最適化 2 (第 3 回)

番原睦則 (banbara@i.nagoya-u.ac.jp)

名古屋大学情報学研究科

SAT と SAT ソルバー

SAT とは

SAT (Boolean satisfiability testing)

与えられた命題論理式を充足する (真にする) 値割当てが存在するかどうかを判定する問題

- 2000 年頃から，SAT 問題を解く **SAT ソルバー**の性能が大幅に向上した.
- 問題を SAT に**符号化** (翻訳) し SAT ソルバーを用いて求解する **SAT 型システム**が様々な分野で成功している.

- Intel core I7 プロセッサ設計 [Kaivola+, CAV 2009]
- Windows 7 デバイス・ドライバー検証 [De Moura and Bjorner, IJCAR 2010] (SMT ソルバー Z3)
- ソフトウェア要素の依存性解析
  - Eclipse [Le Berre and Rapicault, IWOCE 2009]
  - Fedora Linux (および RedHat, CentOS) の dnf

SAT 問題 (SAT Instances)

具体的な SAT の問題は，連言標準形 (CNF) で与えられる．

CNF 式

- **CNF 式**は，複数の節の連言である．
- **節** (clause) は複数のリテラルの選言である．
- **リテラル** (literal) は，命題変数かあるいはその否定である．

標準的フォーマットとしては DIMACS CNF が用いられる．

```
p cnf 3 4       ; Number of variables and clauses
1 2 3 0         ; p1 ∨ p2 ∨ p3
-1 -2 0         ; ¬p1 ∨ ¬p2
-1 -3 0         ; ¬p1 ∨ ¬p3
-2 -3 0         ; ¬p2 ∨ ¬p3
```

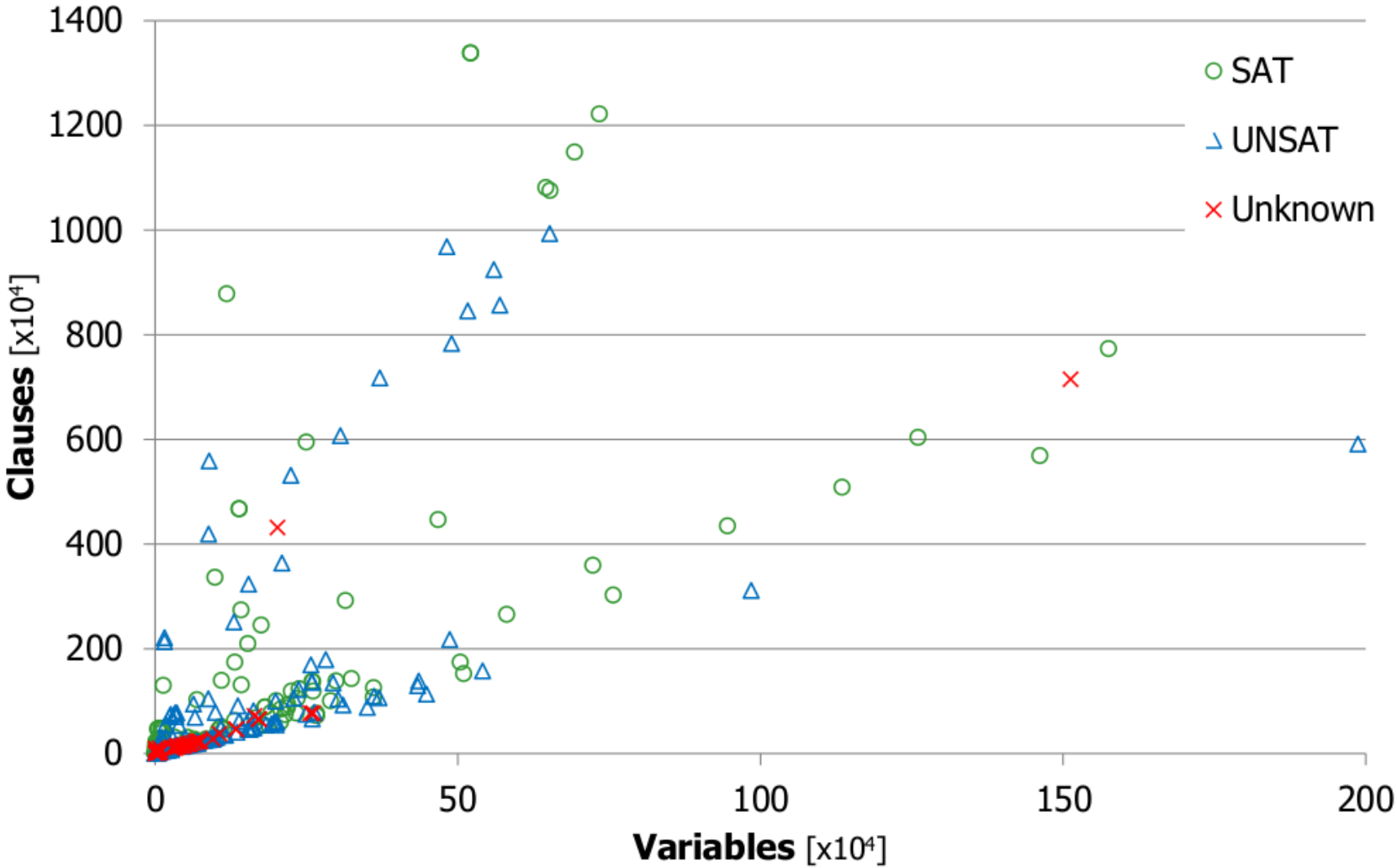
SAT ソルバー (SAT Solvers)

SAT ソルバー

SAT ソルバーは，与えられた SAT 問題が充足可能 (SAT) か充足不能 (UNSAT) かを判定するプログラム．充足可能であればその値割当てを解として出力する．

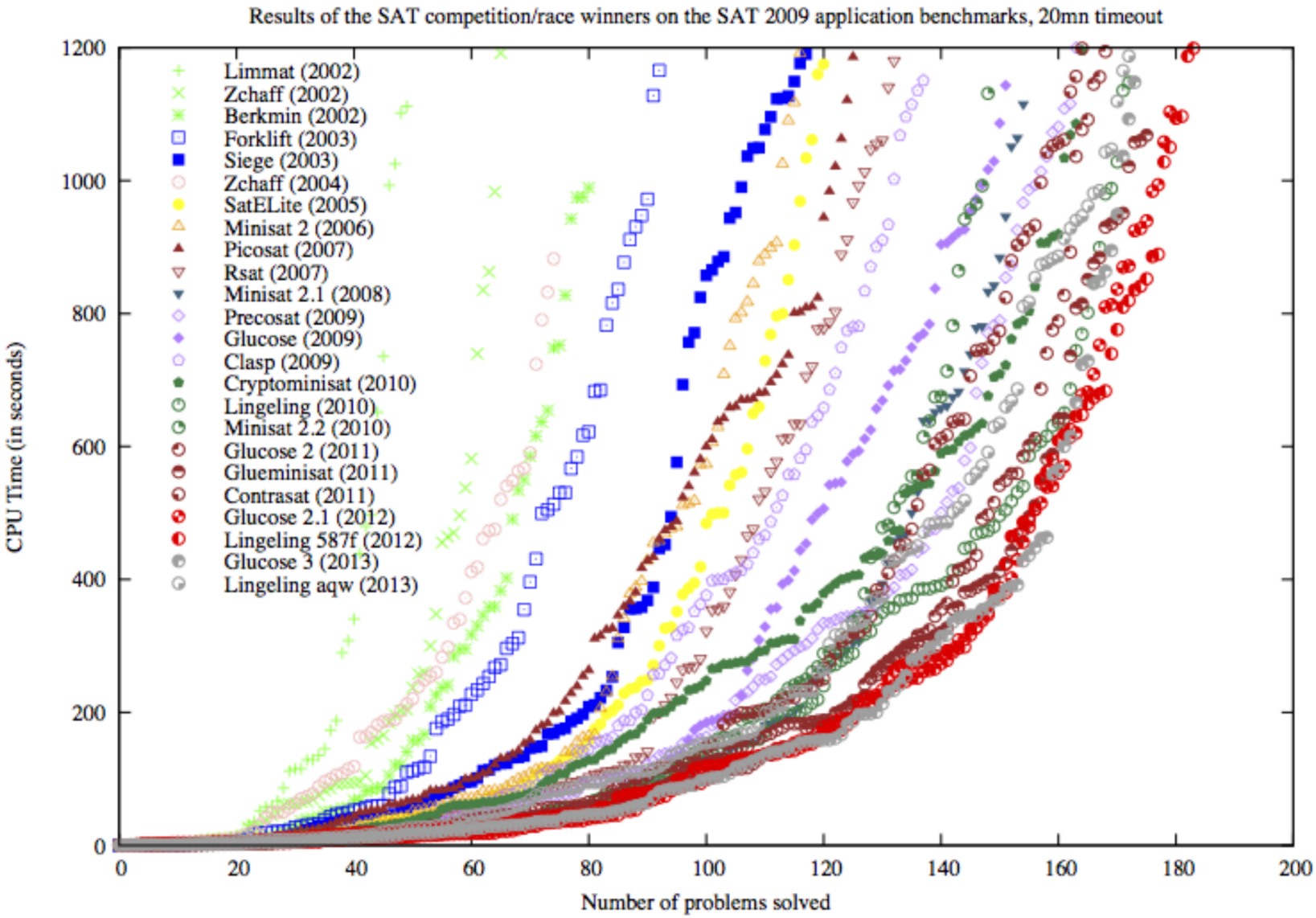
- **近代的 SAT ソルバー**では，DPLL アルゴリズムに様々な技術が導入され大幅な性能向上が実現されている．
  - 矛盾からの節学習 (CDCL), 非時間順バックトラック法, ランダムリスタート, 監視リテラル, 変数選択ヒューリスティック (VSIDS)，リテラルブロック距離 (LBD)
- 近代的 SAT ソルバーは 10^6 個の命題変数 10^7 個の節を含む SAT 問題も取り扱うことができる．
- 2002 年以降**国際 SAT 競技会**が開催され，SAT ソルバーの実装技術の向上に貢献している．オープンソースが義務．

