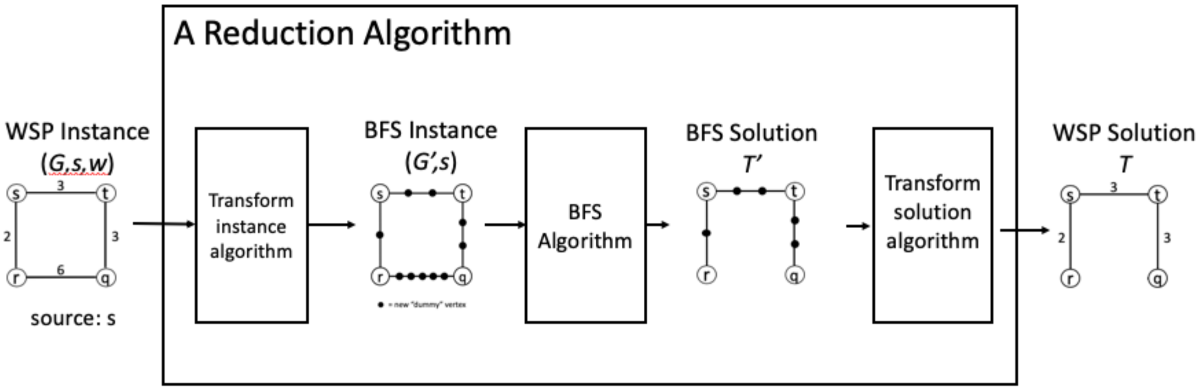
Ruolin Li
31764160

CPSC 320: What's in a Reduction?*

Solution for each questions is shown in page 6

We often use reductions to solve new problems based on problems we can already solve. For example, in an earlier worksheet, we saw a reduction from the Shortest Paths Problem to Breadth First Search:



But…there's another way to use reductions. A more **sinister** way.[1] We'll illustrate this using a famous problem from logic: Satisfiability.

1 Boolean Satisfiability

Boolean satisfiability (SAT) is—as far as Computer Scientists know—a hard problem, in the sense that no-one knows of an algorithm to solve SAT that has worst-case polynomial runtime. In the version of SAT that we discuss here, you're given a propositional logic expression like:

$$(x_1 \vee \overline{x}_2 \vee x_3 \vee x_4) \wedge (x_5) \wedge (\overline{x}_1) \wedge (x_2 \vee \overline{x}_3 \vee \overline{x}_5) \wedge (\overline{x}_2 \vee x_3).$$

You must determine whether any assignment of truth values to variables (the x_i's) makes the expression true, which we call *satisfying* the expression.

Formally, an instance of SAT is a logical statement that is a conjunction (an "and" denoted by $\wedge$) of c clauses. Each clause is a disjunction (an "or" denoted by $\vee$) of one or more literals, and each literal is either a variable x_i or its negation $\overline{x}_i$. For convenience, we'll insist on using the variables $x_1, x_2, \ldots x_n$ for some n, without skipping any. Given an instance I of SAT, we want to know: is the instance I satisfiable or not? The answer is either Yes or No, so we call this a *decision problem.*

1. Is the example SAT instance above satisfiable? If not, explain why not. If so, prove it by giving an assignment that makes the statement true.

[1]Well, OK. Just **another** way.

2. For a SAT instance I, a truth assignment is a potential solution, and the truth assignment is a *good* solution if it satisfies instance I.

   Suppose in addition to instance I you were given a truth assignment, say represented as an array $T[1..n]$ where $T[i]$ is true if and only if x_i is set to true. How long would it take to certify that a truth assignment is good?

3. A brute force algorithm could make a list of the variables $x_1, \ldots, x_n$ in the problem, try every assignment of truth values to these variables, and return `YES` if any satisfies the expression or `NO` otherwise. Asymptotically, how many truth assignments might this algorithm try (in terms of n)?

2 3-SAT and SAT

The 3-SAT problem is just like SAT, except that **every** clause must be **exactly** of length 3. Let's build a reduction from SAT to 3-SAT (so we're solving SAT in terms of 3-SAT). We'll map an instance I of SAT to an instance I' of 3-SAT, working on one clause at a time. Importantly, for our reduction to work, I should be satisfiable if and only if I' is satisfiable.

1. Suppose that I has a clause with one literal, say (x_5). To obtain I' from I, we want to replace this clause by one or more clauses, while ensuring that I is satisfiable if and only if I' is. How can we do this? Hint: one variable can appear multiple times in a clause.

