情報科学専攻 情報基礎講座
ソフトウェア

制約プログラミングと SAT ソルバー

田村直之　　宋剛秀

高度教養セミナー工学部 (情報知能工学)
2019 年 1 月 29 日

研究室のテーマ

論理と推論に基づくプログラミング/ソフトウェア・システム

- 正しいソフトウェアを作るため
- より高性能なソフトウェアを作るため

関連するプログラミング・パラダイム

- 関数プログラミング
  - プログラムは関数から構成される
  - Lisp, Scala, ML, Haskell, OCaml
- 論理プログラミング
  - プログラムは論理式から構成される
  - Prolog, Answer Set Programming
- 制約プログラミング (現在の中心テーマ)
  - プログラムは制約式 (整数上の条件式など) から構成される

制約プログラミングとは

制約プログラミング

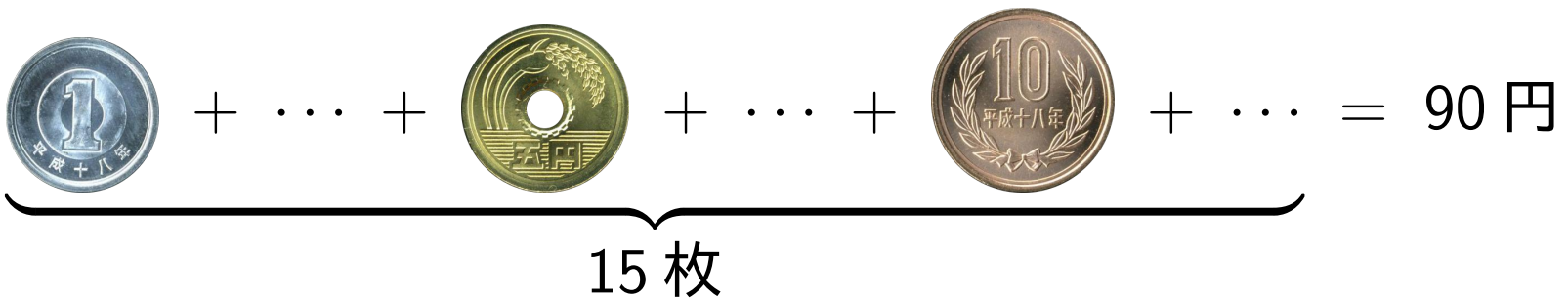
問題を解くための解法をプログラムするのでなく，プログラム中に単に問題の条件 (制約) を記述するだけで問題解決を実現しようとするプログラミング・パラダイム

- 入力から出力を求めるための操作を記述するのではない
- 入力と出力の間で成り立つべき制約を記述する
- したがって，出力から入力を求めることも可能になる (少なくとも原理的には)
- 人工知能の一分野として企業や大学で研究されている
  - IBM, Intel, Google, Microsoft など

制約プログラミングの例: コインの問題の例

コインの問題の例

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?



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

コインの問題の例を Copris で解く

Copris[1] のプログラム (coin.scala)

```
import jp.kobe_u.copris._
import dsl._

int('x, 1, 15)
int('y, 1, 15)
int('z, 1, 15)
add('x + 'y + 'z === 15)
add('x + 'y*5 + 'z*10 === 90)
if (find) println(solution)
```

