Almost Locked Set 入門

Almost Locked Set（以下、ALS）は Wing 系手筋を一般化した手筋です。ALS 自体は大抵の盤面に対して５個程度、削除可能な候補の重複を除いても実質１個～２個存在します。ロジック自体も比較的単純です。まずは最も基本的な ALS である XY-Wing を紹介します。

まずは XY-Wing を理解する

XY-Wing は３つのセルと、３つの候補数字だけを利用する ALS の基本形です。この３つの候補数字を X,Y,Z とします。X,Y,Z は１～９の互いに異なる候補数字です。盤面にこれらの候補数字が次のように配置されていたとします。

```
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
|XY * * | - - - | -XZ - |
| - - - | - - - | - - - |
| -YZ - | - - - | - - - |
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
```

このとき候補数字 XY を含むセルをピボット（pivot）、XZ と YZ を含むセルをピンサー（pincer）と呼びます。ピボットの値は X か Y のどちらかです。したがってピボットがもし X であったときは XZ は Z に確定します。ピボットがもし Y であれば YZ は Z に確定します。つまり**ピボットが X,Y いずれの場合であってもピンサーのどちらか一方は必ず Z** です。したがって両方のピンサーを「同時に見ることが出来る」位置にあるセルに含まれている Z は削除できます。図では「＊」で示したセルにある候補数字 Z は削除できると言うことです。

「同時に見ることができる」という意味は次の通りです。

定義：「候補数字 A が、候補数字 B と C を同時に見ることができる」とは、
・候補数字 A,B,C が同じ数字である。
・A と B が同じユニット[1]内の異なるセルに属している。
・A と C が同じユニット内の異なるセルに属している。
・B と C が同じユニットに属していない。
以上すべてを同時に満たすこと。

もう少し直感的に表現すると、B と C が A のハウス[2]に属していて同じ数字であるということです。ピボットの候補数字を XY としているので、この手筋は XY-Wing と呼ばれます。XY-Wing の紹介は以上で終了です。