SAT 問題のサイズ



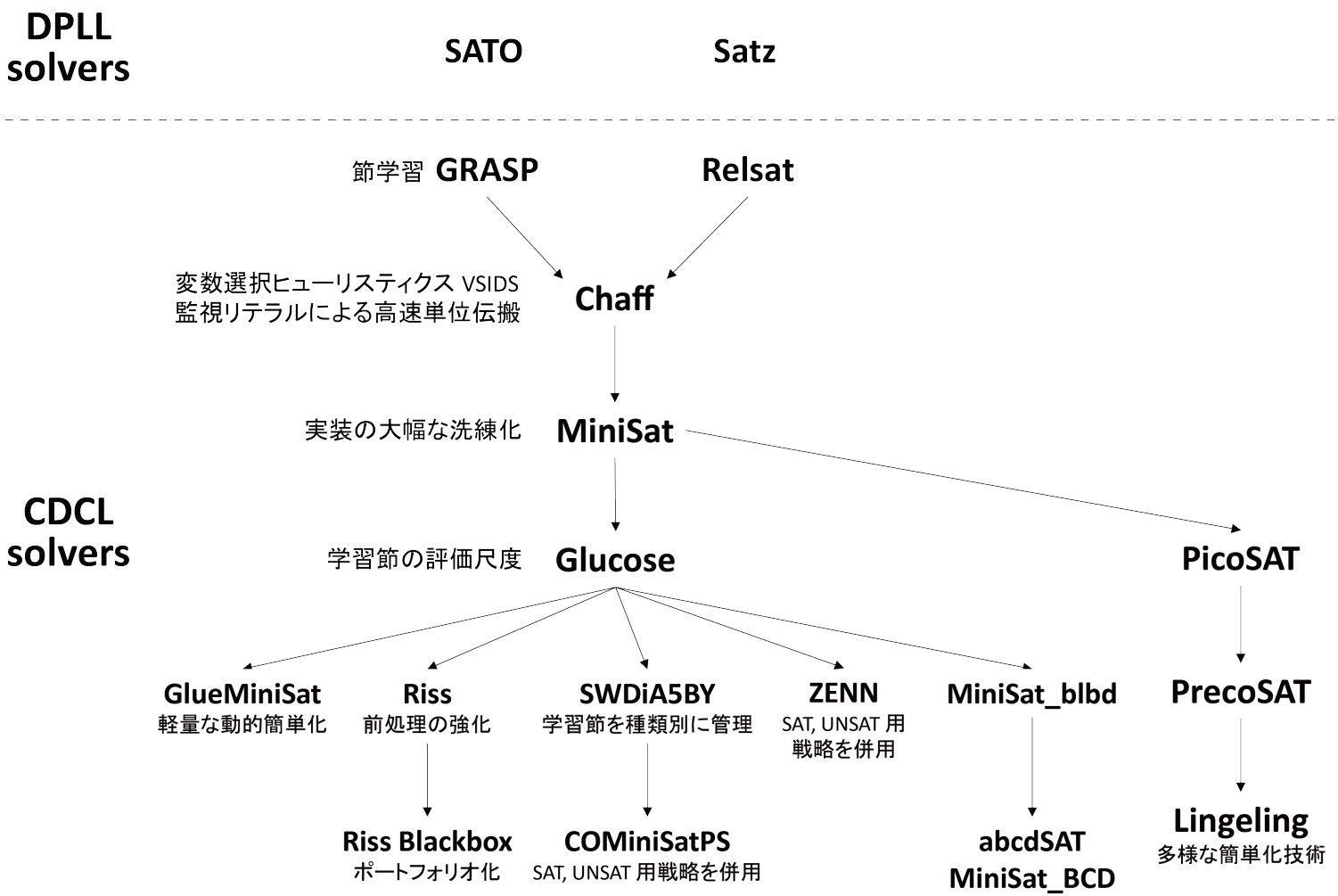
2011 SAT competition, Applications Track

SAT ソルバーの速度向上



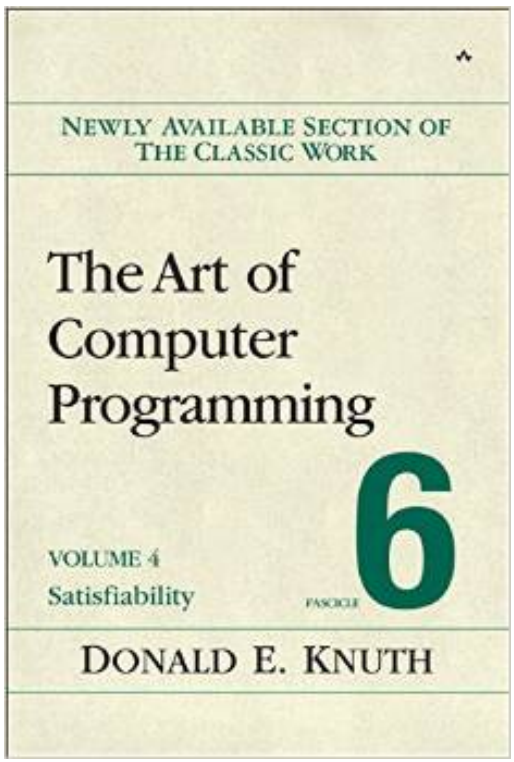
SAT ソルバーの系統樹

歴史的 SAT ソルバーと国際 SAT 競技会 (2013〜2015 年) の上位ソルバー



Knuth: “Satisfiability”, TAOCP, 4 巻, 第 6 分冊

Knuth の有名な教科書 TAOCP の最新分冊は SAT に関する章



Wow—Section 7.2.2.2 has turned out to be the longest section, by far, in The Art of Computer Programming. The SAT problem is evidently a **killer app**, *because it is key to the solution of so many other problems.*

Satisfiability is **important** *chiefly because Boolean algebra is so versatile. Almost any problem can be formulated in terms of basic logical operations, . . .*

The story of satisfiability is the tale of a **triumph** *of software engineering, blended with rich doses of beautiful mathematics.*

Section 7.2.2 explains how such a **miracle** *occurred, . . .*

Knuth: “Satisfiability” 中の例題

Knuth の教科書は 300 ページ以上で 500 問以上の練習問題を含んでいる．最初の 20 ページ余りで以下の例題が紹介されている．

1. ファンデアベルデン数
2. 厳密被覆
3. グラフ彩色
4. 整数の因数分解
5. 回路の故障検査 (32 ビット乗算回路)
6. ブール関数の学習
7. 有界モデル検査 (ライフゲーム)
8. 相互排除アルゴリズムの検証
9. デジタル断層写真