実行結果

```
$ copris coin.scala
Solution(Map(x -> 5, y -> 3, z -> 7),Map())
```

[1] Scala 上の制約プログラミング・システム

制約プログラミングの例: 推理パズルの例

推理パズルの例

明，勇，正，洋の 4 人は，それぞれ傘，靴，紙，糸を買いました．それぞれの色は赤，青，白，黒です．以下のヒントから誰がどの色の何を買ったのか推理してください．

1. 明は白いのを買ったが糸じゃない．
2. 青い紙を買った人がいるが，勇じゃない．
3. 正は靴を買ったが赤じゃない．

推理パズルの例を Copris で解く

Copris のプログラム (puzzle.scala)

```
import jp.kobe_u.copris._; import dsl._

int('明, 1); int('勇, 2); int('正, 3); int('洋, 4)
val items = for (i <- Seq('傘, '靴, '紙, '糸)) yield int(i, 1, 4)
add(Alldifferent(items))
val colors = for (c <- Seq('赤, '青, '白, '黒)) yield int(c, 1, 4)
add(Alldifferent(colors))
def 買った(x: Term, y: Term) = (x === y)

// 明は白いのを買ったが糸じゃない.
add(買った('明, '白) && ! 買った('明, '糸))

// 青い紙を買った人がいるが, 勇じゃない.
int('誰か, 1, 4)
add(買った('誰か, '青) && 買った('誰か, '紙) && '誰か =/= '勇)

// 正は靴を買ったが赤じゃない.
add(買った('正, '靴) && ! 買った('正, '赤))

if (find) println(solution)
```

実行結果

```
$ copris puzzle.scala
Solution(Map(白 -> 1, 青 -> 4, 誰か -> 4, 傘 -> 1, 赤 -> 2, 紙 -> 4,
  勇 -> 2, 糸 -> 2, 明 -> 1, 洋 -> 4, 靴 -> 3, 正 -> 3, 黒 -> 3),Map())
```

本研究室で開発した制約プログラミングのシステム

- **Sugar**：2008,2009 年の制約ソルバー競技会で優勝
- **sCOP**：2018 年の制約ソルバー競技会で優勝
- **Copris**, **Scarab**：Scala 上の制約プログラミングシステム

- いずれも，与えられた制約を SAT 問題に符号化 (一種のコンパイル) し，高速な SAT ソルバーを用いて解を求めている.

SAT 問題とは

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

- 最初に NP-完全であることが証明された問題 [Cook 1971]
- 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，問題を SAT に符号化 (変換) して 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

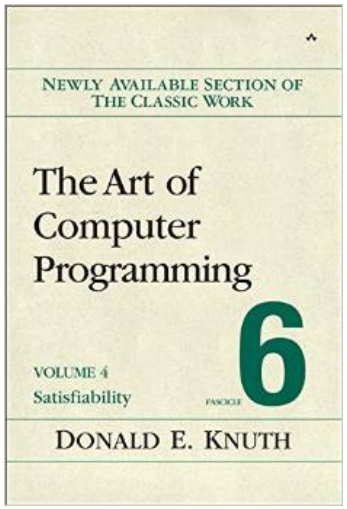
SAT ソルバー (SAT Solvers)

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．
  - $2^{10^6} \sim 10^{300000}$ の大きさの状態空間!

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 に関する章である.

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, . . .

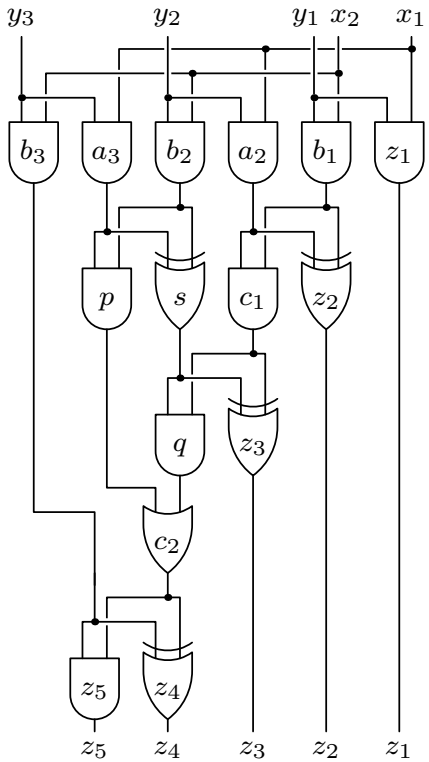
Knuth: “Satisfiability” 中の例題

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おり，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

整数の因数分解

- 右は $x = (x_2x_1)$ と $y = (y_3y_2y_1)$ の積 $z = (z_5z_4z_3z_2z_1)$ を求める乗算回路である.
- この回路を命題論理式で表し, $z = (10101)_2 = 21$ の条件を加えて SAT ソルバーに与えると, $x = (11)_2 = 3$, $y = (111)_2 = 7$ の解が求まる.



