Intelligent Systems (AI-2)

Computer Science cpsc422, Lecture 21

Oct, 30, 2015

Slide credit: some slides adapted from Stuart Russell (Berkeley),
some from Prof. Carla P. Gomes (Cornell)

Lecture Overview

- Finish Resolution in Propositional logics
- Satisfiability problems
- WalkSAT
- Encoding Example

Full Propositional Logics

DEFs.

Literal: an atom or a negation of an atom $p \quad \neg q \quad r$

Clause: is a disjunction of literals $p \vee \neg r \vee q$

Conjunctive Normal Form (CNF): a conjunction of clauses $(p) \wedge (q \vee \neg r) \wedge (\neg q \vee p)$

INFERENCE: $KB \overset{?}{\models} \alpha$ ← formula

- **Convert all formulas in KB and** $\neg\alpha$ **in CNF**
- **Apply Resolution Procedure**

$p \vee q \quad r \vee \neg q \rightarrow p \vee r$

$KB \not\vdash \alpha$

$KB \vdash \alpha$

Resolution Deduction step

Resolution: inference rule for CNF: sound and complete! *

$(A \vee B \vee C)$

$(\neg A)$

$\_\_\_\_\_\_\_\_\_\_\_\_\_\_$

$\therefore (B \vee C)$

"If A or B or C is true, but not A, then B or C must be true."

$(A \vee B \vee C)$

$(\neg A \vee D \vee E)$

$\_\_\_\_\_\_\_\_\_\_\_\_\_$

$\therefore (B \vee C \vee D \vee E)$

"If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true."

$(A \vee B)$

$(\neg A \vee B)$

$\_\_\_\_\_\_\_\_\_\_$

$\therefore (B \vee B) \equiv B$

Simplification

Resolution Algorithm

- The resolution algorithm tries to prove: $KB \models \alpha$
- $KB \wedge \neg\alpha$ is converted in CNF
- Resolution is applied to each pair of clauses with complementary literals $\neg q \vee r \vee s \quad p \vee q \rightarrow r \vee s \vee p$
- Resulting clauses are added to the set (if not already there)

but this is equivalent to prove that $KB \wedge \neg\alpha$ is unsatisfiable

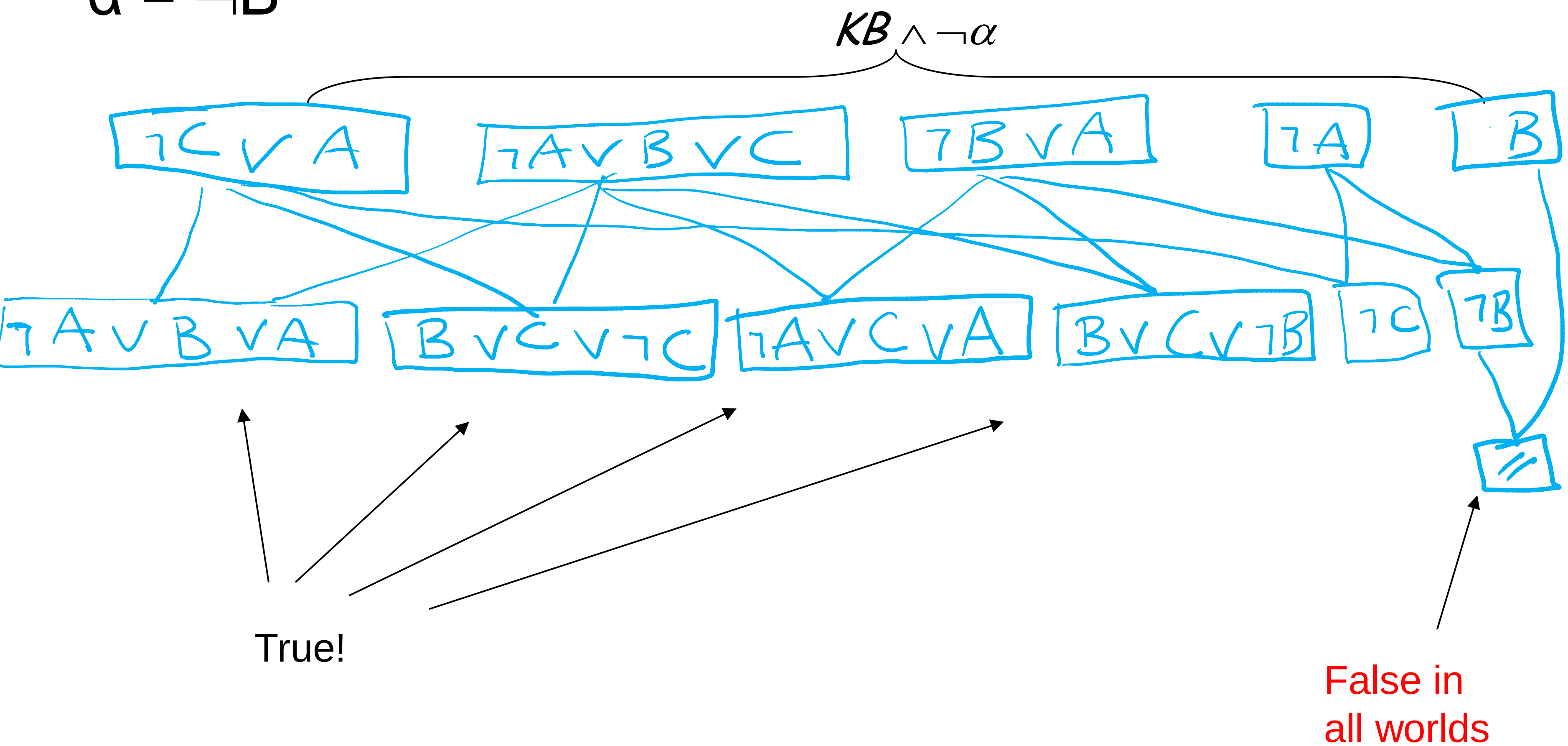
- Process continues until one of two things can happen:

