FAI LAB 7
Logical agents

Paolo Morettin

2024-25

Propositional logic

Ingredients:

- Values: **true** ($\top$), **false** ($\bot$)
- Atomic elements: **propositions**
- Logical connectives: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \oplus$

For the semantics and properties, check Prof. Sebastiani slides

Exercises: truth tables

Write down the full truth table of the following formulas over A, B, C:

- $(\neg A \wedge B)$
- $\neg((\neg A \wedge B) \vee C)$
- $(\neg A \vee B) \wedge (A \vee C) \wedge (\neg B \vee \neg C)$

Propositional reasoning

Propositional reasoning

- **Truth assignment**: $\mu : \mathit{Atoms}(\alpha) \to \{\top, \bot\}$

Propositional reasoning

- **Truth assignment**: $\mu : \mathit{Atoms}(\alpha) \to \{\top, \bot\}$
- **Model**: total TA that satisfies α ($\mu \models \alpha$)

Propositional reasoning

- **Truth assignment**: $\mu : \mathit{Atoms}(\alpha) \to \{\top, \bot\}$
- **Model**: total TA that satisfies α ($\mu \models \alpha$)
- **Set of models**: $M(\alpha) = \{\mu \mid \mu \models \alpha\}$

Propositional reasoning

- **Truth assignment**: $\mu : \mathit{Atoms}(\alpha) \to \{\top, \bot\}$
- **Model**: total TA that satisfies α ($\mu \models \alpha$)
- **Set of models**: $M(\alpha) = \{\mu \mid \mu \models \alpha\}$
- α is **satisfiable** ($\mathit{SAT}(\alpha)$) iff $M(\alpha) \neq \emptyset$

Propositional reasoning

- **Truth assignment**: $\mu : \mathit{Atoms}(\alpha) \to \{\top, \bot\}$
- **Model**: total TA that satisfies α ($\mu \models \alpha$)
- **Set of models**: $M(\alpha) = \{\mu \mid \mu \models \alpha\}$
- α is **satisfiable** ($\mathit{SAT}(\alpha)$) iff $M(\alpha) \neq \emptyset$
- α is **valid**($\models \alpha$) iff $M(\neg\alpha) = \emptyset$

Propositional reasoning

- **Truth assignment**: $\mu : \mathit{Atoms}(\alpha) \to \{\top, \bot\}$
- **Model**: total TA that satisfies α ($\mu \models \alpha$)
- **Set of models**: $M(\alpha) = \{\mu \mid \mu \models \alpha\}$
- α is **satisfiable** ($\mathit{SAT}(\alpha)$) iff $M(\alpha) \neq \emptyset$
- α is **valid**($\models \alpha$) iff $M(\neg\alpha) = \emptyset$
- α **entails** β ($\alpha \models \beta$) iff $M(\alpha) \subseteq M(\beta)$

Propositional reasoning

- **Truth assignment**: $\mu : \mathit{Atoms}(\alpha) \to \{\top, \bot\}$
- **Model**: total TA that satisfies α ($\mu \models \alpha$)
- **Set of models**: $M(\alpha) = \{\mu \mid \mu \models \alpha\}$
- α is **satisfiable** ($\mathit{SAT}(\alpha)$) iff $M(\alpha) \neq \emptyset$
- α is **valid**($\models \alpha$) iff $M(\neg\alpha) = \emptyset$
- α **entails** β ($\alpha \models \beta$) iff $M(\alpha) \subseteq M(\beta)$
- α and β are **equivalent** ($\alpha \equiv \beta$) iff $M(\alpha) = M(\beta)$

Propositional reasoning

- **Truth assignment**: $\mu : \mathit{Atoms}(\alpha) \to \{\top, \bot\}$
- **Model**: total TA that satisfies α ($\mu \models \alpha$)
- **Set of models**: $M(\alpha) = \{\mu \mid \mu \models \alpha\}$
- α is **satisfiable** ($\mathit{SAT}(\alpha)$) iff $M(\alpha) \neq \emptyset$
- α is **valid**($\models \alpha$) iff $M(\neg\alpha) = \emptyset$
- α **entails** β ($\alpha \models \beta$) iff $M(\alpha) \subseteq M(\beta)$
- α and β are **equivalent** ($\alpha \equiv \beta$) iff $M(\alpha) = M(\beta)$
- α and β are **equi-satisfiable** iff $M(\alpha) \neq \emptyset \Leftrightarrow M(\beta) \neq \emptyset$

Exercises: model counting

Consider propositions A, B, C, how many models α has?