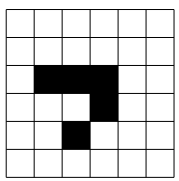
同様にすると以下も 1 秒未満で求まる.

$$z = 11111111111111111111111111011000000000000000000000000010011_2$$
$$= 46116859754777149 63$$
$$x = 11111111111111111111111101101_2 = 2147483629$$
$$y = 11111111111111111111111111111_2 = 2147483647$$

有界モデル検査 (ライフゲーム)

Conway のライフゲーム (Game of Life) は，生命の誕生・死滅をシミュレーションした 2 次元セル・オートマトンである.

- 各セルは生か死のどちらかの状態を持ち，各セルの次の時刻の状態は周囲 8 個のセルの状態で決まる.
  - 現在のセルの状態を x, 周囲 8 個中の生のセルの個数を c, 次時刻のセルの状態を x' で表す.
  - $x =$ 生 $\wedge\, c \leq 1$ なら $x' =$ 死 (過疎)
  - $x =$ 生 $\wedge\, c \geq 4$ なら $x' =$ 死 (過密)
  - $x =$ 死 $\wedge\, c = 3$ なら $x' =$ 生 (誕生)
  - それ以外は $x' = x$

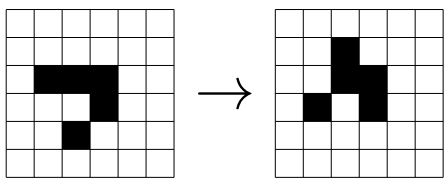


ライフゲームの例 (グライダー)

有界モデル検査 (ライフゲーム)

Conway のライフゲーム (Game of Life) は，生命の誕生・死滅をシミュレーションした 2 次元セル・オートマトンである.

- 各セルは生か死のどちらかの状態を持ち，各セルの次の時刻の状態は周囲 8 個のセルの状態で決まる.
  - 現在のセルの状態を x, 周囲 8 個中の生のセルの個数を c, 次時刻のセルの状態を x' で表す.
  - $x =$ 生 $\wedge\, c \leq 1$ なら $x' =$ 死 (過疎)
  - $x =$ 生 $\wedge\, c \geq 4$ なら $x' =$ 死 (過密)
  - $x =$ 死 $\wedge\, c = 3$ なら $x' =$ 生 (誕生)
  - それ以外は $x' = x$

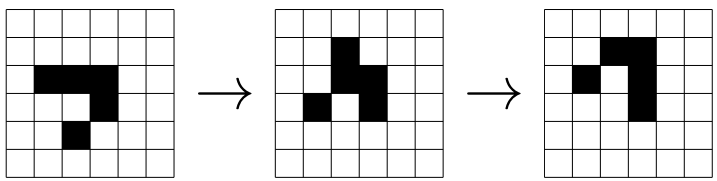


ライフゲームの例 (グライダー)

有界モデル検査 (ライフゲーム)

Conway のライフゲーム (Game of Life) は，生命の誕生・死滅をシミュレーションした 2 次元セル・オートマトンである.