2. What if I has a clause with two literals, say $(\overline{x}_2 \vee x_3)$?

3. Now suppose that I has a clause with four literals, say $(x_1 \vee \overline{x}_2 \vee x_3 \vee x_4)$. What 3-SAT clauses will you put in I' to replace this clause, so that I' is satisfiable if and only if I is? Hints: Break the clause up somehow. Don't try using de Morgan's laws. Instead, create a brand new variable, say x_{n+1}, and integrate that into your new clauses.

4. For your construction of part 3, show that if I is satisfiable then I' must also be satisfiable (and modify your construction if needed to ensure this).

5. Also for your construction of part 3, show that if I' is satisfiable then I must also be satisfiable.

6. Extend your 4-literal clause plan above to a 5-literal clause like $(x_1 \vee \overline{x}_2 \vee x_3 \vee x_4 \vee \overline{x}_5)$. Since new variable x_{n+1} is already "used up" in part 2, index any new variables that you create starting at x_{n+2}.

7. Show, by filling in the blanks below, how you would transform a clause with $k > 3$ literals

$$(l_1 \vee l_2 \vee \ldots \vee l_k)$$

into clauses with three literals (keeping in mind overall reduction correctness). You can use new variables that have not already been "used up", starting with x_{i+1} (where $i \geq n$). How many clauses do you get? What would be the runtime of an algorithm to do this, as a function of k?

$$(l_1 \vee l_2 \vee x_{i+1}) \;\wedge\; (\bar{x}_{i+1} \vee l_3 \vee \underline{\hspace{2em}}) \wedge (\bar{x}_{i+2} \vee l_4 \vee x_{i+3}) \wedge \ldots$$

$$\ldots \wedge \; (\overline{x}_{i+(j-2)} \vee l_j \vee x_{i+(j-1)}) \;\wedge \ldots$$

$$\wedge\, (\bar{x}_{i+(k-4)} \vee l_{k-2} \vee \underline{\hspace{2em}}) \wedge (\bar{x}_{i+(k-3)} \vee l_{k-1} \vee \underline{\hspace{2em}}).$$

8. Let's use the name TRANSFORM-CLAUSE to refer to the algorithm for transforming a clause, as described in part 7. Suppose that I' is obtained from I by transforming clause $(l_1 \vee l_2 \ldots \vee l_k)$ using algorithm TRANSFORM-CLAUSE. Explain why I is satisfiable if and only if I' is satisfiable.

2020/08/05

Step 1

1. It's satisfiable.

$x_1 = F$ $x_2 = F$ $x_3 = F$ $x_4 = F$ $x_5 = T$ $(x_1 \vee \bar{x}_2 \vee x_3 \vee x_4) \wedge (x_5) \wedge (\bar{x}_1) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_5) \wedge (\bar{x}_2 \vee x_3)$

2. $I = (x_1 \cup x_5 \cdots) \wedge (\cdots) \wedge \cdots$

If m = # of the clauses, and there are at ~~least~~ most n literals per clause, then $O(nm)$.

3. 2^n truth assignments

Step 2 3-SAT $I(SAT) \to I'(3\text{-}SAT)$

1. $I = (\bar{x}_1 \vee x_4 \vee x_2) \wedge (x_5) \wedge C_3 \wedge C_4 \cdots$

$I' = (\bar{x}_1 \vee x_4 \vee x_2) \wedge (x_5 \vee x_5 \vee x_5) \wedge C_3 \wedge C_4 \cdots$

Suppose I is satisfiable Claim: T also satisfies I'

Let $T[1 \cdots n]$ satisfy I

2. $(\bar{x}_2 \vee x_3) \to (\bar{x}_2 \vee x_3 \vee x_3)$

3. $I = \cdots (x_1 \vee \bar{x}_2 \vee x_3 \vee x_4) \cdots$ use a new variable

$I' = \cdots (x_1 \vee \bar{x}_2 \vee x_k) \wedge (\bar{x}_k \vee x_3 \vee x_4) \cdots$

4. Let $T[1 \cdots n]$ satisfies I. We need to describe a truth assignment $T'[1 \cdots n+1]$ that satisfies I'.

A) If $T[1 \cdots n]$ sets $x_1 \vee \bar{x}_2$ to true, then we set x_k to false. that ensures all new clauses are true.

B) If $T[1 \cdots n]$ sets $x_3 \cup x_4$ to true, then we set x_k to true. Then both clauses are satisfied.

