Sudoku Players' Forums

Fully supersymmetric chains

Goto page Previous 1, 2, 3 ... 12, 13, 14

newtopic postreply **Sudoku Players' Forums Forum Index -> Advanced solving techniques**

View previous topic :: View next top

Author	Message
denis_berthier Joined: 19 Jun 2007 Posts: 579 Location: Paris, France	Posted: Mon Oct 13, 2008 2:33 am Post subject: quote

ttt wrote:

> Above puzzle I found SK loop for (27 & 16) at start as beloow:
>
> => r3c2<>2, r4c3<>8, r5c3<>48, r9c4<>9, r9c6<>49, r6c8<>8, r5c8<>89, r2c6<>9, r7c1<>6, r8c2<>1, r8c7<>2, r7c9<>7.

Thanks for noticing this.
I usually don't activate my rule for x2y2-belts, my interpretation of the "SK loop", defined here: http://www.sudoku.com/boards/viewtopic.php?t=5894

If I activate it, I find, just after the my first PM, that it eliminates the following candidates:
r5c3 <> 8, r5c3 <> 4, r4c3 <> 8, r3c1 <> 2, r2c6 <> 9, r6c8 <> 8, r5c8 <> 9, r5c8 <> 8, r9c6 <> 9, r9c6 <> 4, r9c4 <> 9, r8c2 <> 1, r7c1 <> 6, r8c7 <> r7c9 <> 7
The difference with you comes entirely from our different starting PMs.

After this, we have the same two eliminations as before, although with simpler rules.
nrczt-whip-rc[12] n9{r1c8 r1c6} - n9{r3c4 r3c1} - n9{r9c1 r9c2} - n9{r7c1 r7c4} - n6{r7c4 r9c4} - n1{r9c4 r9c6} - n7{r9c6 r9c9} - {n7 n8}r7c8 - {n8 n1}r
- n1{r2c9 r2c5} - n6{r2c5 r2c7} - {n6r1c8 .} ==> r8c8 <> 9
nrczt-whip-bn[9] {n9 n6}r1c8 - n6{r2c7 r2c5} - n1{r2c5 r2c6} - n1{r3c5 r3c8} - n9{r3c8 r7c8} - n8{r7c8 r8c8} - n2{r8c8 r9c7} - {n2 n7}r9c6 - {n7r9c9 .}
==> r2c9 <> 9

Then the following:
nrczt-whip-rc[12] n9{r1c8 r1c6} - n9{r3c4 r3c1} - n9{r9c1 r9c2} - n9{r7c1 r7c4} - n6{r7c4 r9c4} - n1{r9c4 r9c6} - n7{r9c6 r9c9} - {n7 n8}r7c8 - {n8 n1}r
- n1{r2c9 r2c5} - n6{r2c5 r2c7} - {n6r1c8 .} ==> r8c8 <> 9
nrczt-whip-bn[9] {n9 n6}r1c8 - n6{r2c7 r2c5} - n1{r2c5 r2c6} - n1{r3c5 r3c8} - n9{r3c8 r7c8} - n8{r7c8 r8c8} - n2{r8c8 r9c7} - {n2 n7}r9c6 - {n7r9c9 .}
==> r2c9 <> 9
nrczt-whip-rn[18] n9{r5c9 r5c7} - n3{r5c7 r1c7} - n4{r1c7 r3c9} - n4{r3c3 r1c3} - n7{r1c3 r2c2} - n9{r2c2 r2c1} - {n9 n8}r3c1 - n8{r5c1 r5c2} - n4{r5c2
r5c1} - {n4 n6}r9c1 - n6{r9c4 r7c4} - {n6 n8}r7c3 - n8{r7c8 r8c8} - n2{r8c8 r9c7} - n4{r9c7 r9c2} - n9{r9c2 r8c2} - n9{r8c6 r7c6} - {n4r7c6 .} ==> r5c9
<> 3
nrczt-whip-cn[19] n4{r7c6 r8c6} - n9{r8c6 r1c6} - {n9 n6}r1c8 - n6{r2c7 r2c5} - n7{r2c5 r2c2} - {n7 n4}r1c3 - n4{r1c7 r9c7} - n4{r9c2 r5c2} - n4{r5c1
r7c1} - n3{r7c1 r8c2} - n9{r8c2 r9c2} - n1{r9c2 r8c3} - n8{r8c3 r7c3} - {n8 n9}r7c8 - n9{r9c9 r5c9} - {n9 n5}r7c9 - {n5 n8}r8c7 - n8{r5c7 r5c1} - {n8r6c
.} ==> r7c6 <> 7
and no other whip.

Remember that my purpose was only to give an example of braids and of a solution with braids of a puzzle that is not solvable by nrczt-whips only.

Back to top profile pm www

Allan Barker Joined: 21 Feb 2008 Posts: 267 Location: Bangkok	Posted: Wed Oct 15, 2008 6:19 pm Post subject: quote

Why braid and whip'em if you can belt them once?

Hi Denis,

Denis B wrote:

> ..... any interesting example of a braid will be obtained for a puzzle with SER > 9.3. As a result, the solution to such a puzzle will require long chains and braids and it will seem very complex.

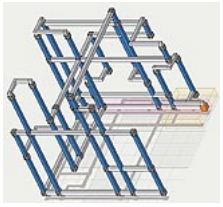
Following ttt's lead, above, I have found several rather simple SK type loops that eliminate large numbers of candidates like the 12 set example below that eliminates 15 candidates. This is probably similar to your belt or ttt's loop. The same candidates can be removed by applying many, often smaller, moves. I also entered one of your braids into my analyzer for comparison, and like you, to see what it looks like.

So the question is in the title. If a method such SK loops or belts can replace so much work for a simpler methods, if braids are relatively large can chainlike, th where do braids fit in the scheme of things? I.e., computational, human-useful, theoretical, etc.?

Denis B wrote:

> The only purpose is to show a few cases of how a braid can look.

They look like this. Click to Enlarge:



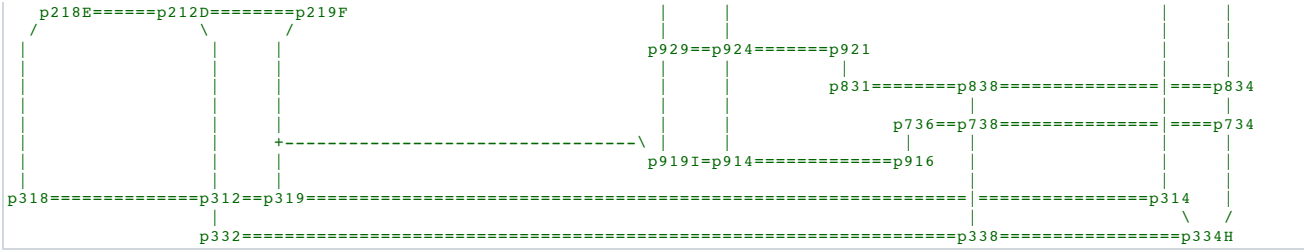
Code:

```
 Easy SK loop
DBT3 34 Candidates, Rank 0:
     12 Sets  = {8r3 7n38 8n38 9n1279 49b1 9b3}
     12 Links = {9r2 49r9 48c3 89c8 3n1 16b7 27b9}
 --> (9r2) => r2c6<>9, (3n1) => r3c1<>2, (8c3) => r4c3<>8, (4c3) => r5c3<>4,
     (8c3) => r5c3<>8, (8c8) => r5c8<>8, (9c8) => r5c8<>9, (8c8) => r6c8<>8,
     (6b7) => r7c1<>6, (7b9) => r7c9<>7, (1b7) => r8c2<>1, (2b9) => r8c7<>2,
     (9r9) => r9c4<>9, (4r9) => r9c6<>4, (9r9) => r9c6<>9

                                                  p279========p189
                                                   |          p389
                                                  p229         |
                           p319==========================p219  |
                            |                                  |
              p338========p318==================================p388   |
               |           |                                   |     |
               |    p134==p314                                 |     |
               |    p334                                       |     |
               |     |                          p787==============p788==p789
               |     |                           |                 |
               |     |                           |    p882========p888
p831========p838==p834                           |     |
 |           |                                   |     |
 |    p736==p738                                 |     |
 |     |                        p994==p999==p997  |     |
 |     |                         |     |         |     |
 |     |                        p974==p979========p972
 |     |                         |     |
 |    p916====================p914==p919
 |                              |     |
p921============================p924==p929


= , strong links
| , weak links
p831, candidate pRCN

   |
 p279==p189  (e.g., two connected candidates: p189 and p389)
       p389  (are form a multi-candidate link between box and line)
       |
```

Your first braid example. Note: these diagrams are sometimes wide, to see them properly you may need to srtetch the Players' Forums window horizontally.