SAT ソルバーの進歩 (個人的見解を含む)

- **様々な探索技術**の導入
  - 矛盾した原因を節として学習 (CDCL) するなど
- **実装技術の進歩**.
  - SAT 競技会の優勝ソルバーはどんどん入れ替わっている.
- **キャシュを意識**した実装 [Zhang & Malik 2003]
  - 例えば，ある 260 万節の SAT 問題は，MiniSat で 4 秒以内で求解でき L2 キャッシュへのヒットレートは 99%以上である.

```
$ valgrind --tool=cachegrind minisat gp10-10-1091.cnf
L2 refs:       42,842,531  ( 31,633,380 rd +11,209,151 wr)
L2 misses:     25,674,308  ( 19,729,255 rd + 5,945,053 wr)
L2 miss rate:         0.4% (        0.4%   +       1.0%  )
```

主な SAT ソルバー

- **MiniSat** [Eén and Sörensson 2003]
  - `http://minisat.se`
- **Glucose** [Audemard and Simon 2009]
  - `http://www.labri.fr/perso/lsimon/glucose/`
- **Lingeling** [Biere 2010]
  - `http://fmv.jku.at/lingeling/`
- **GlueMiniSat** [Nabeshima+ 2011]
  - `http://glueminisat.nabelab.org`
- **Clasp** [Gebser+ 2007]
  - `https://potassco.org/clasp/`
- **Sat4j** [Le Berre 2010]
  - `http://www.sat4j.org`

参考資料 (1/2)

- **特集「SAT 技術の進化と応用〜パズルからプログラム検証まで〜」**
  情報処理学会会誌「情報処理」57 巻 8 号 (2016 年 8 月号).
  - SAT 技術の進化，SAT ソルバーの最近の進展，SAT とパズル，SAT と AI，SAT とラムゼー数，MaxSAT：SAT の最適化問題への拡張，SMT ソルバーによるプログラム検証
- **特集「最近の SAT 技術の発展」**, 人工知能学会誌 25 巻 1 号 (2010)
  - SAT ソルバーの基礎，高速 SAT ソルバーの原理，制約最適化問題と SAT 符号化，SMT：個別理論を取り扱う SAT 技術，モデル列挙とモデル計数，＊-SAT: SAT の拡張，SAT によるプランニングとスケジューリング，SAT によるシステム検証

参考資料 (2/2)

- Donald E. Knuth: Satisfiability, The Art of Computer Programming, Vol.4, Fascicle 6, 2015.
- Handbook of Satisfiability, IOS Press, 2009.
- The international SAT Competitions
  - http://www.satcompetition.org
- SAT/SMT Summer School
  - http://satassociation.org/sat-smt-school.html

制約プログラミング

SAT 技術の広がり

問題を SAT に翻訳し SAT ソルバーを用いて解く手法が様々な分野で成功し，SAT 技術が大きな広がりを見せている．

- 有界モデル検査 [Biere, 2009]
- Intel core I7 プロセッサ設計 [Kaivola+, CAV 2009]
- Windows 7 デバイス・ドライバー検証
  [De Moura+, IJCAR 2010] (SMT ソルバー Z3)
- ソフトウェア要素の依存性解析
  - Eclipse [Le Berre and Rapicault, IWOCE 2009]
  - Fedora Linux (および RedHat, CentOS) の dnf
- プランニング (SATPLAN, Blackbox) [Kautz+, 1992]
- ショップ・スケジューリング [Crawford+, 1994; 田村+, 2009]
- **制約プログラミング** (Sugar) [田村+, 2009]
- 解集合プログラミング (clingo) [Gebser+, 2012]
- 多目的離散最適化問題 (sucre) [宋+, CP 2017]

制約プログラミング (Constraint Programming; CP)

CP とは

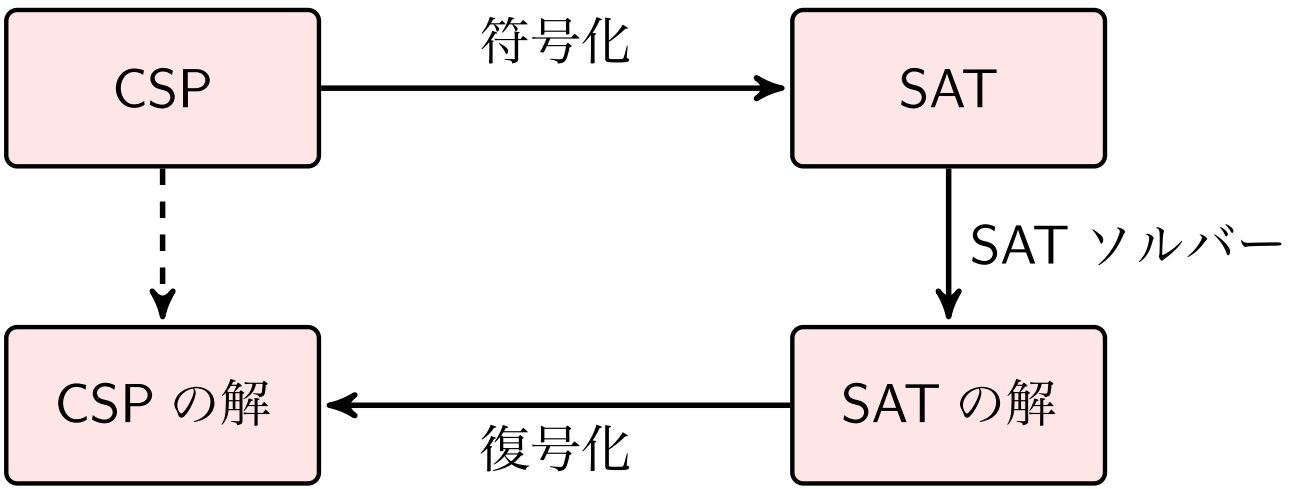
解きたい問題を**制約**として宣言的にプログラム中に記述するプログラミング・パラダイム.

- 問題を解くアルゴリズムを記述する必要はなく，処理系が解を探索してくれる.
- CP の言語は**制約充足問題 (CSP)** と呼ばれる.
- 制約充足問題を解くプログラムは**制約ソルバー**（あるいは CSP ソルバー) と呼ばれ，様々なソルバーが開発されている.

- Abscon (Java)
- Choco (Java)
- Copris (Scala, 神戸大学), **Sugar** (Java, 神戸大学)
- IBM ILOG CP Optimizer (C++)
- Mistral (C++)

SAT 型制約ソルバー Sugar [田村ほか,2006 ▸ Web]

Sugar は，制約充足問題 (CSP) を命題論理式に符号化し，SAT ソルバーを用いて求解を行う SAT 型の制約ソルバー

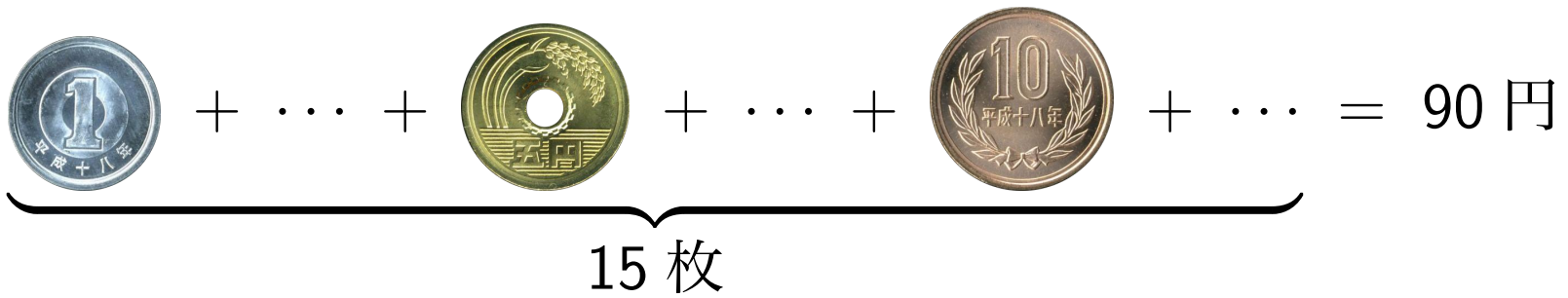


- **順序符号化法** [田村ほか,2006] と呼ばれる SAT 符号化法を使用.
- 制約最適化問題 (COP)，最大制約充足問題 (Max-CSP) にも対応.
- Java, Perl, MiniSat 等を別途インストールしておく必要あり.

