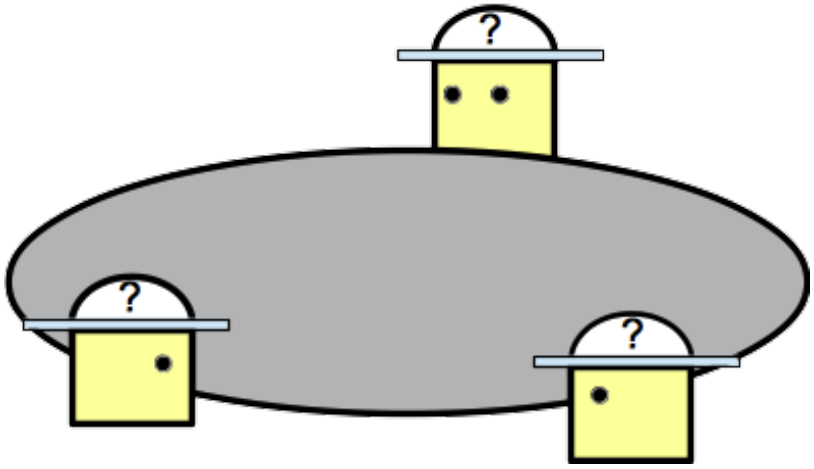
SMTソルバを用いた「竹内の3人の賢者問題」の求解

東京大学情報基盤センター
田中哲朗

「三人の賢者と帽子」パズル

- 3人の賢者に帽子をかぶせる
  - 自分の帽子の色は見えない
  - 残りの2人の帽子の色は見える
- 帽子の色は赤か黒，少なくとも一人は黒だと全員に伝える.
- 自分の帽子の色を当てることを求める
- 賢者たちはしばらく沈黙した後，同時に自分の帽子の色を当てた.
- 3人はそれぞれ何色の帽子をかぶっていたか?



「三人の賢者と帽子」のバリエーション

- The King's Wise Men, Prisoners and Hatなど
- 3人が列に並んで，前の人の帽子しか見えない．後ろの人から順に発言する．
- 人数を増やす

竹内の3人の賢者問題

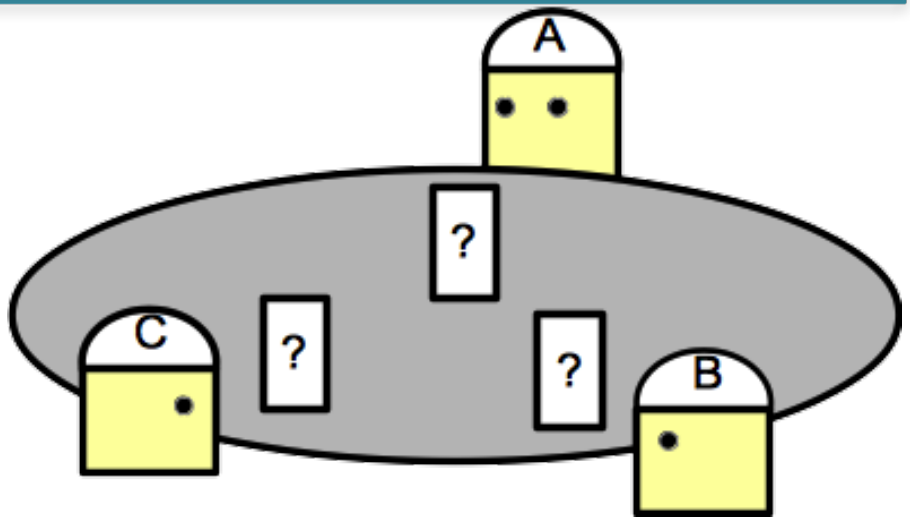
竹内郁雄:「エレガントな解答をもとむ」,数学セミナー,page 6, 2014年9月号より引用

3人の賢者A, B, Cがカード遊びをしています．1から20までの数が書かれた20枚のカードが円卓上に裏向きに置いてあり，賢者たちは1枚ずつ手元に取ります．大小比較で中間の数を取った賢者が勝利します．でも，一斉にカードをオープンするだけでは脳がへたれるというので，賢者たちは次のようにゲームを変形しました．

取ったカードが自分の右隣り(BはA,CはB,AはC)にだけ見せます．そして，Aから順に右回りに勝敗について確定したことを発言させるのです．ただし，勝敗に関する新しい確定情報がない時はニッコリ会釈するだけにします．

最初からニッコリが連続したあと，ある賢者が「じゃあ，わしの勝ちじゃな．ニッコリが最初からこれより長く連続することはなかろうて」と言いました．誰がどのように勝ったのでしょうか?