A) or B) must hold, since $T[1\text{-}n]$ satisfies $(x_1 \vee \bar{x}_2 \vee x_3 \vee x_4)$.

5. Suppose I' is satisfiable, let $T'[1 \cdots n+1]$ satisfy I'.

We will do this by showing that the truth assignment $T[1 \cdots n] = T'[1 \cdots n]$ satisfies I.

This $T[1 \cdots n]$ definitely satisfies all the clauses other than the one we transformed.

Consider $(x_1 \vee \bar{x}_2 \vee x_3 \vee x_4)$.

If x_{n+1} is true, then since $(\bar{x}_6 \vee x_3 \vee x_4)$ is true, ~~then~~ in T', then $x_3 \vee x_4$ is true.

If x_{n+1} is false, then since $(x_1 \vee \bar{x}_2 \vee x_6)$ is true in T', then $x_1 \vee \bar{x}_2$ is true.

6. $(x_1 \vee \bar{x}_2 \vee x_7) \wedge (\bar{x}_7 \vee x_3 \vee x_8) \wedge (\bar{x}_8 \vee x_4 \vee \bar{x}_5)$

7. $\boxed{(l_1 \vee l_2 \vee x_{i+1}) \wedge (\overline{x_{i+1}} \vee l_3 \vee x_{i+2}) \wedge (\overline{x_{i+2}} \vee l_4 \vee x_{i+3}) \wedge \cdots \wedge (\overline{x_{i+(k-4)}} \vee l_{k-2} \vee x_{i+(k-3)}) \wedge (\overline{x_{i+(k-3)}} \vee l_{k-1} \vee l_k)}$

8. If direction: Suppose $T[1 \cdots n]$ satisfies I. We need to show that I' is satisfiable. We can set $T'[1 \cdots n] =$

$T[1 \cdots n]$, then all the clauses of I' other than $(l_1 \vee l_2 \vee l_3 \vee \cdots \vee l_k)$ are satisfied.

We know that $l_1 \vee l_2 \vee \cdots \vee l_k$ is satisfied by $T[1 \cdots n]$ At least one literal is true in the $T[1 \cdots n]$.

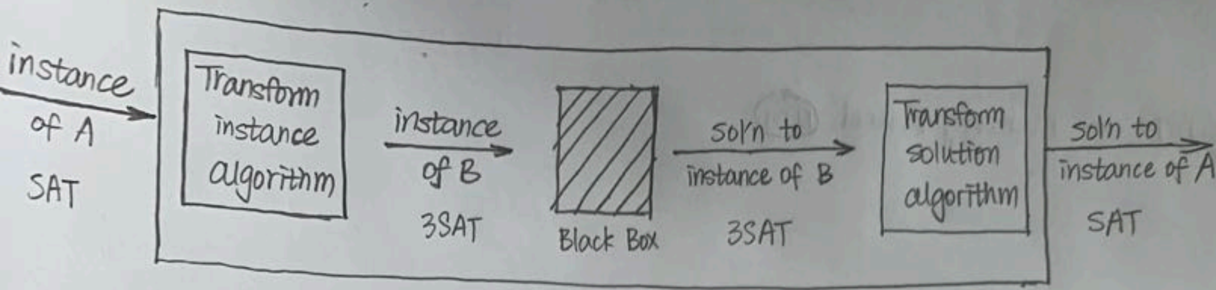
Suppose l_j is the first true one, then from l_1 to l_{j-1} are false.

We'll set $x_{i+1}, x_{i+2}, \cdots, x_{i+(j-2)}$ to true, ensuring that all clauses containing literals up to $j-1$ are satisfied.

We should choose $x_{i+(k-3)}$ to be false, so that the last clause is satisfied. (Assuming only l_j is true)

Working backwards, we keep setting variables to false, ensuring that clause containing $l_{j+1}, \cdots, l_k$ are satisfied.

So we can choose values for the new variables $x_{i+1}, \cdots$ that guarantees that all clauses in I' are satisfied.



solver for A
via reduction from A to B

$I = C_1 \wedge C_2 \wedge \cdots \wedge (l_1 \vee l_2 \vee \cdots \vee l_k) \wedge \cdots \wedge C_{m-1} \wedge C_m$

$I' = C_1 \wedge C_2 \wedge \cdots \wedge (l_1 \vee l_2 \vee x_{i+1}) \wedge (\overline{x_{i+1}} \vee l_3 \vee x_{i+2}) \wedge$
$\cdots (\overline{x_{i+(j-2)}} \vee l_j \vee x_{i+(j-1)}) \wedge \cdots$
$\cdots \wedge (\overline{x_{i+(k-4)}} \vee l_{k-2} \vee x_{i+(k-3)}) \wedge (\overline{x_{i+(k-3)}} \vee l_{k-1} \vee l_k)$

transform one clause to get a new instance I'.
→ translate each clause independently one at a time, eventually we will have that instance I' (SAT) from our original instance I.