- 各セルは生か死のどちらかの状態を持ち，各セルの次の時刻の状態は周囲 8 個のセルの状態で決まる.
  - 現在のセルの状態を x, 周囲 8 個中の生のセルの個数を c, 次時刻のセルの状態を x' で表す.
  - $x =$ 生 $\wedge\, c \leq 1$ なら $x' =$ 死 (過疎)
  - $x =$ 生 $\wedge\, c \geq 4$ なら $x' =$ 死 (過密)
  - $x =$ 死 $\wedge\, c = 3$ なら $x' =$ 生 (誕生)
  - それ以外は $x' = x$

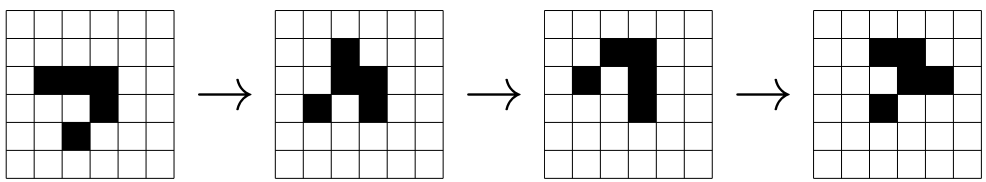


ライフゲームの例 (グライダー)

有界モデル検査 (ライフゲーム)

Conway のライフゲーム (Game of Life) は，生命の誕生・死滅をシミュレーションした 2 次元セル・オートマトンである.

- 各セルは生か死のどちらかの状態を持ち，各セルの次の時刻の状態は周囲 8 個のセルの状態で決まる.
  - 現在のセルの状態を x, 周囲 8 個中の生のセルの個数を c, 次時刻のセルの状態を x' で表す.
  - $x =$ 生 $\wedge\, c \leq 1$ なら $x' =$ 死 (過疎)
  - $x =$ 生 $\wedge\, c \geq 4$ なら $x' =$ 死 (過密)
  - $x =$ 死 $\wedge\, c = 3$ なら $x' =$ 生 (誕生)
  - それ以外は $x' = x$

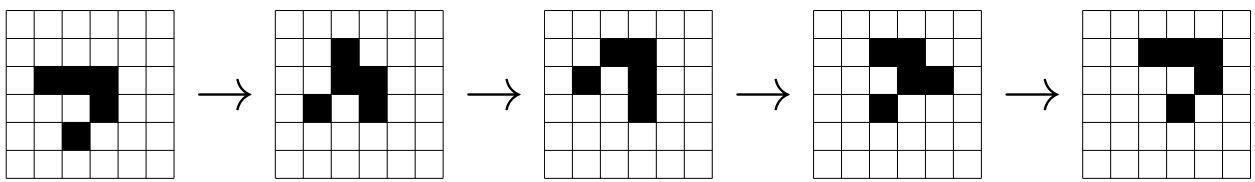


ライフゲームの例 (グライダー)

有界モデル検査 (ライフゲーム)

Conway のライフゲーム (Game of Life) は，生命の誕生・死滅をシミュレーションした 2 次元セル・オートマトンである.

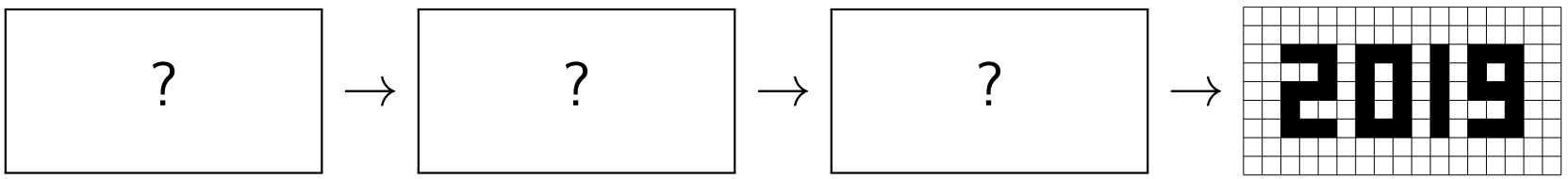
- 各セルは生か死のどちらかの状態を持ち，各セルの次の時刻の状態は周囲 8 個のセルの状態で決まる.
  - 現在のセルの状態を x, 周囲 8 個中の生のセルの個数を c, 次時刻のセルの状態を x' で表す.
  - $x =$ 生 $\wedge\, c \leq 1$ なら $x' =$ 死 (過疎)
  - $x =$ 生 $\wedge\, c \geq 4$ なら $x' =$ 死 (過密)
  - $x =$ 死 $\wedge\, c = 3$ なら $x' =$ 生 (誕生)
  - それ以外は $x' = x$