中央のカードが勝ちという mediocre および最中限というゲームがモデル



発表された解

竹内郁雄:「エレガントな解答をもとむ[解答]」,数学セミナー, pp. 87-90, 2014年12月号

- 最初からニッコリが9回続いたあと，Aがa=10, B=9, C=12 あるいはA = 11, B=12, C=10というカードの組合せで勝った.
- ニッコリした賢者のカードの値域が順に縮まっていくことを利用

計算機による求解

竹内郁雄: 3人の賢者の問題-愚者は賢者をシミュレートできるか?,第57回プログラミング・シンポジウム予稿集, pp. 1-8, 2016年

- ニッコリの連続以外も解くことを目標.
- 独自のマルチパラダイム言語(TAO/ELISのTAO, いまはDAO)で記述
- https://www.dropbox.com/s/qfnddaybtwqv5l5/three-wizards.pdf?dl=0 で公開されている.

もっとシンプルに解けないか?

➔ **satisfiability modulo theories (SMT)**ソルバの利用

satisfiability modulo theories (SMT) problem

- **Boolean satisfiability problem (SAT)** のvariableをtheoryを持つfirst order logicで置き換えたもの
- 主に扱われるtheory
  - empty theory
  - Array, List, Bit vector
  - linear arithmetic
- 近年SAT solverをベースにしたsolverが発達 ➔ programのverification等に利用

SMT Solver : z3

- Microsoft Researchが開発
- https://github.com/Z3Prover/z3 でMITライセンスで公開
- SMTソルバの中では比較的高機能，高速
  - ただし，今回は整数ドメインのlinear arithmeticが対象なのでz3である必要はなし
- 様々な言語のbindingが用意．特にPythonがお勧め ➔ 今回はこれを使用

問題の再定義

- 1からn(元の問題では20)までのカードを用いて，ニッコリが最大いくつ続くか?
- ニッコリの最大回数続いた後の人が自分の勝ちを宣言できるか? また，その時は，数字の組合せを列挙する.

1 – n の相異なるカードをA, B, C に配れるnは?

$$a, b, c \in Z$$
$$1 \leq a, b, c \leq n$$
$$a \neq b, b \neq c, c \neq a$$

を満たすa,b,cはnがいくつの時に存在するか?

```
from z3 import *
import itertools
for n in range(10):
    solver = Solver()
    vs = [Int("a"), Int("b"), Int("c")]
    solver.add([x != y for x, y in
itertools.combinations(v,2)])
    solver.add([And(1 <= x, x <= n) for
x in vs])
    if solver.check() == sat:
        print("n = %d" % n)
        print(solver)
        print(solver.model())
        break
```

プログラム(0.py)

```
n = 3
[a >= 1,
 a <= 3,
 b >= 1,
 b <= 3,
 a != b,
 c >= 1,
 c <= 3,
 a != c,
 b != c]
[b = 2, a = 1, c = 3]
```

出力

Aが最初にニッコリできるnは?

- 「Aの勝ち」と言える ⇔ 「Bの負け」かつ「Cの負け」
  - 「ニッコリ」は「*の負け」と言えない状態を指す.
- Aはa, b は知っていて，Cのカードが未知
  - 「Aの負け」と言えない ⇔ Cのカードを c_1と仮定するとAの勝ちとなる($(b < a \wedge a < c_1) \vee (c_1 < a \wedge a < b)$)
  - 「Bの負け」と言えない ⇔ Cのカードを c_2と仮定するとBの勝ちとなる($(a < b \wedge b < c_2) \vee (c_2 < b \wedge b < a)$)
  - 「Cの負け」と言えない ⇔ Cのカードを c_3と仮定するとCの勝ちとなる($(a < c_3 \wedge c_3 < b) \vee (b < c_3 \wedge c_3 < a)$)

以上を満たすa, b, c, c_1, c_2, c_3 が存在する.

