Lecture 17

Datalog

Intro to Probability

Lecture Overview

- Recap of Lecture 16
- TD: soundness and completeness
- SLD Resolution in Datalog
- Intro to Reasoning Under Uncertainty
- Introduction to Probability
  - ✓Random Variables and Possible World Semantics
  - ✓Probability Distributions (time permitting)

Bottom-up proof procedure

C :={};
repeat
select clause "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" in KB such that $b_i \in C$ for all i, and $h \notin C$;
C:= C $\cup$ { h } until no more clauses can be selected.

$KB \vdash_{BU} G$ if $G \subseteq C$ at the end of this procedure

The C at the end of BU procedure is a fixed point:

- Further applications of our rule of derivation will not change C!

Proved that bottom-up proof procedure is sound and complete

- BU is sound: it derives only atoms that logically follow from KB

- BU is complete: it derives all atoms that logically follow from KB

- Together: it derives exactly the atoms that logically follow from KB

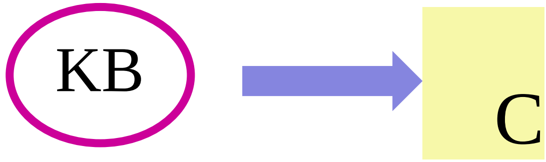
- And, it is efficient!
- Linear in the number of clauses in KB
  - ✓Each clause is used maximally once by BU

4

Bottom-up vs. Top-down

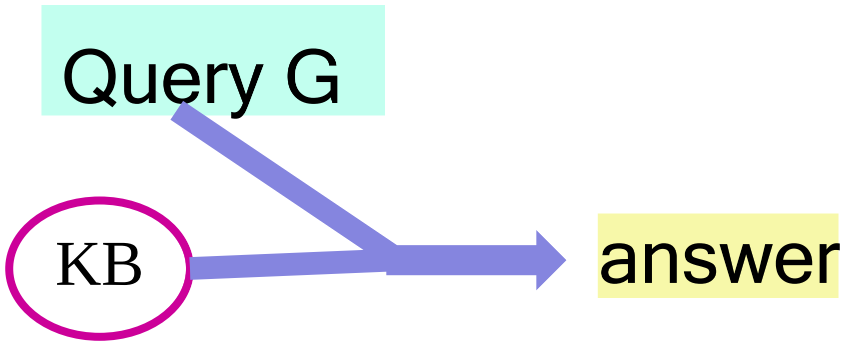
Bottom-up

Top-down



Query g is proven if g ∈ C

- BU derives the same C regardless of the query
- Derivation process not guided by the query



Key Idea of top-down:

search backward from a query g to determine if it can be derived from KB.

SLD Resolution

- Rule of derivation: the SLD Resolutionof clause

$$yes \leftarrow a_1 \wedge \ldots \wedge a_{i-1} \wedge a_i \wedge a_{i+1} \ldots \wedge a_m$$

on atom a_i with the clause:

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

is the answer clause $yes \leftarrow a_1 \wedge \ldots \wedge a_{i-1} \wedge b_1 \wedge \ldots \wedge b_p \wedge a_{i+1} \ldots \wedge a_m$

$yes \leftarrow b \wedge c.$

$b \leftarrow k \wedge f.$

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

SLD resolution

$yes \leftarrow e \wedge f.$

$e.$

$yes \leftarrow f$

Top-down Proof Procedure for PDCL

To solve the query $?\ q_1 \wedge \ldots \wedge q_k$:

ac:= yes ← body, where body is $q_1 \wedge \ldots \wedge$ q_k repeat select $q_i \in$ body; choose clause Cl $\in$ KB, Cl is $q_i \leftarrow b_c$; replace q_i in body by b_c until ac is an answer (fail if no clause with q_i as head)

select: any choice will work choose:
have to pick the right one

We showed soundness and completeness

Top-down/SLD resolution as Search

State: answer clause of the form $yes \leftarrow a_1 \wedge \ldots \wedge a_k$

Successor function: state resulting from substituting first atom a_1 with $b_1 \wedge \ldots \wedge b_m$ if there is a clause $a_1 \leftarrow b_1 \wedge \ldots \wedge b_m$ **Goal test:** is the answer clause empty (i.e. yes $\leftarrow$) ?

