Not All Equal 3SAT

Posted on July 22, 2014 | 48 Comments

This is jumping ahead a little, but this is a variant of 3SAT that will be helpful in the future (actually, for the next problem)

The problem: Not-All-Equal-3SAT (NAE3SAT). This is problem LO3 in the appendix.

The description: Just like 3SAT, except instead of requiring at least one of the literals in each clause to be true, we're requiring at least one literal to be true *and* at least one literal to be false. So we're removing the case where all three literals can be true.

Notice that if we have an assignment of variables that is a NAE3SAT solution, then if we negate *all* of the variables, we *still* have a NAE3SAT solution

The reduction: From regular 3SAT. We're given a formula that is a collection of 3-literal clauses. For each clause $(x_1 \vee x_2 \vee x_3)$ we create a new variable c_i (for clause i) and add 2 clauses to our NAE3SAT instance:

$(x_1 \vee x_2 \vee c_i)$ and $(x_3 \vee \sim c_i \vee F)$ (Where "F" is the constant value false. We'll talk more about this later)

The idea is that c_i is there to "fix" the clause. If the original clause was satisfiable because of x_1 or x_2, then c_i can be false, and its negation can make the second clause true. If the original clause was satisfiable because of x_3, we can make c_i true. If the original clause had *all three* literals satisfiable, we can make c_i false, and become an acceptable NAE3SAT solution.

Note that because the definition of NAE3SAT (and, for that matter, regular 3SAT) is defined in terms of collections of 3 variables, the constant value false we used above isn't strictly legal. We've really just shown "NAE3SAT with some constants replacing variables" is NP-Hard. To show "normal" NAE3SAT is NP-Hard, we need to reduce the fixed constant version into the normal version:

Given an instance of NAE3SAT with fixed constants, we create 2 new variables x_T and x_F. We replace all instances of the constant value true with x_T, and all instances of the constant value false with x_F. We also add one additional clause:

$(x_T \vee x_T \vee x_F)$.

Note that this new clause is only NAESatisfiable if x_T != x_F. We can't directly bind x_T to true and x_F to false because if you take a NAE3SAT solution and negate all literals, you get another NAE3SAT solution. But because of that, this is NAESatisfiable if and only if the original formula is.

Difficulty: It depends on what you give them. If you make them do everything I did above, I'd say it's an 8. If you let them stop at the NAE3SAT with fixed constants, then maybe a 7.

This entry was posted in Appendix-Logic and tagged 3-sat, Difficulty 8, LO3, NAE3SAT, reductions, uncited reduction. Bookmark the permalink.

48 RESPONSES TO "NOT ALL EQUAL 3SAT"

ElnaserAbdelwahab | August 17, 2014 at 5:41 am | Reply

interesting topic and great page ! Can you please provide us with a concrete example showing how to convert an 3-SAT instance to NAE-3-SAT or its monotone variant (maybe also including negated as well as +ve literals in the original 3-SAT). There seems to be no worked out exmaples anywhere out there .. regards and keep up the great work !

Sean McCulloch | August 18, 2014 at 11:23 am | Reply

Here's a simple example (one of the problems with 3SAT is that all the simple examples are pretty trivial to solve. But hopefully this gets the point across).

Suppose we have the following 3SAT instance with 3 clauses:
(X1, X2, X3)
(~X1,~X2,~X3)
(~X1,~X2,X3)

Since we have 3 clauses, we're going to add 3 variables: C1, C2, and C3
and our NAE3SAT instance will have 6 clauses:
(X1, X2, C1)
(X3,C1,F)

(~X1,~X2,C2)
(~X3,C2,F)

(~X1,~X2,C3)
(X3,C3,F)

"F" can either be the constant value false, if your NAE3SAT rules allow it, or a variable (the equivalent of "Xf" above)

Hope this helps!