制約充足問題の例: コインの問題

コインの問題例

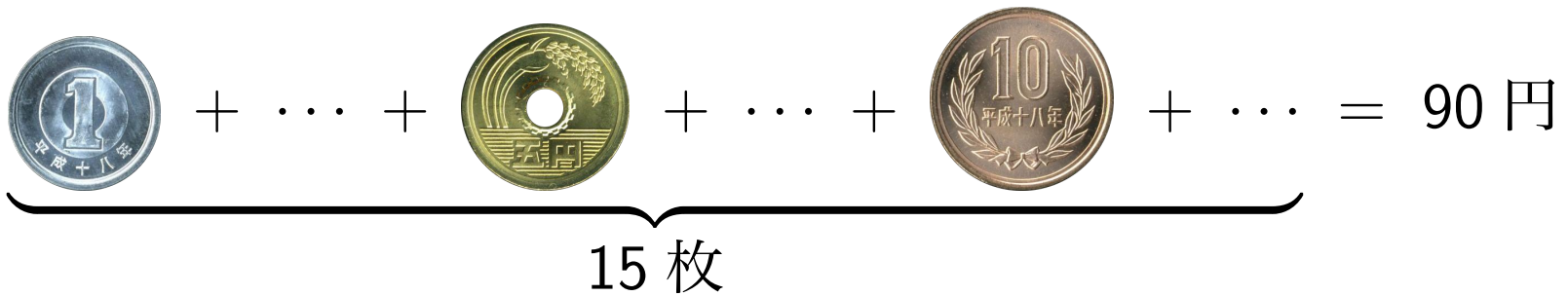
1 円硬貨，5 円硬貨，10 円硬貨が各 1 枚以上，合計で 15 枚ある．金額の合計は 90 円である．それぞれの硬貨は何枚か?



制約充足問題の例: コインの問題

コインの問題例

1 円硬貨，5 円硬貨，10 円硬貨が各 1 枚以上，合計で 15 枚ある．金額の合計は 90 円である．それぞれの硬貨は何枚か?



制約充足問題 (CSP)

$$
\begin{aligned}
x, y, z &\in \{1, 2, \ldots, 15\} \\
x + y + z &= 15 \\
x + 5y + 10z &= 90
\end{aligned}
$$

コインの問題例を Sugar で解く

Sugar への入力ファイル coins.csp

```
(int x 1 15)
(int y 1 15)
(int z 1 15)
(= (+ x y z) 15)
(= (+ x (* 5 y) (* 10 z)) 90)
```

1. 上のファイル `coins.csp` を作成する.
2. Sugar を実行する.

   ```
   $ sugar coins.csp
   ```

3. 以下が出力される. 答え: 1 円 5 枚, 5 円 3 枚, 10 円 7 枚.

   ```
   s SATISFIABLE
   a x 5
   a y 3
   a z 7
   ```

制約充足問題の例: ナップサック問題

ナップサック問題の例

5 つの品物のコストと利益が与えられている．コストの合計が 15 以下で，利益の合計が 18 以上になる場合はあるか?

品物	コスト	利益
品物 1	3	4
品物 2	4	5
品物 3	5	6
品物 4	7	8
品物 5	9	10

制約充足問題の例: ナップサック問題

ナップサック問題の例

5 つの品物のコストと利益が与えられている．コストの合計が 15 以下で，利益の合計が 18 以上になる場合はあるか?

品物	コスト	利益
品物 1	3	4
品物 2	4	5
品物 3	5	6
品物 4	7	8
品物 5	9	10

制約充足問題 (CSP)

$$
\begin{aligned}
p_1, p_2, p_3, p_4, p_5 &\in \{0, 1\} \\
3p_1 + 4p_2 + 5p_3 + 7p_4 + 9p_5 &\leq 15 \\
4p_1 + 5p_2 + 6p_3 + 8p_4 + 10p_5 &\geq 18
\end{aligned}
$$

ナップサック問題の例を Sugar で解く

Sugar への入力ファイル knapsack.csp

```
(int p_1 0 1) (int p_2 0 1)
(int p_3 0 1) (int p_4 0 1) (int p_5 0 1)
(<= (+ (* 3 p_1) (* 4 p_2) (* 5 p_3) (* 7 p_4) (* 9 p_5)) 15)
(>= (+ (* 4 p_1) (* 5 p_2) (* 6 p_3) (* 8 p_4) (* 10 p_5)) 18)
```

1. 上のファイル knapsack.csp を作成する.
2. Sugar を実行する.

   ```
   $ sugar knapsack.csp
   ```

3. 以下が出力される. 答え: 品物 1,3,4 の組合せがある.

   ```
   s SATISFIABLE
   a p_1 1
   a p_2 0
   a p_3 1
   a p_4 1
   a p_5 0
   ```

オープンショップ・スケジューリング (OSS) 問題

- OSS は n 個のジョブと n 個のマシンから成る.
  - $J_0, J_1, \ldots, J_{n-1}$
  - $M_0, M_1, \ldots, M_{n-1}$
- 各ジョブ J_i は n 個のオペレーションから成る.
  - $O_{i0}, O_{i1}, \ldots, O_{i(n-1)}$
- ジョブ J_i のオペレーション O_{ij} は，マシン M_j で処理され，正の処理時間 p_{ij} を要する.
- 同じジョブに対するオペレーションは同時には処理できない.
- 各マシン M_j は複数のオペレーションを同時には処理できない.

OSS 問題の目的

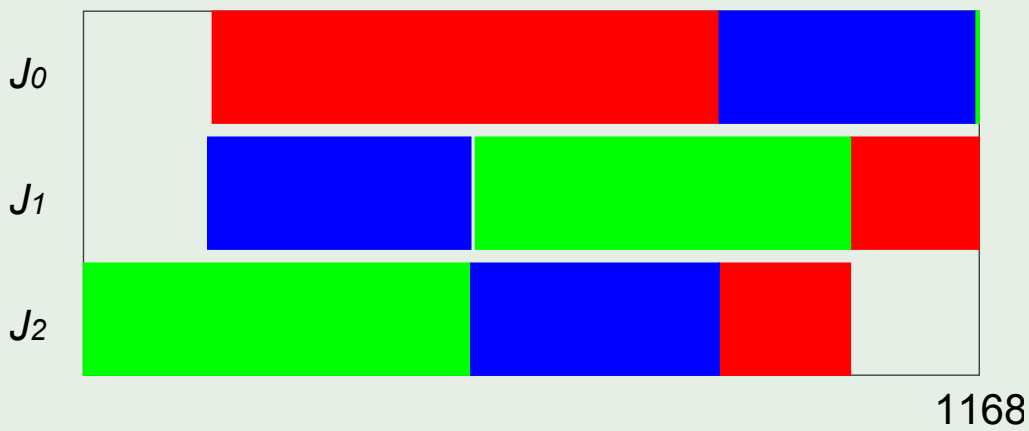
- すべてのジョブを完了するのに必要な時間 *makespan* を最小にする.

- OSS の可能なスケジューリングの組合せは膨大である．n ジョブ n マシンの OSS は，$(n!)^{2n}$ 個の実行可能解を持つ.