1. Two clauses resolve in the empty clause. i.e. query is entailed
$P \quad \neg P \rightarrow \emptyset \quad > KB \vdash_R \alpha \Rightarrow KB \models \alpha$ assuming Resol. is sound
2. No new clauses can be added: We find no contradiction, there is a model that satisfies the sentence and hence we cannot entail the query.
$KB \nvdash_R \alpha \Rightarrow KB \not\models \alpha$ assuming Resol. is complete

Resolution example

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

$\alpha = \neg B$



Resolution algorithm

Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
    inputs: KB, the knowledge base, a sentence in propositional logic
            α, the query,

    clauses $\leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
    new $\leftarrow \{\,\}$
    loop do
        for each C_i, C_j **in** *clauses* **do**
            resolvents $\leftarrow$ PL-RESOLVE(C_i, C_j)
            if *resolvents* contains the empty clause **then return** *true*
            new $\leftarrow$ *new* $\cup$ *resolvents*
        if *new* $\subseteq$ *clauses* **then return** *false* ; no new clauses were created
        clauses $\leftarrow$ *clauses* $\cup$ *new*

Lecture Overview

- Finish Resolution in Propositional logics
- **Satisfiability problems**
- **WalkSAT**
- **Hardness of SAT**
- Encoding Example

Satisfiability problems

Consider a CNF sentence, e.g.,

$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$

Is there an interpretation in which this sentence is true (i.e., that is a model of this sentence)?

Many combinatorial problems can be reduced to checking the satisfiability of propositional sentences (example later)

How can we solve a SAT problem?

Consider a CNF sentence, e.g.,

$(\neg D \vee \neg B \vee C) \wedge (A \vee C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$

Each clause can be seen as a ***constraint*** *that reduces the number of interpretations that can be models*

Eg $(A \vee C)$ eliminates interpretations in which A=F and C=F

So SAT is a Constraint Satisfaction Problem: Find a possible world that is satisfying all the constraints (here all the clauses)

WalkSAT algorithm

(Stochastic) Local Search Algorithms can be used for this task!

