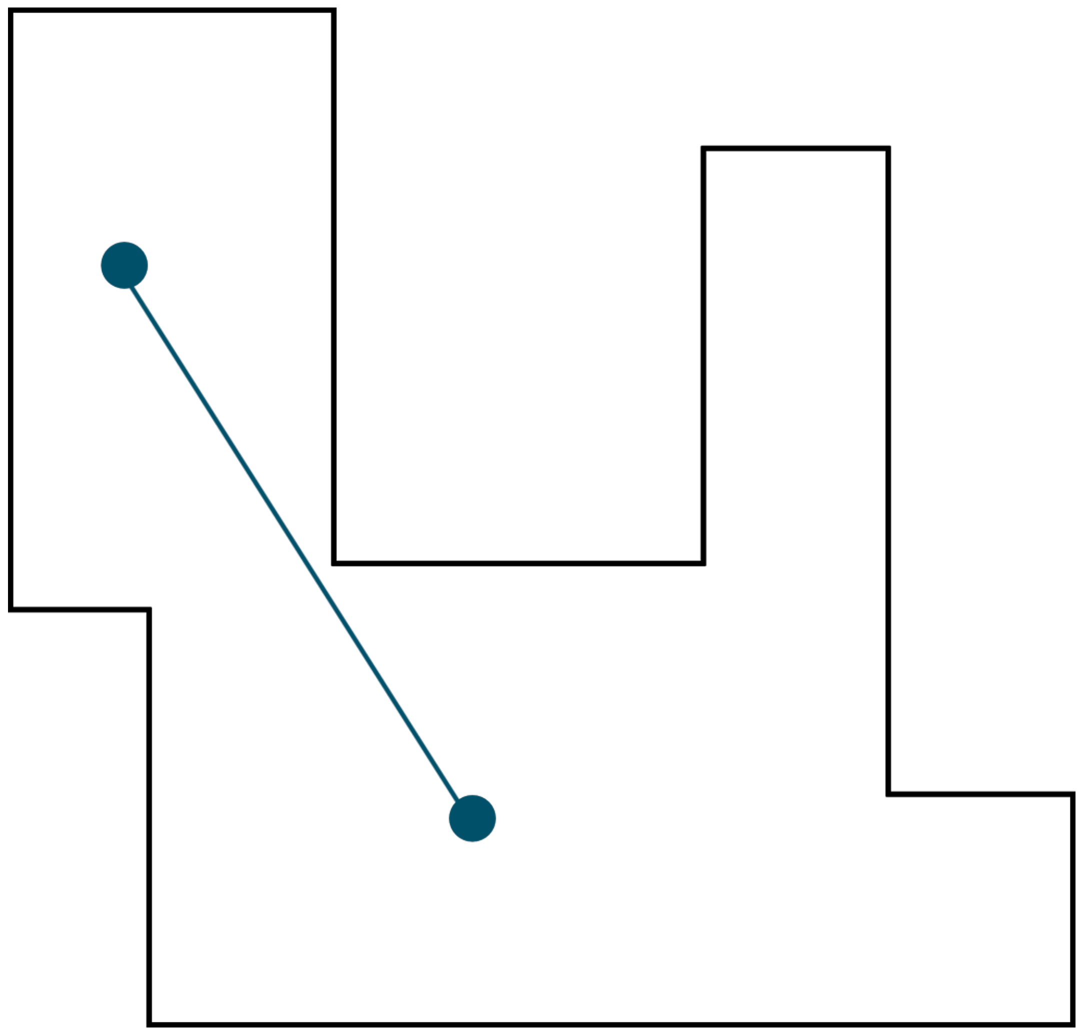
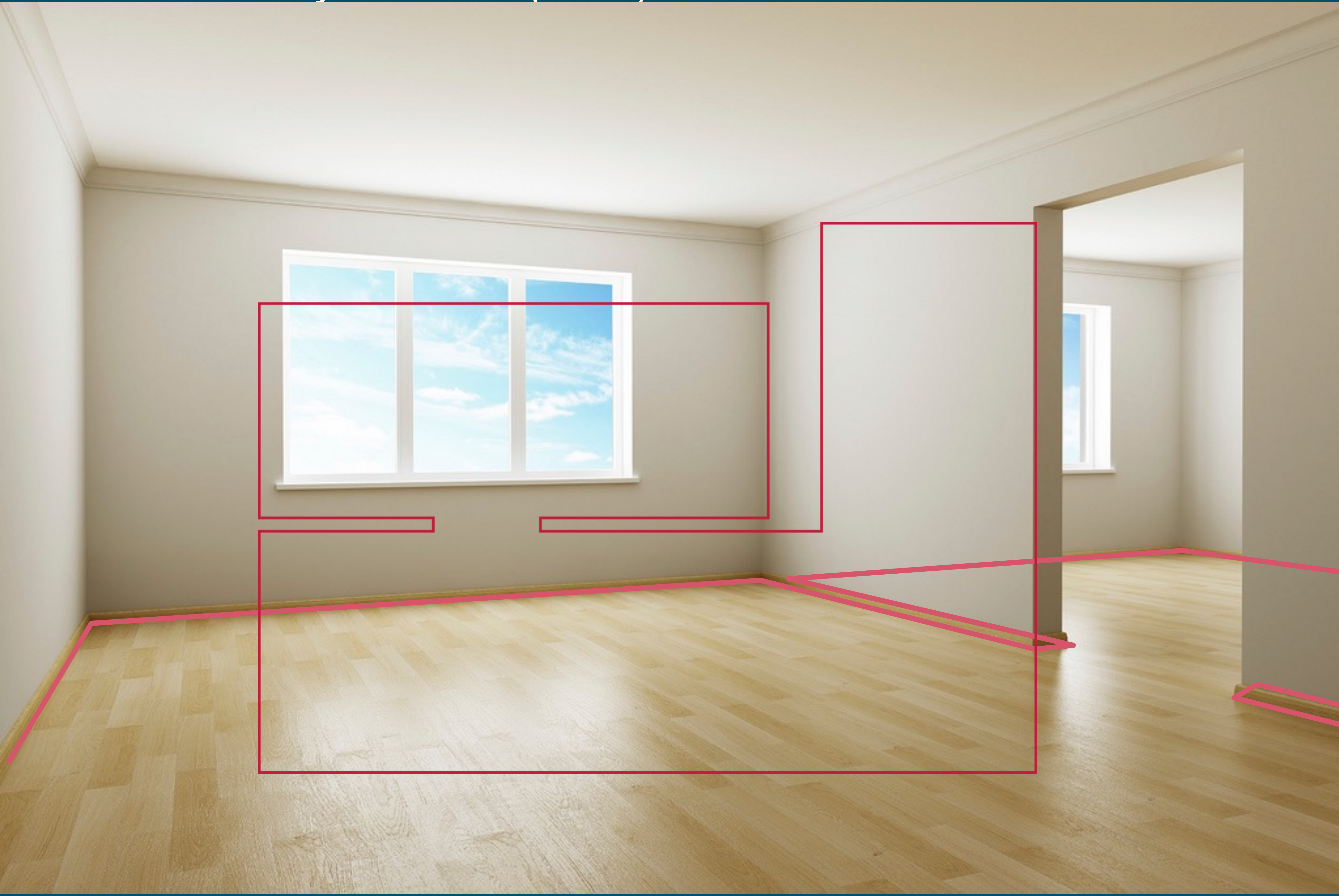
4. Reductions

