Propositional reasoning as a paving problem

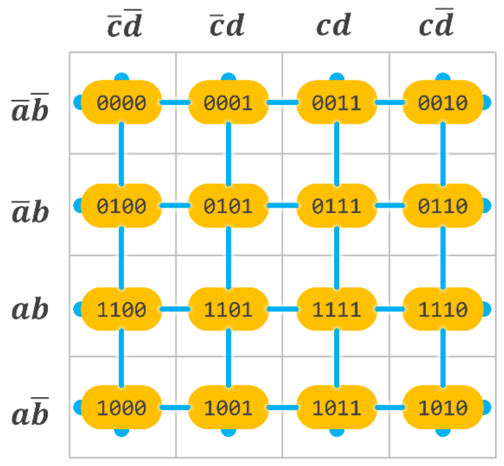
Olivier Bailleux, https://orcid.org/0000-0002-3433-7344
Research idea sheet n°2, january 2018

Abstract

SAT can be seen as a paving problem, where each clause is an hypercubic tile covering a part of the interpretation space, and the goal is to determine whether any interpretation is covered by at least one clause. Such an approach could lead to new algorithms and heuristics for automatic reasoning. In this note, we propose a SAT solver that splits some tiles into smaller ones, and merges them, if applicable, into a large tile representing the empty clause.

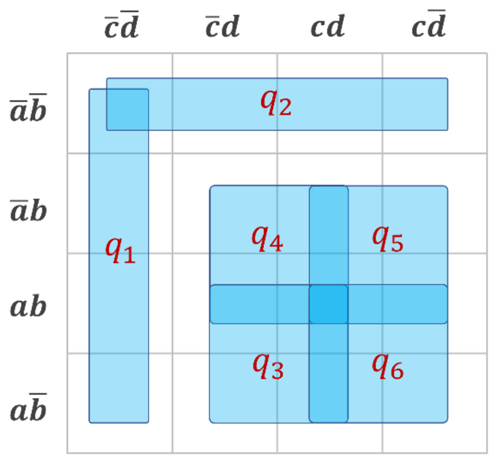
Reasoning is paving

The interpretation space (or assignation space) of a set V of n propositional variables can be represented as an n-dimensional hypercube, where each vertex is an assignation of the variables of V. Here is an example for $V = \{a, b, c, d\}$.



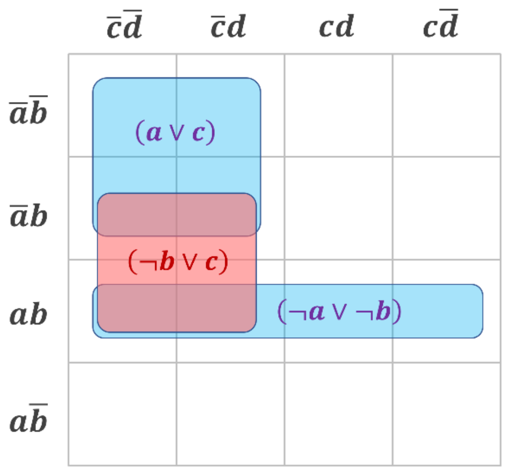
In this context, any clause q can be represented as an hypercubic tile covering the assignations that falsify q. For example, the formula $\Sigma = (\boldsymbol{c} \vee \boldsymbol{d}) \wedge (\boldsymbol{a} \vee \boldsymbol{b}) \wedge (\neg\boldsymbol{a} \vee \neg\boldsymbol{d}) \wedge (\neg\boldsymbol{b} \vee \neg\boldsymbol{d}) \wedge (\neg\boldsymbol{b} \vee \neg\boldsymbol{c}) \wedge (\neg\boldsymbol{a} \vee \neg\boldsymbol{c})$ can be represented as follows.

