

CS 161 Intro. To Artificial Intelligence

Week 5, Discussion 1C

Li Cheng Lan



Today's Topics

- Propositional Logic

- Important Terms
- Syntactic Forms
- Semantics
- Entailment
- Inference

- Proof by enumeration

- Proof by resolution

- SAT solver

- Tractable NNF circuits



- Midterm Review



Important Terms

Model:

- Definition:
 - Formally structured worlds with respect to which truth can be evaluated
 - We say m is a model of a sentence α if α is true in m
- $M(\alpha)$ is the set of models m for which the sentence α is true (α can also be a KB)
 - We say two sentences α and β are equivalent iff $M(\alpha) = M(\beta)$
 - KB is consistent if $M(KB) \neq \emptyset$
 - KB is inconsistent $M(KB) = \emptyset$
 - α_1 and α_2 are two KB, if they are mutually exclusive, then:
 - $M(\alpha_1) \cap M(\alpha_2) = \emptyset$
 - $M(\alpha_1 \wedge \alpha_2) = \emptyset$

Important Terms

Satisfiability:

- α is **satisfiable** if $M(\alpha) \neq \emptyset$
 - There is some assignment (model) that makes α true.
 - E.g. $\alpha \vee \beta$ is satisfiable.
- α is **unsatisfiable** if $M(\alpha) = \emptyset$
 - There is no model that makes α true.
 - E.g. $\alpha \wedge \neg \alpha$ is unsatisfiable.

Validity:

- α is **valid** if α is always true in all models
 - E.g. $\alpha \vee \neg \alpha$ is valid.

Important Terms - Exercise

$$KB = (P \Rightarrow Q) \wedge (Q \Rightarrow P) \wedge \dots$$

- $A = \{p \Rightarrow q, q \Rightarrow p, q \Rightarrow \neg p, p \Rightarrow \neg q\}$

- Is A satisfiable? Yes.

$$p \vee q$$

$$q \vee p$$

$$q \vee \neg p$$

$$p \vee \neg q$$

p	q	$p \rightarrow q$	$q \rightarrow p$	$q \rightarrow \neg p$	$p \rightarrow \neg q$	<u>A all true?</u>
true	true	T	T	F	F	F
true	false	F	T	T	T	F
false	true	T	F	T	T	F
false	false	T	T	T	T	<u>T</u> .

Syntactic Forms – CNF and DNF

Syntax Forms:

CNF (Conjunction Normal Form): $(A \vee \neg B) \wedge (A \vee \neg C \vee D)$

- CNF consists of **clauses** that are connected by conjunction.
 - **Clauses:** disjunctions of **literals** (a symbol or its negation).
- $(A \vee \neg B) \wedge (A \vee \neg C \vee D)$.
 - 2 clauses: $(A \vee \neg B)$, $(A \vee \neg C \vee D)$
 - 4 variables: A, B, C, D
 - Literals: A, $\neg B$, $\neg C$, D

DNF (Disjunction Normal Form): $(A \wedge \neg B) \vee (A \wedge \neg C \wedge D)$

- DNF consists of **terms** that are connected by disjunctions.
 - **Terms:** conjunctions of **literals**
- All propositional sentences can be converted to CNF/DNF.

Syntactic Forms - NNF

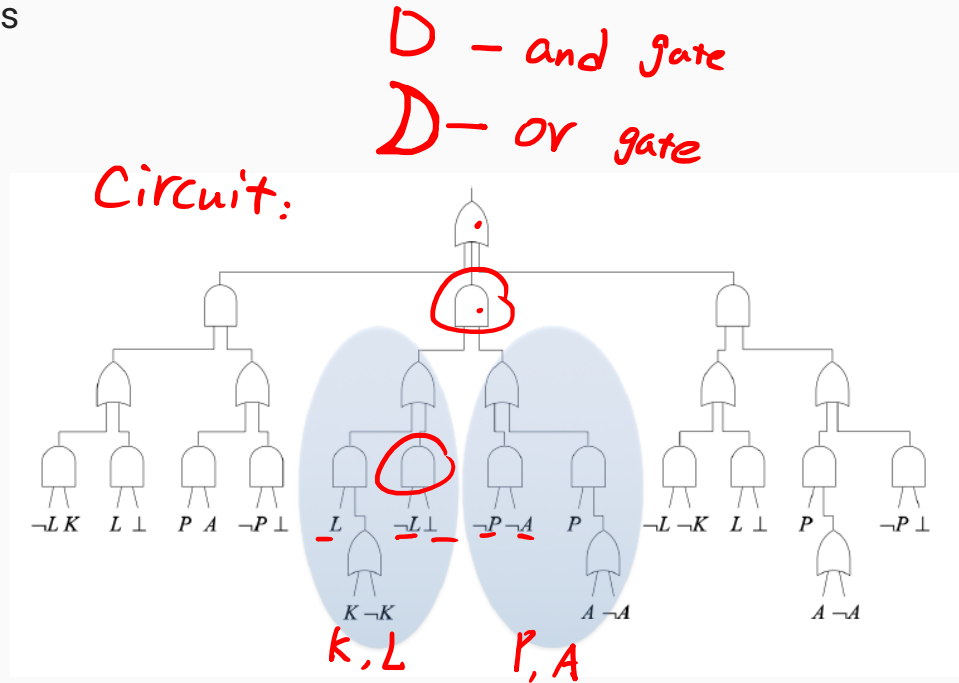
Syntax Forms:

NNF (Negation Normal Form): $(\neg A \vee B) \vee (A \vee \neg C) \wedge D$

- Negation signs only appears next to variables
- E.g. $(\neg(A \vee B) \vee (A \vee \neg C)) \wedge D$ is not NNF

Decomposable NNF: *DNF*

- Subcircuits/Inputs of an and-gate cannot share variables
- Satisfiability of DNNF can be decided in linear time



Syntactic Forms - Horn Clause

Horn Clause:

- A special type of clause that has at most one positive literal.
- Sometimes may also be written in another form only containing:

- Proposition symbol; or
- (conjunction of symbols) \Rightarrow symbol
- E.g. $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

$$A \vee B \vee \neg C \text{ X}$$

$$\neg A \vee \neg B \vee \neg C \checkmark \equiv A \wedge C \Rightarrow B$$

$$\neg A \vee \neg B \vee \neg C \checkmark$$

Another typical form of horn clause

Horn Form: When KB (knowledge base) = **conjunction** of Horn clauses

Why do we care about forms?

- Some systems and algorithms require them, e.g. CNF is required by SAT solver
- Later need them in propositional inference

Syntactic Forms – Tractable and Universal

Tractable:

- A problem is intractable if the time required to solve it grows exponentially with its size
- If a problem is tractable, it becomes easy to solve in some particular form.
 - Expressing knowledge in a tractable form helps with inference
 - E.g. Solving satisfiability is NP-complete on general propositional sentences, but it's linear on horn forms → horn forms is tractable

Universal:

- A form is universal if any sentence of propositional logic can be converted to this form.
 - E.g. CNF is universal because any propositional sentences can be converted to CNF
 - In this class, we just tell you whether a form is universal ← proof can be complicated

Semantics

$$P \Leftrightarrow Q : P \Rightarrow Q \wedge Q \Rightarrow P$$

Truth table of logical connectives:

$P \Rightarrow Q$ is equivalent to $\neg P \vee Q$

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Figure 7.8 Truth tables for the five logical connectives. To use the table to compute, for example, the value of $P \vee Q$ when P is true and Q is false, first look on the left for the row where P is *true* and Q is *false* (the third row). Then look in that row under the $P \vee Q$ column to see the result: *true*.

Semantics

Logical Equivalence:

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	commutativity of \wedge
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	commutativity of \vee
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	associativity of \wedge
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	associativity of \vee
$\neg(\neg\alpha) \equiv \alpha$	double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$	<u>De Morgan</u>
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	<u>De Morgan</u>
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	<u>distributivity of \wedge over \vee</u>
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	<u>distributivity of \vee over \wedge</u>

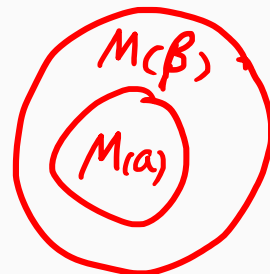
Figure 7.11 Standard logical equivalences. The symbols α , β , and γ stand for arbitrary sentences of propositional logic.

Entailment

Entailment is a relationship between sentences (syntax) that is based on semantics

- Use \models to denote entailment:

- $\alpha \models \beta$: α entails β
- α, β can be single sentence or KB



- Properties:

- $\alpha \models \beta$ iff every model in which α is true, β is also true
- $\alpha \models \beta \Leftrightarrow \underline{M(\alpha) \subseteq M(\beta)}$
- Any two sentences α and β are equivalent only if each of them entails the other
 - i.e., $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$.

$$\alpha \models \beta$$
$$M(\alpha) \subseteq M(\beta)$$

Entailment

More properties:

- KB Δ entails a sentence α is denoted as $\Delta \models \alpha$ if $M(\Delta \wedge \alpha) = M(\Delta)$.
- KB Δ is consistent with sentence α if $M(\Delta \wedge \alpha)$ is non-empty.
- KB Δ contradicts sentence α if $\Delta \wedge \alpha$ is not satisfiable. \rightarrow inference

$\Delta \models \alpha \xleftarrow{\text{show}} \neg \alpha$ $\Delta \wedge \neg \alpha$ gives you an empty clause $\rightarrow \Delta$ contradicts with $\neg \alpha$
($\Delta \wedge \neg \alpha$ is unsatisfiable)

Sanity check: KB entails α iff it contradicts $\neg \alpha$ \rightarrow used in inference

Why is entailment so important?