Code:

```
Big Braid 54 candidates
     18 Sets = {1n3578 2n1267 3n139 7n3 8n3 9n12679}
     24 Links = {346r1 289r2 4r3 2479r9 89c1 48c3 1c6 27b1 7b2 369b3 16b7}
     --> (9r2*9b3) => r2c9<>9


 p228====================p229=============================================================================================================p2
       |                      |
  |
       |                      |               p156====================p157====================p153
  |
       |                      |                |                       |                       \
 |
       |                      |                |                       |                        p173B==================p17
 |
       |                      |                |                       |                            \                  |
 |
       |                      | p189=======p186C                       |                             |                 |
 |
       |                      |  |          |                          |                             |                 |
 |
     p278====================p279G=======p276                          |                             |                 |
 |
       |                      |                                       |                             |                 |
 |
       |    p262==============p269====================p261=============p267                          |                 |
 |
       |     |                |                       |                                            p393==p394        |
 |
       |     |                |                       |                                                   |          /
 |
       |     |                |                       |                                                   |
p134A=====p137
       |     |                |                       |                                                   |     /
       |     |                |                     p961==p969==p964==p962==p967                           |     |
       |     |                |                           |     |     |     |                           |     |
       |     |                |                         p979==p974==p972    |                           |     |
       |     |                |                           |     |           |                           |     |
       |     |                |                         p999==p994========p997                          |     |
       /     \                /                           |     |                                       |     |
```



Back to top profile pm www

denis_berthier

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

Posted: Wed Oct 15, 2008 8:30 pm Post subject: quote

Hi **Allan**,
Seems you haven't read the warning in my post: "I can't guarantee here that the braids used are the shortest ones available. (The contrary is very likely, I didn even try to optimise their length.)" As a result, your comparison with shorter patterns is biased. (Moreover, I explicitly mentioned that my first braid could have been replaced by a much shorter whip, but I kept it for illustrating a 2D braid).

Your 3D representation of a braid is also biased, as your software can't represent linear structures such as chains or braids but it transforms them into complex nets. You loose the main property of chains or braids, which is that they have degree one at any point.

In this example, which is one of the simplest that can't be solved with whips only (SER 9.4), there may be other SK loops (but I'm not sure what your definition an SK loop is). I haven't looked for them, even after ttt's example, because that was not the purpose of this example.

> **Allan Barker wrote:**
> If a method such SK loops or belts can replace so much work for a simpler methods, if braids are relatively large can chainlike, then where do braids fit in the scheme of things? I.e., computational, human-useful, theoretical, etc.?

SK loops don't replace so much work in general. They are very rare patterns. Unless you have another definition of an SK Loop. Do you know of any result abou the scope of SK loops?

Braids are not large or small, they can be of any size. The braids you'll find will be large if you first try all the other patterns, as I did in this example. They will larger still if you don't try to find the shortest ones in any situation.

Are braids human-useful? Contrary to whips, that are chains, I'm tempted to answer no - as is the case for most nets. You shouldn't forget that we are dealing with exceptionally hard puzzles. If you didn't have your solver, would you find any solution of any of these puzzles?

Are braids computational tools?
If I wanted to find the shortest braids in any situation, as I do for all my other patterns, I'd have to put the corresponding rules in SudoRules. Computation time would be large.
If instead, I use T&E and I extract manually the corresponding braid from the output trace, computation time is a few milli-seconds, but I need at least one hou build the braid manually from the trace (as I did in my example above) and I'm quasi certain that this won't be the shortest one: very frustrating in practice but very reassuring about the usefulness of braids as a pattern for this length optimising purpose.
If I wanted to adapt the T&E algorithm so that its output always corresponds to the shortest available braid, that would require so many changes to it that it wo become very slow and it would amount to program braids.

Are braids a theoretical tool? For sure. The question of whether there was a well defined pattern able to do the same eliminations as T&E had been pending for years. We now have it. Moreover, we have the result that almost any puzzle can be solved with braids and any known puzzle can be solved with T&E(braids).
Watch the T&E thread for further results.
Whether we consider braids as "acceptable" from a player POV is another matter. But if we don't accept them, there's no reason to accept any other kind of net.

Back to top profile pm www

Allan Barker

Joined: 21 Feb 2008
Posts: 267
Location: Bangkok

Posted: Thu Oct 16, 2008 8:37 am Post subject: qu

> **Denis B. wrote:**
> Seems you haven't read the warning in my post: "I can't guarantee here that the braids used are the shortest ones available. (The contrary is very likely, I didn't even try to optimise their length.)" As a result, your comparison with shorter patterns is biased.

I read the warning and even included it in your "quote=", but later cut it out for brevity. I often have trouble with not being able to guarantee or even define shortest(ness), so I let the bias pass. The bias has some factual foundation, as you explain later, minimization can be expensive. Granted, the bias is there.

> **Denis B. wrote:**
> Your 3D representation of a braid is also biased, as your software can't represent linear structures such as chains or braids but it transforms them into complex nets. You loose the main property of chains or braids, which is that they have degree one at any point.

I'm not sure of your meaning since of course I can find and display chains, my software make no logical transformations even for displays, and it doesn't know net from a fish. The only transformation I can think of would be from nrtcz chain/braid type notation to sets. To that effect, I think any first order chain must ha a second order logic representation because any rules for building chains are ultimately based on the definition of Sudoku's 324 constraints or sets.

Such logic representations may not show how a chain is ordered or was built, but that is more related to an algorithm on which the logic does not depend. If a logic representation looks like a net, it should not matter what set of rules were used to find it. Forgive me if I haven't correctly understood your statement.

Experimentally, the braid does eliminate the candiate and any alteration to the logic or the 3D representation will casue it to not eliminate the candiate. Since a did was type in your braid, its hard to see how I would have created a net where there was not one.

> **Denis B. wrote:**
> Are braids computational tools? [.....] If instead, I use T&E and I extract manually the corresponding braid from the output trace,

> computation time is a few milli-seconds, but I need at least one hour to build the braid manually from the trace

Here perhaps I am still missing some points. Are you in fact doing this, at least as one of your options? By that, I mean first using some TE algorithm and then extracting a logical elimination from the TE results. This to me is interesting in itself and I have often wondered how to do just such conversions.

Allan

Back to top profile pm www

denis_berthier

Posted: Thu Oct 16, 2008 8:57 am Post subject: quote

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

> **Allan Barker wrote:**
>
> > **Denis B. wrote:**
> >
> > > Your 3D representation of a braid is also biased, as your software can't represent linear structures such as chains or braids but it transforms them into complex nets. You loose the main property of chains or braids, which is that they have degree one at any point.
>
> I'm not sure of your meaning since of course I can find and display chains, my software make no logical transformations even for displays, and it doesn't know a net from a fish. The only transformation I can think of would be from nrtcz chain/braid type notation to sets. To that effect, I think any first order chain must have a second order logic representation because any rules for building chains are ultimately based on the definition of Sudoku's 324 constraints or sets.
> Such logic representations may not show how a chain is ordered or was built, but that is more related to an algorithm on which the logic does not depend. If a 3D logic representation looks like a net, it should not matter what set of rules were used to find it. Forgive me if I haven't correctly understood your statement.
> Experimentally, the braid does eliminate the candiate and any alteration to the logic or the 3D representation will casue it to not eliminate the candiate. Since all I did was type in your braid, its hard to see how I would have created a net where there was not one.

What I mean is that, in the passage from chains/braids to sets, you loose their linear structure. Even if braids are nets, they are very specific ones; fundamenta they still have a linear structure, which is lost in your representation (and the same is true for chains). Said otherwise, they look much more nettish in your representation than they should.
Don't take it as a criticism; your graphics are great. But they are designed to deal with sets, as all your approach; they are not designed to deal specifically wit chains.

> **Allan Barker wrote:**
>
> > **Denis B. wrote:**
> >
> > > Are braids computational tools? [.....] If instead, I use T&E and I extract manually the corresponding braid from the output trace, computation time is a few milli-seconds, but I need at least one hour to build the braid manually from the trace
>
> Here perhaps I am still missing some points. Are you in fact doing this, at least as one of your options? By that, I mean first using some TE algorithm and then extracting a logical elimination from the TE results. This to me is interesting in itself and I have often wondered how to do just such conversions.

