Logic: TD as search, Datalog (variables)

Computer Science cpsc322, Lecture 23

(Textbook Chpt 5.2 &

some basic concepts from Chpt 12)

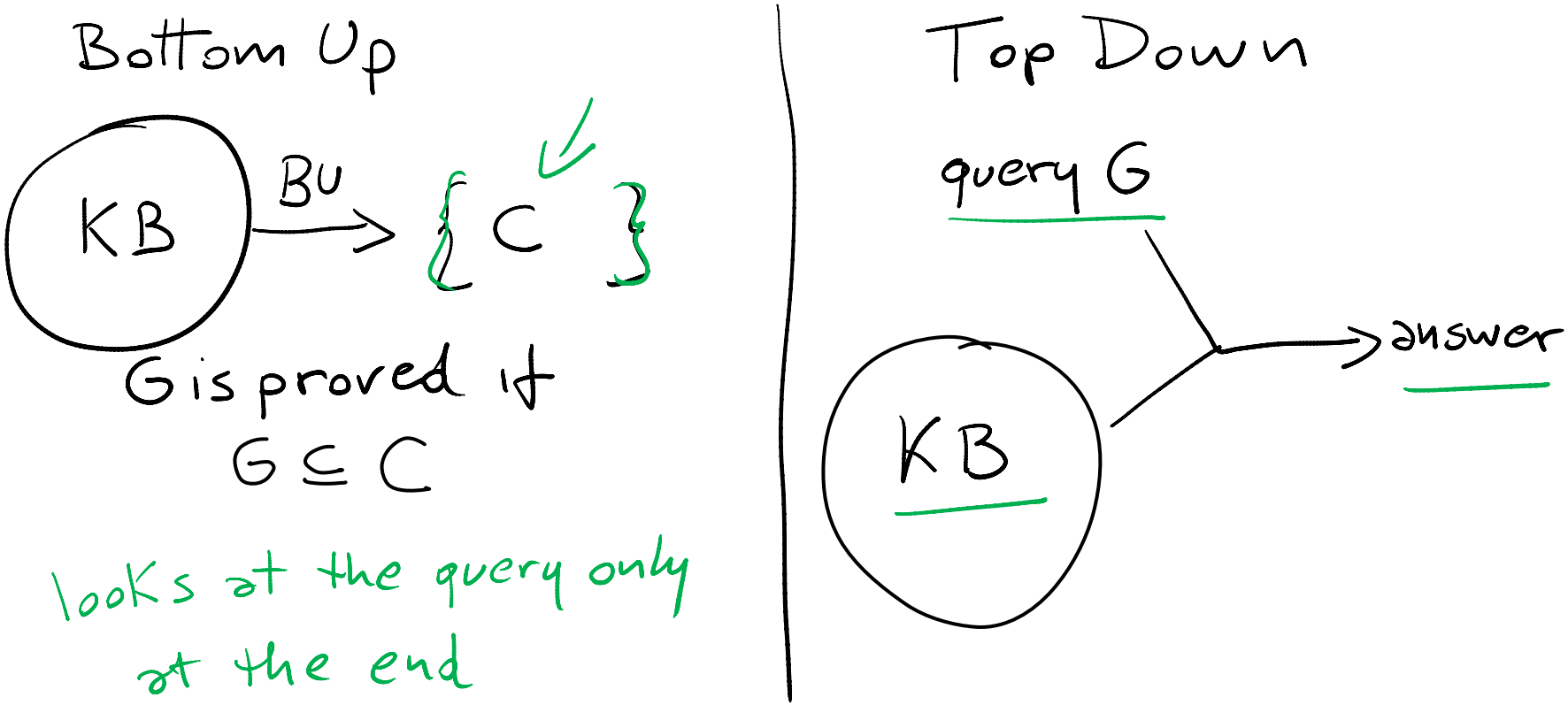
March, 12, 2010

Lecture Overview

- Recap Top Down
- TopDown Proofs as search
- Datalog

Top-down Ground Proof Procedure

Key Idea: search backward from a query *G* to determine if it can be derived from *KB*.