4.1 The Art Gallery Problem (AGP)

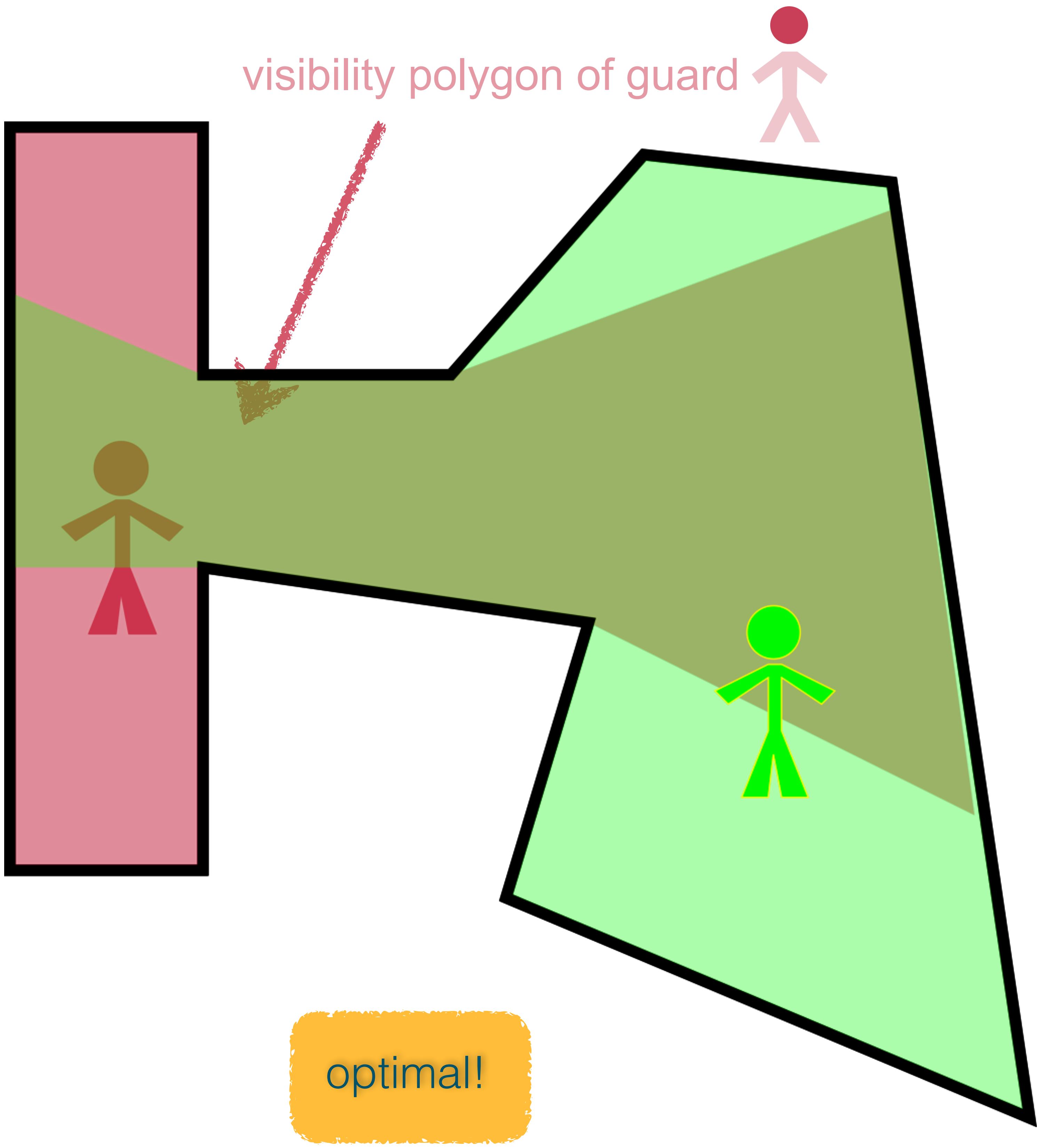
Rectangular vision



The Art Gallery Problem (AGP)



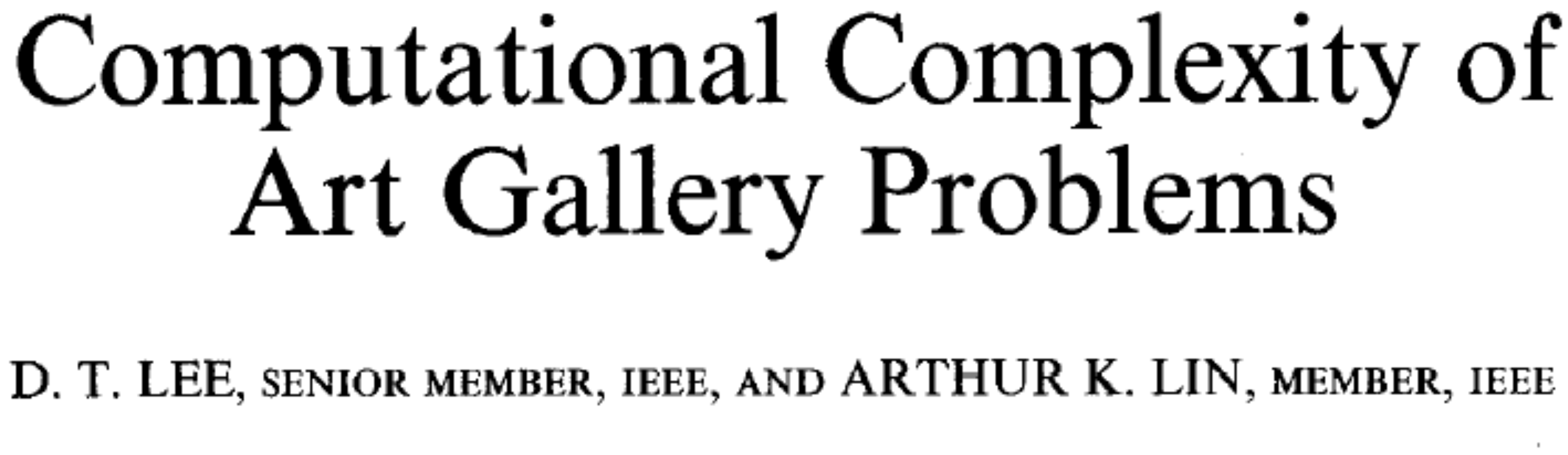
The Art Gallery Problem (AGP)



Given: Polygon P

How many guards do we need to monitor P?

Guard?



Computational Complexity of Art Gallery Problems

D. T. LEE, SENIOR MEMBER, IEEE, AND ARTHUR K. LIN, MEMBER, IEEE

Reduction from 3SAT

Given: 3SAT instance A set $U = \{u_1, u_2, \ldots, u_n\}$ of Boolean variables and a collection $C = \{ c_1, c_2, \ldots, c_m\}$ of clauses over U exist such that $c_i \in C$ is a disjunction of precisely three literals.

Literal pattern: Only vertices a and b can cover the distinguished area of the pattern.

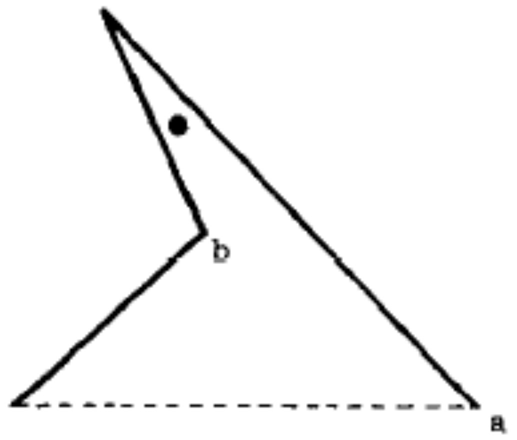


Fig. 1. Literal pattern.

Three of these per clause, each corresponds to one literal.

Reduction from 3SAT

Clause junction: C=A+B+D, A∈{u_i, ¬u_i}, B∈{u_j, ¬u_j}, D∈{u_k, ¬u_k}, u_i, u_j, u_k variables in U

Lemma I: At least three vertices of C_h, are required to cover the region defined by the pattern C_h shown in Fig. 2.
Lemma 2: Only seven three-vertex covers exist that can cover the region defined by the pattern C_h.
Lemma 3: The three vertices selected from the clause pattern C_h = A + B + D cover the region defined by C_h if and only if the truth values represented by the labels of these vertices give a true assignment for C_h.

