Logic: TD as search, Datalog (variables)

Computer Science cpsc322, Lecture 23

(Textbook Chpt 5.2 &

some basic concepts from Chpt 12)

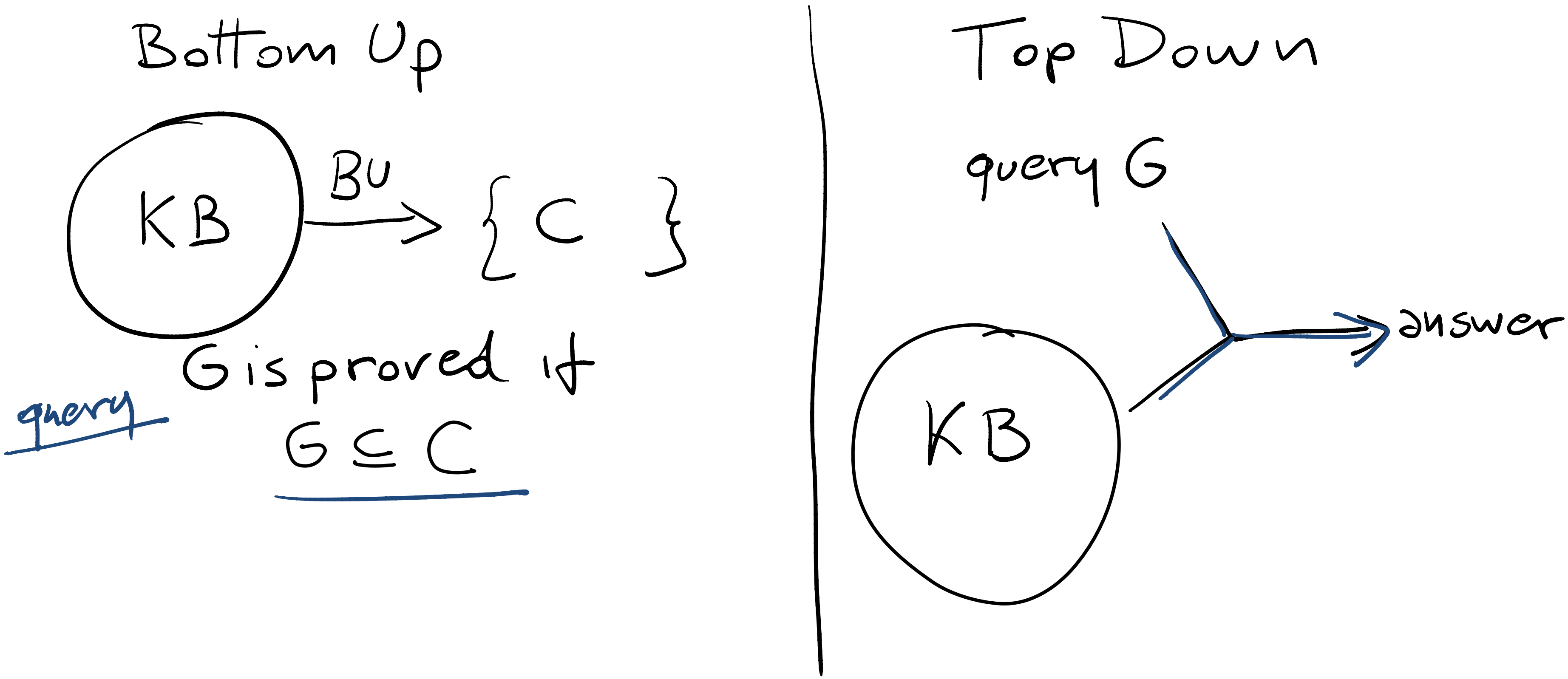
March, 6, 2009

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: from 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 $yes \leftarrow .$ empty body

KB

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

Query: a (two ways)

$yes \leftarrow a.$

$yes \leftarrow e \wedge f$

$yes \leftarrow f$

$yes \leftarrow c$

$yes \leftarrow e$ → $yes \leftarrow$

$yes \leftarrow a.$

$yes \leftarrow b \wedge c$

$yes \leftarrow k \wedge f \wedge c$

fails

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)

start state: query as answer clause

- 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 .... number of atoms in body of state/answer clause