Top-down Proof Procedure: Basic elements

Notation: An answer clause is of the form:

$yes \leftarrow a_1 \wedge a_2 \wedge \ldots \wedge a_m$

Express query as an answer clause

(e.g., query $a_1 \wedge a_2 \wedge \ldots \wedge a_m$)

$yes \leftarrow a_1 \wedge \ldots \wedge a_m$

Rule of inference (called SLD Resolution)

Given an answer clause of the form:

$yes \leftarrow a_1 \wedge a_2 \wedge \ldots \wedge a_m$

and the clause: in KB

$a_i \leftarrow b_1 \wedge b_2 \wedge \ldots \wedge b_p$

You can generate the answer clause

$yes \leftarrow a_1 \wedge \ldots \wedge a_{i-1} \wedge b_1 \wedge b_2 \wedge \ldots \wedge b_p \wedge a_{i+1} \wedge \ldots \wedge a_m$

- **Successful Derivation**: When by applying the inference rule you obtain the answer clause *yes* ← .

$a \leftarrow e \wedge f.$	$a \leftarrow b \wedge c.$	$b \leftarrow k \wedge f.$
$c \leftarrow e.$	$d \leftarrow k.$	$e.$
$f \leftarrow j \wedge e.$	$f \leftarrow c.$	$j \leftarrow c.$

KB

Query: a (two ways)

yes ← a.

a ← e ∧ f

a ← f

a ← c

a ← e

a ←

yes ← a.

a ← b ∧ c

a ← k ∧ f ∧ c

a

Fail

Lecture Overview

- Recap Top Down
- TopDown Proofs as search
- Datalog

Systematic Search in different R&R systems

Constraint Satisfaction (Problems):

- State: assignments of values to a subset of the variables
- Successor function: assign values to a "free" variable
- Goal test: set of constraints
- Solution: possible world that satisfies the constraints
- Heuristic function: *none (all solutions at the same distance from start)*

Planning (forward) :

- State possible world
- Successor function states resulting from valid actions
- Goal test assignment to subset of vars
- Solution sequence of actions
- Heuristic function empty-delete-list (solve simplified problem)

Logical Inference (top Down)

- State answer clause
- Successor function states resulting from substituting one atom with all the clauses of which it is the head
- Goal test empty answer clause
- Solution start state
- Heuristic function ……..