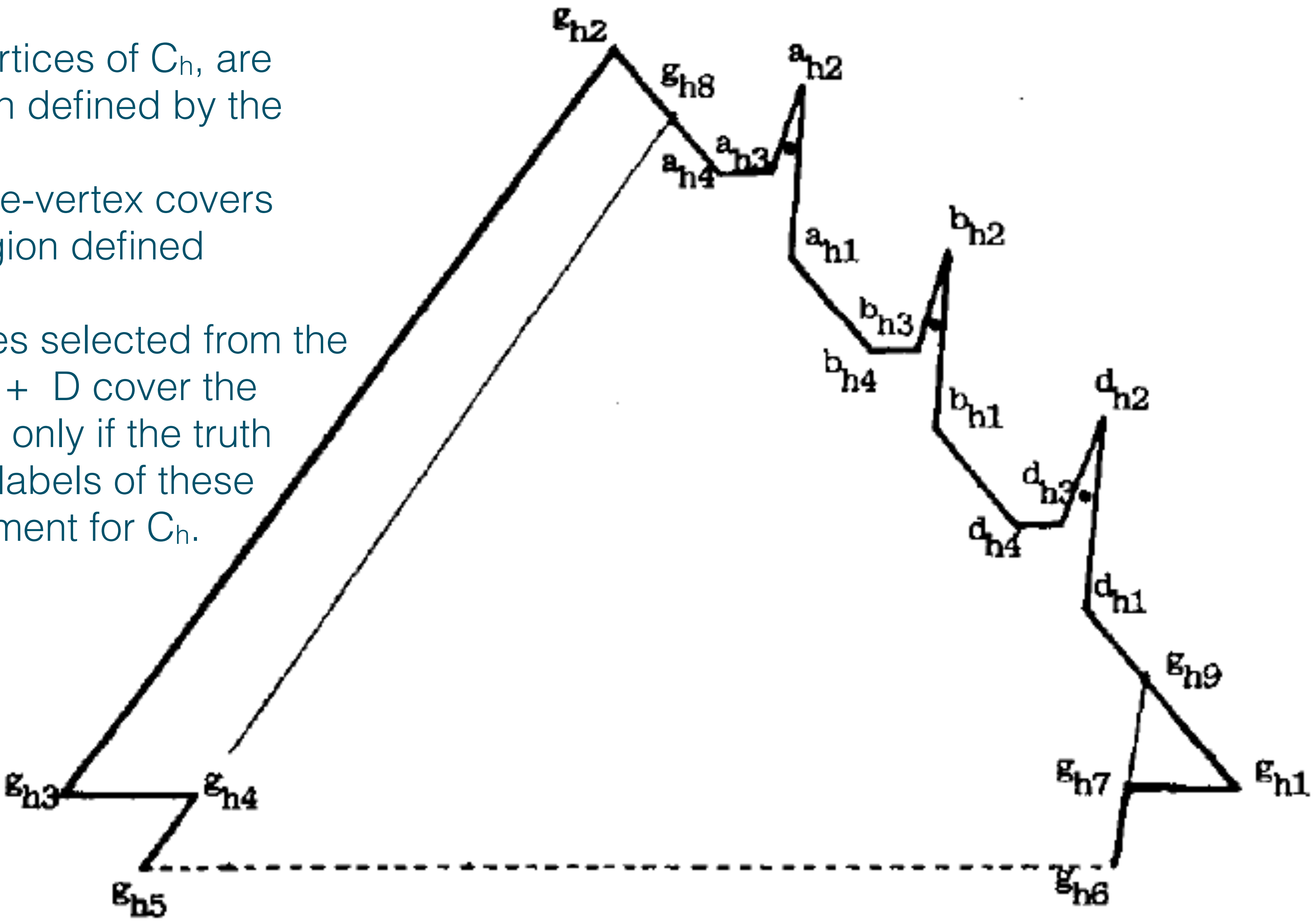


Fig. 2. Clause junction C_h.

Reduction from 3SAT

Variable pattern:

- One such pattern will exist per variable in the final construction.
- The two legs of the variable pattern called *rectangles or rectangular regions*, although they are not really rectangles

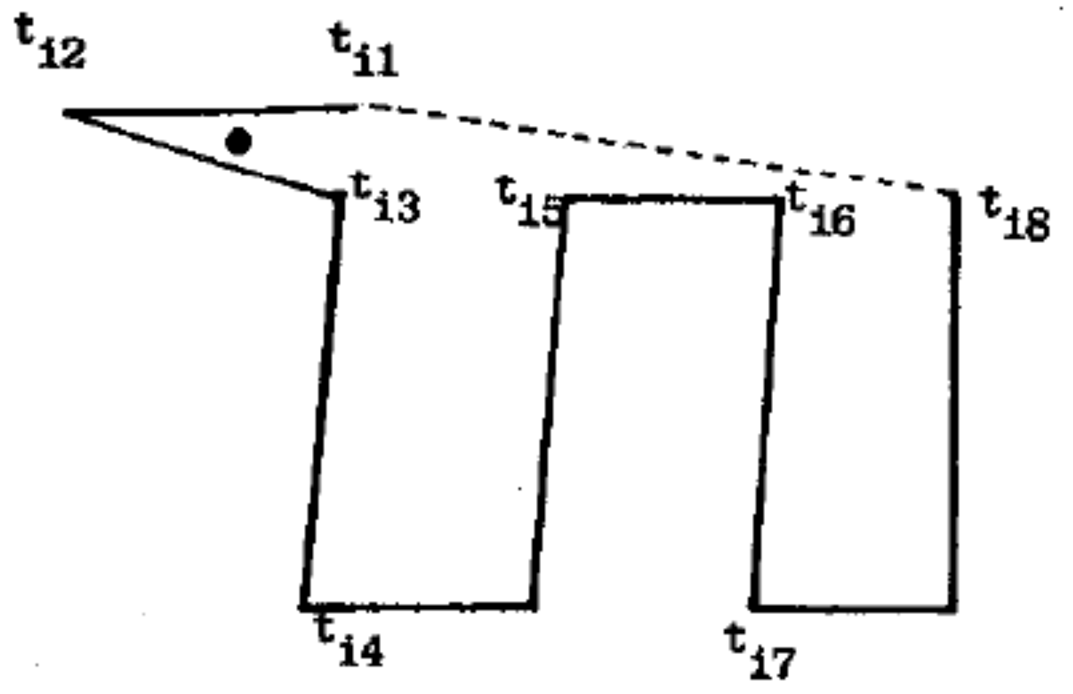


Fig. 3. Variable pattern; $t_{i1}t_{i2}$ and $(t_{i3}, t_{i5}, t_{i6}, t_{i8})$ are parallel.

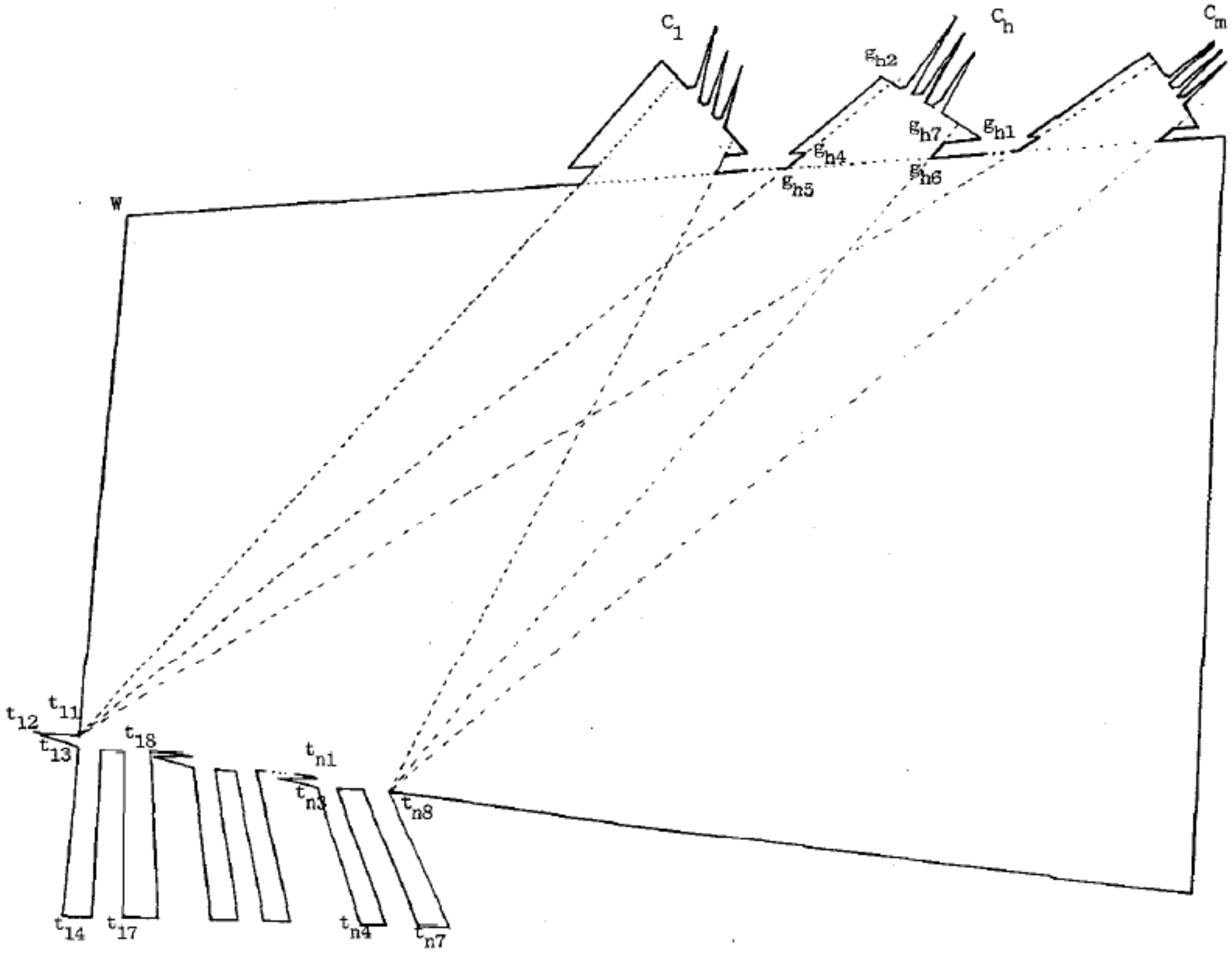


Fig. 4. Putting variable patterns and clause junctions together.