Aが最初にニッコリできるnは?

```
from z3 import *
for n in range(10):
    solver = Solver()
    v = [Int("a"), Int("b"), Int("c")]
    for j in range(3):
        solver.add([1 <= v[j], v[j] <= n])
        for k in range(j):
            solver.add([v[k] != v[j]])
    tmpv = [Int("c_1"), Int("c_2"), Int("c_3")]
    for j in range(3):
        solver.add([1 <= tmpv[j], tmpv[j] <= n])
        for k in range(2):
            solver.add([v[k] != tmpv[j]])
    solver.add(Or(And(tmpv[0] < v[0], v[0] < v[1]),
                  And(v[1] < v[0], v[0] < tmpv[0])))
    solver.add(Or(And(tmpv[1] < v[1], v[1] < v[0]),
                  And(v[0] < v[1], v[1] < tmpv[1])))
    solver.add(Or(And(v[0] < tmpv[2], tmpv[2] < v[1]),
                  And(v[1] < tmpv[2], tmpv[2] < v[0])))
    if solver.check() == sat:
        print("n = %d" % n)
        print(solver)
        print(solver.model())
        break
```

プログラム(1.py)

```
n = 5
[a >= 1, a <= 5, b >= 1, b
<= 5, a != b, c >= 1, c <=
5, a != c, b != c, c_1 >= 1,
c_1 <= 5, a != c_1, b !=
c_1, c_2 >= 1, c_2 <= 5,
a != c_2, b != c_2, c_3 >=
1, c_3 <= 5, a != c_3, b !=
c_3, Or(And(c_1 < a, a <
b),And(b < a, a < c_1)),
Or(And(c_2 < b, b < a),
And(a < b, b < c_2)),
Or(And(a < c_3, c_3 < b),
And(b < c_3, c_3 < a))]
[b = 2, a = 4, c = 5, c_3 =
3, c_1 = 5, c_2 = 1]
```

出力

A, Bと続けてニッコリできるnは?

- Aがニッコリし，かつAがニッコリするでことを知った上でBがニッコリする.
- Bはb,c は知っていて，Aのカードが未知(ただし，Aがニッコリできるカードだということは知っている)
  - 「Aの負け」と言えない ⇔ Aのカードを a_1と仮定するとAの勝ちとなる($(b < a_1 \land a_1 < c) \lor (c < a_1 \land a_1 < b)$)
  - 「Bの負け」と言えない ⇔ Aのカードをa_2と仮定するとBの勝ちとなる($(a_2 < b \land b < c) \lor (c < b \land b < a_2)$)
  - 「Cの負け」と言えない ⇔ Aのカードをa_3と仮定するとCの勝ちとなる($(a_3 < c \land c < b) \lor (b < c \land c < a_3)$)

以上を満たすa, b, c, a_1, a_2, a_3 が存在する(プログラム 2.py).

m回連続してニッコリするのも同様に記述可能

n=20に対して何回ニッコリ可能か? プログラム

```
from z3 import *
import itertools, collections

class SolverGen:
    def __init__(self):
        self.varCount = 0
    def genVar(self, i):
        self.varCount += 1
        return Int("%s_%d" % ("abc"[i], self.varCount))
    def addSolver(self, solver, i, n, vs):
        if i == 0: return
        myIndex, unknownIndex = (i + 2) % 3, (i + 1) % 3
        oldvs = vs[:unknownIndex] + vs[unknownIndex+1:]
        tmpvs = [self.genVar(unknownIndex) for j in range(3)]
        for j in range(3):
            solver.add(And(1 <= tmpvs[j], tmpvs[j] <= n))
            solver.add([tmpvs[j] != y for y in oldvs])
            xs1 = vs[:unknownIndex] + [tmpvs[j]] +
vs[unknownIndex+1:]
            x1, y1, z1 = xs1[j:] + xs1[:j]
            solver.add(Or(And(y1 < x1, x1 < z1), And(z1 < x1,
x1 < y1)))
            self.addSolver(solver, i - 1, n, xs1)
        return
```

```
def makeSolver(self, i, n):
        solver = Solver()
        vs = [Int(x) for x in 'abc']
        solver.add([And(1 <= x, x <= n) for x in
vs])
        solver.add([x != y for x, y in
itertools.combinations(vs, 2)])
        for j in range(i + 1):
            self.addSolver(solver, j, n, vs)
        return solver
n = 20
for i in range(20):
    solver = SolverGen().makeSolver(i, n)
    if solver.check() == sat:
        print(i)
        print(solver)
        print(solver.model())
    else:
        break
```