Start state: query as an answer clause

yes ←

yes ←

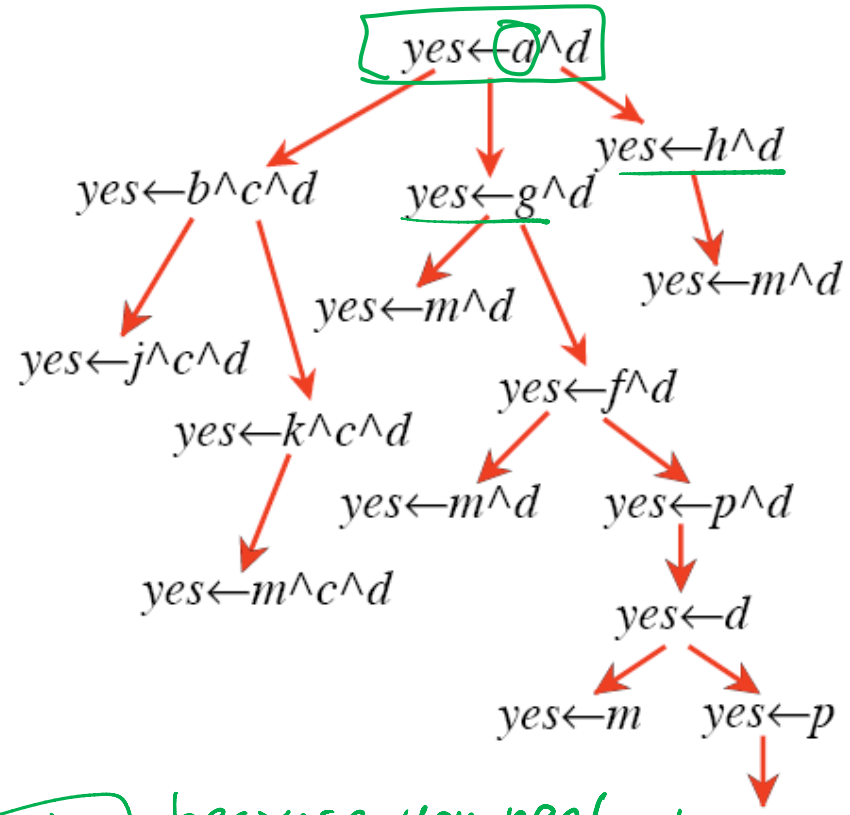
see next slide

Search Graph

Prove: ? ← $a \wedge d$.

KB

$a \leftarrow b \wedge c.$	$a \leftarrow g.$
$a \leftarrow h.$	$b \leftarrow j.$
$b \leftarrow k.$	$d \leftarrow m.$
$d \leftarrow p.$	$f \leftarrow m.$
$f \leftarrow p.$	$g \leftarrow m.$
$g \leftarrow f.$	$k \leftarrow m.$
$h \leftarrow m.$	$p.$



Admissible

because you need at least that number of resolution steps to obtain yes←

Heuristics?

# of atoms in answer clause

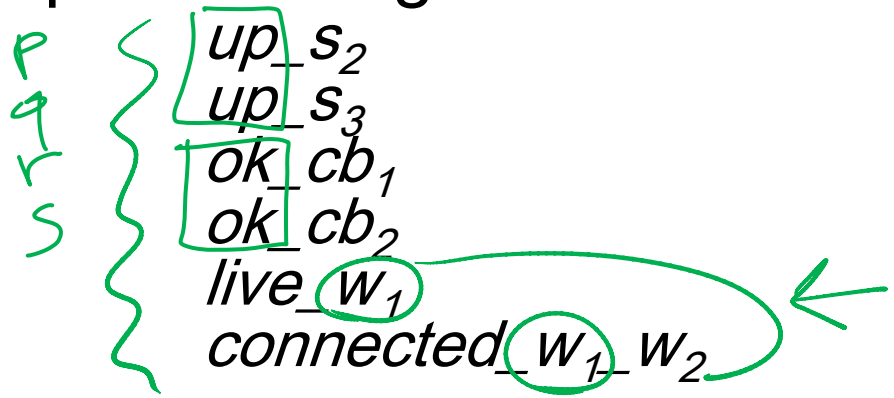
i.e. the goal state

Lecture Overview

- Recap Top Down
- TopDown Proofs as search
- Datalog

Representation and Reasoning in Complex domains

- In complex domains expressing knowledge with **propositions** can be quite limiting



- It is often natural to consider **individuals** and their **properties**

$up(s_2)$
$up(s_3)$
$ok(cb_1)$
$ok(cb_2)$
$live(w_1)$
$connected(w_1, w_2)$

There is no notion that

the system can reason about

up_s_2
up_s_3

up are about the same property

$live\_w_1$
$connected\_w_1\_w_2$

w_1 are about the same individual

What do we gain….

By breaking propositions into relations applied to individuals?

- Express **knowledge** that **holds for set of individuals** (by introducing variables)

  live(W) <- connected_to(W,W1) ∧ live(W1) ∧ wire(W) ∧ wire(W1).