Evaluation Function: number of unsatisfied clauses

WalkSat: One of the simplest and most effective algorithms:

Start from a randomly generated interpretation

- Pick randomly an unsatisfied clause
- Pick a proposition/atom to flip (randomly 1 or 2)
  1. Randomly
  2. To minimize # of unsatisfied clauses

WalkSAT: Example

$(\neg D \vee \neg B \vee C) \wedge (A \vee C) \wedge (\neg C \vee \neg B) \wedge (E \vee \neg D \vee B) \wedge (B \vee C)$

0 1 0 ✓ 0 0 ✗ 1 1 ✓ 0 0 0 ✗ 0 0 ✗

A	B	C	D	E
0	0	0	1	0
0	1	0	1	0

pick randomly unsatisfied clause

assume $(E \vee \neg D \vee B)$

pick randomly 1 or 2

assume 2

flip B $\rightarrow$ B=1

flip E # unsatst. 2
" D
" B # " "

Pseudocode for 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
    max-flips, number of flips allowed before giving up

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

pw = possible world / interpretation

The `WalkSAT` algorithm

If it returns failure after it tries *max-flips* times, what can we say?

A. The sentence is unsatisfiable

B. Nothing

C. The sentence is satisfiable

Typically most useful when we expect a solution to exist

WalkSAT: Example

$(\neg D \vee \neg B \vee C) \wedge (A \vee C) \wedge (\neg C \vee \neg B) \wedge (E \vee \neg D \vee B) \wedge (B \vee C)$

A	B	C	D	E
0	0	0	1	0
0	1	0	1	0

pick randomly unsatisfied clause

assume $(E \vee \neg D \vee B)$

pick randomly 1 or 2

assume 2

flip B → B=1

flip E # unsatst. 2
" D
" B # " 1

2

Hard satisfiability problems

Consider *random* 3-CNF sentences. e.g.,

$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$

m = number of clauses (5)

n = number of symbols (5)

- Under constrained problems:
  - ✓Relatively few clauses constraining the variables
  - ✓Tend to be easy

  E.g. For the above problem16 of 32 possible assignments are solutions
  - (so 2 random guesses will work on average)