Solution: the proof, i.e. the sequence of SLD resolutions

Prove: ?$\leftarrow a \wedge d$.

$a \leftarrow b \wedge c.$ $\quad a \leftarrow g.$

$a \leftarrow h.$ $\quad b \leftarrow j.$

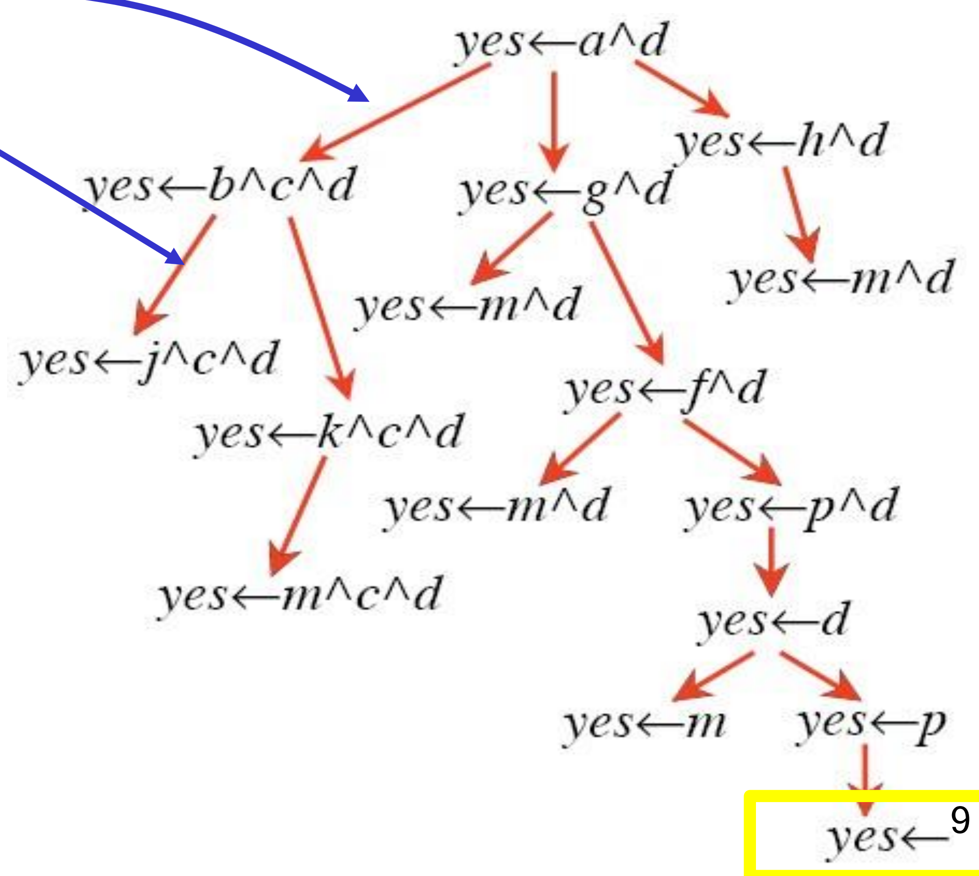
$b \leftarrow k.$ $\quad d \leftarrow m.$