ElnaserAbdelwahab | August 18, 2014 at 12:31 pm | Reply

thanks a lot, Sir, for your prompt reply. I appreciate it :))))
I still have the following problem – even for the strait forward example in the text where we have only one clause (x1 ∨ x2 ∨ x3) – :
The truth tables on the one side for (x1 ∨ x2 ∨ x3) and on the other for ((x1 ∨ x2 ∨ ci) and (x3 ∨ ~ci ∨ F)) which are supposed to be equivalent, are not. See in particular the following assignment(s) : put x1=F,x2=T,x3=F which should give true in the first case while x1=F,x2=T,x3=F,ci=T will give false in the second truth table … How can/shall we explain this ?
regards
Elnaser

Sean McCulloch | August 18, 2014 at 12:48 pm | Reply

Why does ci have to be true? If you make it false, the formula is satisfiable.

ElnaserAbdelwahab | August 18, 2014 at 1:38 pm | Reply

sure we can do this .. my understanding of logical equivalence is more related to truth tables though .. i need to explain what will happen with the combination x1=F,x2=T,x3=F,ci=T in the truth table for ((x1 ∨ x2 ∨ ci) and (x3 ∨ ~ci ∨ F)) . Shall it be not allowed for example because of the NAE condition (although by putting ci=T we dont breach this condition for neither the first nor the second clause) ??

Sean McCulloch | August 18, 2014 at 1:49 pm | Reply

I think you're not really understanding what a reduction is. What we're doing is:
– Taking an instance of 3SAT (For which there are only 2 possible answers: "Yes, it's satisfiable", or "No, it's not satisfiable". "Satisfiable" just means that _some_ combination of variable assignments exists to make all clauses true. Notice that _how_ it's satisfiable or not doesn't matter)
– Building an instance of NAE3SAT. We promise that the instance we build says "Yes, it's satisfiable" exactly when the original instance of 3SAT also said "Yes"
– Showing the conversion happens in polynomial time (which is usually pretty obvious). That way, we can say "Hey, if someone ever comes up with a polynomial time algorithm for NAE3SAT, here's how we can use that algorithm as a function to come up with a polynomial time algorithm for 3SAT"

So, all I have to do is say "If the original 3SAT instance is satisfiable, then the NAE3SAT instance I built is, too" (and vice versa). So your example 3SAT instance is satisfiable (in lots of ways), and so I need to show how the constructed NAE3SAT instance is satisfiable as well. It doesn't matter that _other_ assignments exist to make some clauses false. All I need is _some_ set of variable assignments that makes the clauses true.

In truth table language, I'd say what we're doing is saying: You can find _a_ row (doesn't matter which) in your truth table that makes the 3SAT formula true if and only if you can find _a_ row (doesn't matter which) in your truth table that makes the NAE3SAT formula true.

ElnaserAbdelwahab | August 18, 2014 at 1:58 pm | Reply

great explanation !!! I was understanding reductions wrong indeed .. :))) .. i guess i confused equivalence with reduction !!! This makes the world much more clearer ... thanks for taking your time to answer me Sir. I might come back to you with some other questions if i am stuck again .. grretings. Elnaser

Sean McCulloch | August 18, 2014 at 2:01 pm | Reply

It's sort of equivalence, but equivalence between the "Yes/No" answers of both problems, not between the actual solutions themselves.

Thanks for the discussion, glad someone is actually reading this

ElnaserAbdelwahab | August 18, 2014 at 2:36 pm | Reply

i am actually not only reading, but also doing research related to a new type of NAE-SAT solvers for which i need to understand things right (not only formally) ... eventually preparing a course to be taught to students in adequate time ... greetings

Sean McCulloch | August 18, 2014 at 2:37 pm | Reply

Cool, good luck with it!