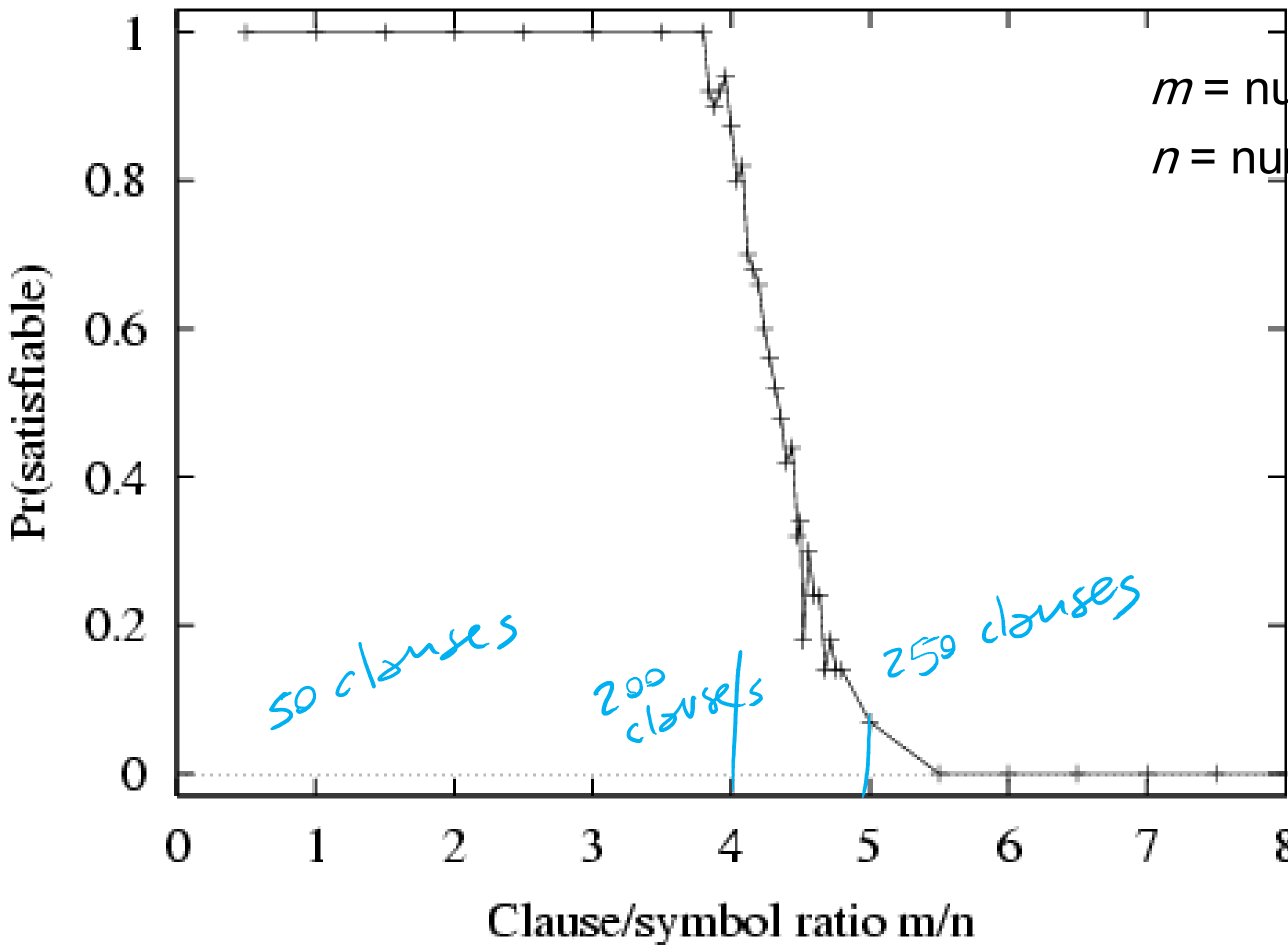
Hard satisfiability problems

What makes a problem hard?

- Increase the number of clauses while keeping the number of symbols fixed
- Problem is more constrained, fewer solutions

- You can investigate this experimentally….

P(satisfiable) for random 3-CNF sentences, n = 50



m = number of clauses
n = number of symbols

- Hard problems seem to cluster near *m/n* = 4.3 (critical point)

Lecture Overview

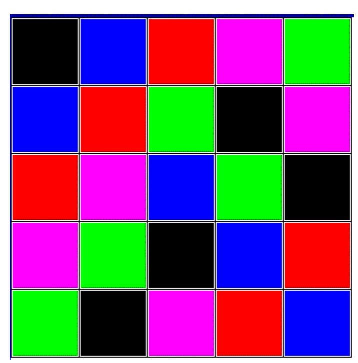
- Finish Resolution in Propositional logics
- Satisfiability problems
- WalkSAT
- **Encoding Example**

Encoding the Latin Square Problem in Propositional Logic

In combinatorics and in experimental design, a **Latin square** is an $n \times n$ array filled with n different symbols, each occurring exactly once in each row and exactly once in each column. Here is an example:

A	B	C
C	A	B
B	C	A

Here is another one:

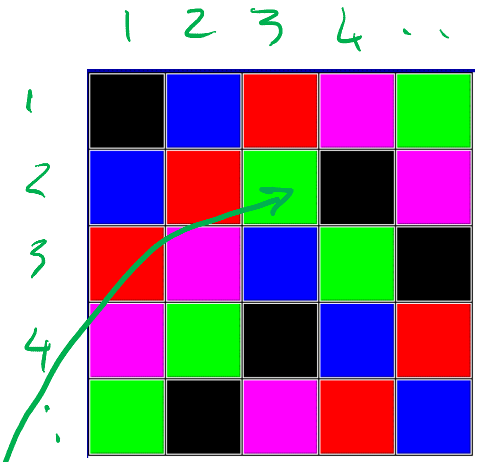


Encoding Latin Square in Propositional Logic: Propositions

Variables must be binary! (They must be propositions)

Each variables represents a color assigned to a cell.

Assume colors are encoded as integers



$$x_{ijk} \in \{0,1\}$$

Assuming colors are encoded as follows

(black, 1) (red, 2) (blue, 3) (green, 4) (purple, 5)

$x_{234} = 1$

$x_{233} = 0$ True or false, ie. 0 or 1 with respect to the interpretation represented by the picture?

How many vars/propositions overall? n^3

Encoding Latin Square in Propositional Logic: Clauses

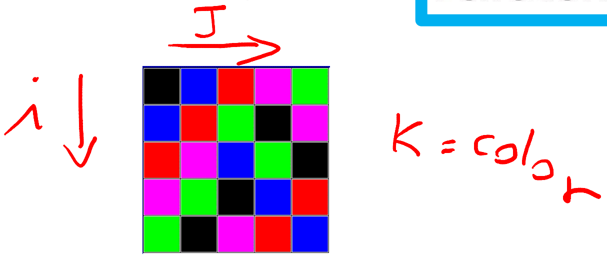
- Some color must be assigned to each cell (clause of length n);

$$\forall_{ij}(x_{ij1} \vee x_{ij2} \ldots x_{ijn})$$

A.

$$\forall_{ik}(x_{i1k} \vee x_{i2k} \ldots x_{ink})$$

B.