It is currently not an option: for braids, can only do this. I haven't implemented braids in SudoRules and I have to build them manually from the trace of the T& procedure. For the details of how I can do this, look at my proof of the T&E vs braids theorem: given a T&E elimination, I show how to build a braid based on it and doing the same elimination.
This is interesting only from a theoretical POV (to prove the theorem) but the practical interest is very limited: T&E doesn't allow to find the shortest braids.
See also my recent answer to ronk, here: http://www.sudoku.com/boards/viewtopic.php?t=6390&start=15

Back to top profile pm www

denis_berthier

Posted: Thu Oct 16, 2008 9:32 am Post subject: quote

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

Allan,
On second thoughts, there is another major difference between my chains and your set representations of them: the additional t- or z- candidates are not part my chains.
This is not only a convention; it has practical consequences: if I notice a chain but I don't use it immediately (e.g. because I've seen a shorter one), then, later, some z- or t- candidates may have disappeared; if the left-linking and right-linking ones haven't changed, this will still be the *same* chain - according to my definition and for SudoRules as well.
But, in your representation, the nets will be different.
Your set representations (unnecessarily) distinguish elements that belong to the same equivalence class: the chain.

Back to top profile pm www

Allan Barker

Posted: Thu Oct 16, 2008 7:35 pm Post subject: qu

Joined: 21 Feb 2008
Posts: 267
Location: Bangkok

Denis,

What started as a semi-humorous response to you seeing what braids 'look like' has hit upon a deep fundamental point. Let's pursue.

> **Denis B. wrote:**
>
> > What I mean is that, in the passage from chains/braids to sets, you loose their linear structure. Even if braids are nets, they are very specific ones; fundamentally, they still have a linear structure, which is lost in your representation (and the same is true for chains). Said otherwise, they look much more nettish in your representation than they should.
> > Don't take it as a criticism; your graphics are great. But they are designed to deal with sets, as all your approach; they are not designed to deal specifically with chains.

Of course, I see no critism at all. This is a very clear statement, and this is why I have adopted ttt's flat dimensionless format, but these are still sets. It seems fundemental difference is between set representations and (DB, possibly other) chain representations. This transformation occurred when I entered your chain into my software.

Allan Barker wrote:

> To that effect, I think any first order chain must have a second order logic representation because any rules for building chains are ultimately based on the definition of Sudoku's 324 constraints or sets.

For me, this is almost a law, the "law of sets", which I assume to be true but would love to know if there are any counter examples (except uniqueness).

In that case, the set rep of your braid is a valid representation of all the logical Sudoku forces required to cause the elimination and removing anything renders elimination invalid. Note, such set reps are not related to any method, only to Sudoku.

Then, let's say that DB finds this logic (braid) by an elegant set of rules that maintains certain logical properties along a path by which the logic is assembled. T someone, XY, finds the same logic by some other set of rules that assembles the logic in a different order. Once done, however, it is the same Sudoku logic. In similar fashion, the starting point of a loop disappears once a loop is found.

On the other hand, your chain notation embeds the rules by which the logic was found, i.e., order among other things. It must also embed, somehow, the Sudo logic that causes the elimination, and the law of sets implies this may be expressed as sets. This may be in the definition of the method and not the notation.

I think the fundamental point at the bottom of all this is: separation of the logic that causes the elimination vs. the logic by which it was found. I'm being driven away from the computer before reading your last reply, but we can continue.

Back to top profile pm www

denis_berthier

Posted: Thu Oct 16, 2008 8:51 pm Post subject: quote

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

Hi Allan,

Allan Barker wrote:

> **Allan Barker wrote:**
>
> > To that effect, I think any first order chain must have a second order logic representation because any rules for building chains are ultimately based on the definition of Sudoku's 324 constraints or sets.
>
> For me, this is almost a law, the "law of sets", which I assume to be true but would love to know if there are any counter examples (except uniqueness).

This is not specific to Sudoku. Any first order formula IS a second order formula. Any sequence (chain) is a set with structure (definable as another set = the or relation).
Or am I misunderstanding what you mean?

Allan Barker wrote:

> In that case, the set rep of your braid is a valid representation of all the logical Sudoku forces required to cause the elimination and removing anything renders the elimination invalid. Note, such set reps are not related to any method, only to Sudoku.

Any representation always conveys representational choices.
Considering a chain as a sequence instead of a set is not removing anything. On the contrary, a sequence is a set PLUS some structure: sequentiality.

Allan Barker wrote:

> Then, let's say that DB finds this logic (braid) by an elegant set of rules that maintains certain logical properties along a path by which the logic is assembled. Then someone, XY, finds the same logic by some other set of rules that assembles the logic in a different order. Once done, however, it is the same Sudoku logic.

What does "logic" mean here? If you mean the proof of the elimination theorem, then it is not the same logic.
Let me take another example, in topology. If you have a metric space X, then you can prove lots of things using the distance (analogue here to the sequentilait a chain) and sequences of points. But, if you have a general topological space Y, most of these proofs will be meaningless in Y because you have lost the concep distance that they relied on. Nevertheless, some theorems that were valid in X will still be valid in Y. The proofs (the "logic") will be different.
One step further: there may even be some heuristic transposition principle, which I used to teach to my students in my topology courses: when you have a proc a metric space, try to replace the sequences of points by filters. Generally, it works. Here, as you transform the chain into sets, the uniqueness of all the rlc is transposed into proper global degree constraints.

Allan Barker wrote:

> On the other hand, your chain notation embeds the rules by which the logic was found, i.e., order among other things. It must also embed, somehow, the Sudoku logic that causes the elimination, and the law of sets implies this may be expressed as sets.

Same as above: the uniqueness of all the rlc is transposed into proper global degree constraints.

Allan Barker wrote:

> This may be in the definition of the method and not the notation.

I agree it is not a matter of notation.
I think the heart of it is: "sequence = set PLUS structure", instead of "sequence = set MINUS what?"

Allan Barker wrote:

> I think the fundamental point at the bottom of all this is: separation of the logic that causes the elimination vs. the logic by which it was found.

I don't think so. Indeed, I'm unable to give this any meaning. Given a chain, its transformation into a set is an embedding into a more general pattern that lead you to justifying the elimination in a different way as the chain itself does.

Back to graphics. I thought of a way you could represent chains and braids (I don't know if this can be done with your software):
- draw a red circle around the target,

- draw dark thin arrows from rlc to llc,
- draw dark thick (or double) arrows from llc to rlc,
- draw very light and thin arrows from rlc to justified additional candidate,
- mark the last rc, rn cn or bn cell with some contradiction symbol.

Back to top profile pm www

denis_berthier

Posted: Fri Oct 17, 2008 12:21 am Post subject: quote

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

HINGES, HINGED-zt-WHIPS, zt-LOCKED-SETS, zt-LS-WHIPS and HINGED-zt-LS-WHIPS

The goal of this post is to show how one can extend the nrczt-whips in such a way that they generalise AICs wth ALSs. This extension is much more general tha merely inserting standard ALSs within a whip or a braid.

1) Hinges and hinged-zt-whips

Hinges were first introduced by Ron Haglund.
(I've been unable to find the thread, but I know there is one, I remember seeing it. If anyone can find the reference, thanks in advance, I'll insert it here).

Define a segment as the intersection of a row or column with a block.
Hinges are the means of using basic interactions (BI) within a chain.
The typical situation is as follows (rows and columns can be interchanged; the roles of rows and blocks can be interchanged):
For some number n, we have the pattern:

Code:

```
?     ?     ?
+     +    (+)    X    X    X    X    X    +in
?     ?    +out
```

? means presence of n irrelevant
+ means presence of n compulsory
(+) means presence of n optional
X means presence of n forbidden
+in means that n is present and is used as a left-linking candidate
+ out means that n is present and is used as the next left-linking candidate

Definition: a segment-candidate is a number together with the cells of a segment in which it is present as a candidate.
Definition: a segment-candidate is nrc-linked to a candidate if both have the same the number-component and if the rc-cell of the candidate is rc-linked to all t rc-cells of the segment-component.

We can now define a hinged zt-whip in the same vein as an nrczt-whip by allowing right-linking candidates to be either an ordinary candidate or a segment-candidate.
As usual, we have the corresponding:
hinged-zt-whip elimination theorem: given a hinged-zt-whip, on can eliminate its target.
the proof of which is obvious, as above.
The only difference is that we need to say what being true means for a segment-candidate: it merely means that one of the occurrences of the number in the segment must be true, which leaves no possibility of being true for the next left-linking candidate.