ElnaserAbdelwahab | August 19, 2014 at 7:59 am | Reply

Here is a followup check of whether i understood the monotone variant of the above NAE-3-SAT.
Converting the above clause set :
{(X1, X2, C1) (X3,C1,F)
(~X1,~X2,C2)(~X3,C2,F)
(~X1,~X2,C3) (X3,C3,F)}
to Monotone-NAE-3SAT gives :
{
{y1,y2,c1}{y3,c1,F}
{z1,z2,c2}{z3,c2,F}
{z1,z2,c3}{y3,c3,F}
{y1,z1,t1}{y1,z1,u1}{y1,z1,v1}{t1,u1,v1} //to guarantee that y1z1
{y2,z2,t2}{y2,z2,u2}{y2,z2,v2}{t2,u2,v2} //to guarantee that y2z2
{y3,z3,t3}{y3,z3,u3}{y3,z3,v3}{t3,u3,v3} //to guarantee that y3z3
}
here : yi is replacing xi and zi is replacing not(xi)
is that correct ? Or did i miss something here ?
Appreciate your effort in advance
Elnaser

Sean McCulloch | August 19, 2014 at 10:21 am | Reply

That looks right, assuming you're allowed the constant F in the monotone version

> **ElnaserAbdelwahab** | August 19, 2014 at 11:33 am | Reply
>
> thanks a lot .. greetings

ElnaserAbdelwahab | August 20, 2014 at 5:07 am | Reply

another small thing : Do we have interesting solver applications based on Monotone-NAE-2SAT formulas (converted from 2SAT in a similar way as we did above) ? or any papers related to those types of formulas for that matter ? (which seem to be strait forward in their resolution)
Elnaser

> **Sean McCulloch** | August 20, 2014 at 1:46 pm | Reply
>
> No idea I don't actually know too much about the problem- I only ran across it when I saw other people using it as the basis for reductions for other problems..

ElnaserAbdelwahab | August 20, 2014 at 1:54 pm | Reply

thanks anyway

ElnaserAbdelwahab | August 27, 2014 at 3:28 am | Reply

Hello again, Sir. Hope you are fine
Here is another related question : assuming that we dont impose the NAE condition, but keep all literals of a 3SAT problem +ve (by means of the transformation above, do we still have an NP

complete problem (i.e. is this Monotone,+ve 3SAT NP-complete as well) ?
regards
Elnaser

Sean McCulloch | August 27, 2014 at 1:12 pm | Reply

Monotone 3SAT means all literals in a clause are the same (either all negated or all non-negated). That;s NP-Complete. If you're trying to make all literals in all clauses positive, wouldn't that be trivially solvable by making all variables true?

ElnaserAbdelwahab | August 27, 2014 at 1:24 pm | Reply

unless we can express inequality between +ve literals somehow (which we can easily do using NAE) ..

Sean McCulloch | August 27, 2014 at 1:40 pm | Reply

So you want NAE _and_ all positive? Yeah, I think that's NP-Complete. There are a lot of people out there doing weird reductions from that for heuristics (search POS-NAE-3SAT to see what I mean)

Ian | April 1, 2015 at 11:37 pm | Reply

I'm trying to convince myself this works and I cannot… Can you tell me what I'm doing wrong?

Consider an expression that is not in satisfiable in 3SAT. It must have at least one clause that cannot be satisfied. I'll say this clause is (x | y | z)… Using the reduction you describe, that would transform to:

(x | y | c) & (z | ~c | x_f) & (x_f | x_t | x_t)

If we consider the case where x,y,z are all FALSE, then could have the case where c is true, x_f is true, and x_t is false. Which becomes:

(F|F|T)&(F|F|T)&(T|F|F)

This is satisfiable in NAE3SAT, but it shouldn't be since it's coming from an unsatisfiable 3SAT clause..

> **Sean McCulloch** | April 2, 2015 at 12:11 am | Reply
>
> I'm not sure what you mean by "this is the clause that's not satisfied". A 3SAT instance is a bunch of clauses, but all of them are sometimes true and sometimes false depending on the assignments of the truth values. There's no _one_ clause that's _always_ false. All you can say is that no matter _what_ the assignments of the variables are, _some_ clause is false. But which clause it is changes depending on what truth values you're using.
>
> So, the simplest 3SAT instance that's not satisfiable is:
> (x1 |x2|x3)
> (x1|x2|~x3)
> (x1|~x2|x3)
> (x1|~x2|~x3)
> (~x1|x2|x3)

(~x1|x2|~x3)
(~x1|~x2|x3)
(~x1|~x2|~x3)

(all possible negations and non-negations of 3 variables.)