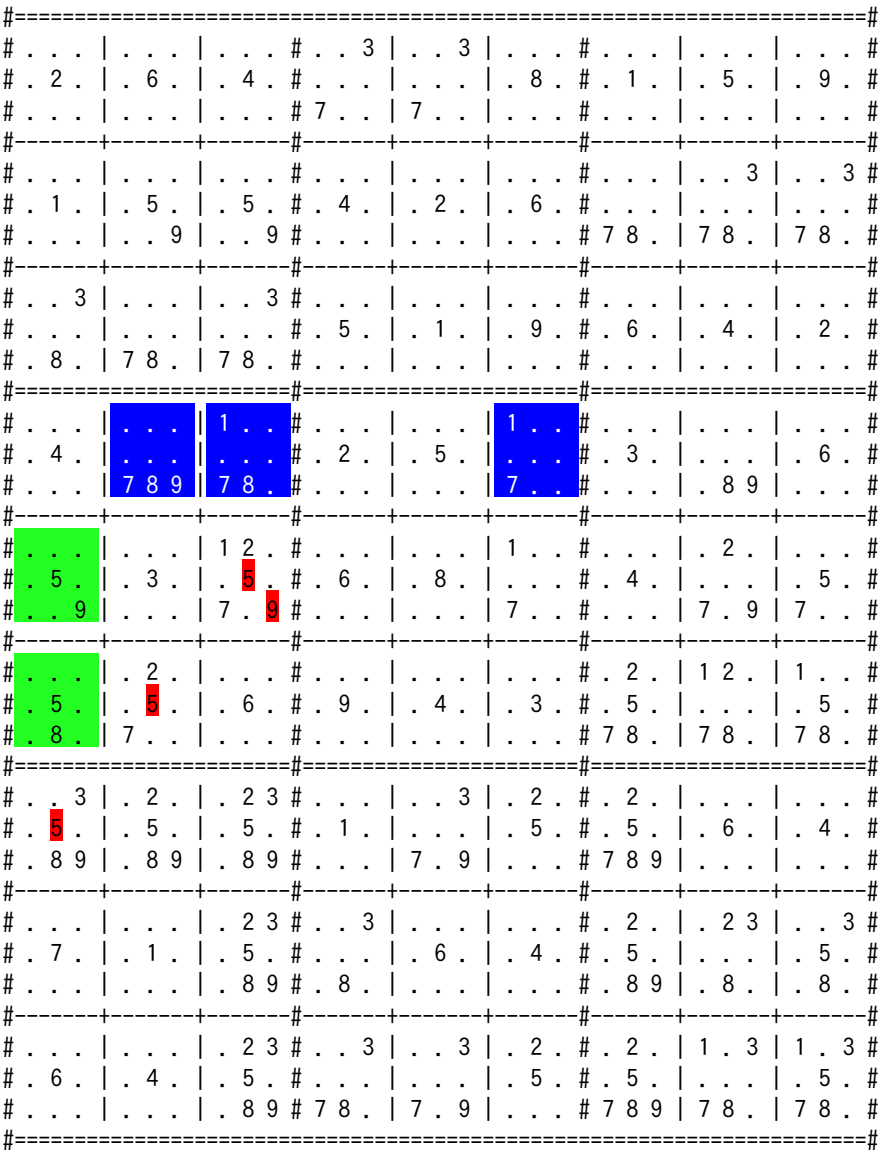
1　ユニット＝あるセルに対する行、列、ブロックのいずれか。

2　ハウス＝あるセルに対する行、列、ブロックの和集合。

XY-Wing を ALS に拡張する

まず基本となる ALS（Almost Locked Set）という考え方を紹介します。ALS とはひとつのハウス内の n 個のセルに n+1 個の候補数字が存在するセルの組のことです。n 個のセルに N 個の候補数字が配置されている組を Locked Set と呼ぶので、n 個のセルに n+1 個の候補数字が配置されている組を Almost Locked Set と呼びます。最小の ALS は 1 個のセルに 2 個の候補数字が配置されている組（＝2 値セル）です。

次の図の盤面には沢山の ALS がありますが、例えば 4 行 2 列（以降では 4 行 2 列を r4c2 と表記します）と r4c3、そして r4c6 は 3 個のセルに対して{1,7,8,9}の 4 個（種類）の候補数字があるので ALS です。また r5c1 と r6c1 は 2 個のセルに対して{5,8,9}の 3 個の候補数字があるので、これも ALS です。



ここで r4c2 と r4c3、そして r4c6 の 3 つのセルで構成される ALS に「A」という名前を付けて、これを A={1,7,8,9}と書き、r5c1 と r6c1 の 2 つのセルで構成される ALS に「B」という名前を付けて B={5,8,9}と書きます。A と B を合わせると 5 つのセルに 5 個の候補数

字が存在するので、この5個の候補数字は必ずこの5個のセルのどこかに配置されることが確定します（Locked Set）。AとBに共通する8と9は特別です。どちらの候補数字も

条件）2つのALSに共通の候補数字であり、互いに同じユニット内に存在する。

つまりそれぞれの候補数字はリンクしています[3]。2つのALS内の候補数字がリンクしている場合、このリンクを「**相互リンク**」と呼びます。また、相互リンクのラベルとなっている候補数字を**restricted common**と呼びます。

セルを共有しない[4]2つのALSに対して**Restricted commonが2つ存在する場合**は特別な条件が成り立ちます。上の例の場合、8がAの要素であると仮定するとAの8とBの8はリンク関係にあるのでBが8を含むことはあり得ません。逆の場合も同様です。8がどちらのALSにも存在しないと仮定するのは条件に反しており矛盾です。

また、8と9の両方がAに存在すると仮定するとAの残り2個のセルに、Aにしか存在し得ない1と7の両方を含むことはできないのでやはり矛盾。Bに両方が存在すると仮定した場合もBにしか存在し得ない5を納めるセルがなくなり同様に矛盾。以上から8と9はAまたはBに同時に存在することはあり得ません。

一般化した場合も同じことが成立します。Aをn個のセルにn+1個の候補数字を持つALS、Bをm個のセルにm+1個の候補数字を持つALS、AとBの2つのrestricted commonをRC（|RC|=2）とします。A∪Bは全体としてLocked Setなので

・RC∈Aの場合
　（1）RCの候補数字が同一のセルに存在（確定）する場合。
　　==> ひとつのセルが2つの確定値を持つことは数独のルール上許されない。
　（2）RCの候補数字がAの2つのセルに分散して存在する場合
　　==> Aの残りのn-2個のセルにAにしか存在し得ないn-1個の候補数字が存在することになり必要なセルが不足するので矛盾。
よってAに2つのrestricted commonが存在することはあり得ない。

RC∈Bの場合も同じ議論が成立するのでRCがAとBの相互リンクであることと併せると、**AとBのそれぞれには2つのrestricted commonのそれぞれが排他的に存在する**ことになります。たとえば8がAに含まれるならば9はAには含まれずBに含まれます。逆に9がAに含まれるならば8はAに含まれずBに含まれるということです。

したがって**AとBのそれぞれのrestricted commonのすべてを同時に見ることのできる位置にあるrestricted commonと同じ値の候補数字は削除できます**。

更にAとBのrestricted commonはA、Bそれぞれにひとつだけが排他的に存在できるので、**restricted commonではないAの要素はAだけに含まれ、restricted commonではないBの要素はBだけに含まれる**ということも分かります。なぜならばAに存在できるrestricted commonはひとつのみなので、Aのn個のセルのひとつにはrestricted