- We can **ask generic queries** (i.e., containing vars variabless)

  ? connected_to(W, w_1)

Datalog vs PDCL (better with colors)

First Order Logic

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

$p(a_1, a_2)$

$\neg q(a_5)$

Propositional Logic

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

p, r

Datalog

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

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

$s(a_1), q(a_2)$

PDCL

$p \leftarrow s \wedge t$

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

r

p

Datalog: a relational rule language

It expands the syntax of PDCL….

A variable is a symbol starting with an upper case letter

X Y

A constant is a symbol starting with lower-case letter or a sequence of digits.

alan w1

A term is either a variable or a constant.

A predicate symbol is a symbol starting with lower-case letter.

in part-of live

Datalog Syntax (cont')

so it includes propositions

An atom is a symbol of the form p or $p(t_1 \ldots. t_n)$ where p is a predicate symbol and t_i are terms

sunny in(alan, X)

A definite clause is either an atom (a fact) or of the form:

$$h \leftarrow b_1 \wedge \ldots \wedge b_m$$

where h and the b_i are atoms (Read this as ``h if b.")

in(X, Y) ← in(X, Z) ∧ part_of(Z, Y)

A knowledge base is a set of definite clauses

Datalog: Top Down Proof

Extension of TD for PDCL.

How do you deal with variables?

Example:

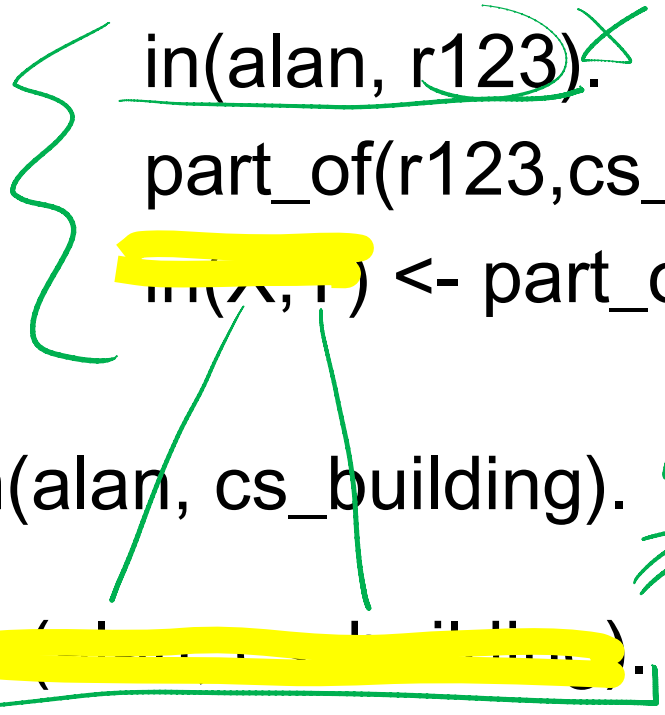
in(alan, r123).

part_of(r123,cs_building).

in(X, Y) <- part_of(Z,Y) & in(X,Z).

Query: in(alan, cs_building).

yes <- [redacted].



AIspace ex 5.9

Datalog: queries with variables

part_of(r123,cs_building).

<- in(X,Z). & part_of(Z,Y)

Query: in(alan, X1).

Yes(X1) <-

one answer

yes(r123)

yes(X1) ← in(alan,Z) & part-of(Z X1)

ex Datalog.