プログラム (3.py)

n=20に対して何回ニッコリ可能か? 実行結果

0

[b = 2, a = 1, c = 3]

1

[b = 4, a = 2, c = 5, c_3 = 3, c_1 = 1, c_2 = 5]

2

[c_8 = 1, c_3 = 3, c_7 = 7, c_1 = 1, c_11 = 5, c_12 = 3, c_15 = 5, a_5 = 2, c_13 = 9, b = 4, a_4 = 6, a = 2, c_2 = 5, c_10 = 1, c_14 = 3, c = 7, c_9 = 5, a_6 = 8]

3

[c_8 = 11, b_16 = 13, c_37 = 13, c_26 = 8, c_22 = 1, c_41 = 11, b_17 = 10, c_15 = 3, c_42 = 9, a_43 = 6, c_49 = 1, c_2 = 1, c_14 = 11, c_53 = 3, c_47 = 1, a_6 = 2, a_45 = 8, c_54 = 5, c_3 = 11, c_29 = 14, c_12 = 11, c_30 = 3, c_50 = 5, c_27 = 14, c_40 = 5, a_32 = 12, b = 10, a_4 = 8, c_48 = 5, a_44 = 2, c_10 = 13, c_9 = 9, c_24 = 12, c_7 = 1, c_52 = 9, c_1 = 13, c_11 = 1, c_28 = 1, a_31 = 8, c_23 = 14, c_51 = 3, a_20 = 15, c_39 = 11, c_34 = 1, a_33 = 6, a_19 = 11, c_36 = 9, b_18 = 4, c_25 = 16, c_38 = 1, a_5 = 12, c_46 = 7, c_13 = 1, a = 12, a_21 = 2, c = 7, c_35 = 11]

最大は3回．公表された解答回数である9回ではない?

公表された正解の問題点

- 公表された解では最初にニッコリできないのは，A, B の中に1, 20 が含まれている時のみとしていた.
- 実際にはA=9, B=10 の時はCは中央値にならないので，Cの負けが確定.
  ニッコリできない.

ニッコリ3回の後に自分の勝ちを宣言できるか?

- 未知のカードが「既知のカードの組合せのもとでニッコリ2回が可能な」いかなるカードであっても自分の勝ち
  - ➔ Universal Quantifier(「∀」) が必要
  - SMT solverで直接扱うのは面倒
  - 候補を作成してから反例をsolverで探す.