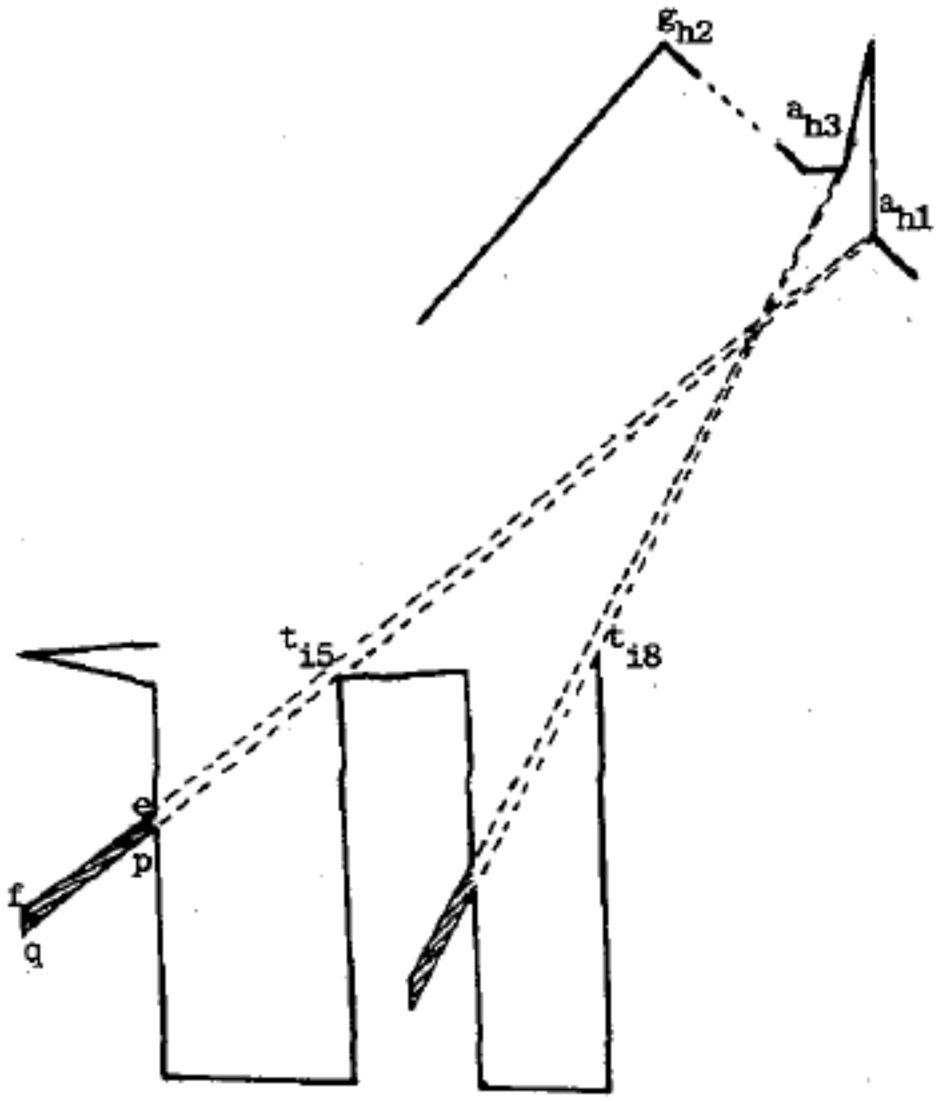


Fig. 7. Each spike is replaced with small region.

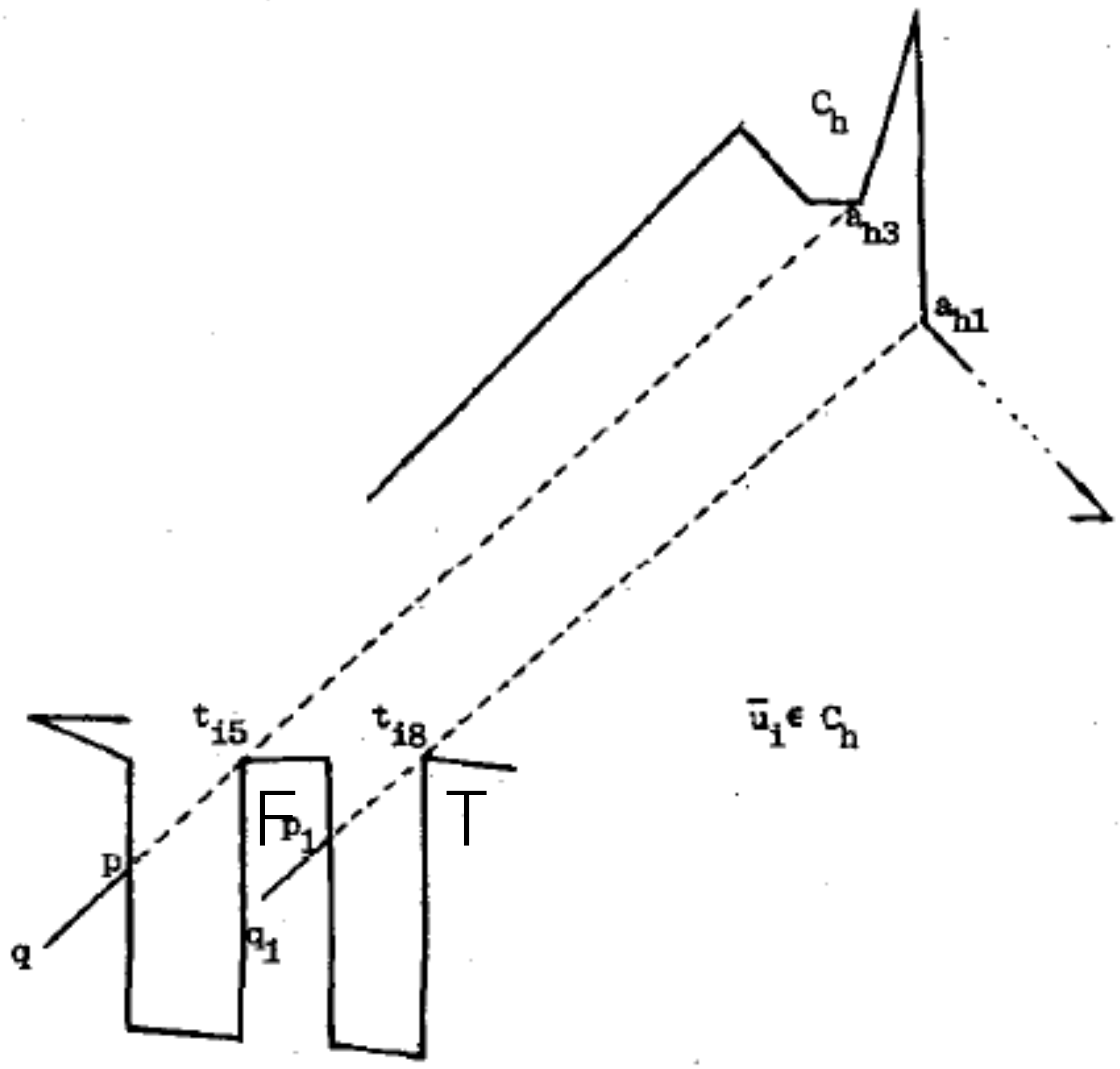


Fig. 6. Augumenting spikes when $\bar{u}_i$ in C_h.

