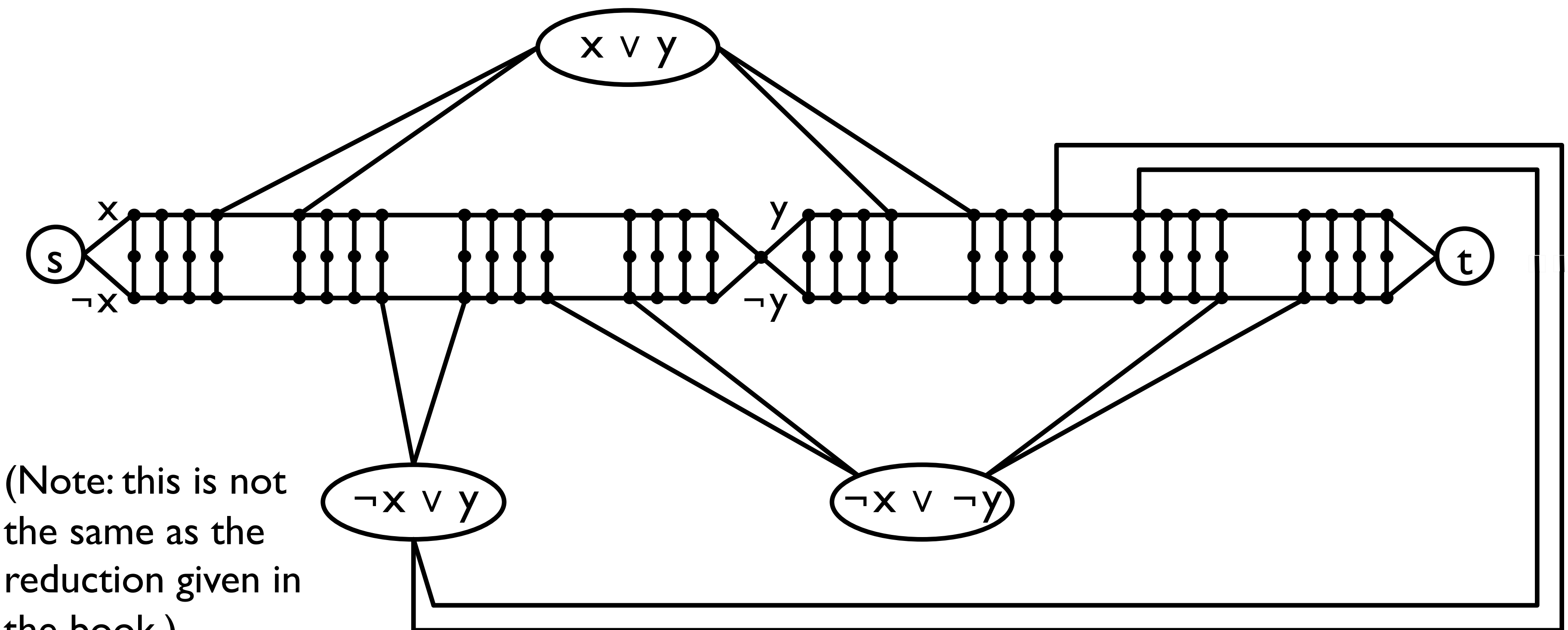
Lecture 24

3-SAT $\leq_p$ UndirectedHamPath

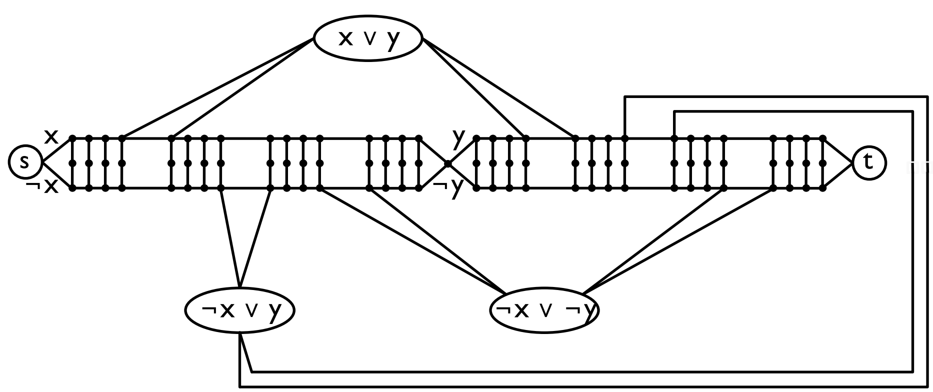
Example: $(x \vee y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$

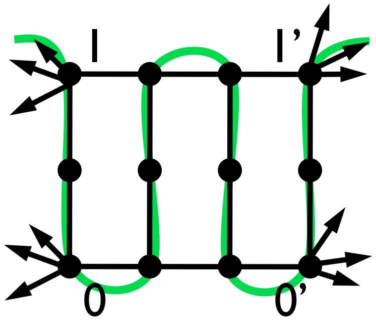


(Note: this is not the same as the reduction given in the book.)

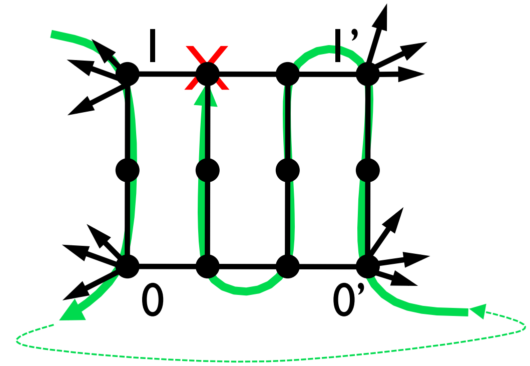
3-SAT $\leq_p$ UndirectedHamPath

Time for the reduction: to be computable in poly time it is necessary (but not sufficient) that G's size is polynomial in n, the length of the formula. Easy to see this is true, since G has q + 12 (p + m) + 1 = O(n) vertices, where q is the number of clauses, p is the number of instances of literals, and m is the number of variables. Furthermore, the structure is simple and regular, given the formula, so easily / quickly computable, but details are omitted. (More detail expected in your homeworks, e.g.)