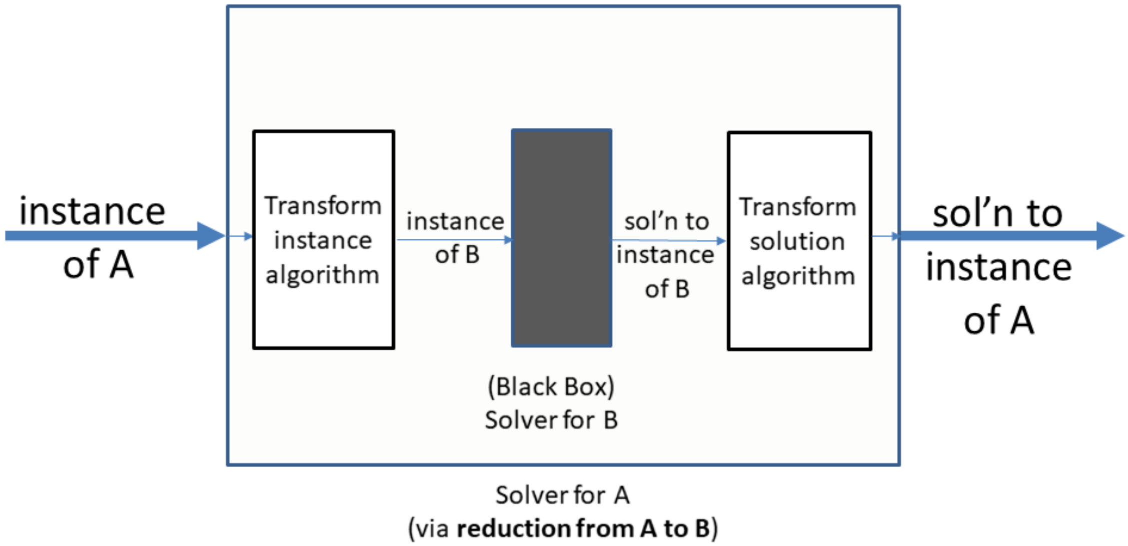
8. Only if direction: I' satisfiable then I is satisfiable.
Let T' be a truth assignment (to variables $x_1, x_2, \ldots, x_n$, plus $x_{i+1} \cdots, x_{i+(k-3)}$) that satisfies I'.
Let T be the same as T' on variables x_1 to x_n. We claim that T satisfies I. Equivalently, in T', at least one of the literals $l_1, l_2, \ldots, l_k$ is true.
Suppose to the contrary that all of $l_1, l_2, \ldots, l_k$ are false. $(l_1 \vee l_2 \vee x_{i+1})$ is true, then x_{i+1} must be true. Then $\overline{x_{i+1}}$ is false, so x_{i+2} must be true, then all up to $x_{i+(k-3)}$ must be true. But $(\overline{x_{i+(k-3)}} \vee l_{k-1} \vee l_k)$ is false since all of the literals are false. The whole thing is false. I' is not satisfied, which is a contradiction.

9. Transform instance algorithm: - transform clause l_1 into $l_1 \vee l_1 \vee l_1$
- transform clause $(l_1 \vee l_2 \vee \cdots \vee l_k)$ using the tranform-clause Procedure.

Transform solution algorithm: - If BBox produces Yes, return Yes
- If BBox produces No, then decide what to return. (No)

10. We know that if I' produced is the output of Transform-Instance (I), then I is satisfiable iff I' is satisfiable. The BBox solver correctly ~~return~~ determines if 3SAT instance I' is satisfiable. Our Transform solution algorithm preserves the answer of the BBox algorithm, so it output yes iff I' is satisfiable,