3　リンク関係はstrong linkでもweak linkでもよい。Strong linkとweak linkについては「Nice Loop入門」を参照。
4　必要な条件です。

common のひとつが、残り n-1 個のセルには restricted　common 以外の n-1 個の候補数字が存在することになります。A と B の共通要素は restricted　common である 2 つだけなので restricted　common 以外の A に含まれる n-1 個の候補数字は A にしか存在できません。B についても同様です。

以上を総合して整理すると、次の定理が成立することが分かります。
定理１：2 つの ALS 同士の間に 2 つの相互リンクが存在するならば、双方の ALS 内の「すべての候補数字 K」を見ることができる位置にある候補数字 K は削除できる。

つまり上の例の場合、赤字で示した 5 と 9 が削除できます。

同じ盤面に対して r8c4、r8c8、r8c9 の候補数字は A={2,3,5,8}という ALS を構成しており、r2c7、r7c7、r8c7、r9c7 の候補数字は B={2,5,7,8,9}という ALS を構成しています。この 2 つの ALS は 2 と 5 により相互にリンクしています。その結果、r8c3 の 3 は ALS 内のすべての 3 を見ることができるので削除できます。また r9c9 の 5 と r6c7 の 7 も同様の理由で削除できます。NumberPlace.lisp ではこれを

```
0:0> Almost Locked Set により[@]の位置から候補を削除できます。
r8c4,r8c8,r8c9 の候補数字[#]は Almost Locked Set [A]=(2 3 5 8)を構成しています。
r2c7,r7c7,r8c7,r9c7 の候補数字[$]は Almost Locked Set [B]=(2 5 7 8 9)を構成しています。
Almost Locked Set[A]と[B]は候補数字(2 5)により相互にリンクしています。
  ==> r8c3 の[3]は Almost Locked Set 内のすべての[3]を見ることができるので削除できます。
  ==> r9c9 の[5]は Almost Locked Set 内のすべての[5]を見ることができるので削除できます。
  ==> r6c7 の[7]は Almost Locked Set 内のすべての[7]を見ることができるので削除できます。
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | $ - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | @ - - |
+-------+-------+-------+
| - - - | - - - | $ - - |
| - - @ | # - - | $ # # |
| - - - | - - - | $ - @ |
+-------+-------+-------+
0:0> Almost Locked Set により[@]の位置から候補を削除できます。
r4c2,r4c3,r4c6 の候補数字[#]は Almost Locked Set [A]=(1 7 8 9)を構成しています。
r5c1,r6c1 の候補数字[$]は Almost Locked Set [B]=(5 8 9)を構成しています。
Almost Locked Set[A]と[B]は候補数字(8 9)により相互にリンクしています。
  ==> r5c3,r6c2,r7c1 の[5]は Almost Locked Set 内のすべての[5]を見ることができるので削除できます。
  ==> r5c3 の[9]は Almost Locked Set 内のすべての[9]を見ることができるので削除できます。
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - # # | - - # | - - - |
| $ - @ | - - - | - - - |
| $ @ - | - - - | - - - |
+-------+-------+-------+
| @ - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
```