2) Locked Sets and zt-LS-whips

Given a set S of candidates, one can easily extend the definitions of Naked, Hidden and Super-Hidden (i.e. fish) Subsets to **Naked, Hidden and Super-Hidde (i.e. fish) subsets modulo S**: what remains when all the candidates that are nrc-linked to at least one element of S are "forgotten", must be a Naked, Hidden and Super-Hidden (i.e. fish) Subset.
More specifically:
Definition: Given a set S of candidates,
- an rc-cell and a number constitute a Naked-Single modulo S if, when all the candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Naked-Single in this rc-cell for this number.
- a set of 2 rc-cells in the same row (resp. column, block) and a set of 2 numbers constitute a Naked-Pairs-in-a-row (resp. column, block) modulo S if, when al the candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Naked-Pairs-in-a-row (resp. column, block) in these 2 rc-cells fo these 2 numbers.
- a set of 3 rc-cells in the same row (resp. column, block) and a set of 3 numbers constitute a Naked-Triplets-in-a-row (resp. column, block) modulo S if, when the candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Naked-Triplets-in-a-row (resp. column, block) in these 3 rc-cells these 3 numbers.
- a set of 4 rc-cells in the same row (resp. column, block) and a set of 4 numbers constitute a Naked-Quads-in-a-row (resp. column, block) modulo S if, when a the candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Naked-Quads-in-a-row (resp. column, block) in these 4 rc-cells f these 4 numbers.

- an rc-cell and a number constitute a Hidden-Single-in-a-row (resp. column, block) modulo S if, when all the candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Hidden-Single-in-a-row (resp. column, block) in this rc-cell for this number.
- a set of 2 rc-cells in the same row(resp. column, block) and a set of 2 numbers constitute a Hidden-Pairs-in-a-row (resp. column, block) modulo S if, when al the candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Hidden-Pairs-in-a-row (resp. column, block) in these 2 rc-cells fo these 2 numbers.
- a set of 3 rc-cells in the same row (resp. column, block) and a set of 3 numbers constitute a Hidden-Triplets-in-a-row (resp. column, block) modulo S if, when all the candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Hidden-Triplets-in-a-row (resp. column, block) in these 3 rc-c for these 3 numbers.
- a set of 4 rc-cells in the same row (resp. column, block) and a set of 4 numbers constitute a Hidden-Quads-in-a-row (resp. column, block) modulo S if, when the candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Hidden-Quads-in-a-row (resp. column, block) in these 4 rc-cells these 4 numbers.

- a set of 2 rc-cells in the same row (resp. column) and a set of 2 columns (resp. rows) constitute an X-Wing-in-a-row (resp. column) modulo S if, when all the candidates that are nrc-linked to at least one element of S are fogotten, what remains is an X-Wing-in-a-row (resp. column) in these 2 rows (resp. columns) fo these 2 columns (resp.rows).
- a set of 3 rc-cells in the same row (resp. column) and a set of 3 columns (resp. rows) constitute a Swordfish-in-a-row (resp. column) modulo S if, when all th candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Swordfish-in-a-row (resp. column) in these 3 rows (resp. columns) f these 3 columns (resp.rows).
- a set of 4 rc-cells in the same row (resp. column) and a set of 4 columns (resp. rows) constitute a Jellysfish-in-a-row (resp. column) modulo S if, when all the candidates that are nrc-linked to at least one element of S are fogotten, what remains is a Jellysfish-in-a-row (resp. column) in these 4 rows (resp. columns) fo these 4 columns (resp.rows).

We call such structures collectively "S Locked Sets" or S-LS.

Remarks:
1) we don't have to consider subsets of size > 4.
2) in compliance with my general view of symmetry and super-symmetry, I make no fundamental distinction between ALS (Almost Locked Sets), AHS (or H-ALS Hidden Almost Locked Sets) and SH-ALS (Super-Hidden ALS, Almost-Fish - I don't know if these are currently used in AICSs), although the definitions I've given here don't explicit these symmetries (they could be made explicit by systematically using the rc-, rn-, cn- and bn- spaces).
3) the number of additional candidates (nrc-linked to S) in any of the (rc-, rn-, cn- or bn-) cells of any of these patterns is totally irrelevant.

One can also define a **candidate nrc-linked to an S-LS**: it is any candidate that could be standardly eliminated by the underlying LS.

Definition: given a target candidate z, a zt-LS-whip built on z is any sequence L1 R1 L2 R2 L3 R3 .... Ln of elements alternatively called left-link and right-linking, where
- L1, L2 ... Ln are candidates, (as usual, it is not necessary to extend the possibilities on left-linking candidates),
- each of R1, R2, R3 ... Rn-1 is a Locked Set modulo the target and the previous right-linking candidates,
- L1 is nrc-linked to the target,
- for each 1 < k <= n, Lk is nrc-linked to Rk-1 (in the above extended sense of "nrc-linked)"; this is the first part of the chain condition,
- for each 1 <= k < n, Lk is nrc-linked to at least one element of one of the rc- (resp. rn-, cn- or bn-) cells containing Rk; this is the second part the chain condition,
- there is an rc-, rn-,- cn, or bn- cell containing Ln such that there is no candidate in this cell compatible with the target and the previous right-linking candidates.

zt-LS-whips are an obvious common generalisation of nrczt-whips and of AICs
(except that they rely only on real candidates, not on ghosts - a useless and misleading notion).

As usual, we have the corresponding
zt-LS-whip elimination theorem: given a zt-LS-whip, on can eliminate its target.
the proof of which is obvious, following the same pattern as usual: if the target was true, then allthe left-linking candidates would be false and all the right-linki ones would be true.
The only difference is that we need to say what being true means for a Subset-candidate: it merely means that the n-candidates of the n-cells are globally true these n cells, which leaves no possibility of being true for the next llc.

3) Combining hinges and locked sets: hinged-zt-LS-whips
We can now define a hinged-zt-LS-whip in the same vein as a zt-LS-whip, by allowing a right-linking candidate to be either as in a zt-LS-whip or a segment-candidate.
As usual, we have the corresponding:
hinged-zt-LS-whip elimination theorem: given a hinged-zt-LS-whip, on can eliminate its target.
the proof of which is obvious, as above.

Back to top profile pm www

denis_berthier

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

Posted: Fri Oct 17, 2008 12:23 am Post subject: quote

HINGED-zt-BRAIDS, zt-LS-BRAIDS and HINGED-zt-LS-BRAIDS

The purpose of this post is to extend the above whips to corresponding braids.

We can now define a hinged-zt-braid, a zt-LS-braid or a hinged-zt-LS-braid in the same vein as a hinged-zt-whip, a zt-LS-whip or a hinged-zt-LS-whips, by allowing left-linking candidates to be linked to the target or to any previous right-linking candidate instead of just the previous right-linking candidate.
As usual, we have the corresponding
hinged-zt-braid, zt-LS-braid and hinged-zt-LS-braid elimination theorem: given a hinged-zt-braid, a zt-LS-braid or a hinged-zt-LS-braid, on can eliminate its target.
the proofs of which are obvious, as above.

Let SubsetRules be the set of resolution rules for (Naked, Hidden and Super-Hidden) Singles, Pairs, Triplets and Quads.
Let BI be the Basic Interaction rules.

It is obvious that any elimination done by a hinged-zt-braid could be done by T&E(ECP+NS+HS+BI). The converse is true and more interesting:
Theorem: Any elimination that can be done by T&E(ECP+NS+HS+BI) can be done by a hinged-zt-braid

It is obvious that any elimination done by a zt-LS-braid could be done by T&E(SubsetRules). The converse is true and more interesting:
Theorem: Any elimination that can be done by T&E(SubsetRules) can be done by a zt-LS-braid

It is obvious that any elimination done by a hinged-zt-LS-braid could be done by T&E(SubsetRules + BI). The converse is true and more interesting:
Theorem: Any elimination that can be done by T&E(SubsetRules + BI) can be done by a hinged-zt-LS-braid

As for nrczt-braids, the theorems in this post allow an easier study of the scope of these new braids.
As they rely on the T&E theorems, experimental results will be reported in the "abominable T&E" thread: http://www.sudoku.com/boards/viewtopic.php?t=6390

Back to top profile pm www

denis_berthier

Posted: Sun Oct 19, 2008 8:46 pm Post subject: quote

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

Let's go one step further.