$d \leftarrow p.$ $\quad f \leftarrow m.$

$f \leftarrow p.$ $\quad g \leftarrow m.$

$g \leftarrow f.$ $\quad k \leftarrow m.$

$h \leftarrow m.$ $\quad p.$



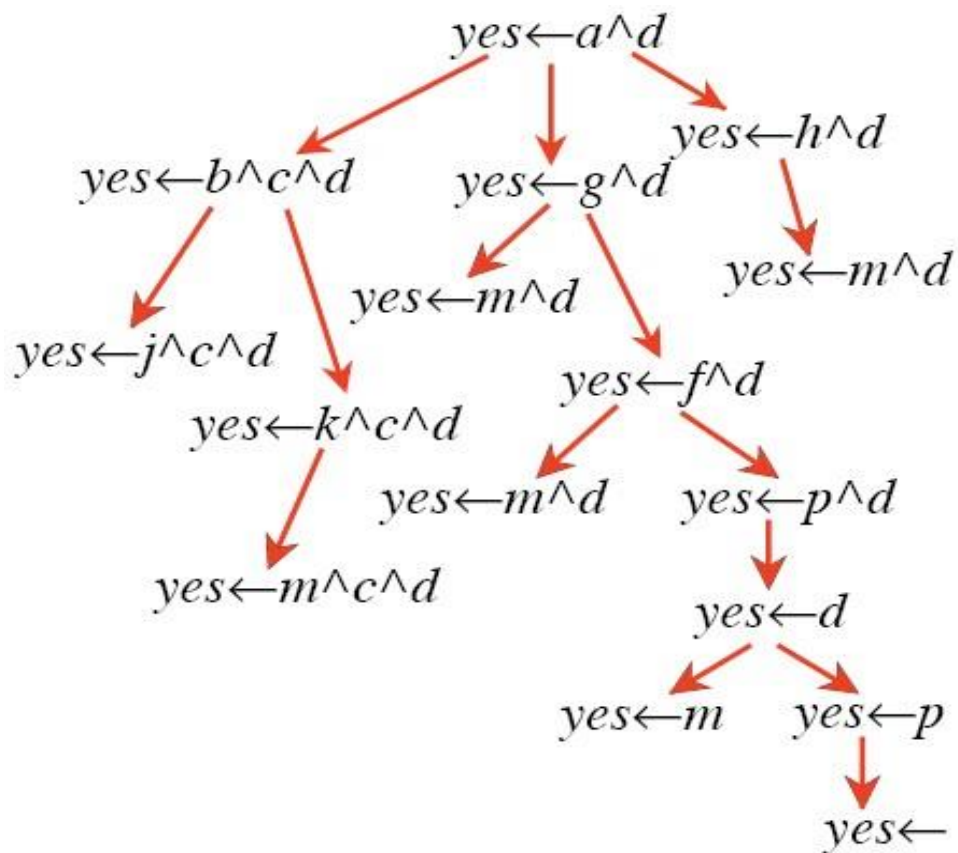
Top-down/SLD resolution as Search

State: answer clause of the form $yes \leftarrow a_1 \wedge \ldots \wedge a_k$

Successor function: state resulting from substituting first atom a_1 with $b_1 \wedge \ldots \wedge b_m$ if there is a clause $a_1 \leftarrow b_1 \wedge \ldots \wedge b_m$ Goal test: is the answer clause empty (i.e. $yes \leftarrow$) ?

Prove: ?← a ∧ d.