ライフゲームの例 (グライダー)

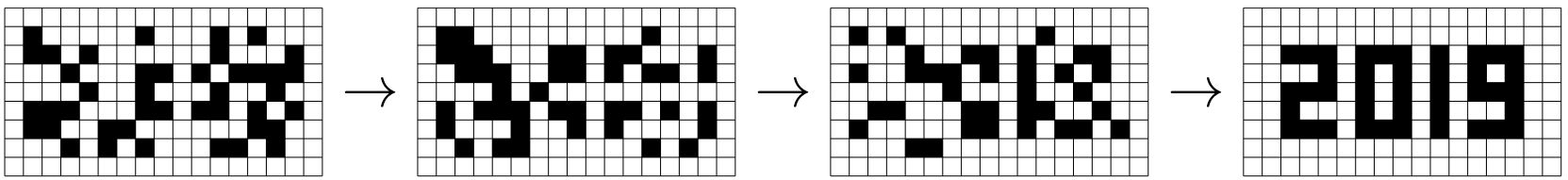
有界モデル検査 (ライフゲーム)

SAT ソルバーを用いれば，前の状態を求めることもできる!

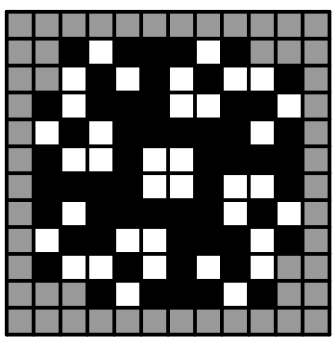


有界モデル検査 (ライフゲーム)

SAT ソルバーを用いれば，前の状態を求めることもできる!



前の状態が存在しない状態を探すことも可能だ (エデンの園問題)



卒業研究の内容

1. 新しいソルバー (制約ソルバーや SAT ソルバー) を作る．あるいは既存のソルバーを改良し，世界一高速なソルバーを目指す．
2. 未解決の問題にチャレンジし，SAT ソルバーを活用した新しい方法で解いてみる．

最近の卒業論文

2018 年度	
大野 周亮	alldifferent 制約のブール基数制約への符号化とその性能評価
岡本 祐樹	命題論理ソルバーを用いた Circuit 制約を含む制約充足問題の解法とその性能評価
中川 大地	SAT ソルバーを用いた多倍長整数制約ソルバーの設計と実装
2017 年度	
飯野 有軌	SAT ソルバーを用いた様相命題論理 S4 の充足可能性判定
生田 哲也	正規制約の SAT 符号化とその性能評価

4年生に期待すること

- 勉強そしてプログラミングにチャレンジ
- 自分の頭で考えて，卒業研究を行う
- 他大学との共同研究への参加

募集内容

プログラミングが好きで，修士課程進学を希望している学生さんを募集します.

- プログラミング・スキルを上達させたい人
- 世界的なソフトウェアを作ってみたい人
- 難しい問題にチャレンジしたい人

4年生に期待すること

- 勉強そしてプログラミングにチャレンジ
- 自分の頭で考えて，卒業研究を行う
- 他大学との共同研究への参加

募集内容

プログラミングが好きで，修士課程進学を希望している学生さんを募集します.

- プログラミング・スキルを上達させたい人
- 世界的なソフトウェアを作ってみたい人
- 難しい問題にチャレンジしたい人

小テスト

制約プログラミングやSATソルバーを用いてどんな問題を解いてみたいか，自分の考えを記述してください.