Search Graph

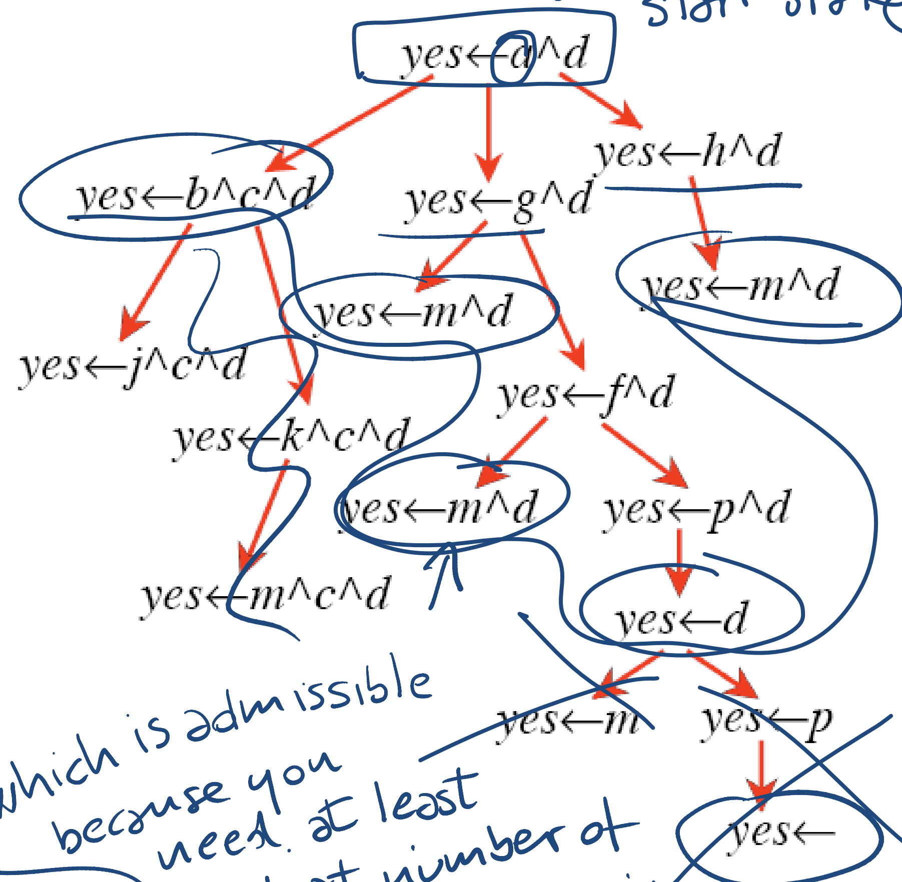
Prove: ?← $a \wedge d$.

KB

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

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

start state



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

Heuristics?

atoms in the answer clause

AIspace

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

up_s_2
up_s_3
ok_cb_1
ok_cb_2
$live\_w_1$
$connected\_w_1\_w_2$

The system can reason about

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

aka relations/predicates

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

There is no notion that

up_s_2
up_s_3

← are about the same property 'up'

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

are about the same individual w1

What do we gain….

By breaking propositions into relations applied to individuals?

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

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

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

  ? connected_to(W, w_1)

Datalog vs PDCL (better with colors)

First Order Logic

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

$p(a_1, a_2)$

$-q(a_5)$

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

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.

w1 alan

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')

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)

includes propositions

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 atomic symbol (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:

KB

in(alan, r123).

part_of(r123,cs_building).

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

first resolution step

Query: in(alan, cs_building).

yes <- in(alan, cs_building).

output of resolution

part-of(Z,cs-bldg) & in(alan,Z)

Datalog: queries with variables

in(alan, r123).

yes(r123) ←

part_of(r123,cs_building).