$$\Sigma \left\{ \begin{array}{l} q_1 : (\boldsymbol{c} \vee \boldsymbol{d}) \\ q_2 : (\boldsymbol{a} \vee \boldsymbol{b}) \\ q_3 : (\neg\boldsymbol{a} \vee \neg\boldsymbol{d}) \\ q_4 : (\neg\boldsymbol{b} \vee \neg\boldsymbol{d}) \\ q_5 : (\neg\boldsymbol{b} \vee \neg\boldsymbol{c}) \\ q_6 : (\neg\boldsymbol{a} \vee \neg\boldsymbol{c}) \end{array} \right.$$

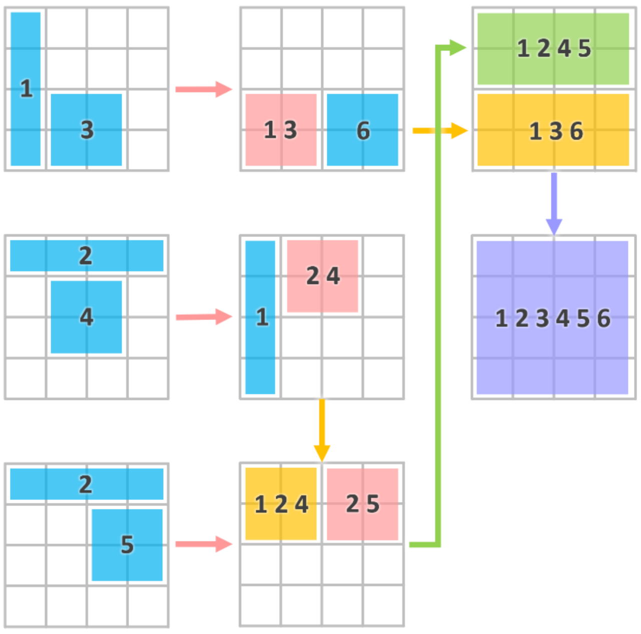


The Resolution rule can be applied to any pair $\{q_1, q_2\}$ of contiguous tiles, i.e., disjoint tiles such that at least one vertex of q_1 is connected to a vertex of q_2 by an edge of the hypercube representing the interpretation space. The resolvent of q_1 and q_2 is then the largest hypercubic tile that is entirely covered by the wo premises q_1 and q_2.

Il the following example, the two contiguous premises $(a \wedge c)$ and $(\neg a \wedge \neg b)$ allow to produce the resolvent $(\neg b \wedge c)$.

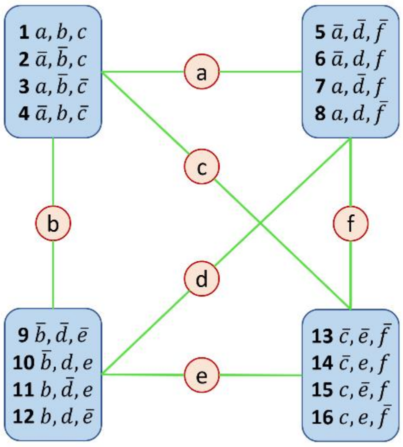


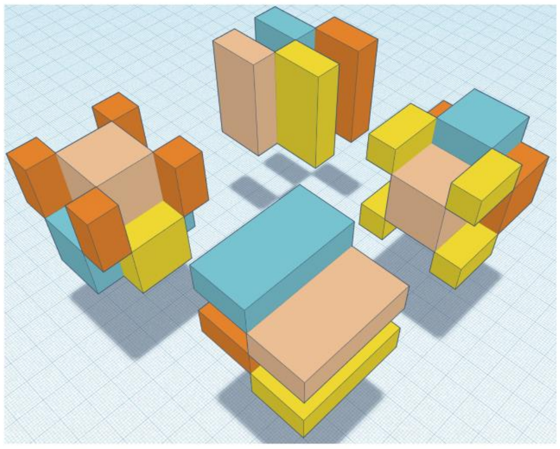
Here are 7 Resolution steps allowing to refute Σ. The blue tiles represent the original clauses. The pink tiles are the first generation resolvents. The yellow tiles are the second generation resolvents, etc. The numbers in each tile represent the initial clauses involved in its production.