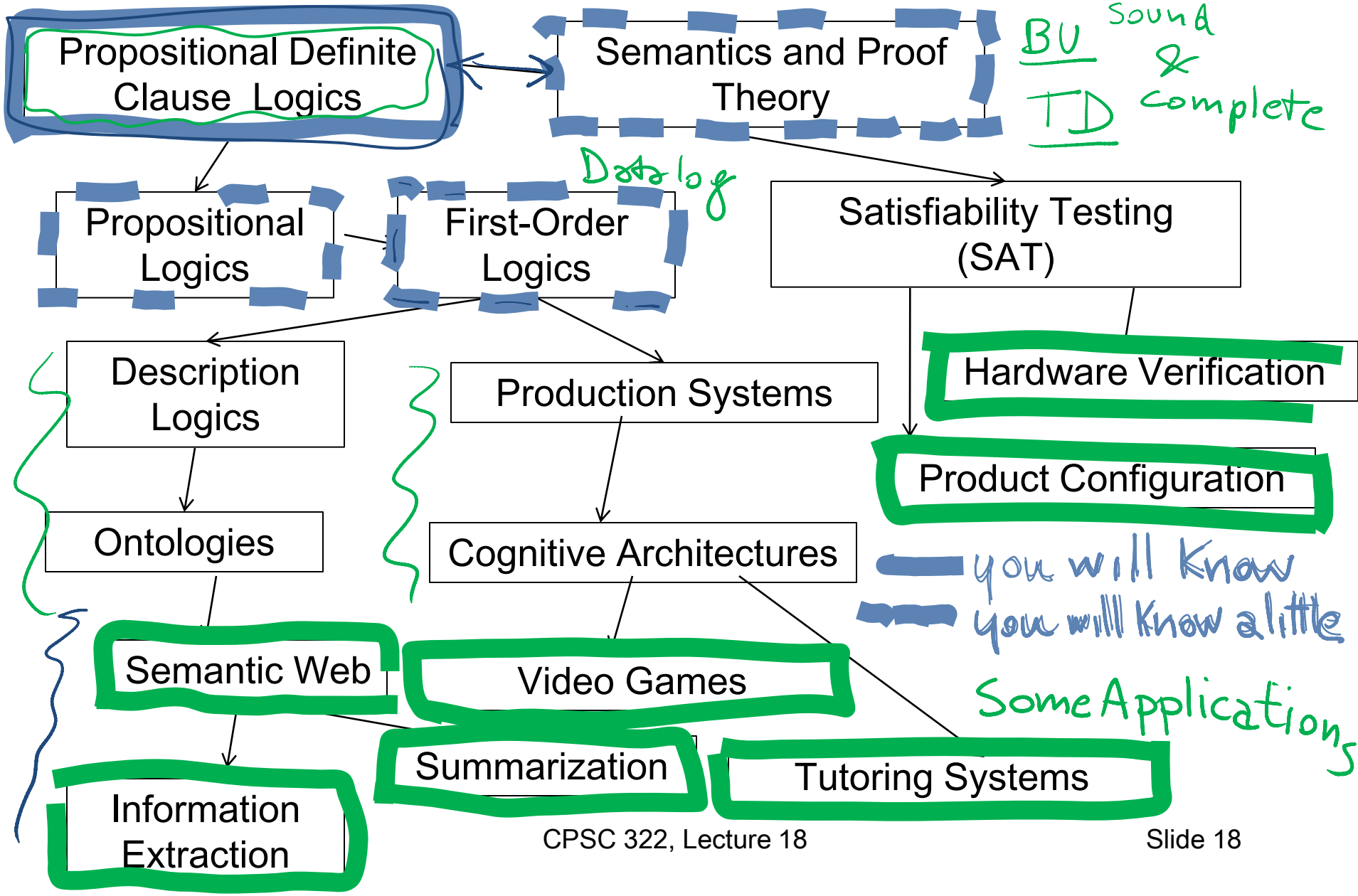
yes(cs_building)