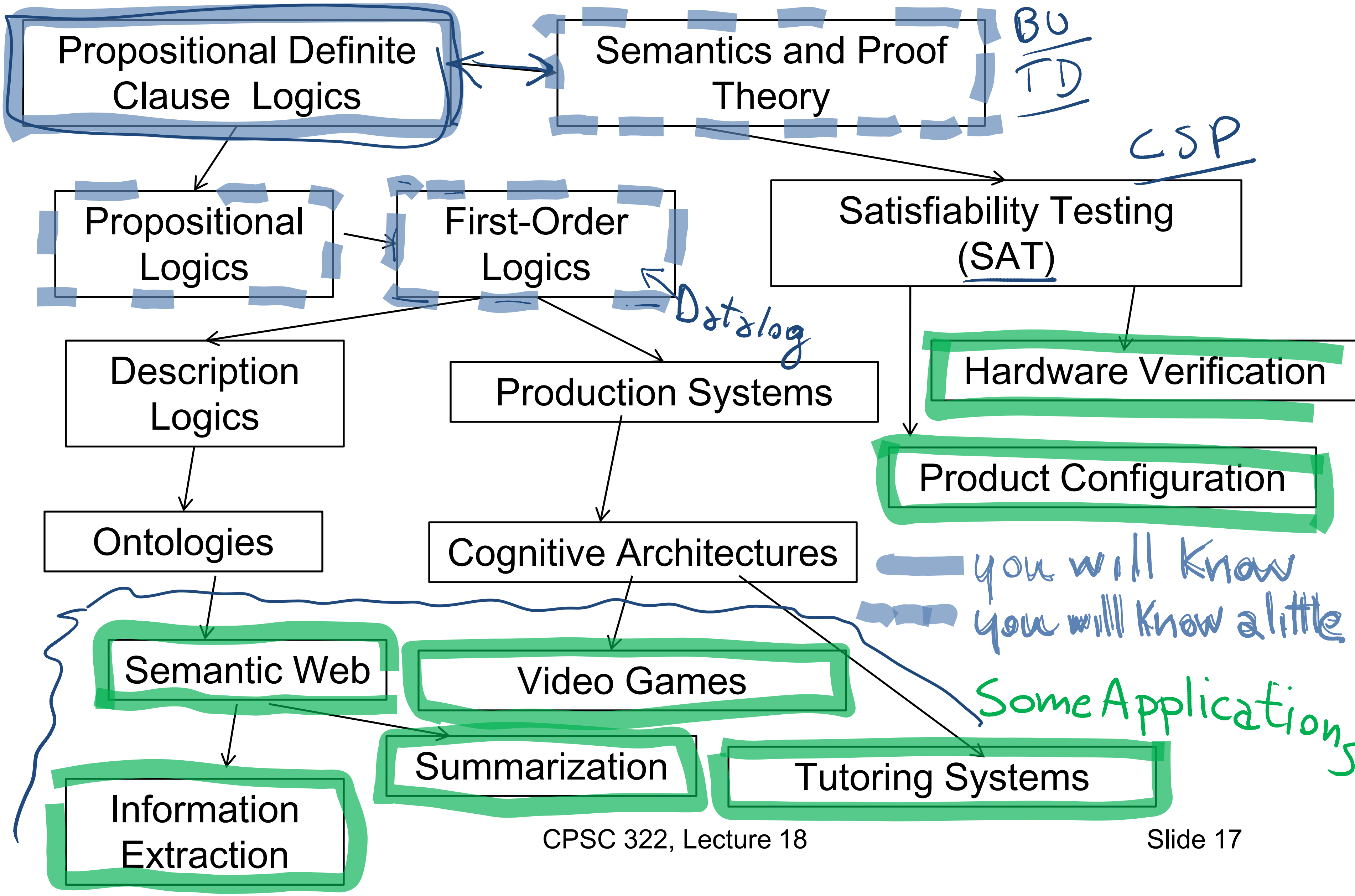
in(X,Y) <- in(X,Z). & part_of(Z,Y)

yes(X1) ← in(alan, Z) & part-of(Z,X1)