Solution: the proof, i.e. the sequence of SLD resolutions



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

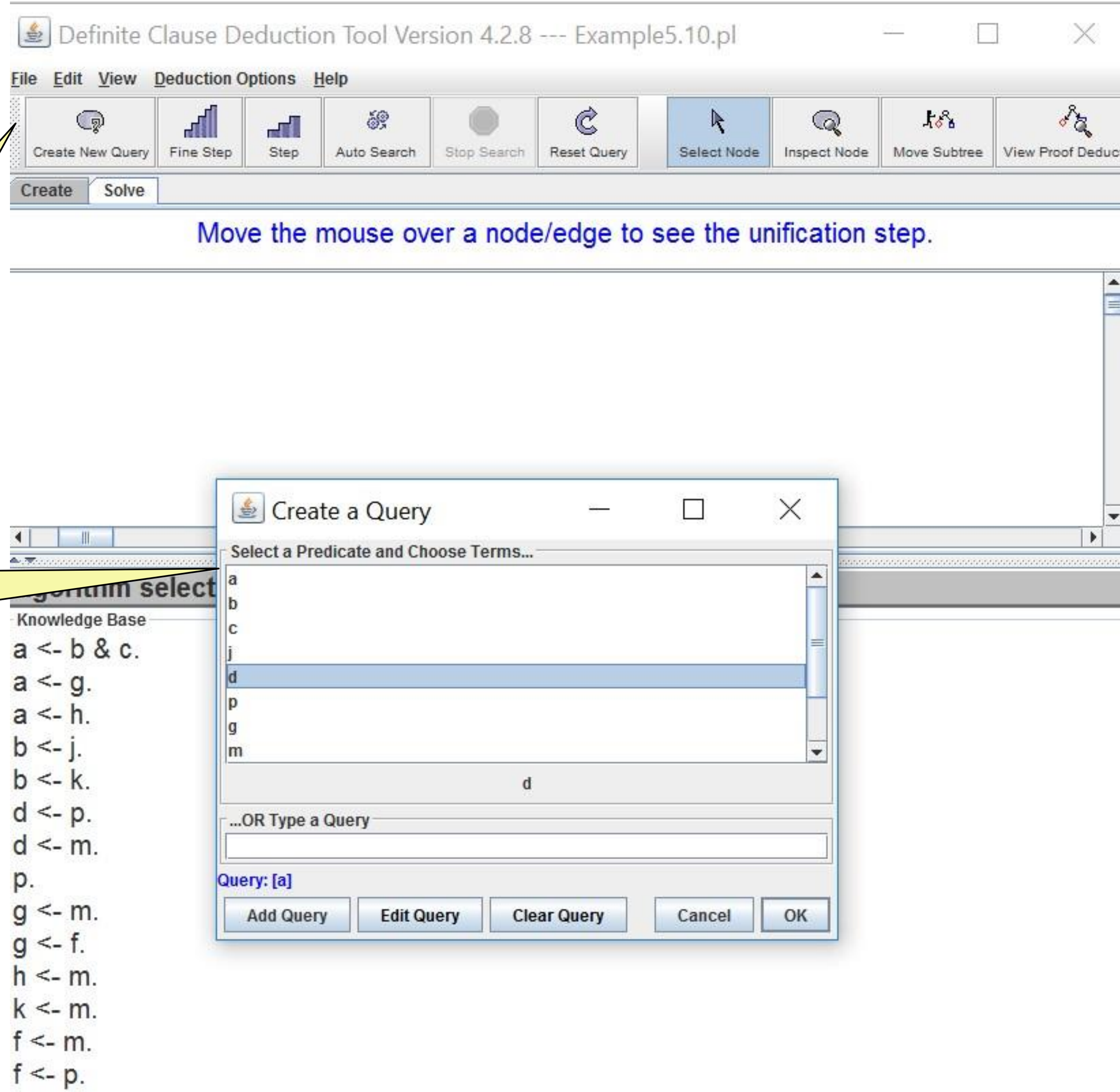
can trace the example in the Deduction Applet at http://aispace.org/deduction/ using file *kb-for-topdown-search* available in course schedule