Ham Path Gadget



Many copies of this 12-node gadget, each with one or more edges connecting each of the 4 corners to other nodes or gadgets (but no other edges to the 8 "internal" nodes).

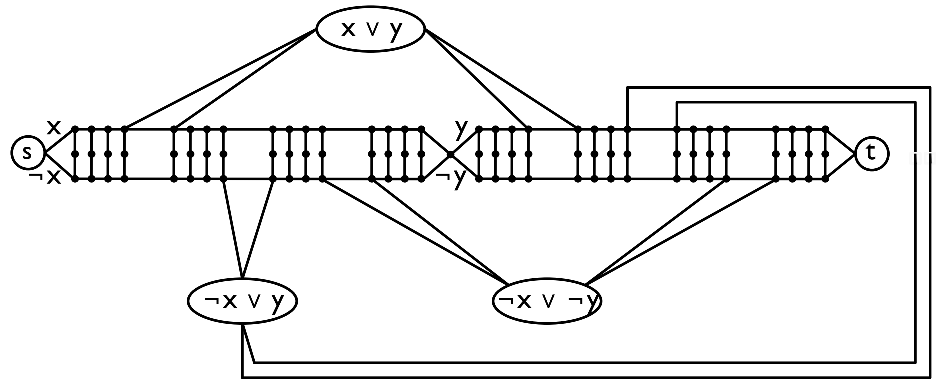
Claim: There are only 2 Ham paths – one entering at 1, exiting at 1' (as shown); the other (by symmetry) 0→0'

Pf: Note *: at 1st visit to any column, must next go to *middle* node in column, else it will subsequently become an untraversable "dead end."
WLOG, suppose enter at 1. By *, must then go down to 0. 2 cases:

Case a: (top left) If next move is to right, then * forces path up, left is blocked, so right again, * forces down, etc; out at 1'.