α	$M(\alpha)$
$\bot$	
$\top$	
$A \wedge B \wedge \neg C$	
$A \vee B \vee \neg C$	
$\neg A$	

Exercises: model counting

Consider propositions A, B, C, how many models α has?

α	$M(\alpha)$
$\bot$	**0**
$\top$	
$A \wedge B \wedge \neg C$	
$A \vee B \vee \neg C$	
$\neg A$	

Exercises: model counting

Consider propositions A, B, C, how many models α has?

α	$M(\alpha)$
$\bot$	**0**
$\top$	**8**
$A \wedge B \wedge \neg C$	
$A \vee B \vee \neg C$	
$\neg A$	

Exercises: model counting

Consider propositions A, B, C, how many models α has?

α	$M(\alpha)$
$\bot$	**0**
$\top$	**8**
$A \wedge B \wedge \neg C$	**1**
$A \vee B \vee \neg C$	
$\neg A$	

Exercises: model counting

Consider propositions A, B, C, how many models α has?

α	$M(\alpha)$
$\bot$	**0**
$\top$	**8**
$A \wedge B \wedge \neg C$	**1**
$A \vee B \vee \neg C$	**7**
$\neg A$	

Exercises: model counting

Consider propositions A, B, C, how many models α has?

α	$M(\alpha)$
$\bot$	**0**
$\top$	**8**
$A \wedge B \wedge \neg C$	**1**
$A \vee B \vee \neg C$	**7**
$\neg A$	**4**

Exercises: entailment

Consider propositions A, B, does α entail β?

α	β	$\alpha \models \beta$
$\top$	$\bot$	
$\bot$	$\top$	
$(A \wedge B)$	$(A \vee B)$	
$(A \vee B)$	$(A \oplus B)$	
$\delta \wedge \gamma$	δ	
$A \vee B$	A	
$A \vee \delta$	A	
$A \vee \delta$	δ	

Exercises: entailment

Consider propositions A, B, does α entail β?

α	β	$\alpha \models \beta$
$\top$	$\bot$	**No**
$\bot$	$\top$	
$(A \wedge B)$	$(A \vee B)$	
$(A \vee B)$	$(A \oplus B)$	
$\delta \wedge \gamma$	δ	
$A \vee B$	A	
$A \vee \delta$	A	
$A \vee \delta$	δ	

Exercises: entailment

Consider propositions A, B, does α entail β?

α	β	$\alpha \models \beta$
$\top$	$\bot$	**No**
$\bot$	$\top$	**Yes**
$(A \wedge B)$	$(A \vee B)$	
$(A \vee B)$	$(A \oplus B)$	
$\delta \wedge \gamma$	δ	
$A \vee B$	A	
$A \vee \delta$	A	
$A \vee \delta$	δ	

Exercises: entailment

Consider propositions A, B, does α entail β?

α	β	$\alpha \models \beta$
$\top$	$\bot$	**No**
$\bot$	$\top$	**Yes**
$(A \wedge B)$	$(A \vee B)$	**Yes**
$(A \vee B)$	$(A \oplus B)$	
$\delta \wedge \gamma$	δ	
$A \vee B$	A	
$A \vee \delta$	A	
$A \vee \delta$	δ	

Exercises: entailment

Consider propositions A, B, does α entail β?

α	β	$\alpha \models \beta$
$\top$	$\bot$	**No**
$\bot$	$\top$	**Yes**
$(A \wedge B)$	$(A \vee B)$	**Yes**
$(A \vee B)$	$(A \oplus B)$	**No**
$\delta \wedge \gamma$	δ	
$A \vee B$	A	
$A \vee \delta$	A	
$A \vee \delta$	δ	

Exercises: entailment

Consider propositions A, B, does α entail β?

α	β	$\alpha \models \beta$
$\top$	$\bot$	**No**
$\bot$	$\top$	**Yes**
$(A \wedge B)$	$(A \vee B)$	**Yes**
$(A \vee B)$	$(A \oplus B)$	**No**
$\delta \wedge \gamma$	δ	**Yes**
$A \vee B$	A	
$A \vee \delta$	A	
$A \vee \delta$	δ	

Exercises: entailment

Consider propositions A, B, does α entail β?