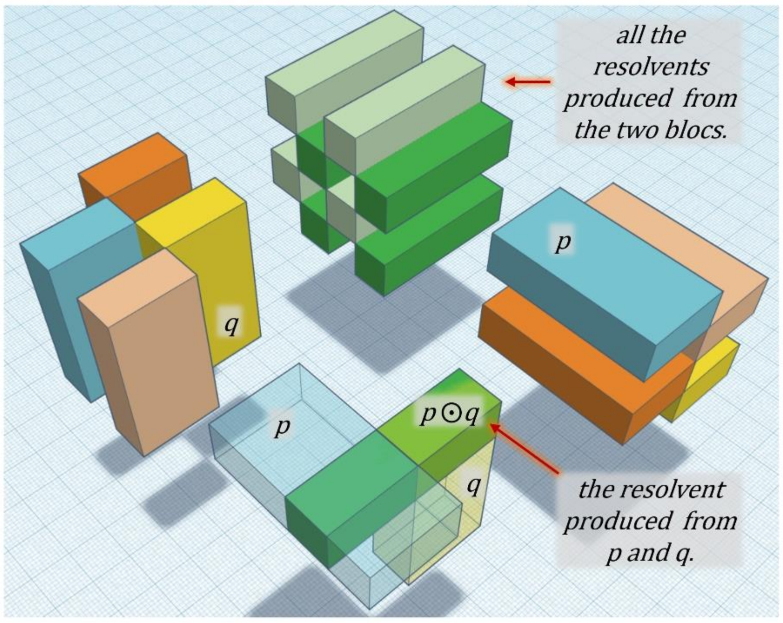
Sometimes, Resolution produces a smallest tile as the used premises. And some formulae can only be refuted by producing smaller tiles than the initial ones. In [Hard Examples for Resolution, Alasdair Urquhart, 1987], a family of such formulae is presented, whose refutation by Resolution requires to produce more and more smaller tiles before it becomes possible to merge these tiles into larger and larger ones until the empty clause is produced.

Here are the initial tiles of such a formula with 6 variables and 16 clauses of 3 literals, grouped into 4 blocks. The interpretation space is the 6-dimensional hypercube represented as a 3-dimensional Karnaugh map.





The clauses belonging to the same block are not contiguous and then can not be resolved together. The only possible resolvents of first generation can be obtained from premises belonging to two different blocks, and the corresponding tiles are smaller than the initial one, as shown bellow.


all the resolvents produced from the two blocs.
p
q
p
$p \odot q$
q
the resolvent produced from p and q.

The relationship between the size of the smallest tiles (the widest clauses) and the size of the shortest refutations is well established in [Short Proofs are Narrow - Resolution made simple, Eli Ben-Sasson and Avi Wigderson, 2000].

Reverse Resolution: a new algorithm for SAT?

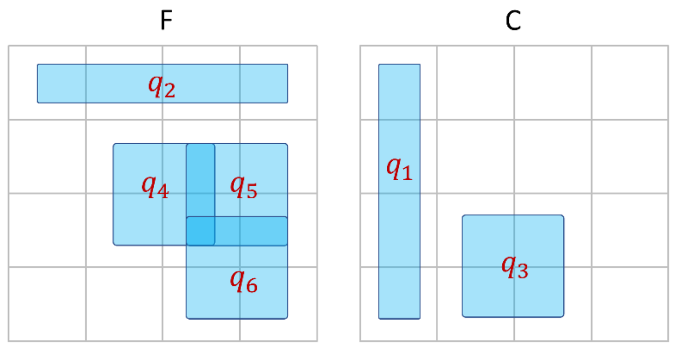
Often, Resolution produces tiles that overlap those that already exist, and somehow increases the average number of tiles covering an assignment. This phenomenon can be limited by the elimination of tiles completely covered by only one other. This operation is called subsumption.

We propose a new approach that consists in gradually repaving the interpretation space with non-overlapping parts of the initial tiles. We work with two sets of clauses: F, which initially contains all the clauses, and C, which is initially empty. Then the following sequence is repeated until either C covers all the space, or F is empty.