Deduction

Button to initiatethe creation a new query

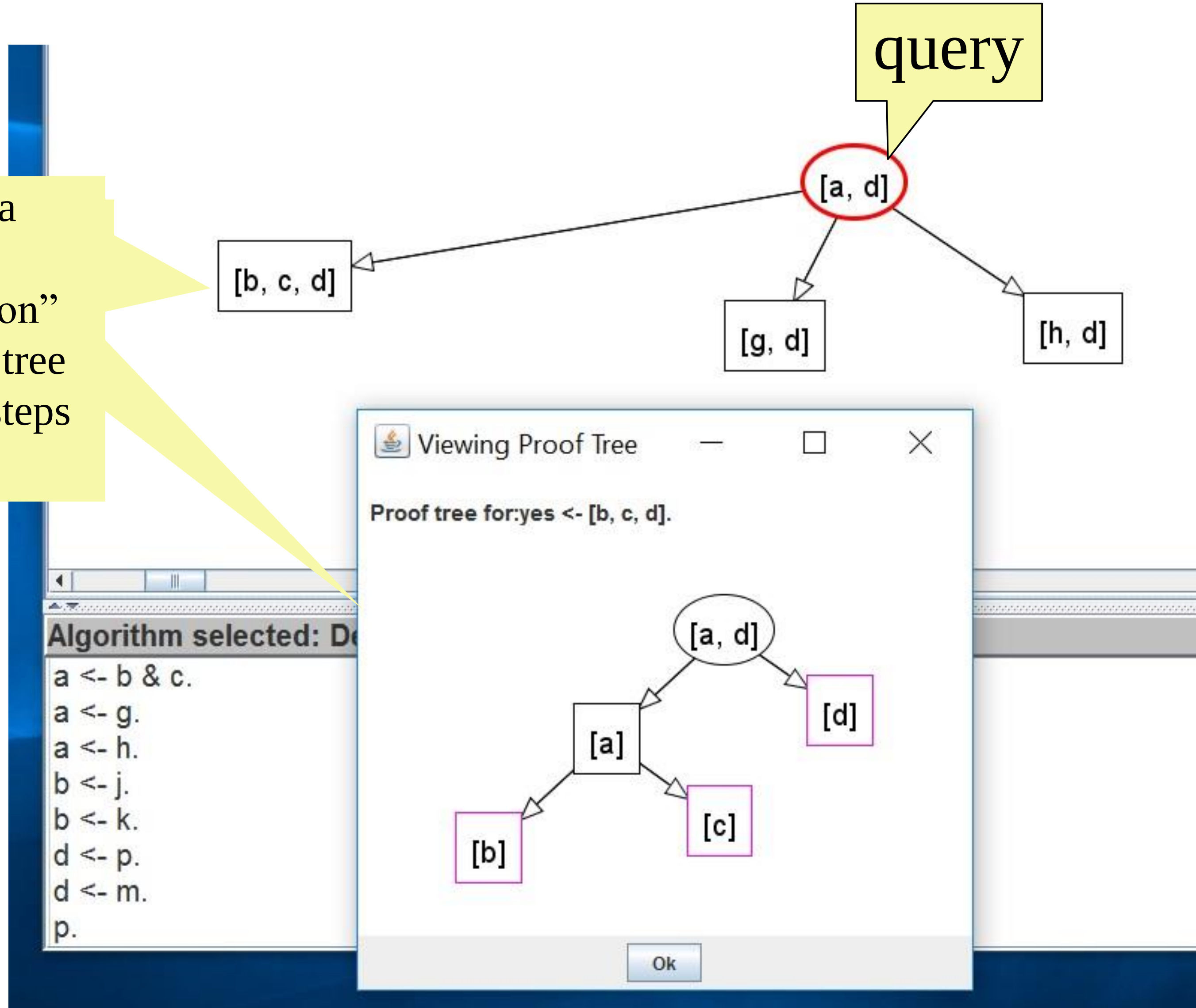
Query creationpanel

Knowledge Base, which can be edited by switching to “create” mode”



Applet

By rightclicking on a node and selecting "view proof deduction" theapplet shows the tree with the resolution steps that led to that node



Top-down/SLD resolution as Search

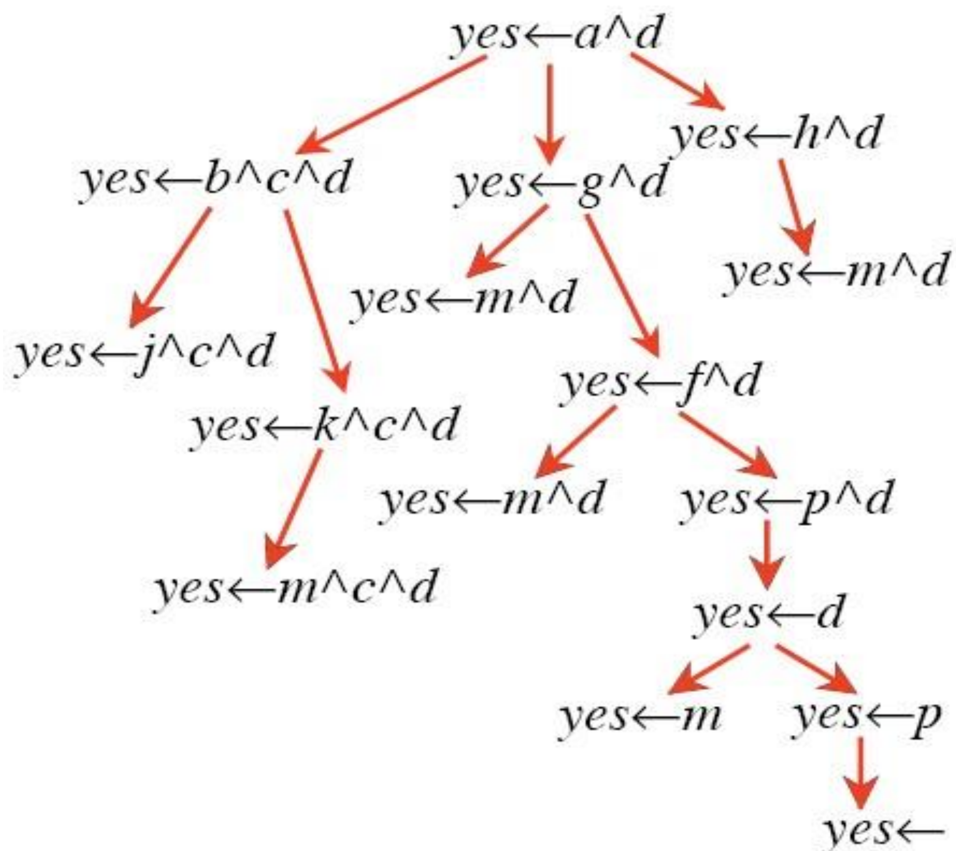
State: answer clause of the form $yes \leftarrow a_1 \wedge \ldots \wedge a_k$