α	β	$\alpha \models \beta$
$\top$	$\bot$	**No**
$\bot$	$\top$	**Yes**
$(A \wedge B)$	$(A \vee B)$	**Yes**
$(A \vee B)$	$(A \oplus B)$	**No**
$\delta \wedge \gamma$	δ	**Yes**
$A \vee B$	A	**No**
$A \vee \delta$	A	
$A \vee \delta$	δ	

Exercises: entailment

Consider propositions A, B, does α entail β?

α	β	$\alpha \models \beta$
$\top$	$\bot$	**No**
$\bot$	$\top$	**Yes**
$(A \wedge B)$	$(A \vee B)$	**Yes**
$(A \vee B)$	$(A \oplus B)$	**No**
$\delta \wedge \gamma$	δ	**Yes**
$A \vee B$	A	**No**
$A \vee \delta$	A	**Maybe**
$A \vee \delta$	δ	

Exercises: entailment

Consider propositions A, B, does α entail β?

α	β	$\alpha \models \beta$
$\top$	$\bot$	**No**
$\bot$	$\top$	**Yes**
$(A \wedge B)$	$(A \vee B)$	**Yes**
$(A \vee B)$	$(A \oplus B)$	**No**
$\delta \wedge \gamma$	δ	**Yes**
$A \vee B$	A	**No**
$A \vee \delta$	A	**Maybe**
$A \vee \delta$	δ	**Maybe**

Conjunctive Normal Form (CNF)

A formula α is in CNF:

- α is a conjunction of **clauses** $\alpha = \bigwedge_{n=1}^{N} C_n$
- each clause is a disjunction of **literals** $C_n = \bigvee_{k=1}^{K_n} \ell_k$
- each literal is a **positive or negative proposition** $\ell_k = [\neg]A$

Resolution rule

Deduction of a *new clause*, given two clauses with:

- common sub-clause α
- **exactly one** incompatible atom A (resolvent)
- two disjoint subclauses β_1, β_2

$$\frac{(\alpha \vee A \vee \beta_1) \qquad (\alpha \vee \neg A \vee \beta_2)}{(\alpha \vee \beta_1 \vee \beta_2)}$$

Resolution

Is α unsatisfiable?:

Resolution

Is α unsatisfiable?:

- We keep applying the resolution rule

Resolution

Is α unsatisfiable?:

- We keep applying the resolution rule
- If we deduce the false clause (empty) then α is UNSAT (**correct**)

Resolution

Is α unsatisfiable?:

- We keep applying the resolution rule
- If we deduce the false clause (empty) then α is UNSAT (**correct**)
- If α is UNSAT resolution will deduce the false clause (**complete**)

Resolution

Is α unsatisfiable?:

- We keep applying the resolution rule
- If we deduce the false clause (empty) then α is UNSAT (**correct**)
- If α is UNSAT resolution will deduce the false clause (**complete**)
- **Both time and memory -inefficient!**

Exercise: resolution

1. $P \vee \neg Q$
2. $\neg P \vee R \vee S$
3. Q
4. $\neg P \vee \neg R$
5. $P \vee Q \vee R$
6. $\neg Q \vee S$
7. R

Exercise: resolution

1. $P \vee \neg Q$
2. $\neg P \vee R \vee S$
3. Q
4. $\neg P \vee \neg R$
5. $P \vee Q \vee R$
6. $\neg Q \vee S$
7. R
8. P (1+3)

Exercise: resolution

1. $P \vee \neg Q$
2. $\neg P \vee R \vee S$
3. Q
4. $\neg P \vee \neg R$
5. $P \vee Q \vee R$
6. $\neg Q \vee S$
7. R

8. P (1+3)
9. $\neg R$ (4+8)

Exercise: resolution

1. $P \vee \neg Q$
2. $\neg P \vee R \vee S$
3. Q
4. $\neg P \vee \neg R$
5. $P \vee Q \vee R$
6. $\neg Q \vee S$
7. R

8. P (1+3)
9. $\neg R$ (4+8)
10. $\bot$ (7+9)

Davis-Putnam-Longemann-Loveland (DPLL)