- No color is repeated in the same row (sets of negative binary clauses);

$$\forall_{ik}(\neg x_{i1k} \vee \neg x_{i2k}) \wedge (\neg x_{i1k} \vee \neg x_{i3k}) \ldots (\neg x_{i1k} \vee \neg x_{ink}) \cdots (\neg x_{ink} \vee \neg x_{i(n-1)k})$$

How many clauses?

Encoding Latin Square in Propositional Logic: Clauses

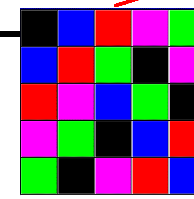
- Some color must be assigned to each cell (clause of length n);

$$\forall_{ij}(x_{ij1} \vee x_{ij2} \ldots x_{ijn}) \qquad \forall_{ik}(x_{i1k} \vee x_{i2k} \ldots x_{ink})$$

A. B.

J

i



K = color

- No color is repeated in the same row (sets of negative binary clauses);

$$\forall_{ik}(\neg x_{i1k} \vee \neg x_{i2k}) \wedge (\neg x_{i1k} \vee \neg x_{i3k}) \ldots (\neg x_{i1k} \vee \neg x_{ink}) \cdots (\neg x_{i(n-1)k} \vee \neg x_{ink})$$

$n * n$

n^2

$(x_{i11} \wedge x_{i21}) \Rightarrow \neg x_{i11} \vee \neg x_{i21}$

i

$\frac{n * n-1}{2}$

same for

How many clauses?

$n^2 * \frac{n * n-1}{2}$ $\quad O(n^4)$

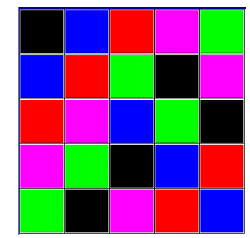
Encoding Latin Square Problems in Propositional Logic: FULL MODEL

n^3

Variab~~les: x_{ijk} *cell i,j has color k*; $i,j,k=1,2,\ldots,n$. $x_{ijk} \in \{0,1\}$~~

Each variables represents a color assigned to a cell.

Clauses: $O(n^4)$



- Some color must be assigned to each cell (clause of length n);

$$\forall_{ij}(x_{ij1} \vee x_{ij2} \ldots x_{ijn})$$

- No color is repeated in the same row (sets of negative binary clauses);