- We have some known facts represented as a knowledge base KB
- Now we make a new claim β
- Does our known facts support this new claim β ?

Inference

What's **inference**?

- Given a KB Δ and β , we want to derive β from Δ with algorithm i
- Denote as: $\underline{KB \vdash_i \alpha}$

Inference methods:

- Proof by enumeration – **Model Checking**
 - List all the models where Δ is True, check whether β is also True
 - E.g. Use truth table
- **Proof by refutation (resolution)**
 - Use resolution rule
 - Often use proof by contradiction
- Search - SAT solver → state-of-art
- Converting sentences to tractable forms (NNF circuits) → state-of-art



Inference

Properties:

● Soundness (correctness):

- Is this inference rule correct in all cases?
- Rule i is **sound** if whenever we can derive α from KB with i ($\text{KB} \vdash_i \alpha$), we know $\text{KB} \models \alpha$

$$\Delta = \frac{\alpha, \alpha \vee \beta}{\alpha} \rightarrow \text{not sound!}$$

$$\gamma = \frac{\alpha \wedge \beta}{\alpha} \quad \Delta \neq \gamma$$

● Completeness:

- Can this inference rule determine entailment for any $\Delta \models \beta$?
- Rule i is **complete** if whenever $\text{KB} \models \alpha$, we can derive α from KB with i ($\text{KB} \vdash_i \alpha$)

Modus Ponens rule:

$$\frac{\alpha, \alpha \rightarrow \beta}{\beta}$$

is not complete.

$$\Delta: A \vee B$$

$$\alpha: A \vee B \vee \neg C$$

$$\Delta \models \alpha, \quad \Delta \not\vdash_i \alpha$$



Inference

What's the difference between: **implication**(\Rightarrow), **entailment**(\models), and **inference**(\vdash)

- **Implication** $A \Rightarrow B$:

- Statement in the object language
- If make it a statement in the meta-language (asserting that the implication is provable) then you get something equivalent to \vdash

- **Entailment** $A \models B$:

- Statement in the meta language
- Asserts that every B holds in every model of A (semantic consequence)

- **Inference** $A \vdash B$:

- Statement in the meta language
- Asserts the existence of a proof of B from A (syntactic consequence)

If the logical system is sound, then we can conclude $A \models B$ from $A \vdash B$.

If the logical system is complete, then we can conclude $A \vdash B$ from $A \models B$.



Inference Methods

Proof by enumeration - Model Checking:

- Enumerate all possible worlds (models), check one by one
 - E.g. Truth table enumeration
- Sound and complete
- $O(2^n)$ for n symbols

$\Delta : \{A, A \vee B \rightarrow C\}$

$\alpha : C$

Determine if $\Delta \models \alpha$

A	B	C	$A \vee B \rightarrow C$
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

Steps:

- Draw a truth table
- For every model (assignment):
 - If Δ is True:
 - if α is True:
 - Continue to next model
 - else (α is False):
 - Return False (no entailment)
 - else (Δ is False):
 - skip
- Return True (after scanning the whole table without returning False)

Inference Methods

Proof by refutation (resolution):

How do we determine whether $\Delta \models \alpha$?

Proof by refutation: $\Delta \models \alpha$ if and only if the sentence $(\Delta \wedge \neg\alpha)$ is unsatisfiable.

proof by contradiction

How do we determine whether $(\Delta \wedge \neg\alpha)$ is unsatisfiable?

Proof by Resolution (a.k.a. a resolution-based algorithm): Use the resolution inference rule. This algorithm is sound and complete. It applies to any kind of Δ and α .

- Typically require translation of sentences into normal forms
- Sound and complete
- Still $O(2^n)$ in worst case

$$\begin{array}{c} \alpha \rightarrow \beta \\ \downarrow \\ \neg\alpha \vee \beta \end{array}$$

Inference Rules

- Modus Ponens:
$$\frac{\alpha, \alpha \rightarrow \beta}{\therefore \beta}$$

- Example: $\Delta = \{A, B, B \vee C, B \rightarrow D\}$

- And-Elimination
$$\frac{\alpha \wedge \beta}{\therefore \alpha}$$

- Resolution
$$\frac{\text{KB: } \alpha \vee \beta, \neg \beta \vee \delta}{\therefore \alpha \vee \delta}$$

Handwritten notes for Resolution:

$\frac{a, \beta}{a \wedge \beta}$ $\frac{\text{KB: } \{a, \beta\}}{\text{KB: 1. } a \text{ 2. } \beta}$