と表示します。この表示の直後には候補数字を削除した結果を整理した盤面が表示されます。

```
#=====================================================================#
#  .  .  .  |  .  .  .  |  .  .  #  .  .  3  |  .  .  3  |  .  .  #  .  .  .  |  .  .  .  |  .  .  .  #
#  .  2  .  |  .  6  .  |  .  4  #  .  .  .  |  .  .  .  |  .  8  #  .  1  .  |  .  5  .  |  .  9  .  #
#  .  .  .  |  .  .  .  |  .  .  #  7  .  .  |  7  .  .  |  .  .  #  .  .  .  |  .  .  .  |  .  .  .  #
#-------+-------+-------#-------+-------+-------#-------+-------+-------#
#  .  .  .  |  .  .  .  |  .  .  #  .  .  .  |  .  .  .  |  .  .  .  #  .  .  .  |  .  .  3  |  .  .  3  #
#  .  1  .  |  .  5  .  |  .  5  #  .  4  .  |  .  2  .  |  .  6  .  #  .  .  .  |  .  .  .  |  .  .  .  #
#  .  .  .  |  .  .  9  |  .  .  9  #  .  .  .  |  .  .  .  |  .  .  .  #  7  8  .  |  7  8  .  |  7  8  .  #
#-------+-------+-------#-------+-------+-------#-------+-------+-------#
#  .  .  3  |  .  .  .  |  .  .  3  #  .  .  .  |  .  .  .  |  .  .  .  #  .  .  .  |  .  .  .  |  .  .  .  #
#  .  .  .  |  .  .  .  |  .  .  .  #  .  5  .  |  .  1  .  |  .  9  .  #  .  6  .  |  .  4  .  |  .  2  .  #
#  .  8  .  |  7  8  .  |  7  8  .  #  .  .  .  |  .  .  .  |  .  .  .  #  .  .  .  |  .  .  .  |  .  .  .  #
#=====================#=====================#=====================#
#  .  .  .  |  .  .  .  |  1  .  .  #  .  .  .  |  .  .  .  |  1  .  .  #  .  .  .  |  .  .  .  |  .  .  .  #
#  .  4  .  |  .  .  .  |  .  .  .  #  .  2  .  |  .  5  .  |  .  .  .  #  .  3  .  |  .  .  .  |  .  6  .  #
#  .  .  .  |  7  8  9  |  7  8  .  #  .  .  .  |  .  .  .  |  7  .  .  #  .  .  .  |  .  8  9  |  .  .  .  #
#-------+-------+-------#-------+-------+-------#-------+-------+-------#
#  .  .  .  |  .  .  .  |  1  2  .  #  .  .  .  |  .  .  .  |  1  .  .  #  .  .  .  |  .  2  .  |  .  .  .  #
#  .  5  .  |  .  3  .  |  .  .  .  #  .  6  .  |  .  8  .  |  .  .  .  #  .  4  .  |  .  .  .  |  .  5  .  #
#  .  .  9  |  .  .  .  |  7  .  .  #  .  .  .  |  .  .  .  |  7  .  .  #  .  .  .  |  7  .  9  |  7  .  .  #
#-------+-------+-------#-------+-------+-------#-------+-------+-------#
#  .  .  .  |  .  2  .  |  .  .  .  #  .  .  .  |  .  .  .  |  .  .  .  #  .  2  .  |  1  2  .  |  1  .  .  #
#  .  5  .  |  .  .  .  |  .  6  .  #  .  9  .  |  .  4  .  |  .  3  .  #  .  5  .  |  .  .  .  |  .  5  .  #
#  .  8  .  |  7  .  .  |  .  .  .  #  .  .  .  |  .  .  .  |  .  .  .  #  .  8  .  |  7  8  .  |  7  8  .  #
#=====================#=====================#=====================#
#  .  .  3  |  .  2  .  |  .  2  3  #  .  .  .  |  .  .  3  |  .  2  .  #  .  2  .  |  .  .  .  |  .  .  .  #
#  .  .  .  |  .  5  .  |  .  5  .  #  .  1  .  |  .  .  .  |  .  5  .  #  .  5  .  |  .  6  .  |  .  4  .  #
#  .  8  9  |  .  8  9  |  .  8  9  #  .  .  .  |  7  .  9  |  .  .  .  #  7  8  9  |  .  .  .  |  .  .  .  #
#-------+-------+-------#-------+-------+-------#-------+-------+-------#
#  .  .  .  |  .  .  .  |  .  2  .  #  .  .  3  |  .  .  .  |  .  .  .  #  .  2  .  |  .  2  3  |  .  .  3  #
#  .  7  .  |  .  1  .  |  .  5  .  #  .  .  .  |  .  6  .  |  .  4  .  #  .  5  .  |  .  .  .  |  .  5  .  #
#  .  .  .  |  .  .  .  |  .  8  9  #  .  8  .  |  .  .  .  |  .  .  #  .  8  9  |  .  8  .  |  .  8  .  #
#-------+-------+-------#-------+-------+-------#-------+-------+-------#
#  .  .  .  |  .  .  .  |  .  2  3  #  .  .  3  |  .  .  3  |  .  2  .  #  .  2  .  |  1  .  3  |  1  .  3  #
#  .  6  .  |  .  4  .  |  .  5  .  #  .  .  .  |  .  .  .  |  .  5  .  #  .  5  .  |  .  .  .  |  .  .  .  #
#  .  .  .  |  .  .  .  |  .  8  9  #  7  8  .  |  7  .  9  |  .  .  .  #  7  8  9  |  7  8  .  |  7  8  .  #
#=====================================================================#
```

