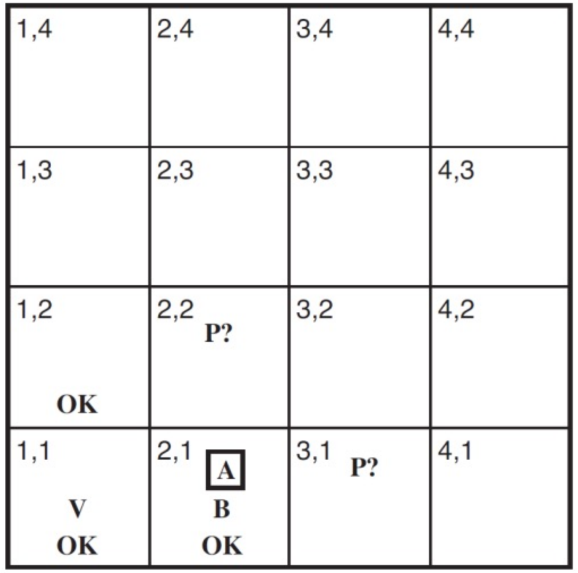
Propositional Logic Review

YAO ZHAO

Wumpus world KB



- Let's build the KB for the reduced Wumpus world.
- Let $P_{i,j}$ be true if there is a pit in $[i, j]$
- Let $B_{i,j}$ be true if there is a breeze in $[i, j]$

  R_1: $\neg P_{1,1}$

- "A square is breezy if and only if there is an adjacent pit"

  R_2: $B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$

  R_3: $B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$

  R_4: $\neg B_{1,1}$

  R_5: $B_{2,1}$

- Questions: Based on this KB, is $KB \models P_{1,2}$? Is $KB \models P_{2,2}$?

Key Points

- Transformation to Conjunctive Normal Form (CNF)
- SAT Problem → PDLL Algorithm

Transformation in to CNF: review

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

The focus of logical expression

- The transformation to conjunctive normal form (CNF)
- The great significance of the conjunction paradigm is that it makes it possible to define a search space for a logical problem. Naturally, the logical problem can be transformed into a problem solved by backtracking.

SAT Problem

SAT:(Boolean) Satisfiability Problem

For example: Is it possible the following expression is true? Which model can make it being true?

```
~P11 & (B11 <=> (P12 | P21)) & (B21 <=>(P11 | P22 | P31)) &  ~B11 & B21
```

A model is an assignment of truth-value to variables making the above expression true.
A possible assignment is
[P11: False, B11: False, P12: False, P21: False, B21: True, P31: True]

Backtracking can be used to find the above assignment of truth-value to variable.

联想CSP中变量
以及对变量赋值的过程

DPLL Review

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

  clauses ← the set of clauses in the CNF representation of s
  symbols ← 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* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols* – P, *model* ∪ {P=*value*})
  P, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols* – P, *model* ∪ {P=*value*})
  P ← FIRST(*symbols*); *rest* ← REST(*symbols*)
  return DPLL(*clauses*, *rest*, *model* ∪ {P=*true*}) **or**
      DPLL(*clauses*, *rest*, *model* ∪ {P=*false*}))

Backtracking in DPLL

- If there is a clause returning false, then return false as a whole.
- If all clauses are determined to be true in the model, return true as a whole.

Forward checking in CSP BTS

- Look for pure symbols first: For example, if there is only A in all clauses, you can directly assign A to true; if there is only ¬B in all clauses, you can directly assign B to false. We can ignore all clauses determined to be true with the built model.
- Find the unit clause in priority: clause with only one literal
  - a clause can also be considered as a unit clause if all other literals are assigned a value except one literal .

  E.g, A is a unit clause, and ¬A is also a unit clause.

  A has been assigned a value of false, A | ¬B is also a unit clause since B must be false here. When B is false, A | B | C is also a unit clause, because C must be true.

  The process is a bit like constraint propagation, called unit propagation.

Minimum Remaining Values heuristic

- If none of the above symbols were found, look for the next symbol. Try all possible assignments of the symbol and recursively call the next layer.

