

Automatic Planning

Jörg Hoffmann

University of Innsbruck
STI

Masterstudium Informatik Vertiefungsmodul (WM 9)

Part II. Constraint-Based Search

Automatic Planning

9. SAT and DPLL

- ▶ **Propositional Formulas**
- ▶ CNFs
- ▶ Unit Propagation
- ▶ DPLL

Definition

Let P be a set of propositions. A propositional formula ϕ is either:

1. a proposition $p \in P$, or
2. the **negation** $\neg\phi'$ of a propositional formula ϕ'
3. the **conjunction** $\phi_1 \wedge \phi_2$ of propositional formulas ϕ_1 and ϕ_2 , or
4. the **disjunction** $\phi_1 \vee \phi_2$ of propositional formulas ϕ_1 and ϕ_2 .

- ▶ A proposition is either 1 or 0 in an “interpretation” (== value assignment == state)
- ▶ Formulas of the form p and $\neg p$ are called **literals**; we denote $\overline{p} = \neg p$ and $\overline{\overline{p}} = p$

Definition

Let P be a set of propositions, ϕ a propositional formula, i an interpretation of P . ϕ **holds in i** , written $i \models \phi$, iff either:

1. $\phi = p \in P$ and $i(p) = 1$, or
2. $\phi = \neg\phi'$ and ϕ' does not hold in i , or
3. $\phi = \phi_1 \wedge \phi_2$ and both ϕ_1 and ϕ_2 hold in i , or
4. $\phi = \phi_1 \vee \phi_2$ and at least one of ϕ_1 and ϕ_2 holds in i .

Propositional Formulas, Examples

$P = \{\text{doof}, \text{mittel}, \text{schlau}\}$

$i = \{\text{doof} \mapsto 0, \text{mittel} \mapsto 1, \text{schlau} \mapsto 0\}$

1. $\text{doof} \wedge \neg \text{mittel} \wedge \neg \text{schlau}$
2. $\text{schlau} \vee \neg \text{schlau}$
3. $(\text{doof} \vee \text{mittel} \vee \text{schlau}) \wedge$
 $(\neg \text{doof} \vee \neg \text{mittel}) \wedge$
 $(\neg \text{mittel} \vee \neg \text{schlau}) \wedge$
 $(\neg \text{doof} \vee \neg \text{schlau})$
4. $(\neg \text{doof} \vee \text{mittel}) \wedge (\neg \text{mittel} \vee \text{schlau}) \wedge (\neg \text{schlau} \vee \text{doof}) \wedge (\text{doof} \vee$
 $\text{schlau}) \wedge (\neg \text{mittel} \vee \neg \text{schlau})$

Automatic Planning

9. SAT and DPLL

- ▶ Propositional Formulas
- ▶ CNFs
- ▶ Unit Propagation
- ▶ DPLL

Which of these are CNFs?

$P = \{\text{doof}, \text{mittel}, \text{schlau}\}$

1. $\text{doof} \wedge \neg \text{mittel} \wedge \neg \text{schlau}$
2. $\text{schlau} \vee \neg \text{schlau}$
3. $(\text{doof} \vee \text{mittel} \vee \text{schlau}) \wedge$
 $(\neg \text{doof} \vee \neg \text{mittel}) \wedge$
 $(\neg \text{mittel} \vee \neg \text{schlau}) \wedge$
 $(\neg \text{doof} \vee \neg \text{schlau})$
4. $(\neg \text{doof} \vee \text{mittel}) \wedge (\neg \text{mittel} \vee \text{schlau}) \wedge (\neg \text{schlau} \vee \text{doof}) \wedge (\text{doof} \vee$
 $\text{schlau}) \wedge (\neg \text{mittel} \vee \neg \text{schlau})$

Definition

Let P be a set of propositions. A propositional formula Φ is called a **CNF** if it has the form $\Phi = \phi_1 \wedge \dots \wedge \phi_n$ where each ϕ_i has the form $\phi_i = l_1 \vee \dots \vee l_k$ where each $l_i \in P \cup \{\neg p \mid p \in P\}$.

- ▶ A conjunction of disjunctions of literals
- ▶ == A set of constraints
- ▶ Which are called **clauses**, each clause being a set of literals
- ▶ $\{\{\text{doof, mittel, schlau}\},$
 $\{\neg\text{doof, } \neg\text{mittel}\},$
 $\{\neg\text{mittel, } \neg\text{schlau}\},$
 $\{\neg\text{doof, } \neg\text{schlau}\}\}$

Automatic Planning

9. SAT and DPLL

- ▶ Propositional Formulas
- ▶ CNFs
- ▶ **Unit Propagation**
- ▶ DPLL

Constraint Propagation

- ▶ Given a set Φ of constraints ϕ over variables (e.g., clauses over propositional variables), **infer** new constraints
- ▶ Inference: some reasoning R that is **sound** – if R infers ϕ from Φ , then ϕ follows from Φ
- ▶ $\Phi \cup \{\phi\}$ is logically equivalent to $\Phi \dots$
- ▶ \dots but: e.g. there may be constraints ψ that R can infer in one step from $\Phi \cup \{\phi\}$, but not from Φ
- ▶ Typically, one computes a fixpoint: **propagation**

You all know the oldest and most wide-spread method of this kind.
What is its name?