$\frac{a \wedge \beta}{a}$ $\frac{a \wedge \beta}{\beta}$

$\frac{(\alpha \vee \beta) \wedge (\neg \beta \vee \delta)}{\therefore \alpha \vee \delta}$

Resolution

Unit resolution inference rule:

- Each l and m is a literal, there exists one l_i such that l_i and m are complementary literals (i.e. one is the negation of the other)

$$\frac{l_1 \vee \cdots \vee l_k, \quad m}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k}$$

$m = \neg l_i$

Full resolution inference rule:

- Each l and each m is a literal, there exists one l_i and one m_i that are complementary literals (i.e. one is the negation of the other)

$$\frac{l_1 \vee \cdots \vee l_k, \quad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

$m_i = \neg l_i$

Resolution

Resolution is sound and complete for propositional logic.

How to **proof** $KB \models \alpha$ with resolution + refutation? $KB \models \alpha$

- As introduced before, **it is equivalent to proof $KB \wedge \neg \alpha$ unsatisfiable**
- **First turn KB into CNF**
- **Then apply resolution rule until an empty clause appears.**
 - If no more resolution can be applied and no empty clause appear (no contradiction is found), then $KB \wedge \neg \alpha$ is satisfiable $\rightarrow KB \not\models \alpha$

Resolution

```
function PL-Resolution(KB,q)
// KB, the knowledge base. a sentence in prop logic.
// q, the query, a sentence in prop logic
  clauses= contra(KB,q) //CNF representation of  $KB \wedge \neg q$ 
  new = {}
  do
    for each pair of clauses  $C_i, C_j$  in clauses
      resolvents=PL-Resolve( $C_i, C_j$ )
      if resolvents contains the empty clause
        return true
      new = new + resolvents
  if new is subset of clauses
    return false
  clauses = clauses + new
```

Resolution – Conversion to CNF

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

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:

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

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

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

$\models B: \{a, a \rightarrow \beta, \neg c\}$
 $\models B: a \wedge (a \rightarrow \beta) \wedge \neg c$
 $\Rightarrow 1. a$
 $2. \frac{a \rightarrow \beta}{a \wedge (a \rightarrow \beta)} \Rightarrow \neg a \vee \beta$
 $3. \neg c$

Resolution - Example

$$\begin{aligned}\Delta : & (A \vee \neg B) \rightarrow C \\ & C \rightarrow (D \vee \neg E) \\ & E \vee D\end{aligned}$$

$$\alpha : A \rightarrow D$$

Determine if $\Delta \models \alpha$

Note: For KB which is a set of sentences $\{\alpha_1, \alpha_2, \dots\}$, we can consider the whole it as a single long sentence $\alpha_1 \wedge \alpha_2 \wedge \dots$

Resolution - Example

$$\Delta : (A \vee \neg B) \rightarrow C$$

$$C \rightarrow (D \vee \neg E)$$

$$E \vee D$$

$$\underline{\alpha : A \rightarrow D}$$

Determine if $\Delta \models \alpha$

1. Convert to CNF:

$$\Delta : (A \vee \neg B) \rightarrow C$$

$$= \neg(A \vee \neg B) \vee C$$

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

$$= (\neg A \vee C) \wedge (B \vee C) \rightarrow \text{CNF}$$

$$\textcircled{2} C \rightarrow (D \vee \neg E)$$

$$= \neg C \vee (D \vee \neg E)$$

$$= \neg C \vee D \vee \neg E$$

$$\textcircled{3} E \vee D$$

$$\underline{\alpha} : A \rightarrow D = \neg A \vee D$$

$$\neg \alpha : \neg(\neg A \vee D) = A \wedge \neg D$$

$KB \wedge \neg \alpha \rightarrow \text{empty clause}$

$$KB : \textcircled{1} A \vee C$$

$$\textcircled{2} B \vee C$$

$$\textcircled{3} \neg C \vee D \vee \neg E$$

$$\textcircled{4} E \vee D$$

$$\textcircled{5} A$$

$$\textcircled{6} \neg D$$

CNF

$$\textcircled{4} + \textcircled{6} = E - \textcircled{7}$$

$$\textcircled{3} + \textcircled{3} = \neg C \vee D - \textcircled{8}$$

$$\textcircled{8} + \textcircled{1} = \neg A \vee D - \textcircled{9}$$

$$\textcircled{9} + \textcircled{5} = D$$

$$D + \textcircled{6} = \phi$$

$$\Rightarrow KB \wedge \neg \alpha \rightarrow \phi$$

Note: For KB which is a set of sentences $\{\alpha_1, \alpha_2, \dots\}$, we can consider the whole it as a single long sentence $\alpha_1 \wedge \alpha_2 \wedge \dots$