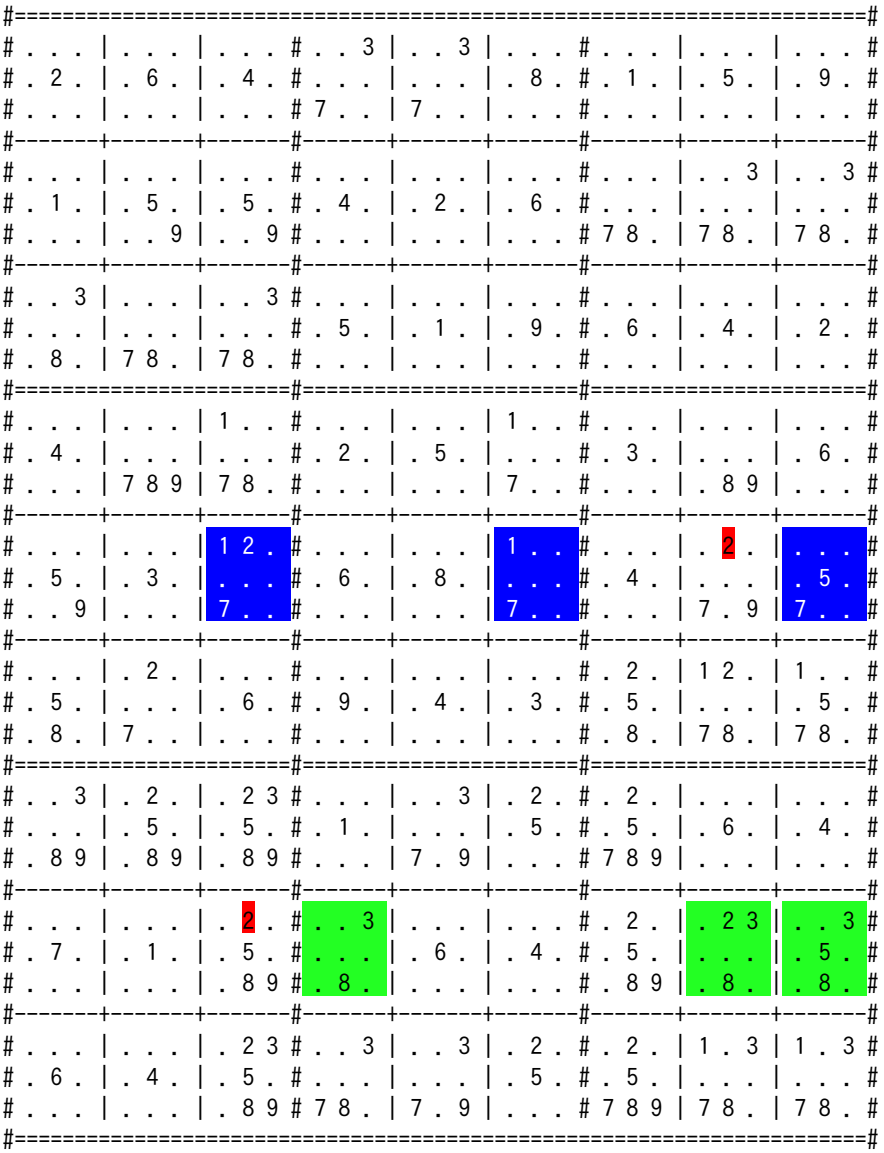
r7c1 の 9 は r4c2 の 9 を見ることはできないので削除できないことに注意して下さい。

共通候補があれば RC がひとつでも成立する

先ほどは 2 つの ALS に対して 2 つの restricted common が存在する場合を紹介しました。「2 つの restricted common」を「ひとつの restricted common とひとつ（以上）の共通候補」に置き換えても ALS が成立します。ただし削除できる候補の条件が若干変わります。そして実は、これから紹介する「ひとつの restricted common とひとつ（以上）の共通候補」が存在するケースの方が高い確率で存在します。

ALS 適用後の上の盤面に別の ALS が存在するので ALS の組に彩色して再掲します。次の図の r5c,r5c6,r5c9 の候補数字は A={1,2,5,7}という ALS を構成しています。そして r8c4, r8c8,r8c9 の候補数字は B={2,3,5,8}という ALS を構成しています。A と B は候補数字 5 に

より相互にリンクし,共通の候補数字２が存在します。



先ほどと同じようにＡをｎ個のセルにn+1個の候補数字が存在するALS、Ｂをｍ個のセルにm+1個の候補数字が存在するALSであるとします。Restricted common(=RC)はひとつだけ存在し、restricted commonではない共通の候補数字が少なくともひとつ存在すると仮定します。

もし、この共通の候補数字がどちらのALSにも存在しない（＝確定値として残らない）と仮定するとＡにはｎ個、Ｂにはｍ個の候補数字が存在することになります。更にrestricted commonがひとつあるので、これはＡとＢのどちらか一方にしか存在しません。したがってＡとＢの候補数字の個数の和からひとつ分の個数を減じる必要があります。故にＡとＢの併せてn+m個のセルにn+m-1個の候補数字しか存在しないことになり矛盾です。共通の候補数字がどちらのALSにも存在しないと仮定したために矛盾が生じたのですから、最初の仮定に誤りがあっということです。すなわち共通の候補数字はＡかＢの少なくともどちらか一方には存在すると言うことです。

以上から、以下の定理が成立することが分かります。

定理２：２つのALSが候補数字「i」を介してリンクしており、「i」とは異なる共通の候補数字「k」が存在するならば、２つのALS内のすべての「k」を見ることができる位置にある、ALSの要素でない「k」は削除できる。