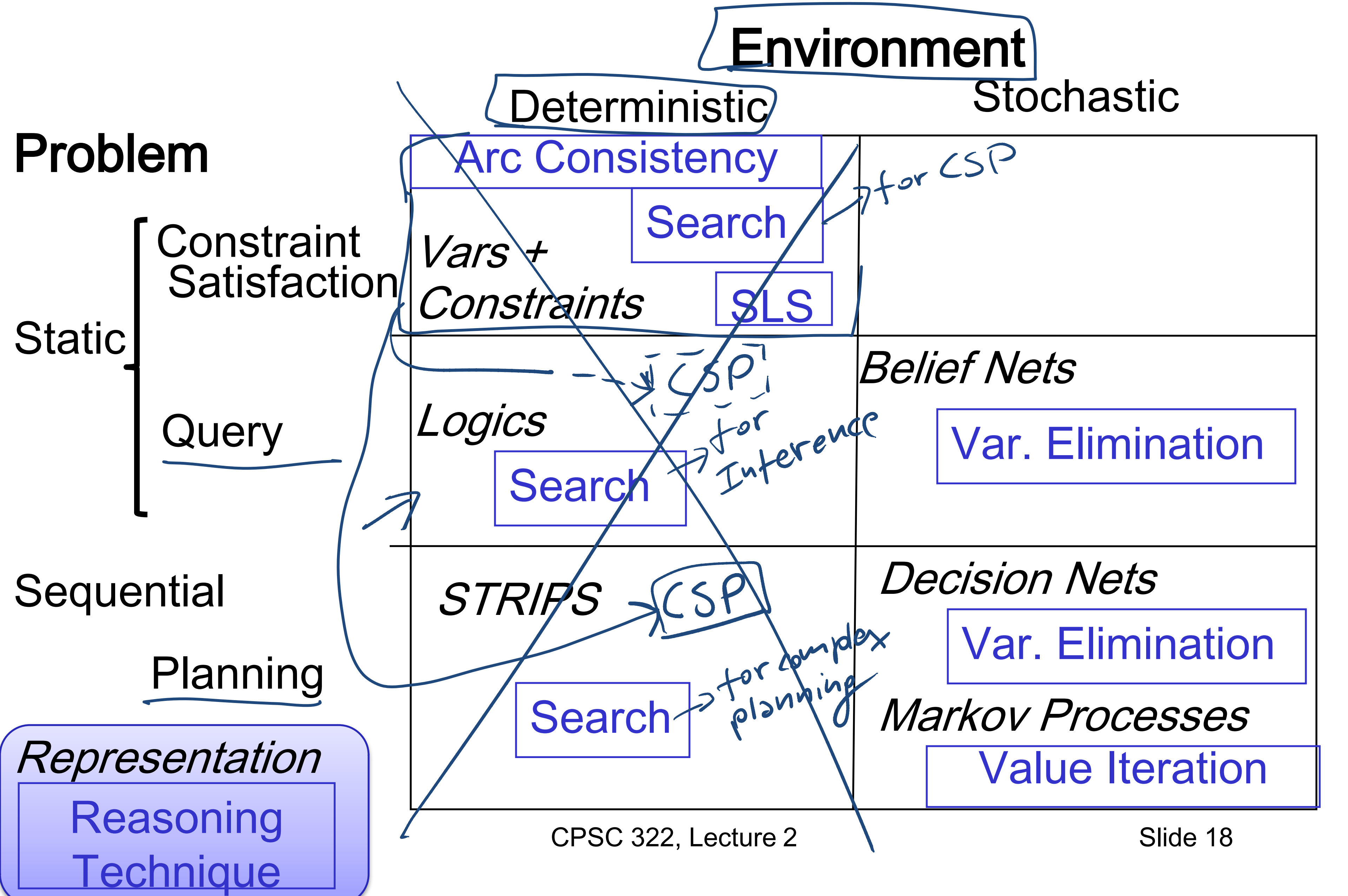
Query: in(alan, X1).

Yes(X1) <- in(alan, X1).

Logics in AI: Similar slide to the one for planning



Big Picture: R&R systems



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**

Midterm review

(without outlier who did 0%)

Average 72% ☺

Best 93%, Worst 23%

Only three <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