Definition: If P is any elementary pattern (e.g. BI, Pairs, Fish, ..., as in the pevious cases, or any new pattern), with an associated resolution rule, a candidate C nrc-linked to P if it satisfies the conditions of the elimination rule.
If S is a set of candidates, one can define as previously being a pattern in the FP family modulo S.

Given any family FP of elementary patterns, one can define a zt-FP-whip and a zt-FP-braid.

Definition: given a target candidate z, a zt-whip(FP) [resp. a zt-braid(FP)] built on z is any sequence L1 R1 L2 R2 L3 R3 .... Ln of elements alternatively called left-linking and right-linking, where
- L1, L2 ... Ln are candidates, (as usual, it is not necessary to extend the possibilities on left-linking candidates),
- each of R1, R2, R3 ... Rn-1 is a pattern from the FP family modulo the target and the previous right-linking candidates,
- L1 is nrc-linked to the target,
- for each 1 < k <= n: Lk is nrc-linked to Rk-1 (in the above extended sense of "nrc-linked)" [resp. nrc-linked to the target or to any previous right-linking candidate]; this is the first part of the chain [resp. braid] condition,
- for each 1 <= k < n, Lk is nrc-linked to at least one element of one of the rc- (resp. rn-, cn- or bn-) cells containing Rk; this is the second part the chain condition,
- there is an rc-, rn-,- cn, or bn- cell containing Ln such that there is no candidate in this cell compatible with the target and the previous right-linking candidates.

As previously, if FP is a family of elementary patterns:
- a zt-braid(FP) is a generalisation of a zt-whip(FP),
- one can define the procedure T&E(FP), T&E with no guessing, based on the resolution rules of the patterns in FP
- and we have the

zt-braid(FP) vs T&E(FP) theorem: let FP be any family of elementary patterns with associated resolution rules; then any elimination done by T&E(FP) can be done by a zt-braid(FP).

The proof works exactly as for the nrczt-braids.

Such theorems could be used in a very practical way. Suppose you want to build a puzzle that ca't be solved with some predefined set of chain rules CT based o family FP of generating patterns. If one knows that if a puzzle P is not in T&E(FP), it can't be solved by CT. Of course, this is putting stronger constraints than needed on P (not solvable by braids instead of not solvable by chains). But checking that P is not in T&E(FP) is much faster than checking that it isn't in CT.

As a player, you may also want to check if a puzzle has a chance of being solvable by some set of techniques. You can do this with an elementary T&E procedu

Such theorems can also be used to evaluate the scope of several types of braids (which are also an outer boundary for the corresponding whips).
Computations for the scope of several theories T are running. I'll report them in the "abominable T&E" thread.

Edited 10/25/08: changed the notation to zt-whip(FP) and zt-braid(FP) - instead of the old zt-FP-whip and zt-FP-braid

Back to top profile pm www

denis_berthier

Posted: Thu Nov 06, 2008 12:41 am Post subject: quote

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

As the previous posts have introduced complex kinds of whips and braids, a word is in order on how to find them.

As for any other pattern, nothing can replace "practice".

But, as for the chains and whips previously introduced, there are also five general recommendations:
1) look first for the 2D counterparts,
2) look for the nrc before the nrct before the nrczt,
3) look for short whips/braids before longer ones,
4) when you've found a whip/braid, try to find other ones by circular permutation of the candidates,
5) use the **composability property of whips/braids**: this is a way of re-using partial chains/braids. Most whips/braids have parts with no z- or t- additional candidates and parts with t-candidates justified by nearby right-linking candidates. Compose longer whips/braids by assembling such shorter ones before you lo for whips/braids with t-candidates justified by right-linking candidates far away before them.

Back to top profile pm www

denis_berthier

Posted: Sun Nov 30, 2008 5:41 pm Post subject: quote

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

BRAID EXAMPLE

I've had a hard time coding braids in SudoRules in a way that doesn't lead to memory overflow. I can't yet say I've done it in a fully satisfactory way (computat times remain very long and lots of optimisations are still possible, but I'm not here to speak of implementation details in CLIPS). At least, I can now find braids decent length.
Once introduced in SudoRules, braids blend perfectly with whips, as shown by the following example.

I had to make some choice about where to put braids in my complexity hierarchy.
As shown in the "rating" thread (here: http://www.sudoku.com/boards/viewtopic.php?t=5995), length remains the best criterion from a statistical POV.
I have therefore extended SudoRules default strategy so as to include braids of length n just after whips of length n and before any chain of length n+1.

(As braids of length n subsume all the chains of the same length in the xy-to-nrczt family, this is a natural choice).
As for the rest of SudoRules, the way braids are coded would allow me to change this strategy in a very straightforward way if an alternative one was needed. B
this is beyond the purpose of this example, which is only to show how resolution rules for braids (here, only the most elementary ones - nrczt-braids) can be us
effectively as any other resolution rule.

The example I'll consider is # 89 (SER = 9.3) in the top1465 collection.

1) # 89 is one of the 5 puzzles in the top1465 collection that can't be solved by nrczt-whips only.
(Notice that this is very rare for a puzzle with SER = 9.3).
More specifically, if I run SudoRules with the usual rules for nrczt-whips (no braids), I get the following resolution path.
Some eliminations are preceded by a capital letter at the beginning of their line.
Corresponding eliminations in the resolution path with braids will be indicated by the same capital letter in the path with braids; they will be primed if the justify
pattern is different.

***** SudoRules version 13.7wbis *****
...3..5...5..1..3...7..4..12.....4...6..9.....1..6..28..7..2...9..8..5...5..9..7
hidden-single-in-a-row ==> r8c1 = 7
nrczt-whip-rn[2] n7{r5c8 r5c6} - {n7r2c6 .} ==> r6c7 <> 7
swordfish-in-columns n2{r3 r9 r1}{c2 c5 c8} ==> r9c4 <> 2, r3c4 <> 2, r1c6 <> 2, r1c3 <> 2
nrc-chain[3] n5{r4c9 r5c9} - n5{r5c1 r6c1} - n9{r6c1 r4c3} ==> r4c9 <> 9
nrczt-whip-cn[3] n9{r4c3 r4c8} - n9{r6c7 r3c7} - {n9r3c4 .} ==> r2c3 <> 9
nrc-chain[4] n9{r4c3 r6c1} - n5{r6c1 r5c1} - n5{r5c9 r4c9} - n6{r4c9 r4c8} ==> r4c8 <> 9
hidden-single-in-a-row ==> r4c3 = 9
nrczt-whip-rn[7] n9{r7c9 r7c8} - n9{r1c8 r1c1} - n1{r1c1 r1c2} - n1{r7c2 r7c6} - n5{r7c6 r7c5} - n5{r3c5 r3c4} - {n9r3c4 .} ==> r2c9 <> 9
;;; end common part
A: nrczt-whip-rn[9] n7{r2c7 r5c7} - n7{r6c8 r1c8} - n9{r1c8 r1c1} - n1{r1c1 r9c1} - n1{r9c7 r8c7} - n1{r8c6 r7c6} - n5{r7c6 r7c5} - n5{r3c5 r3c4} - {n9r
.} ==> r2c7 <> 9
B: nrczt-whip-cn[11] n7{r2c7 r1c8} - {n7 n8}r1c6 - n8{r1c3 r5c3} - n8{r5c9 r4c9} - n8{r6c8 r9c8} - n4{r9c8 r7c8} - n9{r7c8 r7c9} - n9{r1c9 r1c1} - n1{r1
r1c2} - n3{r8c3 r7c2} - {n3r8c3 .} ==> r2c7 <> 8
C: nrczt-whip-cn[18] {n8 n7}r1c6 - n7{r2c6 r2c7} - n7{r5c7 r5c8} - {n7 n9}r6c8 - n9{r7c8 r7c9} - n9{r1c9 r3c7} - n9{r1c8 r1c1} - n1{r1c1 r1c2} - n2{r1c2
r1c5} - n2{r2c6 r2c3} - n8{r2c3 r5c3} - n8{r5c9 r4c9} - n8{r6c7 r9c7} - n6{r9c7 r8c7} - n3{r8c7 r8c9} - n3{r8c3 r7c3} - n6{r7c3 r9c1} - {n1r9c1 .} ==>
r1c8 <> 8
GRID 89 NOT SOLVED. 55 VALUES MISSING.