Is α satisfiable?:

Davis-Putnam-Longemann-Loveland (DPLL)

Is α satisfiable?:

- Recursively generate a model μ

Davis-Putnam-Longemann-Loveland (DPLL)

Is α satisfiable?:

- Recursively generate a model μ
- Before branching, try *deterministic* steps (**unit propagation**):

$$\frac{\Gamma, (\ell), (\neg\ell \vee \alpha)}{\Gamma, (\ell), (\alpha)}\ \text{(unit resolution)} \quad + \quad \frac{\Gamma, (\ell), (\ell \vee \alpha)}{\Gamma, (\ell)}\ \text{(unit subsumption)}$$

Davis-Putnam-Longemann-Loveland (DPLL)

Is α satisfiable?:

- Recursively generate a model μ
- Before branching, try *deterministic* steps (**unit propagation**):

$$\frac{\Gamma, (\ell), (\neg\ell \vee \alpha)}{\Gamma, (\ell), (\alpha)}\ \text{(unit resolution)} \quad + \quad \frac{\Gamma, (\ell), (\ell \vee \alpha)}{\Gamma, (\ell)}\ \text{(unit subsumption)}$$

function DPLL-SATISFIABLE?(s) **returns** *true* or *false*
inputs: s, a sentence in propositional logic

clauses $\leftarrow$ the set of clauses in the CNF representation of s
symbols $\leftarrow$ a list of the proposition symbols in s
return DPLL(*clauses*, *symbols*, { })

---

function DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

if every clause in *clauses* is true in *model* **then return** *true*
if some clause in *clauses* is false in *model* **then return** *false*
P, *value* $\leftarrow$ FIND-UNIT-CLAUSE(*clauses*, *model*)
if P is non-null **then return** DPLL(*clauses*, *symbols* – P, *model* $\cup$ {P=*value*})
$P \leftarrow$ FIRST(*symbols*); *rest* $\leftarrow$ REST(*symbols*)
return DPLL(*clauses*, *rest*, *model* $\cup$ {P=*true*}) **or**
DPLL(*clauses*, *rest*, *model* $\cup$ {P=*false*}))

Exercise: DPLL

1. $P \vee \neg Q$
2. $\neg P \vee R \vee S$
3. Q
4. $\neg P \vee \neg R$
5. $P \vee Q \vee R$
6. $\neg Q \vee S$
7. R

Exercise: DPLL

1. $P \vee \neg Q$
2. $\neg P \vee R \vee S$
3. Q
4. $\neg P \vee \neg R$
5. $P \vee Q \vee R$
6. $\neg Q \vee S$
7. R

Q

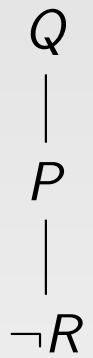
Exercise: DPLL

1. $P \vee \neg Q$
2. $\neg P \vee R \vee S$
3. Q
4. $\neg P \vee \neg R$
5. $P \vee Q \vee R$
6. $\neg Q \vee S$
7. R

$$\begin{array}{c} Q \\ | \\ P \end{array}$$

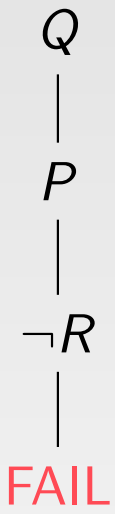
Exercise: DPLL

1. $P \vee \neg Q$
2. $\neg P \vee R \vee S$
3. Q
4. $\neg P \vee \neg R$
5. $P \vee Q \vee R$
6. $\neg Q \vee S$
7. R



Exercise: DPLL

1. $P \vee \neg Q$
2. $\neg P \vee R \vee S$
3. Q
4. $\neg P \vee \neg R$
5. $P \vee Q \vee R$
6. $\neg Q \vee S$
7. R