Which clause is the "one" that's not satisfiable? It depends on the truth values of x1, x2, and x3. But no matter which one you choose, one of the clauses won't work.

Or maybe I'm misunderstanding what you're saying..

Noah | April 8, 2017 at 10:35 pm | Reply

I came to this conversation (a bit late!) because I have the same objection as Ian. He seems to agree with you: for the formula F to be out of 3SAT, just some clause in every possible assignment to F must have been forced to all-false. Now, to reiterate Ian's example, consider the correspondent of such a clause in the transformed formula F', where x, y, and z were the original variables in F:

(x | y | c) & (z | ~c | x_f) & (x_f | x_t | x_t)

Alright, we added the third clause there, even if it really lives way on end of F'.

Well, if (x | y | z) couldn't be made true in F, the only way for its correspondent in F' to be made true starts with binding c to true for the first conjunct. That means x_f must be true in the second conjunct. Is that possible? Yes, it is, if x_t is bound to false. So this conjunction does have an assignment which imparts it with the "not all equal" property, and so it does not force F' to be out of NAE3SAT.

Well, if this conjunction doesn't do the job of putting F' out of NAE3AT, what other part of F' might? If there is not necessarily any such segment in F', then the mapping is not valid, correct? This problem would seem to generalize to any all-false clause you could find in any assignment to an unsatisfiable F, which I think is something Ian was driving at.

I thought for a moment that this issue might not be fatal. (And I'm still hopeful you'll show me it isn't!) I had missed your instruction, "We replace all instances of the constant value true with x_T". But I do not think this is any help here (though I am not sure if you're claiming it is). The mapping can only send 3CNF formulas to 3CNF formulas; it doesn't know anything about assignments in itself, and so there's no mechanism to make the substitution there. And while maybe it helps to know about such a substitution if our goal is to show F' is in NAE3SAT, it does not seem to help us when our goal is to show F' is not in NAE3SAT, since our "adversary" can make any assignment it wants there to foil us.

Noah | April 8, 2017 at 10:38 pm | Reply

Oh, and now I am seeing the later discussion you had with Ian. Seems like it may not be possible to prove this by showing a mapping from "out" to "out". Thank you!

Sean T. McCulloch | April 9, 2017 at 9:23 pm |

Yeah, I think the key mistake is here:

> > > > *(x | y | c) & (z | ~c | x_f) & (x_f | x_t | x_t)*
> > > >
> > > > *Alright, we added the third clause there, even if it really lives way on end of F'.*
> > > >
> > > > *Well, if (x | y | z) couldn't be made true in F, the only way for its correspondent in F' to be made true starts with binding c to true for the first conjunct.*
> > >
> > > Saying "there is no way to make a clause true" is different from saying "the variables in this clause are set to false". So, for example, it's very possible for the variable z to be set to true, and to use that setting to make the formula satisfiable.

Ian | April 2, 2015 at 8:38 am | Reply

I follow this, and I see that for this instance no matter with assignment you choose for the variables x1,x2, x3, there will always be one clause that is false. If we look at assignment FFF, then the first clause causes the rejection. If we look at assignment TTT, then it's the last clause.

I'm stuck on how we can claim that the transformed expression is not satisfiable.

Once we transform to the new form, now we have 17 clauses, 2 for each original clause, plus the (x_f|x_t|x_t) clause:

(x1|x2|c1)&(x3|~c1|x_f)&
(x1|x2|c1)&(~x3|~c1|x_f)&
(x1|~x2|c1)&(x3|~c1|x_f)&
(x1|~x2|c1)&(~x3|~c1|x_f)&
(~x1|x2|c1)&(x3|~c1|x_f)&
(~x1|x2|c1)&(~x3|~c1|x_f)&
(~x1|~x2|c1)&(x3|~c1|x_f)&
(~x1|~x2|c1)&(~x3|~c1|x_f)&
(x_f|x_t|x_t)

This is satisfiable with any assignment that sets c1=T and x_f=T, including:
x1=F
x2=F
x3=F
c1=T
x_f=T
x_t=F

But now I see that this is violating NAE3SAT because (~x1|~x2|c1) is now all T… It just really hard for me to concisely justify that this is the case in general.