Notice that the presence of the final very long whip is, as very often, an indication that this puzzle has few nrczt-chains.
Puzzles with SER 9.3 can usually be solved with shorter whips. For examples of such resolution paths for the hardest puzzles in the top1465 collection, see my w
pages, in particular: http://www.carva.org/denis.berthier/HLS/Results-Classif/magictour-top1465-sol.txt

2) Using an easy implementation of the T&E procedure (independent of SudoRules), it can be checked that # 89 is solvable by T&E = T&E(ECP+NS+HS).
(Indeed, # 89 is one of the 2 puzzles (together with #187, SER 9.4) in the top1465 collection that can't be solved by nrczt-whips but can be solved by nrczt-
braids.)

Thanks to my T&E vs braids theorem (see the "abominable T&E" thread, here http://www.sudoku.com/boards/viewtopic.php?t=6390), we know that there must
a resolution path with nrczt-braids, but we don't yet have such a path. We could get one from the output of the T&E procedure but, as I explained elsewhere, it
would be hard manual work, the path wouldn't be optimised and we wouldn't be able to guarantee any minimality of the braids it uses.

3) Resolution path with nrczt-braids
***** SudoRules version 13.7wbis-B2 *****
;;; Path unchanged down to "end common part"
;;; The next two eliminations are done by slightly shorter braids
A': nrczt-braid-bn[8] n9{r1c8 r1c1} - n9{r3c1 r3c4} - n5{r3c4 r3c5} - n5{r7c5 r7c6} - n1{r1c1 r9c1} - n7{r2c7 r5c7} - n1{r5c7 r8c7} - {n1r9c4 .} ==> r2c
<> 9
B': nrczt-braid-cn[10] n7{r2c7 r1c8} - {n7 n8}r1c6 - n8{r1c3 r5c3} - n8{r9c7 r9c8} - n4{r1c8 r7c8} - n9{r7c8 r7c9} - n9{r1c8 r1c1} - n1{r1c1 r9c1} - {n1
n3}r7c2 - {n3r8c3 .} ==> r2c7 <> 8
;;; Now the two paths diverge completely:
nrczt-braid-cn[11] n9{r7c9 r1c9} - n9{r3c7 r6c7} - n3{r6c7 r5c7} - n3{r5c3 r8c3} - n2{r8c3 r2c3} - n7{r5c7 r2c7} - {n2 n8}r2c6 - {n8 n7}r1c6 - n7{r5c6
r5c8} - n8{r5c9 r6c8} - {n8r5c9 .} ==> r7c9 <> 3
nrczt-braid-cn[10] n6{r4c8 r4c9} - n5{r4c9 r5c9} - n3{r5c9 r8c9} - n7{r5c8 r5c6} - {n7 n8}r1c6 - n8{r5c9 r2c9} - n8{r3c8 r3c2} - {n8 n3}r4c2 - n3{r8c6
r7c6} - {n3r8c3 .} ==> r4c8 <> 7
nrczt-braid-rn[12] n9{r3c7 r6c7} - n8{r9c7 r9c8} - {n9 n7}r6c8 - {n8 n1}r5c8 - {n8 n3}r5c7 - n3{r8c7 r8c9} - n7{r5c8 r5c6} - {n7 n8}r1c6 - {n8 n2}r2c6 -
n3{r8c3 r7c3} - n2{r1c5 r9c5} - {n3r9c5 .} ==> r3c7 <> 8
nrct-chain[8] {n7 n6}r2c7 - {n6 n9}r3c7 - n9{r1c9 r1c1} - n1{r1c1 r9c1} - n6{r9c1 r3c1} - {n6 n4}r2c1 - {n4 n8}r1c3 - {n8 n7}r1c6 ==> r2c6 <> 7
hidden-single-in-a-row ==> r2c7 = 7
nrczt-whip-bn[4] n7{r5c8 r5c6} - {n7 n8}r1c6 - n8{r3c4 r3c2} - {n8r6c2 .} ==> r5c8 <> 8
nrczt-braid-rc[9] {n8 n2}r2c6 - n9{r2c4 r3c4} - {n9 n6}r3c7 - {n8 n4}r2c9 - n2{r2c3 r8c3} - n4{r8c9 r8c4} - n6{r8c4 r9c4} - n6{r9c1 r7c3} - {n8r2c3 .} ==
r2c4 <> 8
nrczt-braid-cn[9] {n2 n8}r2c6 - n9{r2c4 r3c4} - {n9 n6}r3c7 - {n8 n4}r2c9 - {n8 n6}r2c3 - n6{r8c3 r9c1} - n6{r9c4 r8c4} - n4{r8c9 r8c3} - {n2r8c3 .} ==
r2c4 <> 2
nrczt-braid-cn[11] n2{r8c3 r2c3} - {n2 n8}r2c6 - n3{r8c9 r9c7} - n8{r9c7 r9c8} - n8{r3c8 r1c9} - n8{r2c3 r5c3} - {n8 n1}r5c7 - n1{r9c8 r7c8} - {n3 n4}r7
- n4{r1c3 r1c8} - {n4r1c3 .} ==> r8c3 <> 3
nrczt-whip-cn[9] {n6 n9}r3c7 - n9{r1c9 r1c1} - n1{r1c1 r9c1} - n6{r9c1 r3c1} - n3{r3c1 r3c2} - n3{r7c2 r7c3} - n6{r7c3 r8c3} - n6{r8c7 r9c7} - {n6r9c4 .
==> r2c9 <> 6

nrczt-whip-rc[3] n6{r2c3 r2c4} - n9{r2c4 r3c4} - {n9r3c7 .} ==> r3c1 <> 6
nrczt-braid-rc[9] n2{r5c4 r5c6} - n2{r2c6 r2c3} - n7{r5c6 r5c8} - n8{r5c3 r1c3} - n8{r3c4 r3c8} - {n8 n9}r6c8 - {n8 n3}r3c2 - n9{r6c7 r3c7} - {n3r3c1 .} ==> r5c4 <> 8
nrczt-braid-rc[10] n1{r1c1 r9c1} - n6{r9c1 r2c1} - {n6 n8}r1c3 - n9{r2c1 r3c1} - n9{r3c7 r6c7} - n4{r2c3 r2c9} - n8{r2c9 r3c8} - {n9 n7}r6c8 - n7{r5c8 r5c6} - {n8r1c6 .} ==> r1c1 <> 4
nrczt-whip-bn[6] n4{r6c5 r5c4} - n2{r5c4 r5c6} - {n2 n8}r2c6 - {n8 n4}r2c9 - n4{r8c9 r8c3} - {n4r1c3 .} ==> r6c2 <> 4
nrczt-braid-rn[8] n4{r7c2 r1c2} - n3{r7c3 r5c3} - n4{r7c8 r9c8} - n4{r9c5 r6c5} - n3{r6c5 r6c7} - n9{r6c7 r3c7} - n9{r1c9 r1c1} - {n1r1c2 .} ==> r7c3 <>
nrczt-braid-rc[9] n6{r2c3 r2c4} - n1{r1c1 r9c1} - {n6 n4}r9c4 - n1{r9c2 r1c2} - n4{r1c2 r7c2} - n4{r9c8 r1c8} - n2{r1c8 r1c5} - {n4 n8}r2c9 - {n8r2c6 .} ==> r1c1 <> 6
nrczt-braid-cn[8] n6{r9c1 r2c1} - n6{r4c8 r4c9} - n6{r1c9 r1c5} - n6{r7c9 r7c3} - n3{r7c3 r5c3} - n3{r5c9 r6c7} - n6{r9c7 r3c7} - {n9r6c7 .} ==> r9c8 <>
nrczt-braid-rc[9] n4{r5c1 r5c3} - n4{r2c3 r2c9} - n3{r5c3 r7c3} - n6{r9c1 r8c3} - {n6 n3}r8c9 - {n6 n1}r8c7 - {n1 n8}r9c8 - n8{r3c8 r1c9} - {n8r1c3 .} ==> r9c1 <> 4
nrczt-braid-cn[9] n6{r9c1 r2c1} - n2{r9c5 r9c2} - n6{r9c4 r3c4} - n6{r9c7 r8c7} - {n6 n4}r8c3 - n4{r9c2 r1c2} - n1{r1c2 r1c1} - {n6 n9}r3c7 - {n9r3c1 .} ==> r9c5 <> 6
nrczt-whip-rc[10] {n6 n9}r3c7 - n9{r1c9 r1c1} - n1{r1c1 r9c1} - n6{r9c1 r9c7} - n6{r8c9 r8c3} - n2{r8c3 r2c3} - {n2 n8}r2c6 - {n8 n4}r2c9 - n4{r8c9 r8c4} - {n4r9c4 .} ==> r3c4 <> 6
nrczt-whip-cn[8] n2{r1c8 r3c8} - n6{r3c8 r3c5} - n5{r3c5 r3c4} - n8{r3c4 r3c2} - n8{r6c2 r5c3} - n3{r5c3 r7c3} - n6{r7c3 r7c9} - {n6r9c7 .} ==> r1c8 <>