$\alpha_1 \wedge \alpha_2 \wedge \dots$

$KB \models \alpha$

Resolution - Example

$$\Delta : (A \vee \neg B) \rightarrow C$$
$$C \rightarrow (D \vee \neg E)$$
$$E \vee D$$

$$\alpha : A \rightarrow D$$

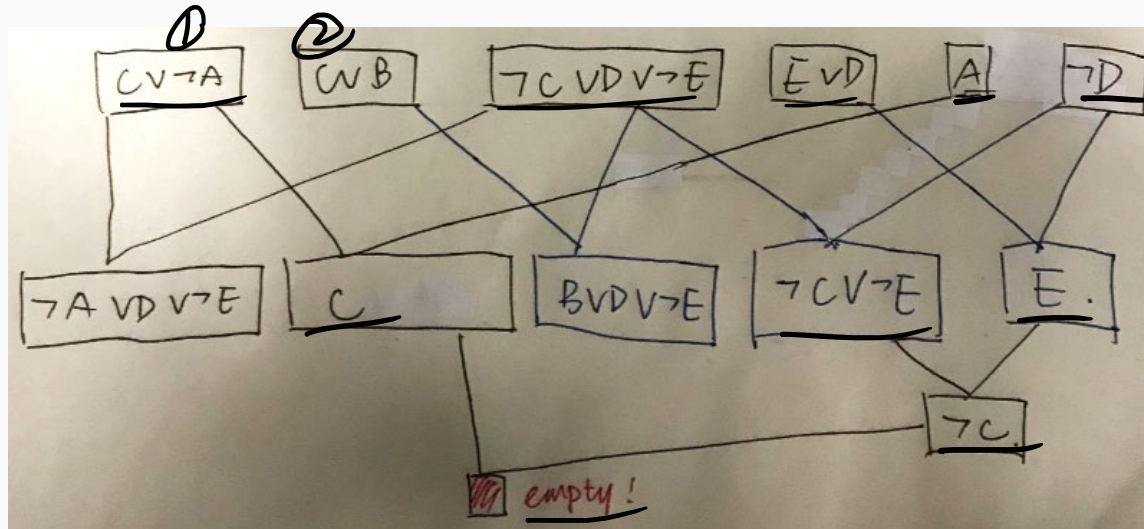
Determine if $\Delta \models \alpha$

Δ :

$$\textcircled{1} \frac{(C \vee \neg A) \wedge (C \vee B)}{(\neg C \vee D \vee \neg E)} \textcircled{2}$$
$$E \vee D$$

$\neg \alpha$:

$$\underline{A} \wedge \underline{\neg D}$$



Resolution - Exercise

$\Delta : P \vee Q, P \rightarrow R, Q \rightarrow R$

$\alpha : R$

Determine if $\Delta \models \alpha$



SAT Solver

Key idea: reducing queries to SAT (a special case of CSP)

$$\text{CNF: } \underline{\Delta} = (A \vee B \vee \neg C) \wedge (\neg A \vee C) \wedge (A \vee C \vee \neg D)$$

Clause 1 Clause 2 Clause 3

Consider each clause in a CNF as constraint

Say if we assign: $A \leftarrow F, B \leftarrow T, C \leftarrow F, D \leftarrow F \rightarrow$ this is a world w

$$\left. \begin{array}{l} \text{Clause 1: } F \vee T \vee T = T \\ \text{Clause 2: } T \vee F = T \\ \text{Clause 3: } F \vee F \vee T = T \end{array} \right\}$$

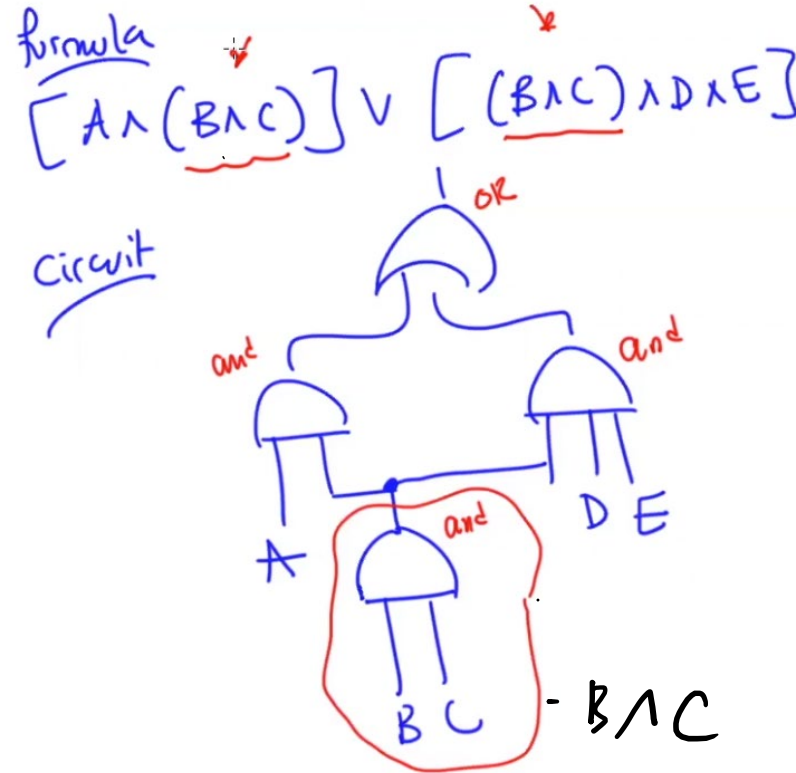
Thus the world $w \models \Delta$, it is proven that Δ is satisfiable.