上の図ではr5c8の２とr8c3の２が２つのALS内のすべての２を見ることができる位置にあるので削除できます。NumberPlace.lispは、これを次のように表示します。なお、この盤面には定理１で紹介したタイプのALSも成立しているので同時に表示されます。

```
0:0> Almost Locked Setにより[@]の位置から候補を削除できます。
r5c3,r5c6,r5c9の候補数字[#]はAlmost Locked Set [A]=(1 2 5 7)を構成しています。
r8c4,r8c8,r8c9の候補数字[$]はAlmost Locked Set [B]=(2 3 5 8)を構成しています。
Almost Locked Set[A]と[B]は候補数字(5)により相互にリンクし,共通の候補数字(2)が存在します。
  ==> r5c8,r8c3の[2]はAlmost Locked Set内のすべての[2]を見ることができるので削除できます。
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - - - | - - - | - - - |
| - - # | - - # | - @ # |
| - - - | - - - | - - - |
+-------+-------+-------+
| - - - | - - - | - - - |
| - - @ | $ - - | - $ $ |
| - - - | - - - | - - - |
+-------+-------+-------+
0:0> Almost Locked Setにより[@]の位置から候補を削除できます。
r5c1,r5c3,r5c6,r5c9の候補数字[#]はAlmost Locked Set [A]=(1 2 5 7 9)を構成しています。
r3c2,r4c2,r6c2の候補数字[$]はAlmost Locked Set [B]=(2 7 8 9)を構成しています。
Almost Locked Set[A]と[B]は候補数字(2 9)により相互にリンクしています。
  ==> r7c2の[8]はAlmost Locked Set内のすべての[8]を見ることができるので削除できます。
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - $ - | - - - | - - - |
+-------+-------+-------+
| - $ - | - - - | - - - |
| # - # | - - # | - - # |
| - $ - | - - - | - - - |
+-------+-------+-------+
| - @ - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
```

以上で手筋としてのAlmost Locked Setの紹介は終了です。ALSは何故成立するのかを理解してしまえば手筋を実際に実行することは比較的容易です。ただし現実的には２つの問題があります。ひとつは盤面中に存在するALSが沢山ありすぎるという点。もうひとつは同一の候補数字を削除できるALSの組み合わせが複数あるという点です。

この記事冒頭の盤面（２ページ参照）には解説に使用しているALSを含めて173個のALSが存在します。これらALSの任意の２つを組み合わせて定理１，あるいは定理２が成立していないかをチェックすることになりますが173個のALSから２個を選ぶ組み合わせは１万4878パターン存在します。この中に実際に候補数字を削除可能な組み合わせは７種類あります。組み合わせによっては一度に複数の候補数字を削除できる場合もありますが、

ひとつの候補数字しか削除できない場合もあります。候補数字を削除可能な組み合わせの7種類は実際には次の通りです。

この盤面には173個のALSが存在します。
これら173個のALSから2個を選ぶ組み合わせ14878パターンをチェックしました。
候補数字を削除可能なパターンは7種類ありました。
0:0> Almost Locked Setにより[@]の位置から候補を削除できます。
r4c2,r4c3,r5c1,r6c1の候補数字[#]はAlmost Locked Set [A]=(1 5 7 8 9)を構成しています。
r4c6の候補数字[$]はAlmost Locked Set [B]=(1 7)を構成しています。
Almost Locked Set[A]と[B]は候補数字(1 7)により相互にリンクしています。
  ==> r5c3,r6c2,r7c1の[5]はAlmost Locked Set内のすべての[5]を見ることができるので削除できます。
  ==> r5c3の[9]はAlmost Locked Set内のすべての[9]を見ることができるので削除できます。

```
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - # # | - - $ | - - - |
| # - @ | - - - | - - - |
| # @ - | - - - | - - - |
+-------+-------+-------+
| @ - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
```

0:0> Almost Locked Setにより[@]の位置から候補を削除できます。
r5c1,r6c1の候補数字[#]はAlmost Locked Set [A]=(5 8 9)を構成しています。
r5c8,r5c9,r6c7,r6c8,r6c9の候補数字[$]はAlmost Locked Set [B]=(1 2 5 7 8 9)を構成しています。
Almost Locked Set[A]と[B]は候補数字(8 9)により相互にリンクしています。
  ==> r5c3の[9]はAlmost Locked Set内のすべての[9]を見ることができるので削除できます。

```
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - - - | - - - | - - - |
| # - @ | - - - | - $ $ |
| # - - | - - - | $ $ $ |
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
```

0:0> Almost Locked Setにより[@]の位置から候補を削除できます。
r4c6,r7c6,r9c6の候補数字[#]はAlmost Locked Set [A]=(1 2 5 7)を構成しています。
r4c2,r4c3,r5c1,r6c1の候補数字[$]はAlmost Locked Set [B]=(1 5 7 8 9)を構成しています。
Almost Locked Set[A]と[B]は候補数字(1 7)により相互にリンクしています。
  ==> r5c3の[9]はAlmost Locked Set内のすべての[9]を見ることができるので削除できます。

```
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - $ $ | - - # | - - - |
| $ - @ | - - - | - - - |
| $ - - | - - - | - - - |
+-------+-------+-------+
| - - - | - - # | - - - |
| - - - | - - - | - - - |
| - - - | - - # | - - - |
+-------+-------+-------+
```