We call the polygonal regions created as described the *consistency-check patterns.*

$F = (u_1 + u_2 + u_3) \wedge (u_1 + \bar{u}_2 + u_3) \wedge (u_1 + \bar{u}_2 + \bar{u}_3)$

sees into all wells

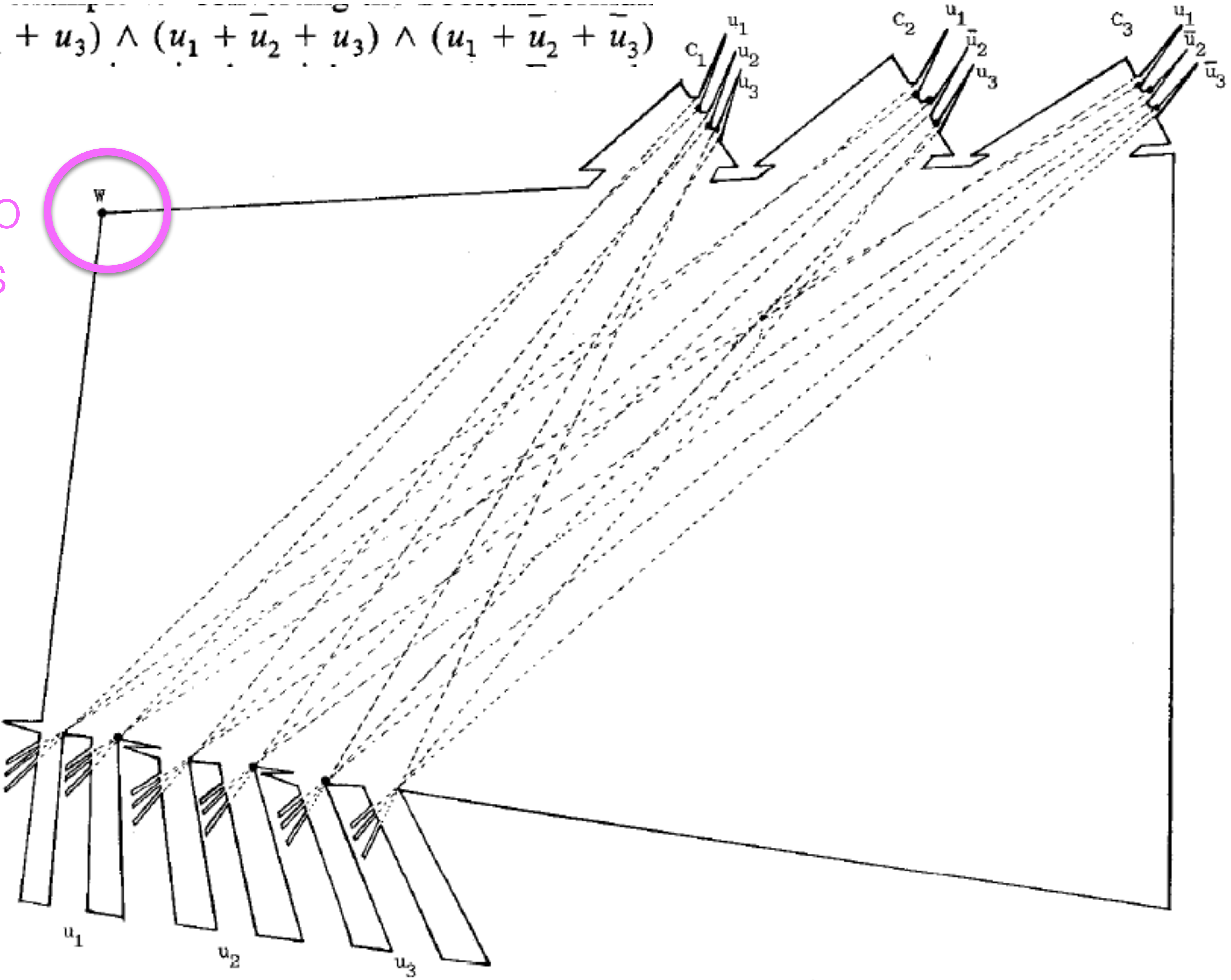


Fig. 8. Example and minimum cover.

$F = (u_1 + u_2 + u_3) \wedge (u_1 + \bar{u}_2 + u_3) \wedge (u_1 + \bar{u}_2 + \bar{u}_3)$

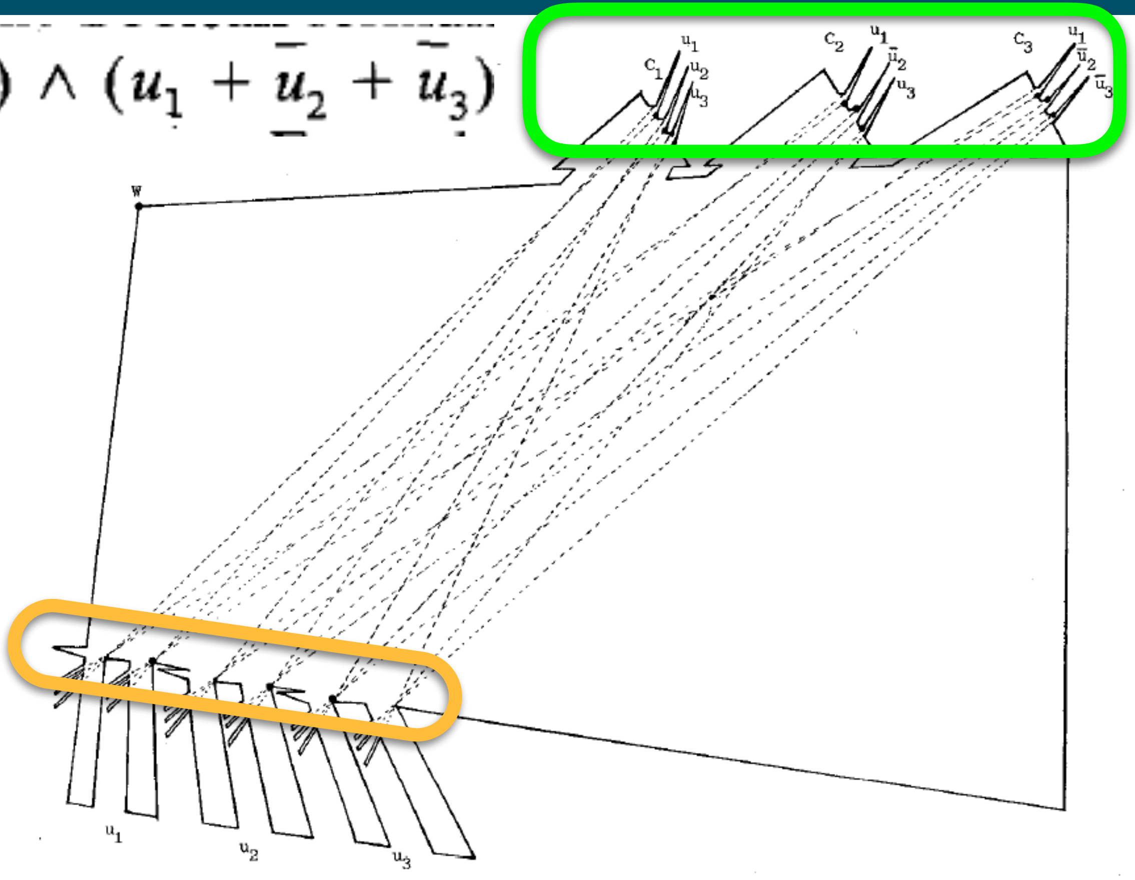


Fig. 8. Example and minimum cover.

Lemma 4: At least K = 3m + n + 1 vertices are needed for covering the simply connected polygonal region.
Proof: At least 3m + n vertices are needed for covering 3m literal patterns, and $\Delta t_{i1} t_{i2} t_{i3}$, i = 1,2; . ., n. At least one more vertex is needed to cover all the variable patterns' rectangles. Therefore, the lemma follows.

Lemma 5: The minimum number of vertices needed to cover the simply connected polygonal region is K = 3m +n + 1 if and only if C is satisfiable.

The Art Gallery Problem (AGP)

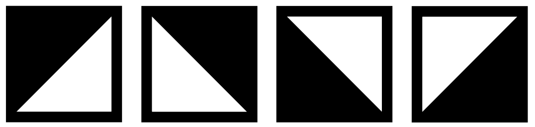
NP-hard
- point guards with holes [O'Rourke & Supowit 1983]
- vertex guards without holes [Lee & Lin 1986]
- point guards without holes [Aggarwal 1986]

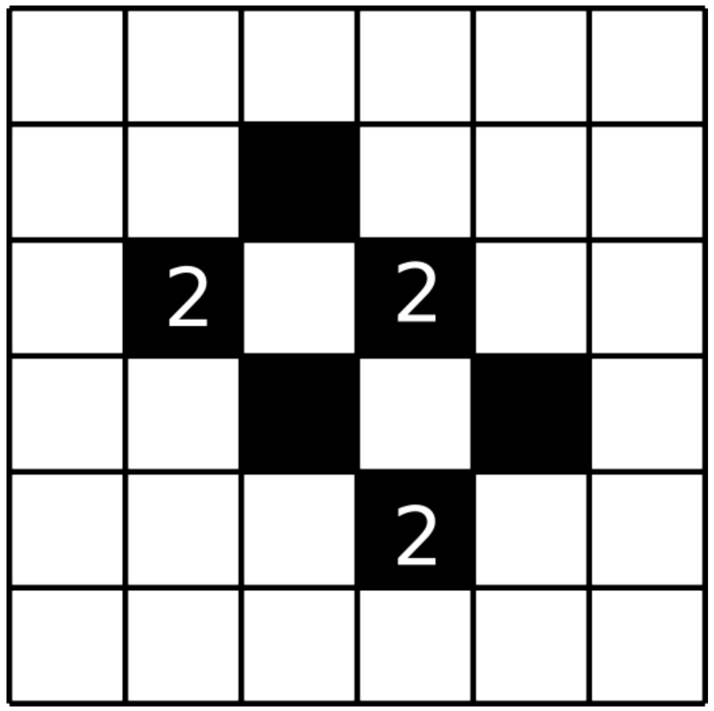
APX-hard [Eidenbenz, Stamm & Widmayer 2001]