How do we solve SAT?

- Backtrack search: DFS + detecting failures early (arc consistency or forward checking)
 - Called **DPLL** (initials of four authors) in the context of SAT
 - Uses unit resolution as an aid to detect failures early and increase efficiency (do resolution in linear time)
 - Use special variable/value ordering
 - Modern SAT solver can “learn” clauses → empower unit resolution
- Local search:
 - Can be **very fast**, but **NOT complete**
 - Steps:
 - Guess a truth assignment (world w)
 - Check whether $w \models \Delta$: If yes → done. If no → find another w to try
 - Method: Pick a clause that is violated (unsatisfied), then flip a variable that either maximize # of satisfied clauses or minimize # of unsatisfied clauses → can go back and forth, thus not complete

Compile Sentences into Tractable Circuits

Formula/sentence vs **circuit**: circuit is more compact (no repeat portion) than formula



Compile Sentences into Tractable Circuits

Methods:

- Decomposable NNF (DNNF) circuit
 - **Decomposability**: subcircuits feeding into an **and-gate** share no variables
- Deterministic decomposable NNF (d-DNNF)
 - **Determinism**: at most one high input for each **or-gate** under any circuit input

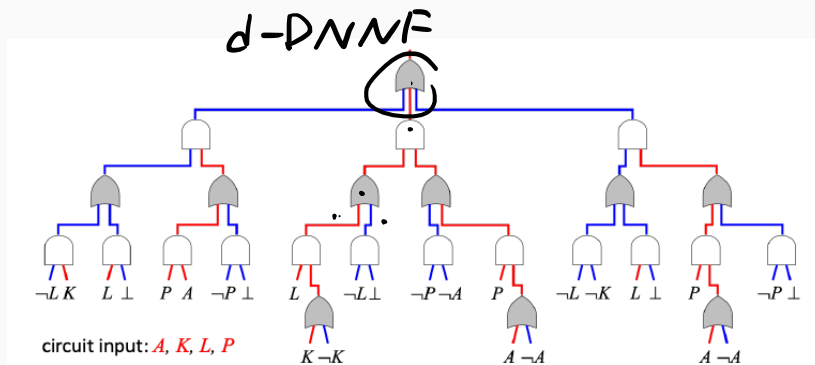
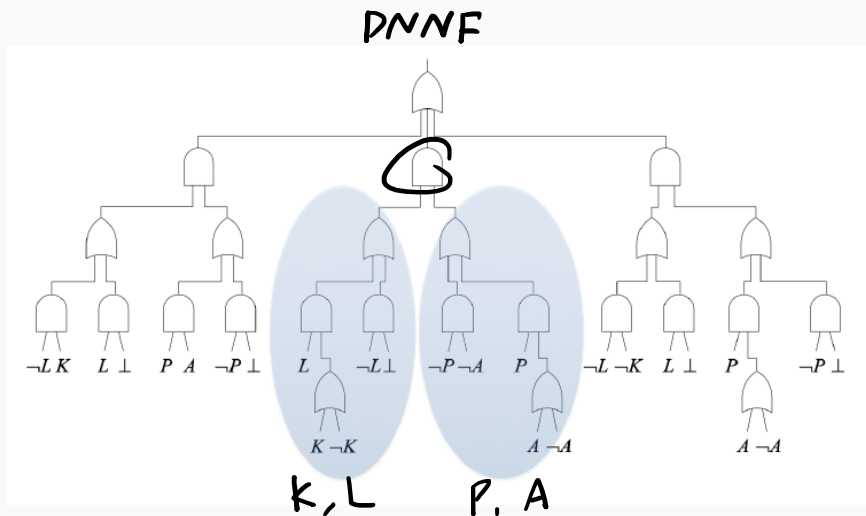


Figure 7: Illustrating the *determinism* property of NNF circuits. Red wires are high and blue ones are low.

Midterm Review – Study Guide

Midterm will be **open book and note (electronic versions are fine), but not open internet!**

The midterm will have the following questions. This applies to both offerings of the midterm, but the questions will be different for each midterm.