Normal backtracking

DPLL Example

Transform into CNF and extract all conjunction sub-clauses

`~P11 & (B11 <=> (P12 | P21)) & (B21 <=>(P11 | P22 | P31)) & ~B11 & B21`

~P11

~P12 | B11

~P21 | B11

~P22 | B21

~P31 | B21

~P11 | B21

~B11

B21

clauses

P12 | P21 | ~B11

P11 | P22 | P31 | ~B21

symbols:

P31, P22, B21, P12, P21, P11, B11

Extract each clause of the conjunction paradigm.
The entire expression is true if each clause is true.

DPLL Example(initial)

Put priority in selecting pure literal and unit clauses

~P11	~P12 | B11	~P21 | B11	~P22 | B21	~P31 | B21	~P11 | B21	~B11	B21

unknown clauses

P12 | P21 | ~B11

P11 | P22 | P31 | ~B21

symbols:

P31, P22, B21, P12, P21, P11, B11

model:

{}

DPLL Example

Put priority in selecting pure literal and unit clauses

Pure symbols?

Unit Clause

~P11

~P12 | B11

~P21 | B11

~P22 | B21

~P31 | B21

~~~P11 | B21~~

true

~B11

B21

unknown clauses

P12 | P21 | ~B11

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21

symbols:

P31, P22, B21, P12, P21, ~~P11,~~ B11

model:

{P11:False}
~~~

DPLL Example

Unit Clause

~P11

~P12 | B11

~P21 | B11

~P22 | B21

~P31 | B21

~B11

B21

~P11 | B21

~P12

~P21

unknown clauses

P12 | P21 | ~B11

True

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21

symbols:

P31, P22, B21, P12, P21, ~~P11, B11~~

model:

{P11:False, B11:False}

DPLL Example

~P11

~P11 | B21

~B11

P12 | P21 | ~B11

~P12 | B11

~P12

~P21 | B11

~P21

~P22 | B21

True

~P31 | B21

True

Unit Clause

B21

unknown clauses

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21

P22 | P31

symbols:

P31, P22, ~~B21~~, P12, P21, ~~P11, B11~~

model:

{P11:False, B11:False, B21:True}

DPLL Example

~P11

~P11 | B21

~B11

P12 | P21 | ~B11

~P22 | B21

~P31 | B21

B21

Unit Clause

unknown clauses

~P12 | B11

~P12

~P21 | B11

~P21

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21

P22 | P31

symbols:

P31, P22, ~~B21~~, ~~P12~~, P21, ~~P11, B11~~

model:

{P11:False, B11:False, B21:True， P12:False}

DPLL Example

~P12 | B11

~P11

~P11 | B21

~B11

P12 | P21 | ~B11

~P22 | B21

~P31 | B21

B21

Unit Clause

~P21 | B11

~P21

unknown clauses

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21

P22 | P31

symbols:

P31, P22, ~~B21~~, ~~P12~~, ~~P21~~, ~~P11, B11~~

model:

{P11:False, B11:False, B21:True,P12:False,P21:False}

DPLL Example

No pure literals and no unit clauses. Select a literal and assign it to be true

~P11

~P11 | B21

~P12 | B11

~B11

~P21 | B11

P12 | P21 | ~B11

~P22 | B21

~P31 | B21

B21

No Unit Clause, choose the first symbol in symbols.

unknown clauses

P11 | P22 | P31| ~B21

P22 | P31| ~B21 → P22 | P31

True

symbols:

Choose and set true

P31, ~~P22~~, ~~B21~~, ~~P12~~, ~~P21~~, ~~P11, B11~~

model:

{P11:False, B11:False, B21:True,P12:False,P21:False,P22:True}

DPLL Example

~P11

~P11 | B21

~P12 | B11

~B11

~P21 | B11

P12 | P21 | ~B11

~P22 | B21

~P31 | B21

B21

P11 | P22 | P31| ~B21

No clauses and return model

unknown clauses

symbols:

P31, ~~P22~~, ~~B21~~, ~~P12~~, ~~P21~~, ~~P11, B11~~

A solution

model:

{P11:False, B11:False, B21:True,P12:False,P21:False,P22:True}

Summary

- The above example is relatively simple, but you can understand two points:
  - Early termination of backtracking, such as the last remaining symbol P31, but its value has not affected the result.
  - Preemptive selection of pure symbols and unit clauses, for the role of pruning , the process of the unit's communication.
- For large problems, there are more directions for optimization.