プログラムはスライドでは省略
https://github.com/u-tokyo-gps-tanaka-lab/three_wise_men で公開

答え

- 70通り

```
[[(a, 6), (b, 4), (c, 10)], [(a, 13), (b, 15), (c, 9)], [(a, 12), (b, 14), (c, 8)],
[(a, 12), (b, 16), (c, 8)], [(a, 12), (b, 17), (c, 8)], [(a, 11), (b, 16), (c, 7)],
[(a, 12), (b, 16), (c, 7)], [(a, 11), (b, 17), (c, 7)], [(a, 11), (b, 15), (c, 7)],
[(a, 11), (b, 14), (c, 7)], [(a, 11), (b, 13), (c, 7)], [(a, 12), (b, 17), (c, 7)],
[(a, 13), (b, 17), (c, 7)], [(a, 14), (b, 17), (c, 7)], [(a, 15), (b, 17), (c, 7)],
[(a, 13), (b, 15), (c, 7)], [(a, 14), (b, 16), (c, 7)], [(a, 12), (b, 14), (c, 7)],
[(a, 12), (b, 15), (c, 7)], [(a, 13), (b, 16), (c, 7)], [(a, 13), (b, 15), (c, 8)],
[(a, 12), (b, 15), (c, 8)], [(a, 15), (b, 17), (c, 8)], [(a, 15), (b, 17), (c, 9)],
[(a, 15), (b, 17), (c, 10)], [(a, 15), (b, 17), (c, 11)], [(a, 14), (b, 16), (c,
10)], [(a, 14), (b, 17), (c, 10)], [(a, 14), (b, 17), (c, 9)], [(a, 14), (b, 17),
(c, 8)], [(a, 13), (b, 17), (c, 8)], [(a, 13), (b, 17), (c, 9)], [(a, 13), (b, 16),
(c, 9)], [(a, 14), (b, 16), (c, 9)], [(a, 13), (b, 16), (c, 8)], [(a, 14), (b, 16),
(c, 8)], [(a, 10), (b, 8), (c, 14)], [(a, 9), (b, 7), (c, 13)], [(a, 8), (b, 6),
(c, 12)], [(a, 7), (b, 5), (c, 12)], [(a, 6), (b, 4), (c, 12)], [(a, 7), (b, 4),
(c, 11)], [(a, 7), (b, 4), (c, 12)], [(a, 8), (b, 5), (c, 12)], [(a, 10), (b, 5),
(c, 14)], [(a, 10), (b, 6), (c, 14)], [(a, 9), (b, 6), (c, 13)], [(a, 8), (b, 6),
(c, 13)], [(a, 8), (b, 6), (c, 14)], [(a, 9), (b, 6), (c, 14)], [(a, 7), (b, 5),
(c, 13)], [(a, 6), (b, 4), (c, 13)], [(a, 7), (b, 4), (c, 13)], [(a, 8), (b, 4),
(c, 13)], [(a, 8), (b, 5), (c, 13)], [(a, 9), (b, 5), (c, 13)], [(a, 9), (b, 4),
(c, 13)], [(a, 10), (b, 7), (c, 14)], [(a, 9), (b, 7), (c, 14)], [(a, 7), (b, 5),
(c, 11)], [(a, 8), (b, 5), (c, 14)], [(a, 9), (b, 5), (c, 14)], [(a, 7), (b, 5),
(c, 14)], [(a, 6), (b, 4), (c, 11)], [(a, 8), (b, 4), (c, 12)], [(a, 10), (b, 4),
(c, 14)], [(a, 9), (b, 4), (c, 14)], [(a, 8), (b, 4), (c, 14)], [(a, 6), (b, 4),
(c, 14)], [(a, 7), (b, 4), (c, 14)]]
```

nを変更した時のニッコリの最大値

- n = 3 .. 4 ➔ 0　(すべて勝ち宣言可能)
- n = 5 .. 8 ➔ 1　(すべて勝ち宣言可能)
- n = 9 .. 14 ➔ 2　(すべて勝ち宣言可能)
- n = 15 .. 22 ➔ 3 (n=15では勝ち宣言不可)
- n = 23 .. 32 ➔ 4 (n=23,24では勝ち宣言不可)
- n = 33 .. 44 ➔ 5 (n=33 ~ 35では勝ち宣言不可)
- n = 45 .. 58 ➔ 6 (n=45 ~ 48では勝ち宣言不可)
- n = 59 .. 74 ➔ 7 (n=59 ~ 63では勝ち宣言不可)

$N(n) = \left\lfloor \frac{\sqrt{4n-11}-1}{2} \right\rfloor$ と予想される(未証明)

まとめ

- SMTソルバを用いて「竹内の3人の賢者問題」を解いた
  - 効率はともかく簡潔に書ける(おそらくSATソルバ，CSPソルバ，IPソルバでも解ける).
- nを変更した時のニッコリの最大値
  - 予想はできるが未証明
- 元の問題に「エレガントな解法」があるか? ➔ 不明
- プログラム及びスライドは(https://github.com/u-tokyo-gps-tanaka-lab/three_wise_men) で公開します.