Given clauses $\phi = A \cup \{p\}$ and $\psi = B \cup \{\neg p\}$, we allow the inference:

$$\frac{A \cup \{p\} \quad B \cup \{\neg p\}}{???$$

That is, ??? can be added as a new clause.

- ▶ Since p cannot be 0 *and* 1, we have to make at least one of ??? true
- ▶ Resolution is **complete**: Φ is unsatisfiable iff $\{\} \in R^+(\Phi)$

- ▶ $\{\{A, B\}, \{\neg A, \neg B\}\}$
- ▶ $\{\{\neg A, B\}, \{\neg B, C\}, \{\neg C, A\}, \{A, C\}, \{\neg B, \neg C\}\}$

Blackboard.

Unit propagation is a restricted form of resolution. *Do you know what the restriction is?*

Unit Propagation, I

- ▶ A full (complete) constraint propagation is exponentially costly: it solves the decision problem
- ▶ Assume we only want *some information/simplification*
- ▶ More restricted reasoning! One option: in

$$\frac{A \cup \{p\} \quad B \cup \{\neg p\}}{A \cup B}$$

require that either $|A \cup \{p\}| \leq k$ or $|B \cup \{\neg p\}| \leq k$

- ▶ ***k-Resolution***, parameterized trade-off between inference power and runtime cost
- ▶ Unit propagation == ???-Resolution, most wide-spread technique in implemented systems

Unit Propagation, II

Fixpoint of application of

$$\frac{A \cup \{\bar{l}\} \quad \{l\}}{A}$$

```
while TRUE do  
   $\Phi' := \Phi$   
  for all  $\phi \in \Phi, \phi = \{l\}$  do  
    for all  $\psi \in \Phi, \bar{l} \in \psi$  do  
       $\Phi' := \Phi' \cup \{\psi \setminus \{\bar{l}\}\}$   
    endfor  
  endfor  
  if  $\Phi = \Phi'$  then stop endif  
   $\Phi := \Phi'$   
endwhile
```

Unit Propagation, III

```
while TRUE do  
   $\Phi' := \Phi$   
  for all  $\phi \in \Phi, \phi = \{l\}$  do  
    for all  $\psi \in \Phi, \bar{l} \in \psi$  do  
       $\Phi' := \Phi' \cup (\psi \setminus \{\bar{l}\}), \Phi' := \Phi' \setminus \psi$   
    endfor  
    for all  $\phi \neq \psi \in \Phi', l \in \psi$  do  
       $\Phi' := \Phi' \setminus \psi$   
    endfor  
  endfor  
  if  $\Phi = \Phi'$  then stop endif  
   $\Phi := \Phi'$   
endwhile
```

What is the justification for the clause removal operations?

Examples

- ▶ $\{\{\neg A, \neg B, \neg C, D\}, \{\neg A, B\}, \{A\}, \{\neg A, \neg B, \neg C, \neg D\}, \{\neg A, \neg B, C\}\}$
- ▶ $\{\{\neg A, B\}, \{\neg B, C\}, \{\neg C, A\}, \{A, C\}, \{\neg B, \neg C\}\}$

Blackboard.

Automatic Planning

9. SAT and DPLL

- ▶ Propositional Formulas
- ▶ CNFs
- ▶ Unit Propagation
- ▶ **DPLL**

Constraint-Based Search

Backtracking over variable values:

```
Bool solve( $\phi$ , partial assignment  $a$ )  
  ( $\phi'$ ,  $a'$ ) := constraint-propagation( $\phi$ ,  $a$ )  
  if  $\phi'$  self-contradictory then return FALSE endif  
  select a variable  $v$  not assigned by  $a'$   
  if no such  $v$  exists then return TRUE endif  
  for all  $c \in \text{dom}(v)$  do  
    if solve( $\phi'$ ,  $a' \cup \{v \mapsto c\}$ ) then return TRUE endif  
  endfor  
  return FALSE
```

The Davis-Putnam-Logeman-Loveland Algorithm:

```

Bool DPLL( $\Phi$ , partial assignment  $a$ )
  ( $\phi'$ ,  $a'$ ) := unit-propagation( $\phi$ ,  $a$ )
  if  $\phi'$  contains empty clause then return FALSE endif
  select a variable  $v$  not assigned by  $a'$ 
  if no such  $v$  exists then return TRUE endif
  if DPLL( $\phi'$ ,  $a' \cup \{v \mapsto 1\}$ ) then return TRUE endif
  if DPLL( $\phi'$ ,  $a' \cup \{v \mapsto 0\}$ ) then return TRUE endif
  return FALSE
  
```

Here, unit propagation extends a with the assignments enforced by unit clauses.

- ▶ $\{\{A, B, C\}, \{\neg A, \neg B\}, \{\neg A, \neg C\}, \{\neg B, \neg C\}\}$
- ▶ $\{\{\neg A, B\}, \{\neg B, C\}, \{\neg C, A\}, \{A, C\}, \{\neg B, \neg C\}\}$

Blackboard.