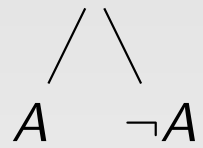


Exercise: DPLL

1. $\neg A \vee B$
2. $\neg B \vee \neg C$
3. $\neg A \vee C$
4. $A \vee D$

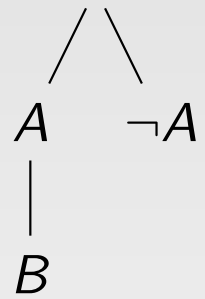
Exercise: DPLL

1. $\neg A \vee B$
2. $\neg B \vee \neg C$
3. $\neg A \vee C$
4. $A \vee D$



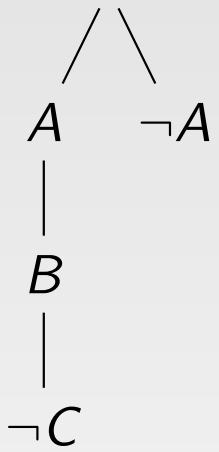
Exercise: DPLL

1. $\neg A \vee B$
2. $\neg B \vee \neg C$
3. $\neg A \vee C$
4. $A \vee D$



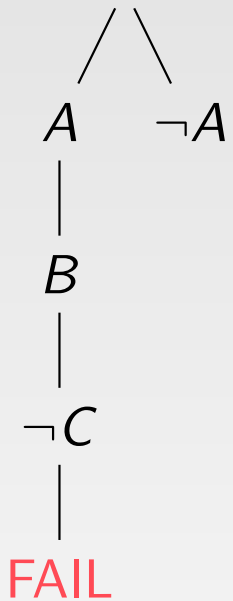
Exercise: DPLL

1. $\neg A \vee B$
2. $\neg B \vee \neg C$
3. $\neg A \vee C$
4. $A \vee D$



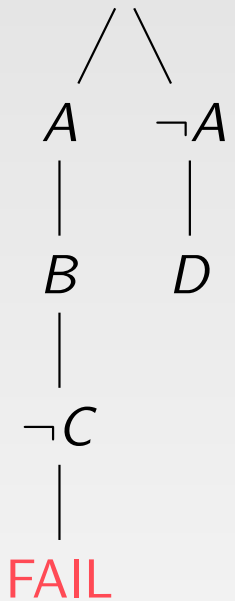
Exercise: DPLL

1. $\neg A \vee B$
2. $\neg B \vee \neg C$
3. $\neg A \vee C$
4. $A \vee D$



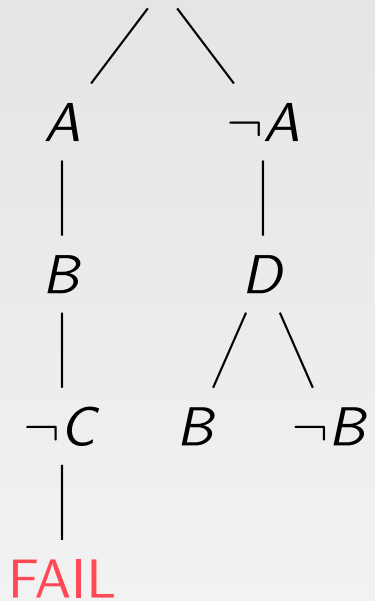
Exercise: DPLL

1. $\neg A \vee B$
2. $\neg B \vee \neg C$
3. $\neg A \vee C$
4. $A \vee D$



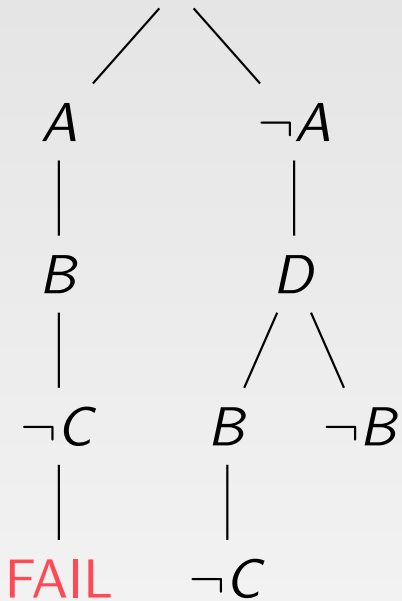
Exercise: DPLL

1. $\neg A \vee B$
2. $\neg B \vee \neg C$
3. $\neg A \vee C$
4. $A \vee D$



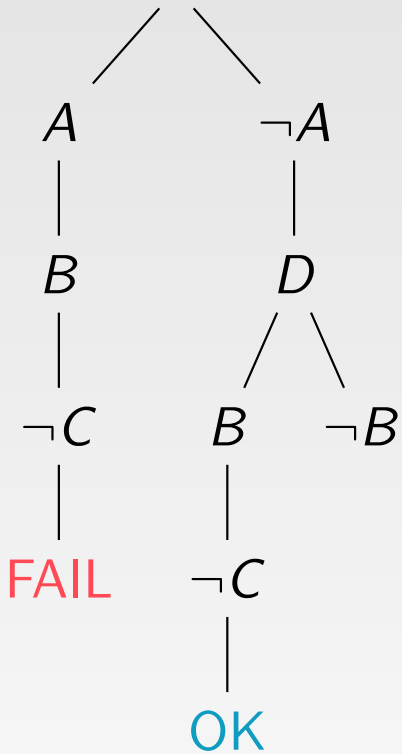
Exercise: DPLL

1. $\neg A \vee B$
2. $\neg B \vee \neg C$
3. $\neg A \vee C$
4. $A \vee D$



Exercise: DPLL

1. $\neg A \vee B$
2. $\neg B \vee \neg C$
3. $\neg A \vee C$
4. $A \vee D$



WalkSAT

function WALKSAT($clauses, p, max\_flips$) **returns** a satisfying model or *failure*
  inputs: $clauses$, a set of clauses in propositional logic
    p, the probability of choosing to do a "random walk" move, typically around 0.5
    max_flips, number of flips allowed before giving up

  $model \leftarrow$ a random assignment of *true*/*false* to the symbols in $clauses$
  for $i = 1$ **to** max_flips **do**
    if $model$ satisfies $clauses$ **then return** $model$
    $clause \leftarrow$ a randomly selected clause from $clauses$ that is false in $model$
    with probability p flip the value in $model$ of a randomly selected symbol from $clause$
    else flip whichever symbol in $clause$ maximizes the number of satisfied clauses
  return *failure*

WalkSAT

function WALKSAT(*clauses*, *p*, *max_flips*) **returns** a satisfying model or *failure*
  inputs: *clauses*, a set of clauses in propositional logic
    p, the probability of choosing to do a "random walk" move, typically around 0.5
    max_flips, number of flips allowed before giving up

  model ← a random assignment of *true*/*false* to the symbols in *clauses*