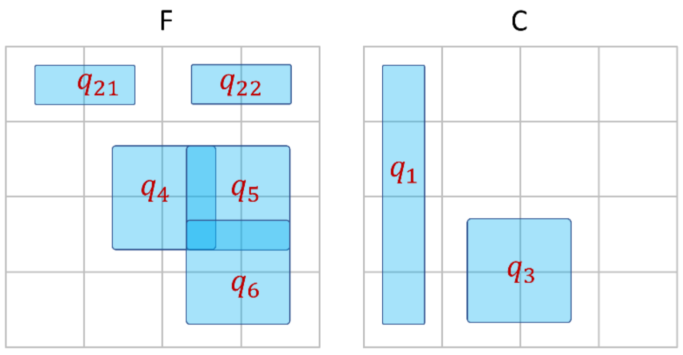
1. Choose a clause q in F that does not overlap any clause of C, if applicable, then remove q from F and add it to C.

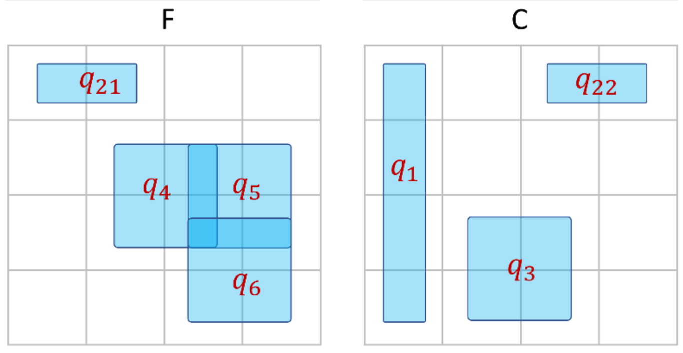
2. If step 1 fails, choose a clause q in F and split it into two clauses q1 = q ∨ x and q2 = q ∨ ¬x, where x is a variable not used in q. If q1 or q2 is covered (subsumed) by either another clause of F or a clause of C, then remove it from F.

Here, we can see F and C after 2 iterations from the formula Σ.

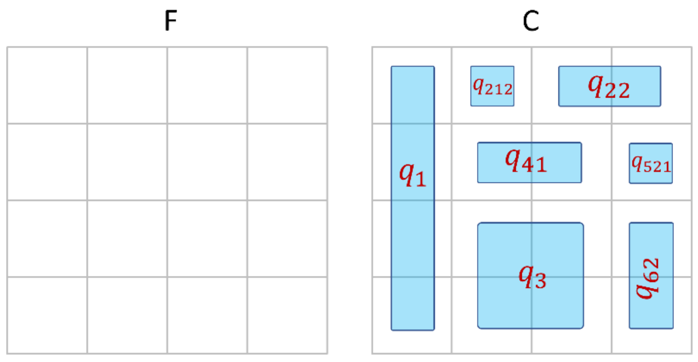


Here are the states after 3 and 4 iterations.





And here is the final state, where the tiles in C cover all the interpretation space with no overlapping, which can be checked quickly and proves that the formula is inconsistent.



Clearly enough, the way in which the clauses to be cut are chosen and how they are split is likely to impact the efficiency of this algorithm.

Interestingly, the splitting operation is like a reverted Resolution. The two clauses obtained from a clause q can be used as premises to produce q by Resolution, more precisely by a so called self-subsuming Resolution. For this reason, we propose to call this operation *reverse Resolution*.

New heuristics for SAT solvers?

If the clauses of a formula Σ does not overlap, then it is very easy to determine if this formula is consistent or not. In a way, what makes SAT difficult is clause overlapping. On this basis, we can imagine different heuristics to guide Resolutions or other forms of deductions; heuristics specifically aimed at growing a set of non-overlapping clauses. The total volume of the tiles in this set would act as an indicator of search progress.

In the previously presented reverse Resolution algorithm, the set C represents the knowledge accumulated since the beginning of the search, and the progress indicator is the sum of the volumes of the clauses in C. But such a cumulated knowledge could be maintained by another SAT solving algorithm, such as a CDCL solver or a DPLL solver with restarts, and then used to guide the production of new clauses. We present this as a very general idea that deserves to be refined.