Learning Goals for today's class

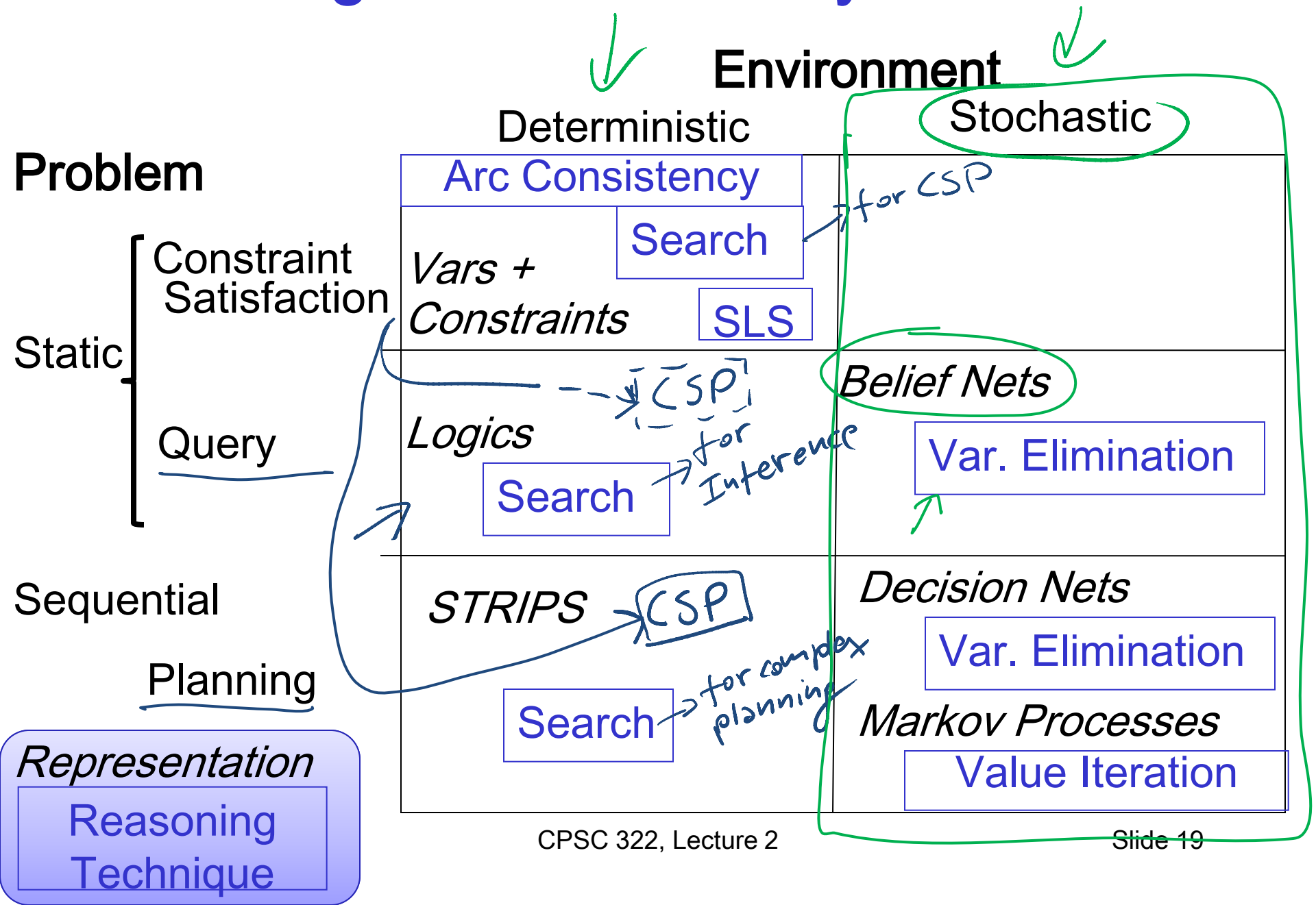
You can:

- Define/read/write/trace/debug the **TopDown** proof procedure (as a **search** problem)
- Represent simple domains in **Datalog**
- Apply **TopDown** proof procedure in **Datalog**

Logics in AI: Similar slide to the one for planning



Big Picture: R&R systems



Midterm review

Average 75 ☺

Best 104

Five <50%

How to learn more from midterm

- Carefully examine your mistakes (and our feedback)
- If you still do not see the correct answer/solution go back to your notes, the slides and the textbook
- If you are still confused come to office hours with specific questions

Need to talk with student: Huan Dong 11519071