,
  mul(M1,A,C,X), mul(M2,B,D,Y),
  mul(M3,C,E,Z),
  add(A1,X,Y,F), add(A2,Y,Z,G).

sum([],Sum) :- Sum#=0.
sum([S|Ss],Sum) :- sum(Ss,Sum0), Sum#=S+Sum0.
```

素子

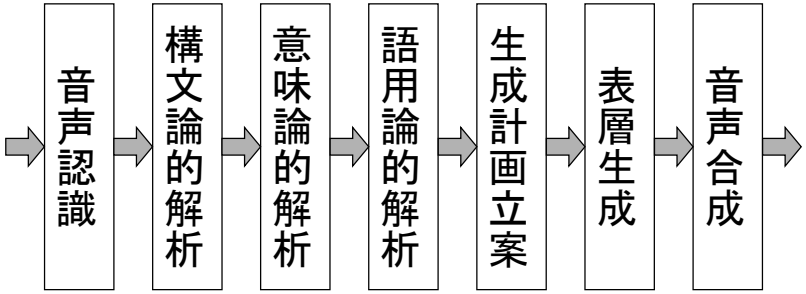
回路



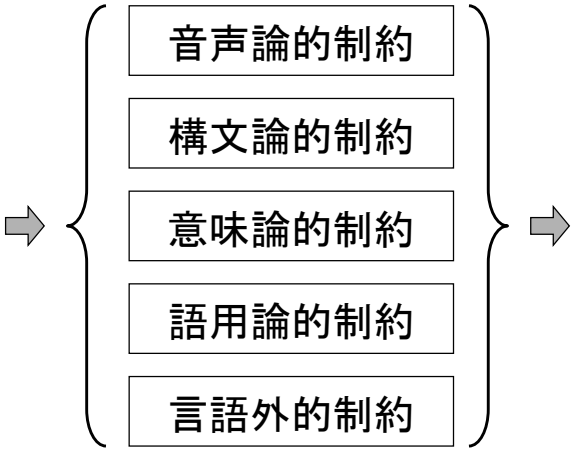
診断 (diagnosis) 問題

- 例：回路の故障診断
- 与えるもの：
  - 各素子の正常時の動作
  - 回路の構成
  - 回路への入力
  - 観測された出力
- 以上を与えると，診断システムは故障箇所についての仮説を計算する（仮説推論）

直列型自然言語処理



並列型自然言語処理



制約の下での談話理解

- 会話１
  - “What are they?”
  - “They are flying airplanes.”
- 会話２
  - “What are they *doing*?”
  - “They are flying airplanes.”

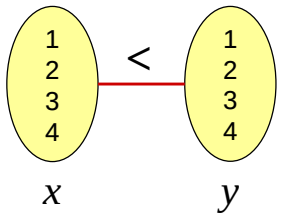
制約の下での文字認識

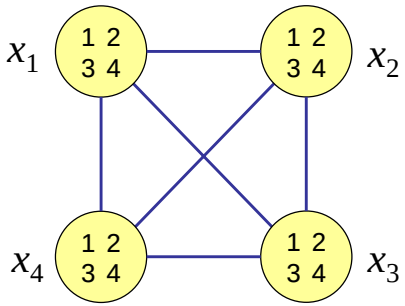
THE CAT

制約の下での文字認識

制約伝播アルゴリズム

- すべての制約が２変数間の制約である場合，制約充足問題は制約ネットワークで表現できる





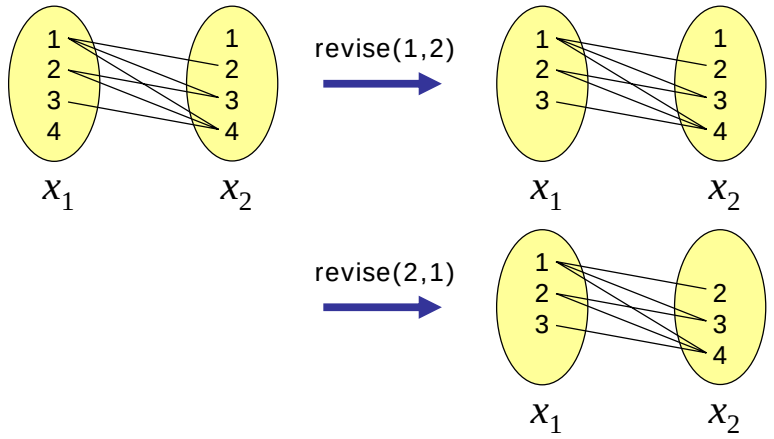
$x_i \neq x_j,\ x_j \neq x_i \pm (j-i)$

アーク整合性 (arc consistency)

- 制約充足問題