0:0> Almost Locked Set により[@]の位置から候補を削除できます。
r8c4,r8c8,r8c9 の候補数字[#]は Almost Locked Set [A]=(2 3 5 8)を構成しています。
r4c8,r5c8,r5c9,r6c8,r6c9 の候補数字[$]は Almost Locked Set [B]=(1 2 5 7 8 9)を構成しています。
Almost Locked Set[A]と[B]は候補数字(2 5)により相互にリンクしています。
  ==> r8c3 の[3]は Almost Locked Set 内のすべての[3]を見ることができるので削除できます。
  ==> r9c9 の[5]は Almost Locked Set 内のすべての[5]を見ることができるので削除できます。
  ==> r6c7 の[7]は Almost Locked Set 内のすべての[7]を見ることができるので削除できます。
```
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - - - | - - - | - $ - |
| - - - | - - - | - $ $ |
| - - - | - - - | @ $ $ |
+-------+-------+-------+
| - - - | - - - | - - - |
| - - @ | # - - | - # # |
| - - - | - - - | - - @ |
+-------+-------+-------+
```
0:0> Almost Locked Set により[@]の位置から候補を削除できます。
r8c4,r8c8,r8c9 の候補数字[#]は Almost Locked Set [A]=(2 3 5 8)を構成しています。
r2c7,r7c7,r8c7,r9c7 の候補数字[$]は Almost Locked Set [B]=(2 5 7 8 9)を構成しています。
Almost Locked Set[A]と[B]は候補数字(2 5)により相互にリンクしています。
  ==> r8c3 の[3]は Almost Locked Set 内のすべての[3]を見ることができるので削除できます。
  ==> r9c9 の[5]は Almost Locked Set 内のすべての[5]を見ることができるので削除できます。
  ==> r6c7 の[7]は Almost Locked Set 内のすべての[7]を見ることができるので削除できます。
```
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | $ - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | @ - - |
+-------+-------+-------+
| - - - | - - - | $ - - |
| - - @ | # - - | $ # # |
| - - - | - - - | $ - @ |
+-------+-------+-------+
```
0:0> Almost Locked Set により[@]の位置から候補を削除できます。
r5c1,r5c8,r5c9 の候補数字[#]は Almost Locked Set [A]=(2 5 7 9)を構成しています。
r6c1,r6c7,r6c8,r6c9 の候補数字[$]は Almost Locked Set [B]=(1 2 5 7 8)を構成しています。
Almost Locked Set[A]と[B]は候補数字(2 7)により相互にリンクしています。
  ==> r5c3 の[9]は Almost Locked Set 内のすべての[9]を見ることができるので削除できます。
```
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - - - | - - - | - - - |
| # - @ | - - - | - # # |
| $ - - | - - - | $ $ $ |
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
```
0:0> Almost Locked Set により[@]の位置から候補を削除できます。
r4c2,r4c3,r4c6 の候補数字[#]は Almost Locked Set [A]=(1 7 8 9)を構成しています。
r5c1,r6c1 の候補数字[$]は Almost Locked Set [B]=(5 8 9)を構成しています。
Almost Locked Set[A]と[B]は候補数字(8 9)により相互にリンクしています。

```
  ==> r5c3,r6c2,r7c1 の[5]は Almost Locked Set 内のすべての[5]を見ることができるので削除できます。
  ==> r5c3 の[9]は Almost Locked Set 内のすべての[9]を見ることができるので削除できます。
+-------+-------+-------+
| - - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
| - # # | - - # | - - - |
| $ - @ | - - - | - - - |
| $ @ - | - - - | - - - |
+-------+-------+-------+
| @ - - | - - - | - - - |
| - - - | - - - | - - - |
| - - - | - - - | - - - |
+-------+-------+-------+
```

候補数字を削除できるすべてのパターンを確認したい場合は便利ですが、別の削除可能パターンに含まれているパターンまで表示するのは冗長な場合もあります。そこで NumberPlace.lisp では（青字が入力部分）

```
cl-user> (als-show-all t)
```

とすると ALS のすべての削除可能パターンを表示し、

```
cl-user> (als-show-all nil)
```

とすると冗長な削除可能パターンを省略して、一度に多数の候補数字を削除可能で、なおかつ可能な限り少ないセル数で ALS を構成できる効率的なパターンだけを表示します。その場合でも削除可能な候補数字が抜けることはありません。この盤面での「効率的なパターン」は 4 ページに示している 2 種類です。この 2 種類だけで他の 5 種類の ALS で削除可能な候補数字すべてをカバーしています。確認してみて下さい。

盤面中に存在する ALS の数と、NumberPlace.lisp が実際にチェックしたパターン数を表示したい場合は

```
cl-user> (als-show-stat t)
```

とします。表示させたくない場合は

```
cl-user> (als-show-stat nil)
```

とします。(als-show-stat t)と設定した状態では

```
この盤面には 173 個の ALS が存在します。
これら 173 個の ALS から 2 個を選ぶ組み合わせ 14878 パターンをチェックしました。
候補数字を削除可能なパターンは 7 種類ありました。
```