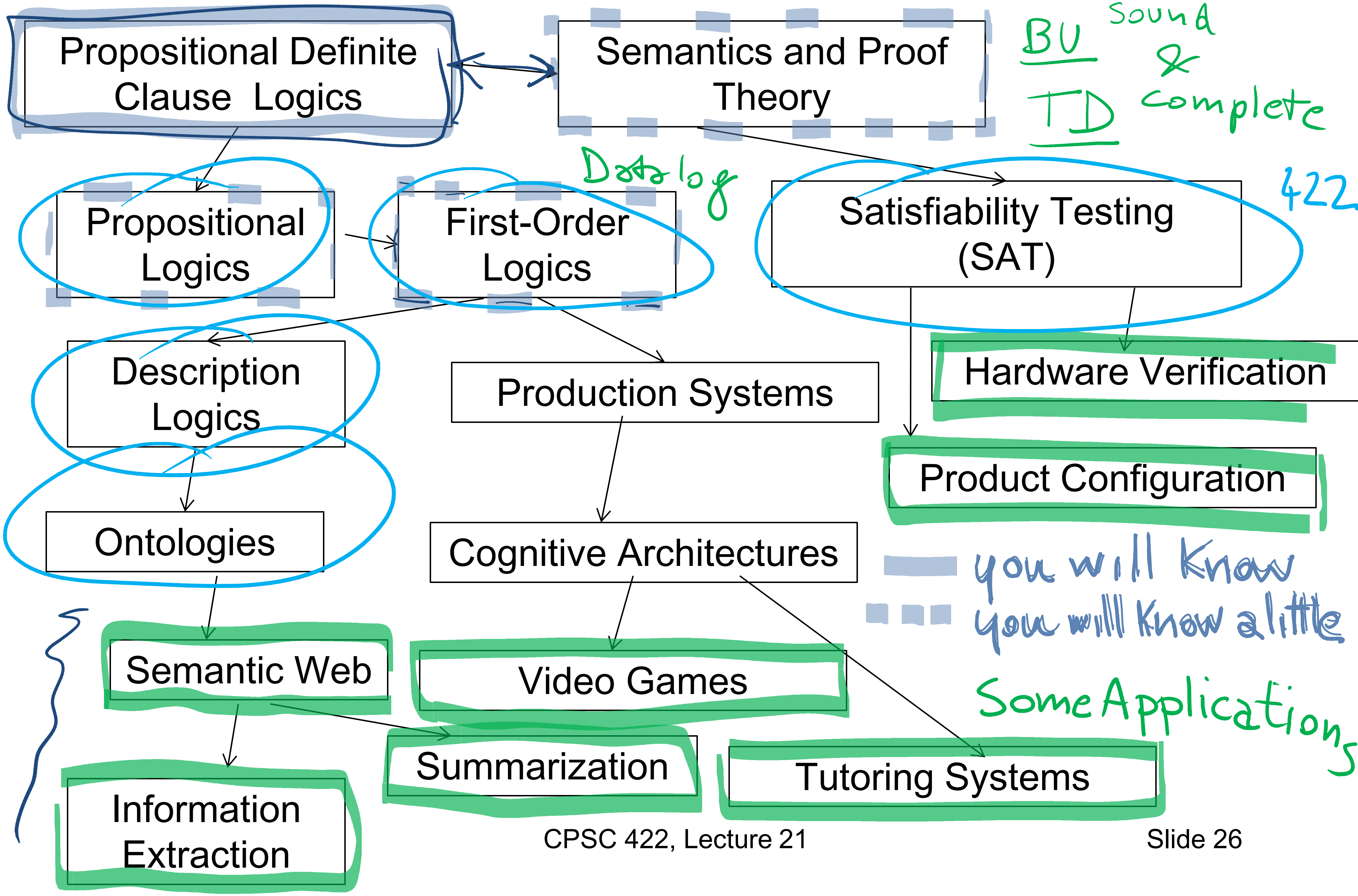
$$\forall_{ik}(\neg x_{i1k} \vee \neg x_{i2k}) \wedge (\neg x_{i1k} \vee \neg x_{i3k}) \ldots (\neg x_{i1k} \vee \neg x_{ink}) \cdots (\neg x_{i(n-1)k} \vee \neg x_{ink})$$

- No color is repeated in the same column (sets of negative binary clauses);

$$\forall_{jk}(\neg x_{1jk} \vee \neg x_{2jk}) \wedge (\neg x_{1jk} \vee \neg x_{3jk}) \ldots (\neg x_{1jk} \vee \neg x_{njk}) \cdots (\neg x_{njk} \vee \neg x_{(n-1)jk})$$

same as for rows

Logics in AI: Similar slide to the one for planning



Relationships between different Logics

(better with colors)

First Order Logic

$\forall X \exists Y p(X,Y) \Longleftrightarrow \neg q(Y)$

$p(a_1, a_2)$

$-q(a_5)$

...

Datalog

$p(X) \leftarrow q(X) \wedge r(X,Y)$

$r(X,Y) \leftarrow s(Y)$

$s(a_1), q(a_2)$

Propositional Logic

$\neg(p \vee q) \rightarrow (r \wedge s \wedge f),$

p, r

PDCL

$p \leftarrow s \wedge f$

$r \leftarrow s \wedge q \wedge p$

r

p

Learning Goals for today's class

You can:

- Specify, Trace and Debug the resolution proof procedure for propositional logics
- Specify, Trace and Debug WalkSat
- Explain differences between Proposition Logic and First Order Logic

Announcements

Midterm

- Avg 72 Max 103 Min 13
- If score below 70 need to very seriously revise all the material covered so far
- You can pick up a printout of the solutions along with your midterm

Next class Mon

- First Order Logic
- Extensions of FOL

- Assignment-3 will be posted next week!