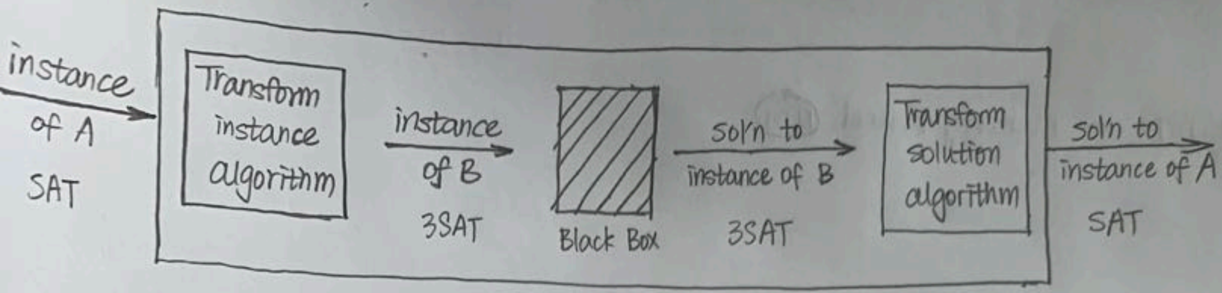
9. Give a reduction from SAT to 3-SAT. Recall that a reduction consists of two algorithms that "connect" one problem to another, as in this diagram:



Transform instance algorithm:

Transform solution algorithm:

10. Why is the reduction correct?



solver for A
via reduction from A to B

$I = C_1 \wedge C_2 \wedge \cdots \wedge (l_1 \vee l_2 \vee \cdots \vee l_k) \wedge \cdots \wedge C_{m-1} \wedge C_m$

$I' = C_1 \wedge C_2 \wedge \cdots \wedge (l_1 \vee l_2 \vee x_{i+1}) \wedge (\overline{x_{i+1}} \vee l_3 \vee x_{i+2}) \wedge$
$\cdots (\overline{x_{i+(j-2)}} \vee l_j \vee x_{i+(j-1)}) \wedge \cdots$
$\cdots \wedge (\overline{x_{i+(k-4)}} \vee l_{k-2} \vee x_{i+(k-3)}) \wedge (\overline{x_{i+(k-3)}} \vee l_{k-1} \vee l_k)$

transform one clause to get a new instance I'.
→ translate each clause independently one at a time, eventually we will have that instance I' (SAT) from our original instance I.

8. Only if direction: I' satisfiable then I is satisfiable.
Let T' be a truth assignment (to variables $x_1, x_2, \dots, x_n$; plus $x_{i+1} \cdots, x_{i+(k-3)}$) that satisfies I'.
Let T be the same as T' on variables x_1 to x_n. We claim that T satisfies I. Equivalently, in T', at least one of the literals $l_1, l_2, \dots, l_k$ is true.
Suppose to the contrary that all of $l_1, l_2, \dots, l_k$ are false. $(l_1 \vee l_2 \vee x_{i+1})$ is true, then x_{i+1} must be true. Then $\overline{x_{i+1}}$ is false, so x_{i+2} must be true, then all up to $x_{i+(k-3)}$ must be true. But $(\overline{x_{i+(k-3)}} \vee l_{k-1} \vee l_k)$ is false since all of the literals are false. The whole thing is false. I' is not satisfied, which is a contradiction.

9. Transform instance algorithm: - transform clause l_1 into $l_1 \vee l_1 \vee l_1$
- transform clause $(l_1 \vee l_2 \vee \cdots \vee l_k)$ using the tranform-clause procedure.

Transform solution algorithm: - If BBox produces Yes, return Yes
- If BBox produces No, then decide what to return. (No)

10. We know that if I' produced is the output of Transform-Instance (I), then I is satisfiable iff I' is satisfiable. The BBox solver correctly ~~return~~ determines if 3SAT instance I' is satisfiable. Our Transform solution algorithm preserves the answer of the BBox algorithm, so it output yes iff I' is satisfiable.

3 What does a reduction tell us?

Here, consider a reduction from problem A to problem B, as illustrated in the figure of part 9.

1. **SCENARIO #1 (how we've used reductions prior to this worksheet):** Say our reduction's two algorithms take $O(f(n))$ time and the black box solver for B also takes $O(f(n))$ time. What can we say about the running time to solve problem A?

2. **SCENARIO #2 (what we usually think of NP-completeness as meaning):** Say our reduction's two algorithms take $O(g(n))$ time and we know that there is **no algorithm** for problem A that runs in $O(g(n))$ time. What do we know about the the running time for problem B? Why?

3. **SCENARIO #3 (what NP-completeness technically means):** Say that we know (which we do) that if SAT can be solved in polynomial time, then **any** problem in the large set called "NP" can also be solved in polynomial time. What does our redution from SAT to 3-SAT tell us? Why?