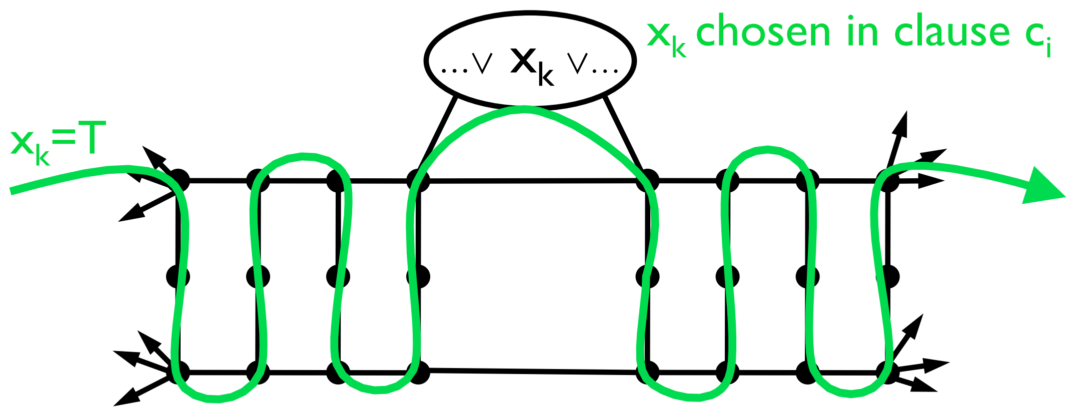
Case b: (top rt) if exit at 0, then path must eventually reenter at 0' or 1'. * forces next move to be up/down to the other of 0'/1'. Must then go left to reach the 2 middle columns, but there's *no exit* from them. So case b is impossible.

Correctness, I

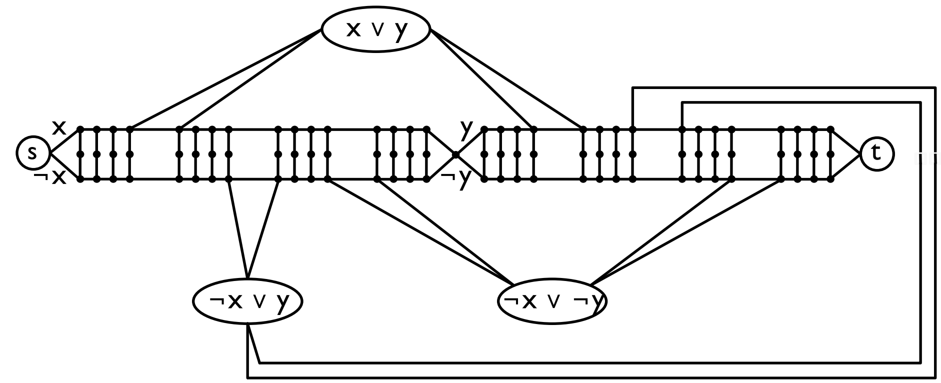


Ignoring the clause nodes, there are 2^m s-t paths along the "main chain," one for each of 2^m assignments to m variables.