のように ALS 実行毎に情報が表示されます。ここで紹介した盤面は「als-01」という名前で NumberPlace.lisp に登録してあるので

```
cl-user> (teach als-01)
```

と入力することですべての出力を得ることが出来ます。その他のコマンドについて知りたい場合は

cl-user> (help)

と入力してみて下さい。

```
NumberPlace.lisp 5.0.3

(help '?)                       個別に解説可能な項目の一覧を表示する。
(teach board)                   解と解法過程を表示する。
(plot board)                    解法過程の難易度をグラフ表示する。
(stat  board)                   解と試行錯誤回数等の情報だけを表示する。
(simple-answer board)           問題自身と手筋適用後の盤面＋解を表示する。
(numberplace board)             解のリストを返す。エンジン部分。
(pencil-mark board)             刈り込みだけを行って盤面を表示する。
(enter-board)                   候補数字の初期値を入力する。空マスには[0]を入力。
(edit-board board)              ボードの候補数字を編集する。

表示制御関連の関数：
(print-mini {t|nil})            盤面をコンパクトなサイズで表示するか設定する。
(print-normal {t|nil})          盤面を候補数字付きのサイズで表示するか設定する。
(pencil-mark {t|nil})           盤面の候補数字を固定位置に表示するか設定する。
(color-mode {0|1|2})            Advanced Coloring の盤面表示に使用するカラー表示レベル。
(als-show-stat {t|nil})         Almost Locked Set に関する統計情報を表示するか設定する。
(set-parity color-1 color-2)    Advanced Coloring の盤面表示で使用する色指定。
(pause {nil|1..n})              盤面を指定した数出力するごとに一時停止する。[nil]は一時停止なし。
(print-check {t|nil}            手筋を解説する小さな盤面を随時表示する。
(check-backtrack-point {t|nil}) 仮置きの際の盤面を出力するかどうかを設定する。
(need-multiple-answer {t|nil})  複数解を探索するかどうかを設定する。
(explanation-level {bn})        解法過程の解説表示レベルを設定する。
                                10 の位[b]が盤面表示レベル。1 の位[n]が解説表示レベル。
                                それぞれ[0]が出力なし,[1]が標準,[2]が全項目出力。標準は[11]。

動作制御関連の関数：
(n-grid-limit {nil|0..n})       n-grid の上限を設定する。[nil]は上限なし。
(tuples-limit {nil|0..n})       n 国同盟の上限を設定する。[nil]は上限なし。
(max-nice-length {nil|3..n})    Nice Loop の連鎖セル数上限を設定する。[nil]は上限なし。
(max-nice-loops {nil|1..n})     盤面ごとに採用する Nice Loop の最大数。[nil]は上限なし。
(als-show-all {t|nil})          ALS で効率的パターンのみを表示するかを設定する。
(gb-als-show-all {t|nil})       GB-ALS で効率的パターンのみを表示するかを設定する。
(easy-method-first {t|nil})     易しい手筋を最優先で適用する。
(think-depth {nil|1..n})        [n]手先まで読んで最善の手筋を適用する。[nil]は先読みなし。
(find-logical-path board)       すべての手筋の組み合わせを尽くして解に到達できる手筋を探す。
(evil-boards)                   find-logical-path でも解けなかった盤面を呼び出す。
(print-env)                     現在の設定値を表示する。

動作制御関数のプリセット：
(novice-level)                  初心者向けの動作設定。使用する手筋を限定。
(middle-level)                  初級から中級者向けの設定。
(senior-level)                  中級から上級者向けの設定。
(advanced-level)                上級者向けの設定。
(machine-level)                 超上級者向けの設定。
(speed-first)                   速度最優先の設定。途中経過表示も一切なし。
nil
cl-user> (help '?)
(set-parity color-mode nice-loop advanced-coloring almost-locked-set)
cl-user> (help 'almost-locked-set)
```

定義：Almost Locked Set とは n+1 個の候補数字が n 個のセルに配置されているセルの集合である。

定理１：２つの Almost Locked Set 同士の間に２つの link が存在するならば、双方の集団内のすべての候補数字[k]を見ることができる位置にある候補数字[k]は削除できる。

定理２：２つの Almost Locked Set が候補数字[i]を介して link しており、[i]とは異なる共通の候補数字[k]が存在するならば、２つの Almost Locked Set 内のすべての[k]を「見ることができる」位置にある Almost Locked Set の要素でない[k]は削除できる。

```
t
cl-user>
```

上記のような簡単なヘルプ・メッセージが表示されます。

参考 url

http://www.stolaf.edu/people/hansonr/sudoku/explain.htm （例題出典もこちら）

http://sudopedia.org/wiki/Solving_technique （用語の定義等）

（mail2daigo@gmail.com）■