OSS 問題 gp03-01

$$(p_{ij}) = \begin{pmatrix} 661 & 6 & 333 \\ 168 & 489 & 343 \\ 171 & 505 & 324 \end{pmatrix}$$

gp03-01 の最適解 (makespan=1168)



gp03-01 の制約モデル

整数変数の定義

- m: makespan
- s_{ij}: オペレーション O_{ij} の開始時刻

制約の定義

- 各 s_{ij} に対して

$$s_{ij} + p_{ij} \leq m$$

- 同一ジョブ J_i の各オペレーション O_{ij}, O_{il} に対して

$$(s_{ij} + p_{ij} \leq s_{il}) \vee (s_{il} + p_{il} \leq s_{ij})$$

- 同一マシン M_j の各オペレーション O_{ij}, O_{kj} に対して

$$(s_{ij} + p_{ij} \leq s_{kj}) \vee (s_{kj} + p_{kj} \leq s_{ij})$$

gp03-01 の制約モデル

gp03-01 の CSP 表現

$$s_{00} + 661 \leq m$$
$$s_{01} + 6 \leq m$$
$$s_{02} + 333 \leq m$$
$$\cdots\cdots\cdots$$
$$s_{22} + 324 \leq m$$
$$(s_{00} + 661 \leq s_{01}) \vee (s_{01} + 6 \leq s_{00})$$
$$(s_{00} + 661 \leq s_{02}) \vee (s_{02} + 333 \leq s_{00})$$
$$(s_{01} + 6 \leq s_{02}) \vee (s_{02} + 333 \leq s_{01})$$
$$\cdots\cdots\cdots$$
$$(s_{02} + 333 \leq s_{12}) \vee (s_{12} + 343 \leq s_{02})$$
$$(s_{02} + 333 \leq s_{22}) \vee (s_{22} + 324 \leq s_{02})$$
$$(s_{12} + 343 \leq s_{22}) \vee (s_{22} + 324 \leq s_{12})$$

gp03-01 を Sugar で解く

```
$ sugar oss-gp03-01.csp

a makespan  1168
a s_0_0  12
a s_0_1  1162
a s_0_2  829
a s_1_0  1000
a s_1_1  511
a s_1_2  0
a s_2_0  829
a s_2_1  0
a s_2_2  505
a
s OPTIMUM FOUND
```

gp03-01 を Sugar で解く

充足可能の場合 ($m \leq 1168$)

- MiniSat が解を求めるまでの処理
  - 12 decisions
  - 1 conflict (backtrack).

充足不能の場合 ($m \leq 1167$)

- MiniSat が充足不能を示すまでの処理
  - 6 decisions
  - 5 conflicts (backtracks).

- MiniSat の単位伝播により高速な範囲伝播が実現されている.

解集合プログラミングと時間割問題

SAT 技術の広がり (再掲)

問題を SAT に翻訳し SAT ソルバーを用いて解く手法が様々な分野で成功し，SAT 技術が大きな広がりを見せている．

- 有界モデル検査 [Biere, 2009]
- Intel core I7 プロセッサ設計 [Kaivola+, CAV 2009]
- Windows 7 デバイス・ドライバー検証
  [De Moura+, IJCAR 2010] (SMT ソルバー Z3)
- ソフトウェア要素の依存性解析
  - Eclipse [Le Berre and Rapicault, IWOCE 2009]
  - Fedora Linux (および RedHat, CentOS) の dnf
- プランニング (SATPLAN, Blackbox) [Kautz+, 1992]
- ショップ・スケジューリング [Crawford+, 1994; 田村+, 2009]
- 制約プログラミング (Sugar) [田村+, 2009]
- **解集合プログラミング** (clingo) [Gebser+, 2012]
- 多目的離散最適化問題 (sucre) [宋+, CP 2017]

解集合プログラミング (Answer Set Programming; ASP)

ASP は比較的新しい宣言的プログラミングパラダイムの一つである.

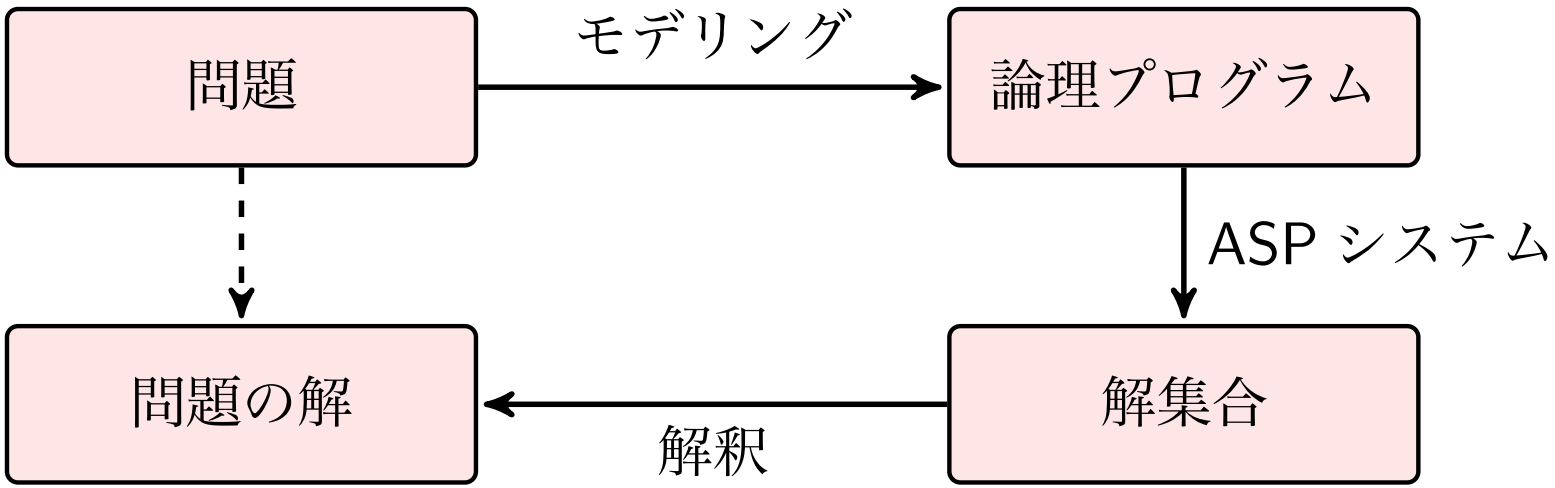
- **ASP 言語**は一階論理に基づく知識表現言語の一種
- **ASP システム**は安定モデル意味論 [Gelfond and Lifschitz, '88] に基づく**解集合**を計算するシステム
- ASP の起源
  - 演繹データベース
  - 論理プログラミング (否定付き)
  - 知識表現 & 非単調推論
  - 制約充足 (特に，SAT)

解集合プログラミング (続き)

近年，SAT 技術を応用した高速な ASP システムが開発され，人工知能分野への実用的応用が急速に拡大している．

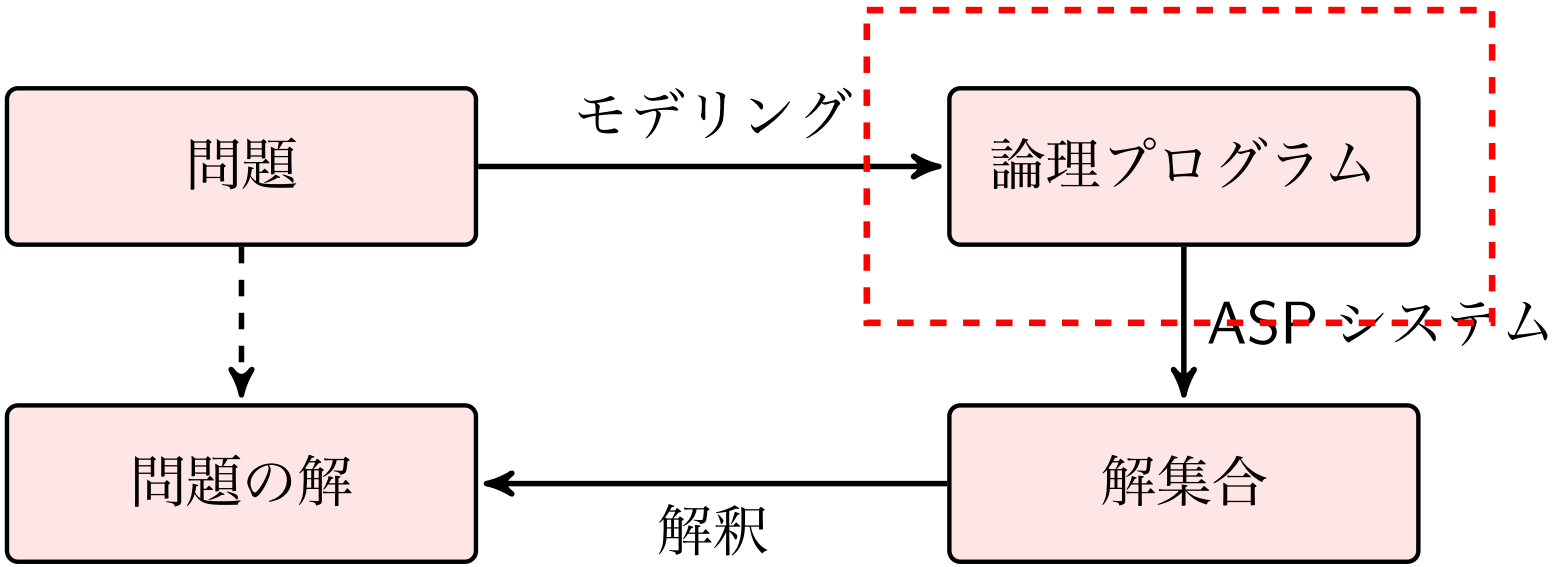
- ASP システム
  - **clingo** ▸ Web, WASP ▸ Web, DLV ▸ Web, etc.
- 応用
  - ロボット工学，システム生物学，モデル検査，
  - マルチエージェント，チーム編成，テストケース生成，
  - プランニング，スケジューリング，etc.
- 国際会議
  - IJCAI, AAAI, ICLP, KR, LPNMR, etc.