;;; Here, in order to make computations faster, I've deactivated braids; there's nothing new after this point.
nrczt-whip-rc[10] {n6 n9}r3c7 - n9{r1c8 r1c1} - n1{r1c1 r1c2} - n1{r9c2 r9c1} - {n1 n4}r9c4 - n4{r9c2 r7c2} - n4{r7c8 r1c8} - n2{r1c8 r1c5} - {n2 n8}r2c6 - {n8r2c9 .} ==> r9c7 <> 6
nrczt-whip-cn[11] {n8 n7}r1c6 - n7{r5c6 r5c8} - {n7 n9}r6c8 - n9{r6c7 r3c7} - n9{r1c9 r1c1} - n1{r1c1 r1c2} - n2{r1c2 r1c5} - n2{r9c5 r9c2} - n4{r9c2 r7c2} - n6{r8c7 r8c3} - {n6r8c7 .} ==> r1c8 <> 8
nrczt-whip-rc[8] n8{r1c9 r3c8} - n8{r3c4 r6c4} - n8{r6c2 r1c2} - n8{r2c3 r2c6} - n2{r2c6 r2c3} - {n2 n3}r3c2 - {n3 n7}r6c2 - {n7r4c2 .} ==> r4c9 <> 8
nrczt-whip-rn[7] n8{r4c2 r5c3} - n8{r5c9 r2c9} - {n8 n2}r2c6 - n2{r1c5 r1c8} - n4{r1c8 r1c9} - n9{r1c9 r1c1} - {n1r1c1 .} ==> r1c2 <> 8
nrczt-whip-rn[11] n3{r5c3 r7c3} - n3{r9c2 r3c2} - {n3 n9}r3c1 - n9{r3c7 r6c7} - n3{r6c7 r6c5} - n3{r9c5 r9c7} - n8{r9c7 r9c8} - {n8 n7}r6c8 - n7{r5c8 r5c6} - n8{r3c4 r1c6} - {n8r3c4 .} ==> r5c1 <> 3
nrczt-whip-rc[11] {n3 n6}r7c3 - {n6 n1}r9c1 - n1{r9c2 r1c2} - n4{r1c2 r9c2} - n2{r9c2 r3c2} - n2{r3c8 r1c8} - n4{r1c8 r7c8} - {n4 n8}r9c8 - n8{r3c8 r3c4} - n5{r7c5 r3c5} - {n5r7c5 .} ==> r7c2 <> 3
nrczt-whip-cn[7] n9{r6c7 r3c7} - n9{r1c8 r1c1} - n1{r1c1 r9c1} - {n1 n3}r9c7 - n3{r9c1 r7c3} - n6{r7c3 r8c3} - {n6r8c7 .} ==> r6c7 <> 8
nrczt-whip-cn[9] {n3 n9}r6c7 - {n9 n6}r3c7 - {n6 n1}r8c7 - {n1 n8}r5c7 - {n8 n7}r6c8 - n7{r5c8 r5c6} - {n7 n8}r1c6 - n8{r1c9 r2c9} - {n8r2c3 .} ==> r9c<> 3
interaction block b9 with row r8 for number 3 ==> r8c6 <> 3
nrct-chain[4] {n6 n3}r7c3 - n3{r7c5 r9c5} - n2{r9c5 r9c2} - n2{r8c3 r2c3} ==> r2c3 <> 6
hidden-pairs-in-a-row {n6 n9}r2{c1 c4} ==> r2c1 <> 4
interaction column c1 with block b4 for number 4 ==> r5c3 <> 4
hidden-pairs-in-a-block {n4 n5}{r5c1 r6c1} ==> r6c1 <> 3
x-wing-in-rows n6{r2 r9}{c1 c4} ==> r8c4 <> 6
nrct-chain[4] n4{r2c9 r2c3} - n2{r2c3 r8c3} - {n2 n1}r8c6 - {n1 n4}r8c4 ==> r8c9 <> 4
nrczt-whip-rn[4] {n1 n2}r8c6 - {n2 n4}r8c4 - {n4 n6}r8c3 - {n6r9c1 .} ==> r9c4 <> 1
xyzt-chain[5] {n6 n3}r8c9 - {n3 n5}r4c9 - {n5 n8}r5c9 - {n8 n3}r5c3 - {n3 n6}r7c3 ==> r7c9 <> 6
nrczt-whip-rn[5] n5{r7c5 r7c6} - n3{r7c6 r7c3} - n6{r7c3 r7c8} - n6{r8c9 r8c3} - {n4r8c3 .} ==> r7c5 <> 4
nrct-chain[5] {n8 n2}r2c6 - n2{r3c5 r9c5} - n4{r9c5 r6c5} - {n4 n5}r6c1 - {n5 n8}r6c4 ==> r5c6 <> 8
nrct-chain[5] {n8 n2}r2c6 - n2{r3c5 r9c5} - n4{r9c5 r6c5} - {n4 n5}r6c1 - {n5 n8}r6c4 ==> r4c6 <> 8
interaction column c6 with block b2 for number 8 ==> r3c4 <> 8
swordfish-in-columns n8{r5 r2 r1}{c3 c6 c9} ==> r5c7 <> 8
hidden-single-in-a-column ==> r9c7 = 8
nrct-chain[2] n1{r5c7 r8c7} - n1{r8c6 r7c6} ==> r5c6 <> 1
nrc-chain[3] n1{r5c7 r8c7} - {n1 n2}r8c6 - n2{r8c4 r5c4} ==> r5c4 <> 1
interaction row r5 with block b6 for number 1 ==> r4c8 <> 1
nrc-chain[5] {n7 n8}r1c6 - {n8 n2}r2c6 - {n2 n1}r8c6 - n1{r8c7 r5c7} - {n1 n7}r5c8 ==> r5c6 <> 7
hidden-singles ==> r5c8 = 7, r5c7 = 1
interaction row r8 with block b8 for number 1 ==> r7c6 <> 1
naked-pairs-in-a-block {n3 n6}{r8c7 r8c9} ==> r7c8 <> 6
interaction block b9 with row r8 for number 6 ==> r8c3 <> 6
nrc-chain[4] n3{r3c1 r3c2} - n8{r3c2 r3c8} - {n8 n9}r6c8 - n9{r6c7 r3c7} ==> r3c1 <> 9
naked-single ==> r3c1 = 3
hidden-pairs-in-a-row {n2 n3}r9{c2 c5} ==> r9c5 <> 4
naked and hidden singles
914378526
658912734
327564981
289137465
463295178
571846392
846753219
792481653
135629847

Back to top profile pm www

denis_berthier

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

Posted: Mon Dec 01, 2008 8:28 pm Post subject: quote

As I mentioned in the previous post, #187 (SER = 9.4) is one of the two grids in the top1465 collection that can't be solved by nrczt-whips but can by nrczt-bra (as can be verified with a simple T&E procedure).

First tentative solution with whips:
***** SudoRules version 13.7w-bis *****
6.....3...5..9..8...2..6..98.....7...7..5..4.....1..51..3..5...4..2..6...8..7..2
hidden-singles ==> r4c3 = 5, r7c2 = 2

interaction row r4 with block b5 for number 4 ==> r6c5 <> 4, r6c4 <> 4
nrczt-whip-cn[11] {n4 n1}r3c7 - {n1 n9}r9c7 - {n9 n8}r8c7 - n8{r8c9 r5c9} - n1{r5c9 r5c3} - n1{r4c2 r1c2} - n9{r1c2 r1c3} - n9{r8c3 r8c1} - {n9 n5}r8c6
n1{r9c5 r8c4} - {n1r9c5 .} ==> r2c7 <> 4
;;; end common part
nrczt-whip[21] n8{r8c9 r7c9} - n4{r7c9 r9c7} - {n4 n1}r3c7 - n1{r8c7 r8c9} - n7{r8c9 r7c8} - {n7 n5}r3c8 - {n5 n2}r1c8 - n1{r1c8 r4c8} - n1{r4c2 r1c2} -
n9{r1c2 r1c3} - {n9 n6}r7c3 - {n6 n4}r7c5 - {n4 n9}r7c6 - {n9 n5}r8c6 - n5{r9c4 r1c4} - n1{r1c4 r2c4} - n2{r2c4 r2c6} - n2{r4c6 r4c4} - n9{r4c4 r4c2} -
{n9 n3}r9c2 - {n3r9c8 .} ==> r8c4 <> 8
GRID 187 NOT SOLVED. 55 VALUES MISSING.
Here again, we notice a very long whip (length 21), an indication that this puzzle has few chains.