Thanks for this resource and for your willingness to help people understand.

> **Sean McCulloch** | April 2, 2015 at 12:52 pm | Reply
>
> Yeah, I think since it's so hard to come up with an "easy" SAT instance (for any variant of SAT, really) that is not satisfiable, the easiest way to show the reduction is to show

yes(3SAT) => yes(NAESAT)
and
yes(NAESAT) => yes(3SAT)

In other words, instead of focusing on how to make things _not_ work, think about the contrapositive- show that if there _is_ a way to set the variables to satisfy the NAE instance, that gives you a way to satisfy the regular 3SAT instance. (and vice versa)

Ian | April 2, 2015 at 12:57 pm | Reply

I was so set on solving it using the pattern I'm used to I didn't even consider this approach.

Thanks again,

Ian

Eva Jelinkova | April 20, 2015 at 9:58 am | Reply

What is the source of this reduction please? I would like to cite it in my paper because I am doing something very similar as a lemma… In the paper "The Complexity of Satisfiability Problems" by T. Schaefer, the problem is claimed to be NP-complete but the reduction is not there.
Thanks in advance,
Eva

Sean McCulloch | April 20, 2015 at 10:03 am | Reply

Hmm, I just kind of did it from the top of my head (though admittedly the idea was put there from seeing it in a bunch of classes and papers along the way).

If all you need is the fact that it's NP-Complete, you can probably just reference G&J, which is what most people do.

If you need to cite the reduction itself, hmm. I'll see if I can look around and find something.

Sean McCulloch | April 20, 2015 at 2:54 pm | Reply

The Schaefer paper does show it, but it's a corollary of a wacky theorem that doesn't do it they way it was done here.

I found a version that does it similar to my way in a Theory textbook: "The Theory of Computation", by Bernard Moret, Addison-Weslay, 1998. That's probably where you want to look.

Eva Jelinkova | April 22, 2015 at 9:37 am | Reply

Thanks a lot!
Eva

Ciechowoj | June 2, 2015 at 7:08 pm | Reply

I think you have mistake here. Consider simple 3SAT instance
(x ∨ y ∨ z)

and NAE-SAT equivalent
(x ∨ y ∨ c) ∧ (z ∨ ¬c ∨ f) ∧ (f ∨ f ∨ t)

When SAT is false x = 0, y = 0 and z = 0 then NEA-SAT still can be satisfied
x = 0, y = 0, c = 1
z = 0, ¬c = 0, f = 1
f = 1, f = 1, t = 0.

I think you should replace that last clause with (¬f ∨ ¬f ∨ ¬f) to make it work.

> **Sean McCulloch** | June 3, 2015 at 2:16 pm | Reply
>
> Remember, that SAT asks "can I find _any_ way to set the variables to make the formula true". So your SAT instance is satisfiable.

Marten | February 11, 2016 at 1:12 am | Reply

I think if found another possible reduction that may be a little better to understand. For each clause (x1 ∨ x2 ∨ x3) just add the clause (~x1 ∨ ~x2 ∨ ~x3) to the 3SAT instance. (x1 ∨ x2 ∨ x3) and (~x1 ∨ ~x2 ∨ ~x3) are only satisfiable if either x1 != x2, x2 != x3 or x1 != x3. Furthermore at least one of the literals has to be true. In (x1 ∨ x2 ∨ x3) each literal can be replaced with its negation and it still works.