  for $i = 1$ **to** *max_flips* **do**
    if *model* satisfies *clauses* **then return** *model*
    clause ← a randomly selected clause from *clauses* that is false in *model*
    with probability *p* flip the value in *model* of a randomly selected symbol from *clause*
    else flip whichever symbol in *clause* maximizes the number of satisfied clauses
  return *failure*

- **Stochastic local search** algorithm for SAT

WalkSAT

function WALKSAT(*clauses*, *p*, *max_flips*) **returns** a satisfying model or *failure*
  inputs: *clauses*, a set of clauses in propositional logic
    p, the probability of choosing to do a "random walk" move, typically around 0.5
    max_flips, number of flips allowed before giving up

  model ← a random assignment of *true*/*false* to the symbols in *clauses*
  for $i = 1$ **to** *max_flips* **do**
    if *model* satisfies *clauses* **then return** *model*
    clause ← a randomly selected clause from *clauses* that is false in *model*
    with probability *p* flip the value in *model* of a randomly selected symbol from *clause*
    else flip whichever symbol in *clause* maximizes the number of satisfied clauses
  return *failure*

- **Stochastic local search** algorithm for SAT
- **Only detects satisfiability!**

WalkSAT

function WALKSAT(*clauses*, *p*, *max_flips*) **returns** a satisfying model or *failure*
  inputs: *clauses*, a set of clauses in propositional logic
    p, the probability of choosing to do a "random walk" move, typically around 0.5
    max_flips, number of flips allowed before giving up

  model ← a random assignment of *true*/*false* to the symbols in *clauses*
  for $i = 1$ **to** *max_flips* **do**
    if *model* satisfies *clauses* **then return** *model*
    clause ← a randomly selected clause from *clauses* that is false in *model*
    with probability *p* flip the value in *model* of a randomly selected symbol from *clause*
    else flip whichever symbol in *clause* maximizes the number of satisfied clauses
  return *failure*

- **Stochastic local search** algorithm for SAT
- **Only detects satisfiability!**
- Multiple variants, we consider:
  - Clause selection: uniform sample among UNSAT clauses
  - Variable selection: flip at random with p, minimize UNSAT clauses with $1 - p$