Let's now solve it with braids:
***** SudoRules version 13.7wbis-B2 *****
6.....3...5..9..8...2..6..98.....7...7..5..4.....1..51..3..5...4..2..6...8..7..2
;;; Same resolution path down to "end common part"
;;; We now get braids much shorter than the whip obtained in the previous path:
nrczt-braid-rc[11] n3{r8c9 r9c8} - {n7 n9}r7c8 - {n3 n2}r6c8 - n2{r1c8 r2c7} - n6{r2c7 r2c9} - {n2 n1}r4c8 - n1{r2c9 r1c9} - n1{r1c2 r3c2} - n1{r1c5 r9c5
n4{r9c7 r7c9} - {n4r9c7 .} ==> r8c9 <> 7
interaction row r8 with block b7 for number 7 ==> r7c3 <> 7
nrczt-braid-cn[4] {n6 n9}r7c3 - n4{r6c3 r6c1} - n9{r9c1 r5c1} - {n2r6c1 .} ==> r6c3 <> 6
nrczt-braid-cn[11] n7{r6c4 r6c5} - n8{r6c5 r6c7} - n6{r6c7 r6c2} - n2{r4c6 r4c8} - n2{r6c7 r2c7} - n6{r2c7 r5c7} - n9{r6c7 r6c8} - n3{r6c8 r9c8} - n2{r6
r5c1} - n9{r9c1 r9c2} - {n9r9c1 .} ==> r6c4 <> 2
nrczt-braid-rc[12] n1{r9c4 r9c5} - {n1 n4}r3c7 - n4{r9c5 r9c4} - n6{r9c4 r9c2} - {n6 n9}r7c3 - n9{r1c3 r1c2} - n1{r3c2 r4c2} - n1{r9c8 r1c8} - {n4 n7}r1
{n9 n3}r6c2 - {n1 n4}r1c3 - {n4r6c3 .} ==> r3c4 <> 1
;;; Even with braids activated, the next eliminations use only whips
nrczt-whip-cn[13] {n1 n4}r3c7 - {n4 n9}r9c7 - {n9 n8}r8c7 - n8{r8c9 r5c9} - n1{r5c9 r5c3} - n6{r5c3 r7c3} - n9{r7c3 r7c6} - n8{r7c6 r1c6} - n8{r3c5 r3c
- n1{r3c2 r3c5} - n3{r3c5 r3c1} - n3{r5c1 r5c6} - {n3r6c5 .} ==> r2c7 <> 1
nrczt-whip-cn[8] n1{r5c9 r5c3} - n1{r2c3 r2c4} - n1{r9c4 r9c5} - n1{r9c8 r8c7} - {n1 n4}r3c7 - n4{r9c7 r9c4} - n6{r9c4 r9c2} - {n6r7c3 .} ==> r4c9 <> 1
nrczt-whip-rc[6] n9{r1c2 r1c3} - {n9 n6}r7c3 - n6{r9c2 r4c2} - {n6 n3}r4c9 - n3{r8c9 r9c8} - {n3r9c2 .} ==> r6c2 <> 9
nrczt-whip-cn[8] n1{r4c8 r4c2} - n1{r3c2 r3c5} - {n1 n4}r3c7 - {n4 n7}r1c9 - {n7 n6}r2c9 - {n6 n3}r4c9 - n3{r4c5 r6c5} - {n7r6c5 .} ==> r1c8 <> 1
nrczt-whip-rn[11] n6{r9c5 r9c4} - n6{r9c2 r4c2} - {n6 n3}r4c9 - n3{r4c6 r5c6} - {n3 n4}r4c5 - n4{r9c5 r7c6} - {n4 n2}r2c6 - n2{r2c7 r1c8} - {n2 n9}r6c8
{n6 n9}r7c3 - {n6r7c3 .} ==> r6c5 <> 6
;;;; (I checked that a solution with whips only can be obtained from here, which shows that this puzzle needs only a few eliminations to be solvable by whips).
nrczt-braid-bn[11] n6{r9c2 r7c3} - n6{r7c5 r9c5} - {n6 n3}r4c9 - {n6 n3}r6c2 - n3{r6c5 r3c5} - n1{r9c5 r1c5} - n1{r4c2 r3c2} - n3{r5c9 r5c6} - n7{r3c5
r6c5} - n8{r6c4 r3c4} - {n8r6c4 .} ==> r4c2 <> 6
nrczt-braid-rc[8] n6{r5c3 r6c2} - {n6 n3}r4c9 - n3{r8c9 r9c8} - {n6 n9}r9c2 - {n9 n1}r4c2 - n1{r5c9 r5c7} - {n9 n4}r9c7 - {n4r3c7 .} ==> r5c9 <> 6
;;; For shorter computation times, I de-activated braids here. There's nothing new in the sequel (whips of length <= 12). Details can be seen on my Web pages

As a conclusion of this example and the previous ones:

On the positive side:
- braids can lead to solutions that can't be found when only whips are used (which is not a scoop, as they are more general),
- in all the cases, braids can lead to solutions with much shorter maximum length than whips (in this example braid[13] instead of the whip[21
that didn't even lead to a solution),
- the T&E vs braids theorem is useful in practice as an oracle: using it, we can be confident that there'll be a solution with braids.

On the negative side:
- from a programming POV, useful braids are more difficult to find than whips because there are more useless ones,
- braids are less "beautiful" than whips as there is some branching (although mild compared to some solutions proposed elsewhere with very complex nets),
- from a player POV, I think some strategies of latest branching may make them easier to find than for a computer.

This last idea is the same as the one I explained long ago for t- or z- candidates: these candidates are impurities in what would otherwise be pu
xy- or nrc- chains. We use them when we can't do otherwise. This is (conceptually) how rules with t- and/or z- candidates are implemented in SudoRules.
The difference is that for branching, the way I've found for doing this efficiently is still partial.
(I've solved memory oveflow problems but computation times remain very long).

Back to top profile pm www

denis_berthier

Posted: Thu Jun 04, 2009 3:43 am Post subject: quote edit

Joined: 19 Jun 2007
Posts: 579
Location: Paris, France

denis_berthier wrote:

The scope of nrczt-braids (concrete results)
.....
1) Random collections
All the 10,000 minimal puzzles in the sudogen0 collection randomly generated by suexg are solvable by braids; that's no fresh news, as they were already all solvable by the simpler nrczt-whips (chains and lassos if you prefer the old style).

I've therefore generated and tested 1,000,000 minimal puzzles with the same generator: all of them are solvable by nrczt-braids.
Open question: are all of them solvable by nrczt-whips (as is the case for the first 10,000)? It would be too long to check directly with SudoRules.

The aim of this post is to give a preliminary answer to the last question.

Fisrtly, I have checked that all the puzzles in this extended "sudogen0-1M" collection are different. (I'ven't checked that any two of them are not essentially equivalent, but this is pointless in the present case.)

Secondly, checking with SudoRules is long but not impossible (and, above all, not too time consuming for me). I've left SudoRules run in the background for 3 weeks.

Thirdly, computations are not over, but I've already done them for the first 600,000 puzzles and the answer for them is YES, all of them can be solved by nrczt-whips.

Fourthly, I've also computed the SER of all these puzzles and the correlation coefficient between the SER and the NRCZT-rating and I find 0.89, consistent with result obtained from the first 10,000.

Finally, considering many more puzzles leads to an extended range of complexity (whether you measure it with SER or NRCZT) and the range of validity for the correlation results.

Detailed results forthcoming (hopefully) in 2 weeks.

Back to top profile pm www

Display posts from previous: All Posts Oldest First Go

newtopic postreply **Sudoku Players' Forums Forum Index -> Advanced solving techniques**

All times are
Goto page Previous 1, 2, 3 ... 12, 13,

Page 14 of 14

Stop watching this topic

Jump to: Advanced solving techniques

You **can** post new topics in this fo
You **can** reply to topics in this fo
You **can** edit your posts in this fo
You **can** delete your posts in this fo
You **can** vote in polls in this fo