ASP による問題解法



1. 解きたい問題を**論理プログラム**としてモデリングする.
2. ASP システムを用いて，論理プログラムの**解集合** (一種の最小モデル) を計算する.
3. 解集合を解釈して元の問題の解を得る.

ASP による問題解法



1. 解きたい問題を**論理プログラム**としてモデリングする.
2. ASP システムを用いて，論理プログラムの**解集合** (一種の最小モデル) を計算する.
3. 解集合を解釈して元の問題の解を得る.

論理プログラムとルール

ASP 言語は論理プログラムをベースとしている.

- **論理プログラム** P とは，以下の形式の**ルール** r の有限集合である.

$$\underbrace{a_0}_{\text{ヘッド}} \leftarrow \underbrace{a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n}_{\text{ボディ}}$$

$0 \leq m \leq n$ であり，各 a_i はアトム，$\sim$ は**デフォルトの否定**，“,” は連言を表す [1].

[1] 本発表では標準論理プログラムを単に論理プログラムと呼ぶ.

論理プログラムとルール

ASP 言語は論理プログラムをベースとしている.

- **論理プログラム** P とは，以下の形式の**ルール** r の有限集合である.

$$\underbrace{a_0}_{\text{ヘッド}} \leftarrow \underbrace{a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n}_{\text{ボディ}}$$

  $0 \leq m \leq n$ であり，各 a_i はアトム，$\sim$ は**デフォルトの否定**，“,” は連言を表す[1].

- **直観的な意味は**「$a_1, \ldots, a_m$ がすべて成り立ち，$a_{m+1}, \ldots, a_n$ のそれぞれが成り立たないならば，
  a_0 が成り立つ」となる.

[1]本発表では標準論理プログラムを単に論理プログラムと呼ぶ.

論理プログラムとルール (続き)

- ボディが空のルールを**ファクト**と呼び，← を省略可.

$$\underbrace{a_0}_{\text{ヘッド}}$$

- ヘッドが空のルールを **(一貫性) 制約**と呼ぶ.

$$\leftarrow \underbrace{a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n}_{\text{ボディ}}$$

ボディのリテラル (a_i あるいは ${\sim}a_i$) の連言 (AND) が成り立たないことを表す.

拡張構文

ASP 言語には，組合せ問題を解くために便利な構文が用意されている．

- **選択子**

$$\{a_1; \ldots; a_n\}$$

アトム集合 $\{a_1, \ldots, a_n\}$ の任意の部分集合が成り立つことを意味する．

- **個数制約**

$$s\ \{a_1; \ldots; a_n\}\ k$$

$a_1, \ldots, a_n$ のうち，s 個以上 k 個以下が成り立つことを意味する．

拡張構文

ASP 言語には，組合せ問題を解くために便利な構文が用意されている．

- **選択子**

  $$\{a_1; \ldots; a_n\}$$

  アトム集合 $\{a_1, \ldots, a_n\}$ の任意の部分集合が成り立つことを意味する．

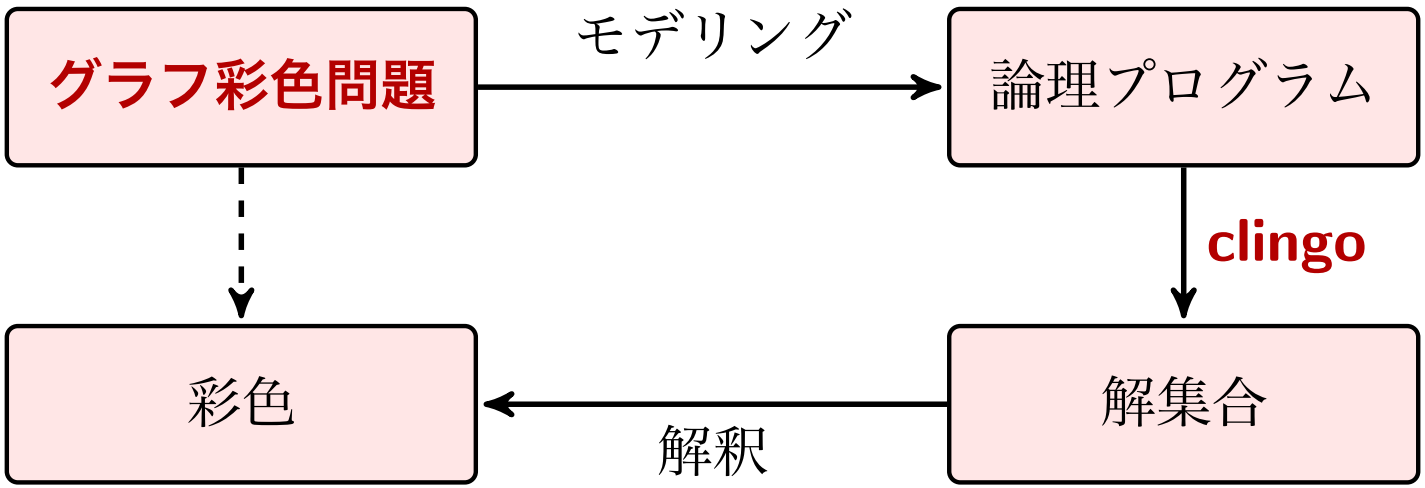
- **個数制約**

  $$s\ \{a_1; \ldots; a_n\}\ k$$

  $a_1, \ldots, a_n$ のうち，s 個以上 k 個以下が成り立つことを意味する．

これら以外にも，組合せ最適化問題を解くための**最小化関数・最大化関数**なども用意されている．

例: グラフ彩色問題

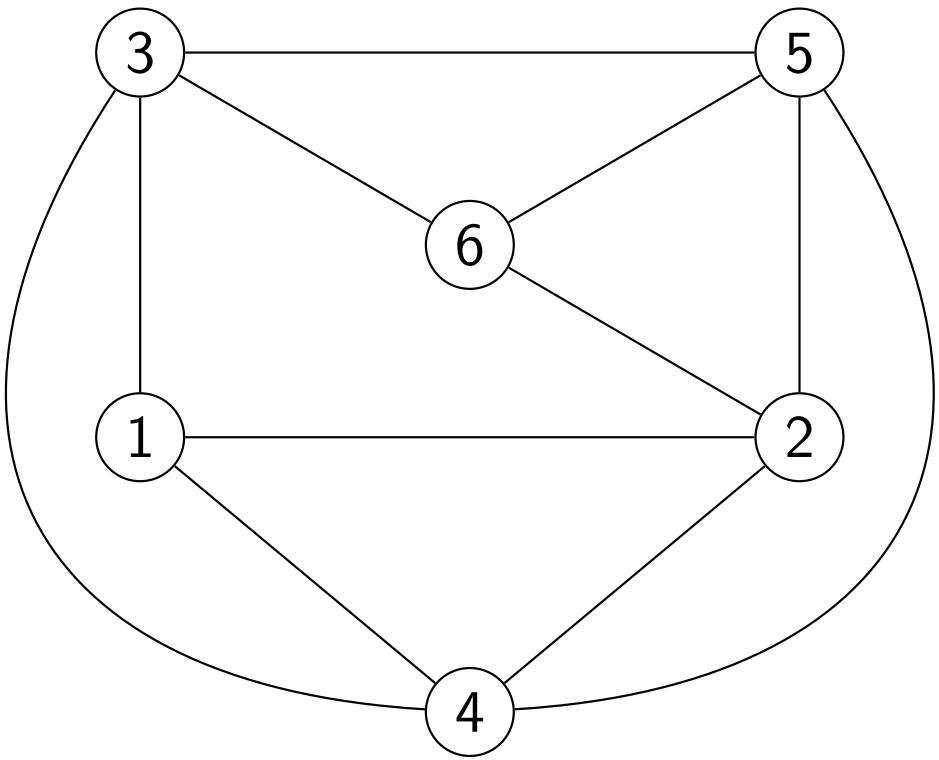


- 表記法

論理プログラム	←	,	;	~
ソースコード	:-	,	;	not

グラフ彩色問題

- 辺で結ばれたノードが同じ色にならないように，各ノードを塗り分ける．
- 言い換えると，
  1. 各ノードは，一つの色で塗られる．
  2. 辺で結ばれたノードは，同じ色で塗られない．