1. Mark nodes according to their order of expansion by different search strategies.
2. Properties of various search algorithms.
3. Minimax and alpha-beta pruning (evaluate nodes according to minimax and prune nodes according to alpha-beta pruning).
4. Formulating a problem as A* search – admissible heuristics.
5. Converting propositional sentences to CNF.
6. Answering queries using the method of enumerating models.
7. Answering implication questions using resolution.
8. Write a lisp function.
9. Various true/false questions.

The midterm may slightly deviate from the above list, but the list should give you a very good idea of what to expect.

Midterm Review

LISP

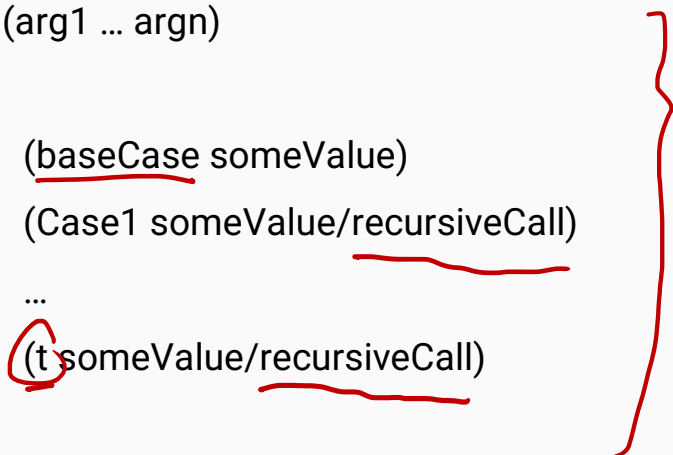
- quote or ' : everything under it is kept symbolic
- nil or (): empty list
- car: first element of the list, cdr: the rest of the list (always a list)
- (cons arg1 arg2): reverse of car+cdr → only take 2 args., 2nd arg should be a list
(cons 2 3) ⇒ (2 3) ×
(cons 2 '(3)) ⇒ (2 3) ✓
- list { • (list arg1 ... argn): construct a list '(arg1 ... argn)
- (append '(l11 l12 ...) '(l21 l22 ...) ... '(ln1 ln2 ...)): '(l11 l12 ... l21 l22 ... ln1 ln2 ...)
- predicates: atom, listp, null, equal
- (cond (cond1 value1) (cond2 value2) ... (condn valuen))
- (let ((var1 value1) ... (varn valuen)) (expression)) → local variable
 - let: parallel assignment, let*: sequential assignment

Midterm Review

LISP

- (defun functionName (arg1 ... argn) (expression))
- Calling a function: (functionName arg1 ... argn)
- General form for a lisp recursion function

```
(defun functionName (arg1 ... argn)
  (cond
    (baseCase someValue)
    (Case1 someValue/recursiveCall)
    ...
    (t someValue/recursiveCall)
  )
)
```



Midterm Review

SEARCH

- Search Problem Formulation

- Initial state, State space, Actions, Transition model, Goal Test
- 8 queens - complete formulation and incremental formulation

- State space and search tree

- Solution

- A path from initial state to goal state

- Node generation and expansion

- Fringe

- Nodes to expand. Keep in memory

- **Properties** of search strategies

- Completeness, Optimality, Time complexity, Space complexity

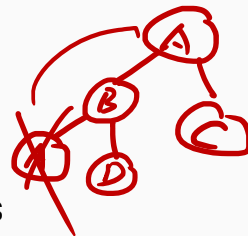
- Tree search and graph search

- Graph search maintains an “explored” set and does not re-expand states



BFS : goal test in generation

UCS, - - in expansion



Midterm Review

Uninformed Search: Properties

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes ^a	Yes ^{a,b}	No	No	Yes ^a
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes ^c	Yes	No	No	Yes ^c

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: ^a complete if b is finite; ^b complete if step costs $\geq \epsilon$ for positive ϵ ; ^c optimal if step costs are all identical; ^d if both directions use breadth-first search.

Midterm Review

Uninformed Search:

- **Breadth-first search:** expands the shallowest nodes first
 - Complete, optimal for unit step costs, exponential space complexity.
- **Uniform-cost search:** expands the node with lowest path cost
 - Complete, optimal
- **Depth-first search:** expands the deepest unexpanded node first.
 - Neither complete nor optimal, but has linear space complexity.
- **Depth-limited search:** adds a depth bound to DFS
- **Iterative deepening search:** calls depth-first search with increasing depth limits until a goal is found.
 - Complete, optimal for unit step costs, time complexity comparable to breadth-first search, linear space complexity.

Midterm Review

Informed Search:

- Informed search methods have access to heuristic function $h(n)$

- Evaluate cost from node n to goal
- Admissible, consistent

hcn: admissible \rightarrow tree-search is optimal
consistent \rightarrow graph-search is optimal.