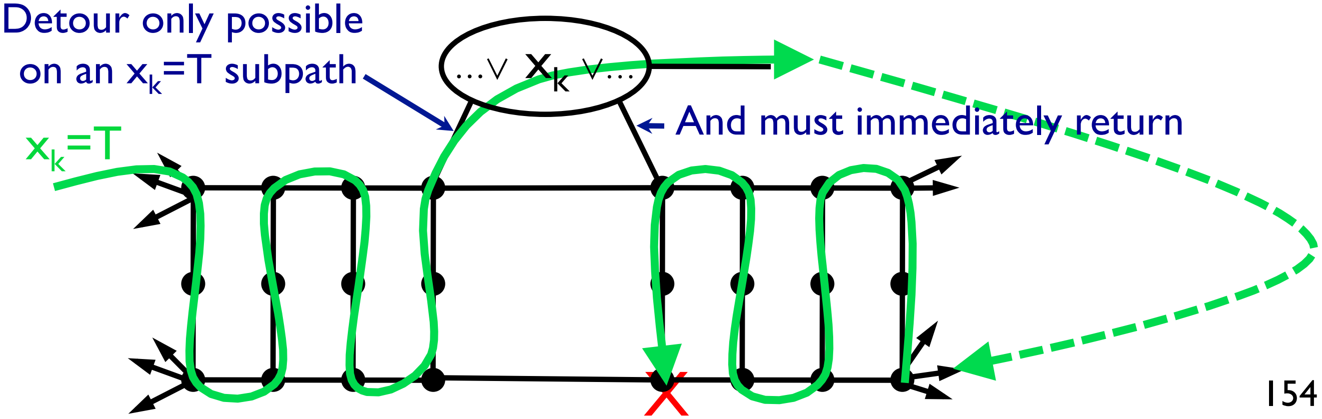
If f is satisfiable, pick a satisfying assignment, and pick a true literal in each clause. Take the corresponding "main chain" path; add a detour to/from c_i for the true literal chosen from clause i. Result is a Hamilton path.



Correctness, II



Conversely, suppose G has a Ham path. Obviously, the path must detour from the main chain to each clause node c_i. If it does not return *immediately* to the next gadget on main chain, then (by gadget properties on earlier slide), that gadget cannot be traversed. Thus, the Ham path must consistently use "top chain" or consistently "bottom chain" exits to clause nodes from each variable gadget. If top chain, set that variable True; else set it False. Result is a satisfying assignment, since each clause is visited from a "true" literal.



Subset-Sum, AKA Knapsack

KNAP = { (w_1, w_2, …, w_n, C) | a subset of the w_i sums to C }

w_i's and C encoded in radix $r \geq 2$. (Decimal used in following example.)

Theorem: 3-SAT $\leq_P$ KNAP

Pf: given formula with p variables & q clauses, build KNAP instance with 2(p+q) w_i's, each with (p+q) decimal digits. For the 2p "literal" weights, H.O. p digits mark which variable; L.O. q digits show which clauses contain it. Two "slack" weights per clause mark that clause. See example below.

3-SAT $\leq_p$ KNAP

Formula: $(x \vee y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$

		Variables		Clauses		
		x	y	$(x \vee y)$	$(\neg x \vee y)$	$(\neg x \vee \neg y)$
Literals	w_1 (x)	1	0	1	0	0
	w_2 $(\neg x)$	1	0	0	1	1
	w_3 (y)		1	1	1	0
	w_4 $(\neg y)$		1	0	0	1
Slack	w_5 (s_{11})			1	0	0
	w_6 (s_{12})			1	0	0
	w_7 (s_{21})				1	0
	w_8 (s_{22})				1	0
	w_9 (s_{31})					1
	w_{10} (s_{32})					1
	C	1	1	3	3	3

Correctness

Poly time for reduction is routine; details omitted

If formula is satisfiable, select the literal weights corresponding to the true literals in a satisfying assignment. If that assignment satisfies k literals in a clause, also select (3 – k) of the "slack" weights for that clause. Total will equal C.

Conversely, suppose KNAP instance has a solution. Note ≤ 5 one's per column, so no "carries" in sum (recall – weights are decimal); i.e., columns are decoupled. Since H.O. p digits of C are 1, exactly one of each pair of literal weights included in the subset, so it defines a valid assignment. Since L.O. q digits of C are 3, but at most 2 "slack" weights contribute to it, at least one of the selected literal weights must be 1 in that clause, hence the assignment satisfies the formula.