グラフ彩色問題: color.lp

```
node(1).    node(2).    node(3).
node(4).    node(5).    node(6).

edge(1,2).  edge(1,3).  edge(1,4).
edge(2,4).  edge(2,5).  edge(2,6).
edge(3,4).  edge(3,5).  edge(3,6).
edge(4,5).  edge(5,6).

col(r).   col(b).   col(g).
```

問題インスタンス

グラフ彩色問題: color.lp

```
node(1).    node(2).    node(3).
node(4).    node(5).    node(6).

edge(1,2).  edge(1,3).  edge(1,4).
edge(2,4).  edge(2,5).  edge(2,6).
edge(3,4).  edge(3,5).  edge(3,6).
edge(4,5).  edge(5,6).

col(r).   col(b).   col(g).
```

問題インスタンス

```
1 { color(X,C) : col(C) } 1 :- node(X).


:- edge(X,Y), color(X,C), color(Y,C).
```

制約の記述

グラフ彩色問題: color.lp

```
node(1).    node(2).    node(3).
node(4).    node(5).    node(6).

edge(1,2).  edge(1,3).  edge(1,4).
edge(2,4).  edge(2,5).  edge(2,6).
edge(3,4).  edge(3,5).  edge(3,6).
edge(4,5).  edge(5,6).

col(r).   col(b).   col(g).
```

問題インスタンス

```
1 { color(X,r);color(X,b);color(X,g) } 1
        :- node(X).
:- edge(X,Y), color(X,C), color(Y,C).
```

制約の記述

グラフ彩色問題: 実行

```
$ clingo color.lp -n 0
```

グラフ彩色問題: 実行

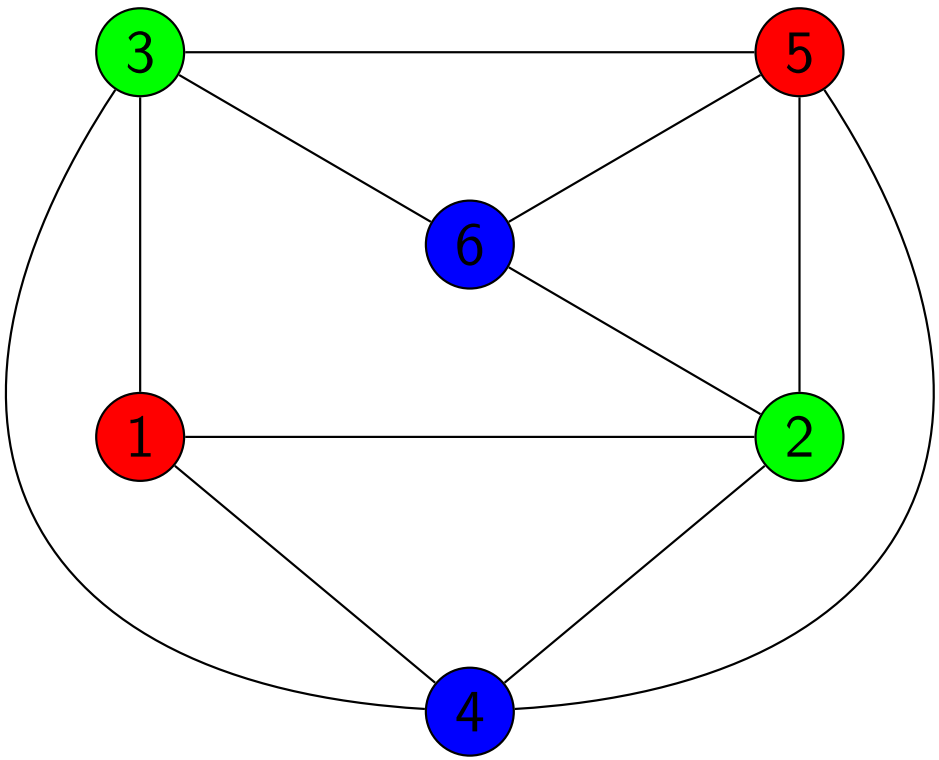
```
$ clingo color.lp -n 0
```

```
clingo version 5.0.0
Reading from color.lp
Solving...
Answer: 1
node(1) ... col(r) ... edge(1,2) ... color(1,g) color(2,b) color(3,b) color(4,r) color(5,g) color(6,r)
Answer: 2
node(1) ... col(r) ... edge(1,2) ... color(1,r) color(2,b) color(3,b) color(4,g) color(5,r) color(6,g)
Answer: 3
node(1) ... col(r) ... edge(1,2) ... color(1,r) color(2,g) color(3,g) color(4,b) color(5,r) color(6,b)
Answer: 4
node(1) ... col(r) ... edge(1,2) ... color(1,b) color(2,g) color(3,g) color(4,r) color(5,b) color(6,r)
Answer: 5
node(1) ... col(r) ... edge(1,2) ... color(1,g) color(2,r) color(3,r) color(4,b) color(5,g) color(6,b)
Answer: 6
node(1) ... col(r) ... edge(1,2) ... color(1,b) color(2,r) color(3,r) color(4,g) color(5,b) color(6,g)
SATISFIABLE

Models       : 6
Calls        : 1
Time         : 0.001s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

彩色

```
Answer: 3
node(1) ... col(r) ... edge(1,2) ...
color(1,r) color(2,g) color(3,g)
color(4,b) color(5,r) color(6,b)
```



時間割編成 (Timetabling)

求解が困難とされる組合せ最適化問題の一種.

- 時間割に関する**国際会議** PATAT が 1995 年から開催.
  - 教育時間割編成 (educational timetabling)
  - 輸送時間割編成 (transport timetabling)
  - 従業員時間割編成 (employee timetabling)
  - スポーツ時間割編成 (sports timetabling) など
- 近年，**国際的な時間割編成競技会**が開催されており，時間割ソルバーの性能向上に貢献している.
  ITC2007 ▸ Web ITC2011 ▸ Web ITC2019 ▸ Web INRC ▸ Web
  INRC-II ▸ Web

コース時間割問題

カリキュラムベースのコース時間割問題 (CB-CTT) は大学の 1 週間の講義スケジュールを作成する問題である．この CB-CTT は最も研究されている教育時間割の一つである．

- 日時，科目，教員，教室，課程に対する**ハード制約**と**ソフト制約**から構成される．
- ソフト制約に違反した場合はペナルティが課せられる．
- ハード制約をすべて満たし，かつ，ソフト制約のペナルティの合計を最小化することが目的．

制約とシナリオ

- **シナリオ**とはソフト制約の集合である.
- ITC2007 競技会では UD2 が使用された.

制約	UD1	UD2	UD3	UD4	UD5
H_1. Lectures	H	H	H	H	H
H_2. Conflicts	H	H	H	H	H
H_3. RoomOccupancy	H	H	H	H	H
H_4. Availability	H	H	H	H	H
S_1. RoomCapacity	1	1	1	1	1
S_2. MinWorkingDays	5	5	-	1	5
S_3. IsolatedLectures	1	2	-	-	1
S_4. Windows	-	-	4	1	2
S_5. RoomStability	-	1	-	-	-
S_6. StudentMinMaxLoad	-	-	2	1	2
S_7. TravelDistance	-	-	-	-	2
S_8. RoomSuitability	-	-	3	H	-
S_9. DoubleLectures	-	-	-	1	-

CB-CTT インスタンス: toy.ectt