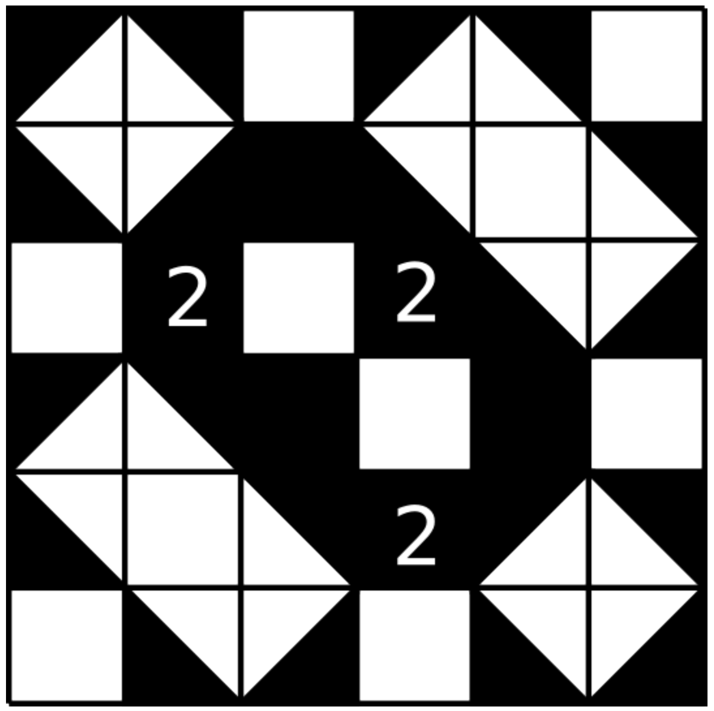
What to do?
-Approximation
-Exact Solutions
-Heuristics

4.2 Shakashaka

Shakashaka

- Shakashaka is a pencil-and-paper puzzle, proposed by Guten in 2008 and popularized by the Japanese publisher Nikoli
- An instance of Shakashaka consists of an mxn rectangle of unit squares. Initially, each square is colored either black or white, and black squares may also contain an integer between 0 and 4, inclusive. The solver proceeds by filling in the initially white squares with squares consisting of a black and a white triangle in one of four orientations: (b/w squares) 
- The white squares may also be left blank.
- The numbers written in black squares constrain the solver by specifying the number of b/w squares that must neighbor the given square (in its four vertically and horizontally neighboring squares).
- An instance is considered solved if every maximal con nected white region on the board is a rectangle (axis-aligned or rotated by 45) and each numbered black square has exactly as many b/w square neighbors as is specified by its number. Example and its (unique) solution:





Shakashaka

Demaine et al. proved that Shakashaka is NP-complete. They used a reduction from planar 3-SAT, and the black squares in the reduction either contained the number 1 or remained blank.
We (Aviv Adler, Michael Biro, Erik Demaine, Mikhail Rudoy, Christiane Schmidt) showed that Shakashaka without numbers in the black squares is NP-complete by a reduction from POSITIVE PLANAR 1-IN-3 SAT, a variant of the well known PLANAR 3-SAT problem, shown to be NP-complete by Mulzer and Rote. The reduction is parsimonious, and, hence, also shows #P-completeness.

For now: think about how a variable gadget could look like.

Shakashaka

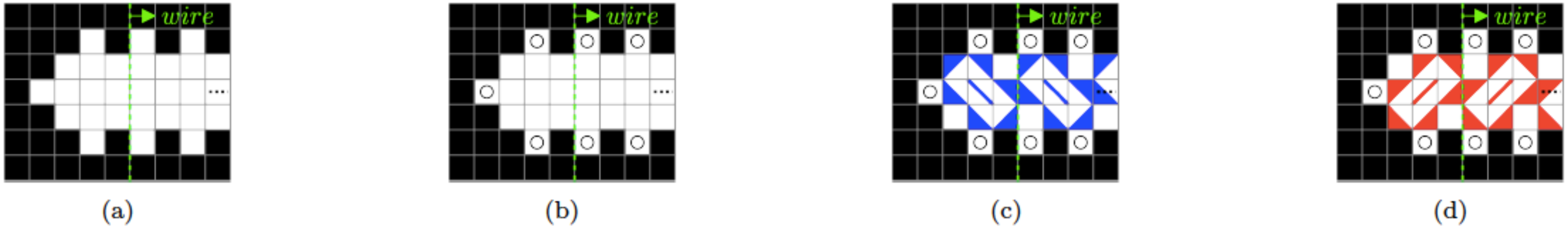


Figure 2: (a) The variable gadget, (b) with enforced white pixels and (c),(d) the two possible feasible solutions. We associate the “kite” in blue with a truth setting of “false” and the “kite” in red with a truth setting of “true”.

Shakashaka

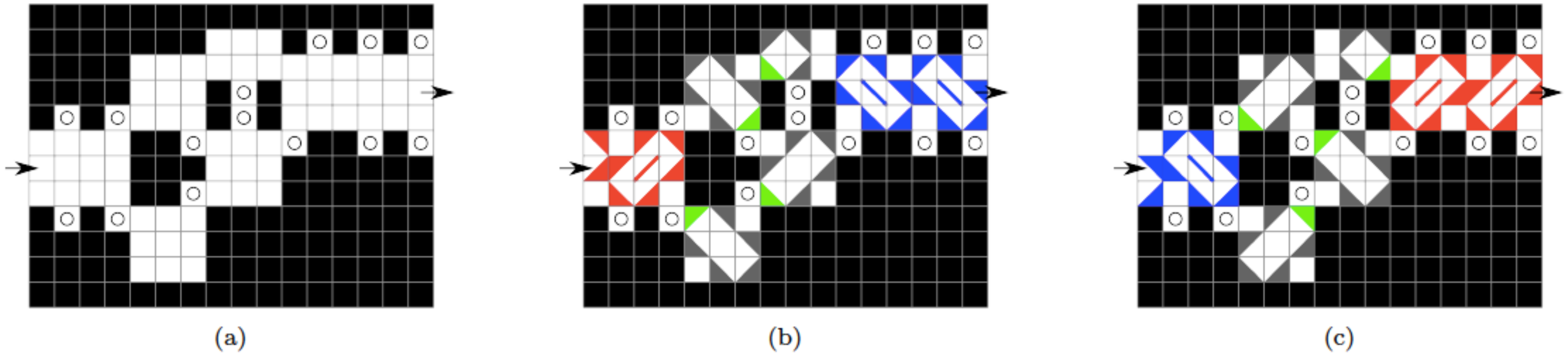


Figure 3: (a) The NOT gadget. (b),(c) The wires connected by the NOT gadget always satisfy opposite truth assignments. In (b) the gadget is entered with a truth assignment corresponding to "true" and left with a truth assignment corresponding to "false". Those roles are reversed in (c). Some enforced triangles are shown in green to facilitate understanding.

Shakashaka

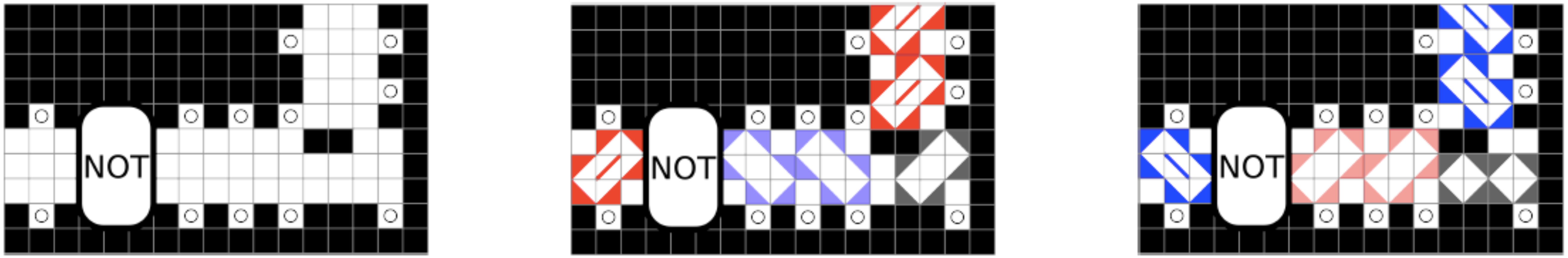


Figure 4: (a) The bend gadget. (b),(c) The wires connected by the bend always satisfy the same truth assignment.