- **Greedy search** expands nodes with minimal $h(n)$

- Not always optimal but efficient

\rightarrow admissible $\rightarrow A^*$ is optimal

- **A-star search** expands nodes with minimal $g(n) + h(n)$

- Complete and optimal
- Tree-search version when h is admissible
- Graph-search version when h is consistent

Midterm Review

Constraint Satisfaction: *CSP*

- Constraint satisfaction problem formulation
 - Variables, Domains, Constraints
- **Backtracking DFS**
 - Fail and backtrack when a consistent assignment is not possible
- **Heuristics**: increase the efficiency of backtracking DFS
 - Variable selection
 - Most constraint variable / Minimum Remaining Values heuristic *(MRV)*
 - Most constraining variable / Degree heuristic *→ tie-breaker*
 - Value selection
 - Least constraining value

Midterm Review

Constraint Satisfaction:

- Constraint propagation → *detects failures early*
 - Node consistency and arc consistency
 - AC-3 (push all the arcs) ← Arc consistency ← *used before Backtracking DFS*
 - Forward checking (variable-level arc consistency) ← *used in Backtracking DFS each time after assign a value to a variable*
- Problem Structure
 - Tree-structured CSP

Midterm Review

Game Playing: Basics

- Minimax: \rightarrow deterministic game

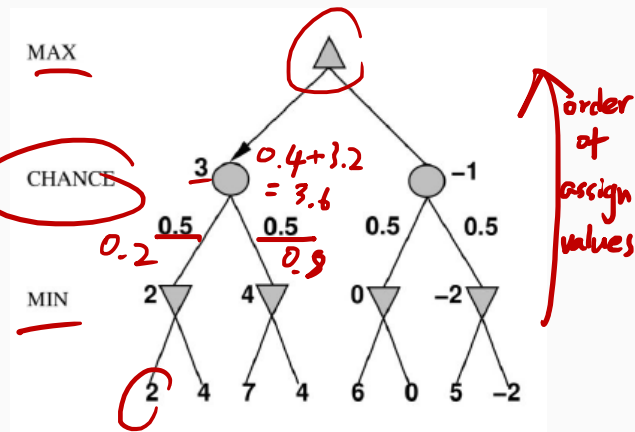
- a utility value for all goal states (leaves)
- max player: value is max of its children
- min player: value is min of its children
- value of the root: the value of the game outcome

- Expected-minimax: \rightarrow non-deterministic game (w/ probability)

- calculate the expectation over children

Δ : max (may not be same in exam!)

∇ : min



Midterm Review

Game Playing: Alpha-beta pruning

- Motivation: skip branches that won't matter to improve efficiency

- A generic algorithm:


- (min)
child: 2 } not change
α-parent: 1 } tighten the bound
Δ (max) 2
- During the DFS search, each node carries an lower bound α and an upper bound β .
 - Pushing bound upward: when a child returns, it pushes its value onto the parent (**always** **tighten the bound**). Min-child pushes onto max-parent's α . Max-child pushes onto min-parent's β . (How to remember: Max player will modify its lower bound, and min player will modify its upper bound.)
 - Pushing bound downward: right before analyzing a child, parent pushes its bound (**Both α and β**) onto that child.
 - Prune all unsearched children when parent has $\alpha \geq \beta$

- A website for alpha-beta pruning practice:

△ http://inst.eecs.berkeley.edu/~cs61b/fa14/ta-materials/apps/ab_tree_practice/

Midterm Review

Propositional Logic:

- Syntax and semantics
- Important terms: model, satisfiability, validity, entailment, etc.
- **Syntactic forms:** CNF, DNF, Horn clauses, NNF, DNNF
 - All but horn clauses are universal. DNF, horn and DNNF are tractable
- **Propositional Inference:**
 - Inference rules
 -  ◦ Method 1: Proof by enumeration - model checking: E.g. Using a truth table
 - Method 2: Proof by refutation (resolution):
 - Step 1: Convert KB to CNF
 - Step 2: Keep applying inference rules until an empty clause appear
 - Showing $\Delta \wedge \neg\alpha$ is unsatisfiable!
 - Method 3 (SAT solver) and 4 (NNF circuits) won't be covered in the midterm.

Questions?

- Midterm is next Tuesday (5/4) **10:00 am ~11:50 am** or **8:00 pm ~9:50 pm** depending on your choice in survey. Good luck!
- My slides take the following materials as references:
 - Prof. Darwiche's paper "Three Modern Roles for Logic in AI"
https://ccle.ucla.edu/pluginfile.php/3418250/mod_resource/content/1/PODS-darwiche.pdf
 - Shirley Chen's slides
 - Yewen Wang's (Winter 2020's TA) slides
 - Prof. Quanquan Gu's (Winter 2020) slides

Thank you!