```
Name: Toy
Courses: 4
Rooms: 3
Days: 5
Periods_per_day: 4
Curricula: 2
Min_Max_Daily_Lectures: 2 3
UnavailabilityConstraints: 8
RoomConstraints: 3

COURSES:
SceCosC Ocra 3 3 30 1
ArcTec Indaco 3 2 42 0
TecCos Rosa 5 4 40 1
Geotec Scarlatti 5 4 18 1

ROOMS:
rA 32 1
rB 50 0
rC 40 0

CURRICULA:
Cur1 3 SceCosC ArcTec TecCos
Cur2 2 TecCos Geotec

UNAVAILABILITY_CONSTRAINTS:
TecCos 2 0
TecCos 2 1
TecCos 3 2
TecCos 3 3
ArcTec 4 0
ArcTec 4 1
ArcTec 4 2
ArcTec 4 3

ROOM_CONSTRAINTS:
SceCosC rA
Geotec rB
TecCos rC

END.
```

ASP ファクト: toy.lp

```
name("Toy").
courses(4).
rooms(3).
days(5).
periods_per_day(4).
curricula(2).
min_max_daily_lectures(2,3).
unavailabilityconstraints(8).
roomconstraints(3).

course("SceCosC","Ocra",3,3,30,1).
course("ArcTec","Indaco",3,2,42,0).
course("TecCos","Rosa",5,4,40,1).
course("Geotec","Scarlatti",5,4,18,1).

room(rA,32,1).
room(rB,50,0).
room(rC,40,0).

curricula("Cur1","SceCosC").
curricula("Cur1","ArcTec").
curricula("Cur1","TecCos").
curricula("Cur2","TecCos").
curricula("Cur2","Geotec").

unavailability_constraint("TecCos",2,0).
unavailability_constraint("TecCos",2,1).
unavailability_constraint("TecCos",3,2).
unavailability_constraint("TecCos",3,3).
unavailability_constraint("ArcTec",4,0).
unavailability_constraint("ArcTec",4,1).
unavailability_constraint("ArcTec",4,2).
unavailability_constraint("ArcTec",4,3).

room_constraint("SceCosC",rA).
room_constraint("Geotec",rB).
room_constraint("TecCos",rC).
```

CB-CTT の解 (シナリオ UD2 の場合)

アトム assigned(C,R,D,P) は，科目 C の講義が D 日の P 時限に，教室 R で開講されることを意味する．

partial answer set

```
assigned("TecCos", rB,0,1).
assigned("TecCos", rB,0,3).
assigned("TecCos", rB,1,3).
assigned("TecCos", rB,2,3).
assigned("TecCos", rB,4,3).
assigned("SceCosC",rB,3,0).
assigned("SceCosC",rB,2,2).
assigned("SceCosC",rB,4,2).
assigned("ArcTec", rB,3,1).
assigned("ArcTec", rB,0,2).
assigned("ArcTec", rB,1,2).
assigned("Geotec", rA,4,1).
assigned("Geotec", rA,0,2).
assigned("Geotec", rA,1,2).
assigned("Geotec", rA,2,2).
assigned("Geotec", rA,4,2).
```

Cur1

	Day0	Day1	Day2	Day3	Day4
0				SceCosC rB	
1	TecCos rB			ArcTec rB	
2	ArcTec rB	ArcTec rB	SceCosC rB		SceCosC rB
3	TecCos rB	TecCos rB	TecCos rB		TecCos rB

Cur2

	Day0	Day1	Day2	Day3	Day4
0					
1	TecCos rB				Geotec rA
2	Geotec rA	Geotec rA	Geotec rA		Geotec rA
3	TecCos rB	TecCos rB	TecCos rB		TecCos rB

論理プログラム: ハード制約 H_1

H_1. Lectures

各科目のすべての講義は異なる日時で開講される.

```
N {assigned(C,D,P) : d(D), ppd(P)} N :- course(C,_,N,_,_,_).
```

- アトム `assigned(C,D,P)` は，科目 `C` の講義が `D` 日の `P` 時限に開講されることを意味する.

論理プログラム: ハード制約 H_2

H_2. Conflicts

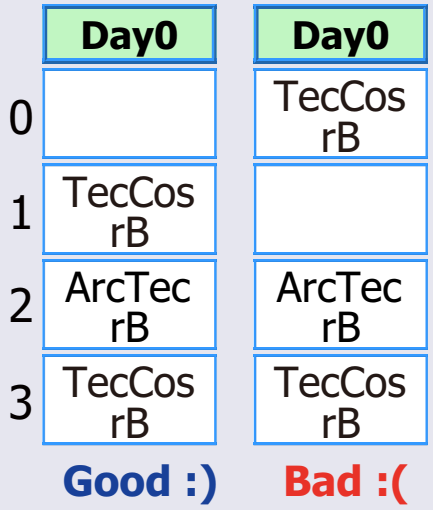
- 同一教員が担当する科目のすべての講義は，異なる日時で開講される．
- 同一課程に属する科目のすべての講義は，異なる日時で開講される．

```
:- not {assigned(C,D,P) : course(C,T,_,_,_,_)} 1,t(T),d(D),ppd(P).
:- not {assigned(C,D,P) : curricula(Cu,C)} 1,cu(Cu),d(D),ppd(P).
```

論理プログラム: ソフト制約 S_3

S_3. IsolatedLectures

- 同一課程に属する講義は，連続した時限に開講されるべきである．
- 同一曜日に同一課程に属する他のどの講義とも隣接していない (孤立した) 講義がある場合に違反となり，孤立した講義毎にペナルティが課される．




```
scheduled_curricula(Cu,D,P) :- assigned(C,D,P), curricula(Cu,C).
penalty("IsolatedLectures",isolated_lectures(Cu,D,P),weight_s3) :-
        scheduled_curricula(Cu,D,P),
        not scheduled_curricula(Cu,D,P-1),
        not scheduled_curricula(Cu,D,P+1).
```

参考文献 (ASP)

- **Answer Set Solving in Practice**
  M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. 238 pages, 2012, Morgan and Claypool Publishers. ▸ Web
- **AI MAGAZINE** Vol.37, No.3, Fall Issue, 2016
  - ASP の特集．8 編の記事が掲載． ▸ Web
- **論理プログラミングから解集合プログラミングへ**
  井上克巳，坂間千秋．コンピュータソフトウェア, Vol.25, No.3, pp.20-32, 2008. ▸ Web
- **解集合プログラミング**
  坂間千秋，井上克巳．人工知能学会誌, Vol.25, No.3, pp.368-378, 2010. ▸ Web

演習

数独パズル (藤原博文氏作)

9 つある各行，9 つある各列，9 つある 3×3 のブロックに 1 から 9 の数が 1 回ずつ出るように数を埋めるパズル

	4	3				6	7	
5			4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

数独パズル (藤原博文氏作)

9 つある各行，9 つある各列，9 つある 3×3 のブロックに 1 から 9 の数が 1 回ずつ出るように数を埋めるパズル

	4	3				6	7	
5			4		2			8
8				6				1
2								5
	5						4	
		6				7		
			5		1			
				8				

7	2	8	9	3	6	5	1	4
9	4	3	1	5	8	6	7	2
5	6	1	4	7	2	9	3	8
8	3	4	7	6	5	2	9	1
2	1	7	8	4	9	3	6	5
6	5	9	2	1	3	8	4	7
1	8	6	3	2	4	7	5	9
3	7	2	5	9	1	4	8	6
4	9	5	6	8	7	1	2	3

Sugar のダウンロード

- Sugar のダウンロード

  http://bach.istc.kobe-u.ac.jp/sugar/ ▸ Web

- パズルを Sugar 制約ソルバーで解く
  - http://bach.istc.kobe-u.ac.jp/sugar/puzzles/ ▸ Web
  - 数独，カックロ，美術館，四角に切れ，ナンバーリンク，ましゅ，スリザーリンク，橋をかけろ，ヤジリン，ぬりかべ，ひとりにしてくれ，フィルオミノ，へやわけ，お絵かきロジック