> **Marten** | February 11, 2016 at 1:44 am | Reply
>
> Nevermind I found my mistake. If each literal is true, (x1 ∨ x2 ∨ x3) and (~x1 ∨ ~x2 ∨ ~x3) is not an acceptable NAE3SAT solution because it is not NAESatisfiable.
>
> > **Sean McCulloch** | February 11, 2016 at 6:59 pm | Reply
> >
> > This is a pretty tricky problem to imagine- at least for me. There are so many troublesome cases..

elnaser | February 26, 2016 at 7:36 am | Reply

Just wanted to share with you the following recent article related to 3SAT :
http://arxiv.org/abs/1602.04955
Hope you enjoy the read !
regards
Elnaser

> **Sean McCulloch** | February 26, 2016 at 9:09 am | Reply
>
> Wow, that looks fascinating. Very cool!
>
> > **elnaser** | February 26, 2016 at 9:42 am | Reply
> >
> > thanks. Please share with me any comments. regards

Prateek | April 19, 2016 at 2:56 pm | Reply

It seems that Ian's idea for a construction of a contradiction would work, if you could find an unsatisfiable 3SAT formula and an assignment of the variables that doesn't set any clause to T,T,T.

Starting from such a formula and assignment, your reduction would fail, and you would construct a NAE-satisfiable NAE formula out of an unsatisfiable 3SAT formula.

Since I buy your reduction, it seems like this is a proof of:

Every assignment of an unsatisfiable 3SAT formula must set some clause to (T,T,T)

My question is: why didn't I already know that?

> **Sean McCulloch** | April 19, 2016 at 3:41 pm | Reply
>
> Hmm, it is the case that each assignment of an unsatisfiable 3SAT instance has a way to set a clause to _all_ 8 combinations of trues and falses?
>
> We know that each unsatisfiable assignment sets a clause to (F,F,F) (because that's what makes it unsatisfiable).
>
> Is it the case that if no clause is assigned, say, (T,T,F), we can reverse some truth values to make a clause have that truth value, and at the same time, change our (F,F,F) clause to be satisfiable?
>
> That's how it works if you have less than the 8 contradictory clauses in the minimal unsatisfiable instance anyway (If you have 7 clauses, some assignment will not be there. You just need to rearrange assignments so that (F,F,F) is the one not there). I don't know if that works in more general cases too, though..

Steven | October 28, 2016 at 1:14 am | Reply

In any 3-CNF, no clause can contain the same literal twice. So I take issue with your final clause (x_t, x_t, x_f).

> **Sean McCulloch** | October 28, 2016 at 11:26 am | Reply
>
> I'm not familiar with that kind of restriction- it seems like there are lots of different definitions of 3SAT out there. Luckily, it's not hard to replace a repeated variable with 2 variables that are bound (by adding extra clauses) to have the same truth value).

Cyriac Antony | May 18, 2017 at 2:43 pm | Reply

This 3-SAT <= NAE 3-SAT reduction is wrong.
Suppose x_F is used in place of F. The issue is that the constructed NAE 3-SAT instance is always an yes instance. For instance, the assignment A given by :
set all x_i = False, all c_j=True, x_F=True
(so first kind (so second kind
of clauses satisfied) of clauses satisfied)
The clause (x_T,x_T,x_F) has no role at all. (same as using ~(x_F) for x_T).

Sean T. McCulloch | May 18, 2017 at 3:14 pm | Reply

Well, keep in mind that in a "real" SAT instance, there will be lots of instances of clauses that have negated x_i literals. So sure, what you're doing is an example of how to make a formula with no negated literals NAE satisfiable, but all formulas like that really _are_ satisfiable. For a formula to be unsatisfiable, you need negated literals (and so some of those x_i will really be ~x_i and so will evaluate to true)

Cyriac Antony | May 18, 2017 at 11:13 pm | Reply

Terribly sorry for my previous (18-05-17) comment.
The reduction is fine.
If assignment A for SAT instance satisfies all clauses in it, we obtain assignment B for NAE-3-SAT by reusing truth values for original variables (from A), False for F and True/False for c_i appropriately as explained in the post.
If assignment B for NAE-3-SAT instance satisfies all clauses in it, (i) we flip truth assignment for all variables so that F=False under B afterwards, and then, (ii) reuse the truth assignment for original variables.

> **Sean T. McCulloch** | May 19, 2017 at 1:16 am | Reply
>
> Yeah, I think that's the right way to think about it. Thanks for checking on me!