  - 変数（未知数） $x_1,\ldots, x_n$
  - 各変数の領域 $D_1,\ldots, D_n$
  - 制約の集合 $\{C_{ij}\}$
- アーク C_{ij} がアーク整合している：
  - $\forall a_i \in D_i\ \exists a_j \in D_j\ (C_{ij}(a_i, a_j))$ かつ
  - $\forall a_j \in D_j\ \exists a_i \in D_i\ (C_{ij}(a_i, a_j))$
- 制約ネットワークがアーク整合：
  - すべてのアークがアーク整合している

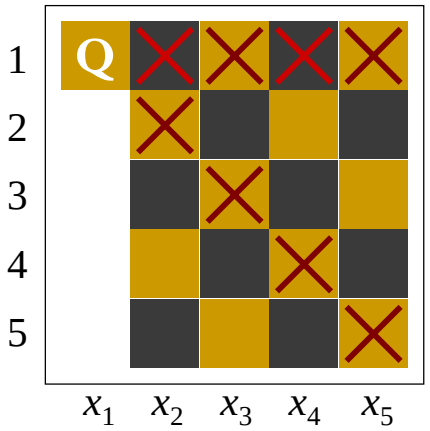
アーク整合性

- $x_1 < x_2$ $(x_i \in \{1,2,3,4\})$ の場合



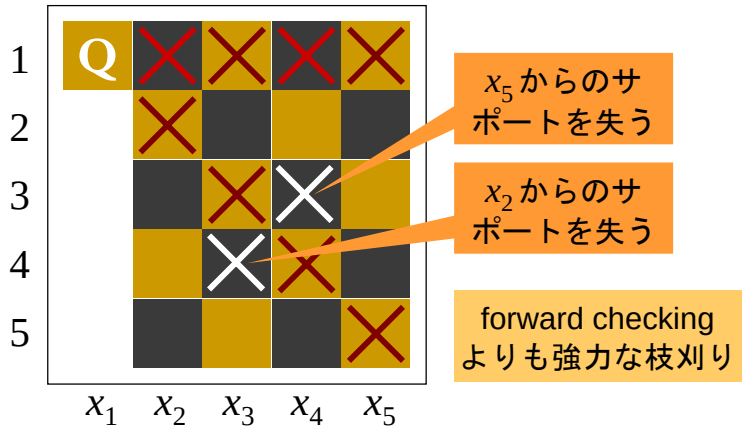
アーク整合性

- $x_i \neq x_j$, $x_j \neq x_i \pm (j-i)$ $(x_1 \in \{1\}, x_{2\ldots5} \in \{1,2,3,4,5\})$



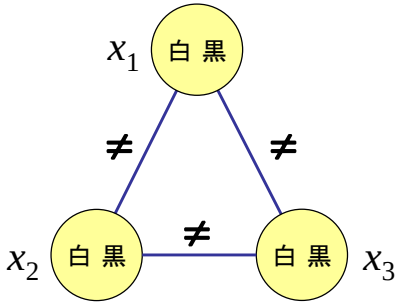
アーク整合性

- $x_i \neq x_j$, $x_j \neq x_i \pm (j-i)$ $(x_1 \in \{1\}, x_{2\ldots5} \in \{1,2,3,4,5\})$



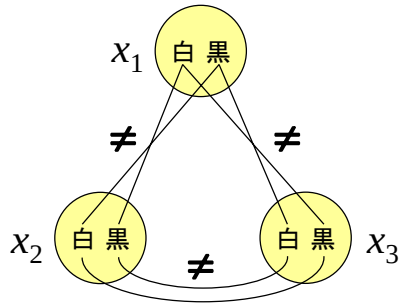
アーク整合性

- アーク整合性を保証できても解があるとは限らない



アーク整合性

- アーク整合性を保証できても解があるとは限らない



- より高度な整合性（パス整合性）を検査するか，最後はしらみつぶし探索を行って具体的な組合せを求める