Shakashaka

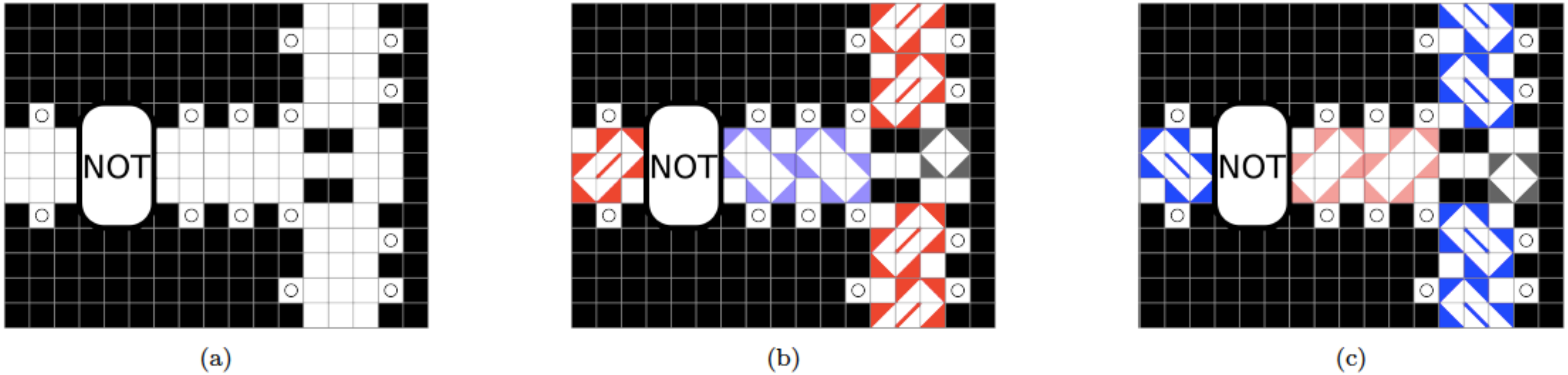


Figure 5: (a) A split of the corridor. (b),(c) The wires connected by the split always satisfy the same truth assignment.

Shakashaka

The at-most gadget enables us to enforce that at most one of a pair of truth assignments is true.

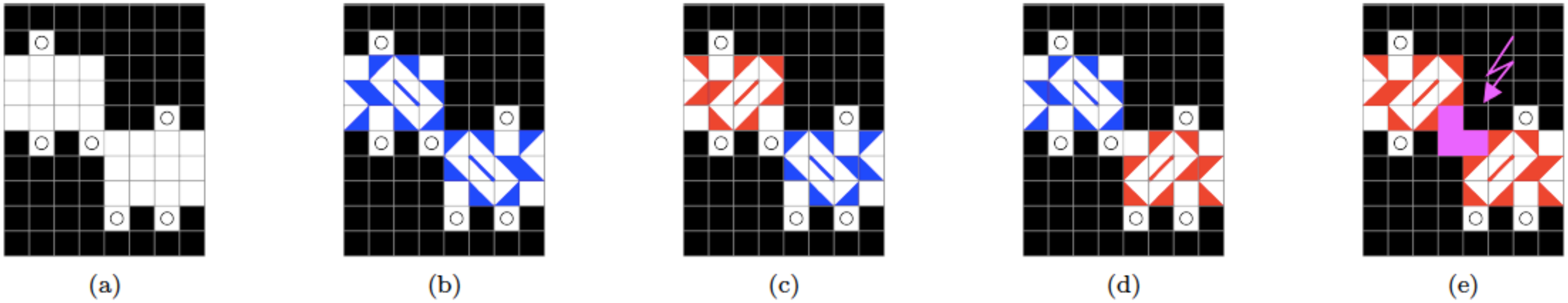


Figure 6: (a) The “at most” gadget. (b) with two false inputs, (c)/(d) with one true and one false input, (e) with two true inputs the board cannot be completed.

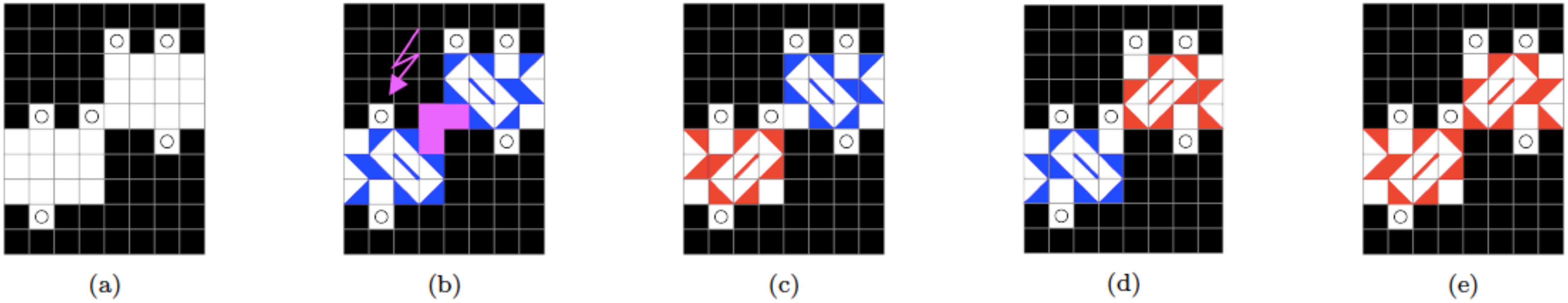


Figure 7: (a) The “at least” gadget: (b) with two false inputs the board cannot be completed, (c)/(d) with one true and one false input, (e) with two true inputs.

The at-most gadget enables us to enforce that at least one of a pair of truth assignments is true.

Shakashaka

The "**XOR**" gadget, shown in Figure 8, takes two wires as input and outputs:

false/false	$\rightarrow$	false
false/true	$\rightarrow$	true or false possible
true/false	$\rightarrow$	true or false possible
true/true	$\rightarrow$	infeasible.

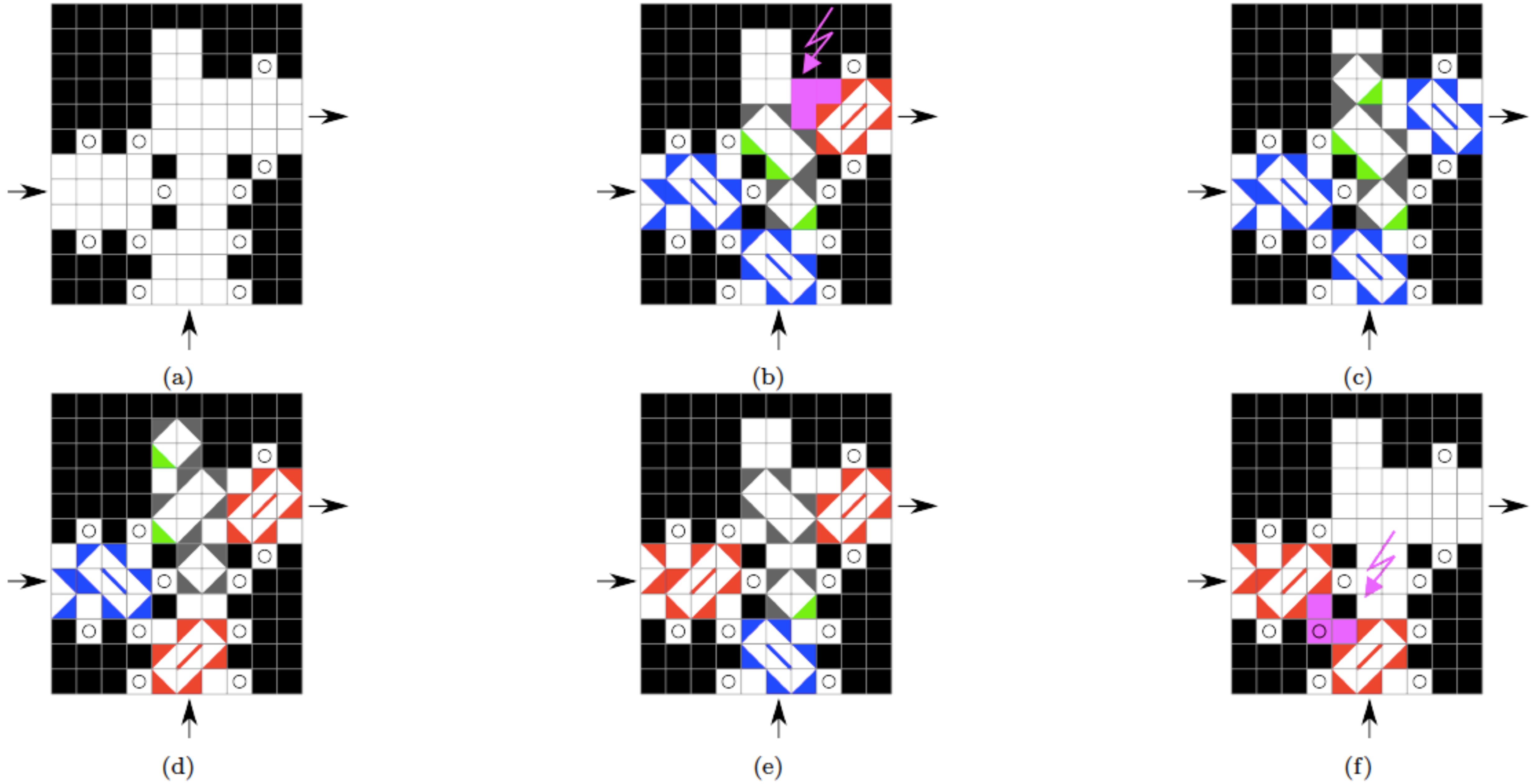


Figure 8: (a) The "XOR" gadget. (b)/(c) two false inputs cannot be completed for a true output (infeasible Shakashaka board indicated in purple), but may be completed for a false output. (d)/(e) both true/false false/true combinations allow a true output, (f) two inputs of true result in an infeasible Shakashaka board. Enforced triangles are shown in green.

Shakashaka

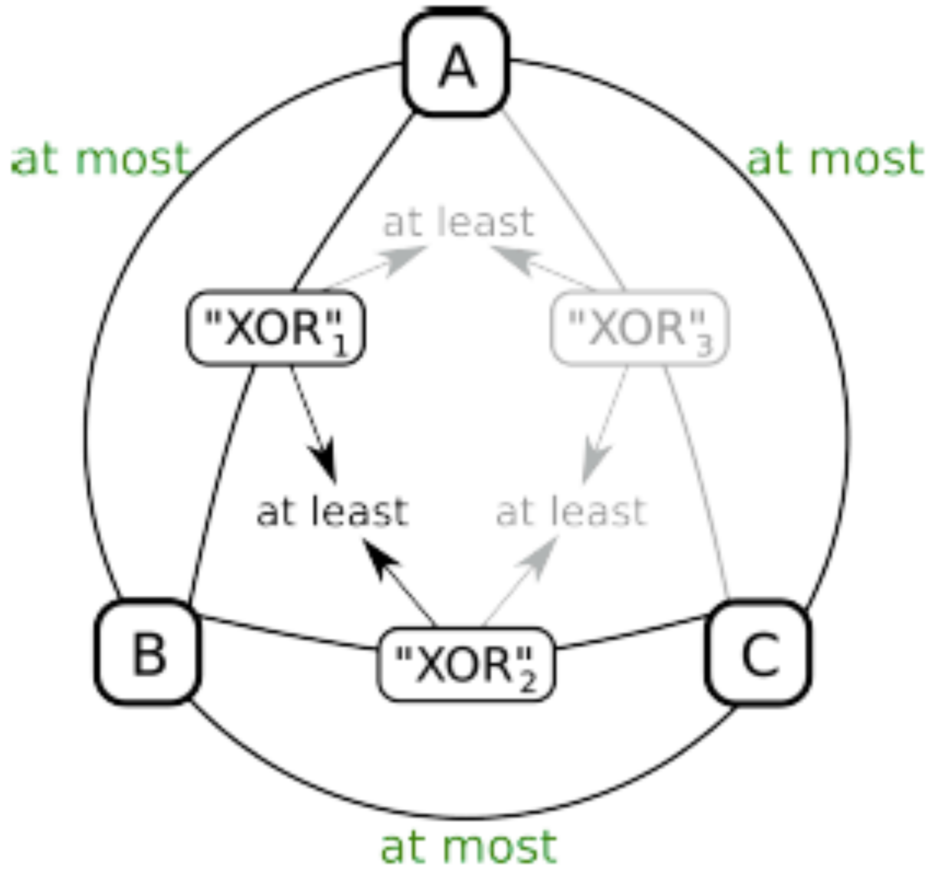


Figure 9: The clause gadget. The gray components ensure that the reduction is parsimonious.

Three variables, represented by A, B and C are pairwise combined by the at-most gadget.
This combination can only be solved if there is at most one true variable among A, B, and C (i.e. the possibilities are false/false/false, true/false/false, false/true/false, and false/false/true).
⟹we only need to exclude the false/false/false case.
We combine each of two pairs of variables with an XOR gadget (XOR1 and XOR2) and combine the results in the at-least gadget.
Note that XOR gadgets would yield an infeasible Shakashaka board for two true inputs, but this case has already been excluded.
If all variables are set to false, both XOR gadgets must output false.
The subsequent combination of the two XOR outputs with an at-least gadget results in an infeasible Shakashaka board.
If one variable is set to true, at least one XOR gadget can output true.
⟹subsequent combination of the two XOR outputs with an at-least gadget is possible and does not render the board infeasible.

5. Approximation Algorithms

Approximation Algorithms

Wie have an NP-hard/NP-complete problem:

What to do?
- Approximation
- Exact Solutions
- Heuristics

Definition 5.1: A polynomial time algorithm A is called a **c-approximation**, when for each problem instance I with optimal value OPT(I) we have:

$$R_A = \frac{A(I)}{OPT(I)} \leq c \qquad \text{for a minimisation problem } (c \geq 1)$$
$$R_A = \frac{A(I)}{OPT(I)} \geq c \qquad \text{for a maximisation problem } (c \leq 1)$$

c is called the **approximation factor.**

What do we need?
- A solution
- A lower bound on OPT, which we can determine in polynomial time

Approximation Algorithms

Example 5.2: Vertex Cover
What is a good bound?
—> max matching! Because we have max Matching ≤ min VC
(In a matching M no two edges can be covered by the same vertex, hence, we already need |M| vertices for M)

Idea here:
1. Compute a bound (with some object)
2. Compute a feasible solution for our problem, which is not much larger…
Here: one vertex per edge in matching, plus possibly additional vertices
Observation: For the additional vertices we can restrict to the non-chosen vertices of matching edges
→ Choose all vertices of matching edges

Approximation Algorithms

Algorithm 5.3 (Approximation for VC)

Input: Graph G=(V,E)

Output: approximation for VC

(1) Determine a maximal Matching: M

(2) Choose all vertices of edges in M: S_M

What is the approximation factor of algorithm 5.3?

2!

We have:

$|S_M| = 2*|M|$

$OPT \geq |M|$

$\Longrightarrow |S_M| = 2*|M| \leq 2\ OPT$

Approximation Algorithms

Can we do better? Find other solution approach?

1. Can the approximation guarantee of Algorithm 5.3 be improved by a better analysis?
2. Can an approximation algorithm with a better guarantee be designed using the lower bounding scheme of Algorithm 5.3, that is, the size of a maximal matching?
3. Is there some better bounding method that can lead to an improved approximation guarantee for VC?

ad 1. NO, this is tight: family of instances where algorithm is factor 2 away from optimum
Complete bipartite graphs

Short insertion on the TSP (for HW#3)

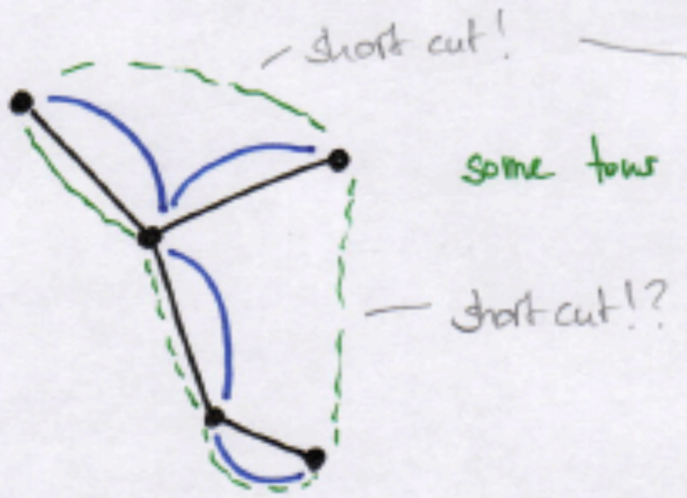
In general: given a graph – idea for building a tour? VI

Let |MST| denote the size of an optimal MST,
let |OPT| denote the size of the optimal tour.

$\Rightarrow \quad |MST| \leq |OPT|$

Can we use the MST to construct a tour?

Example:



only, if the graph describes something geometric (a map...) – we rely on the Δ-inequality!!

idea: double the MST

observation: a Eulertour exists (only vertices of even degree)

approx. factor: homework set