Successor function: state resulting from substituting first atom a_1 with $b_1 \wedge \ldots \wedge b_m$ if there is a clause $a_1 \leftarrow b_1 \wedge \ldots \wedge b_m$

Goal test: is the answer clause empty (i.e. $yes \leftarrow$) ?

Solution: the proof, i.e. the sequence of SLD resolutions

Prove: $? \leftarrow a \wedge d.$



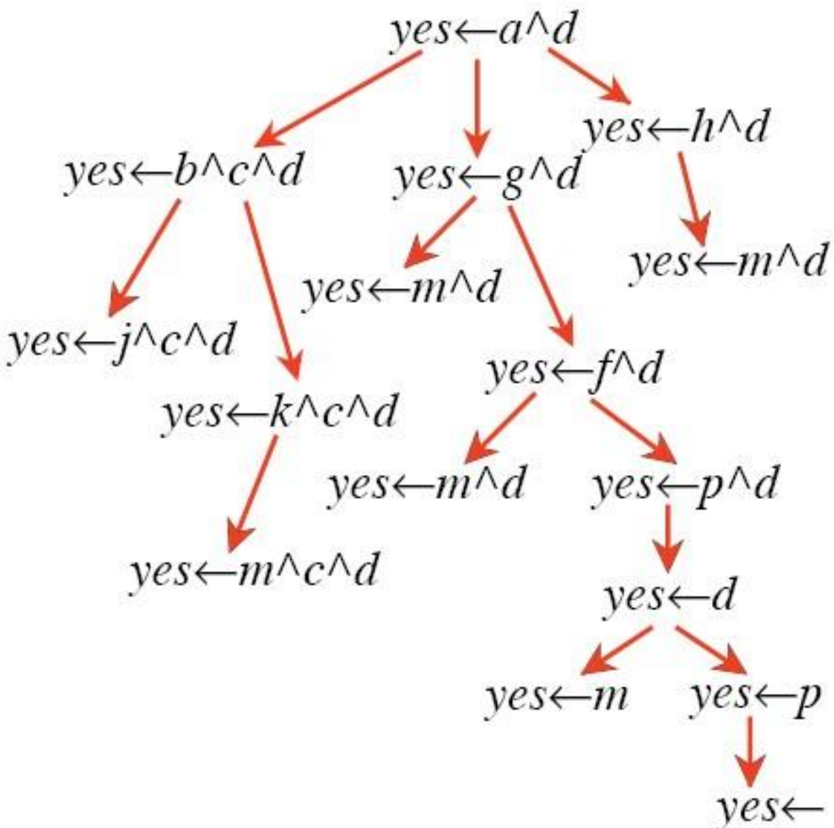
$a \leftarrow b \wedge c.$ $a \leftarrow g.$ $a \leftarrow h.$ $b \leftarrow j.$ $b \leftarrow k.$ $d \leftarrow m.$ $d \leftarrow p.$ $f \leftarrow m.$ $f \leftarrow p.$ $g \leftarrow m.$ $g \leftarrow f.$ $k \leftarrow m.$ $h \leftarrow m.$ $p.$

Possible Heuristic?

Search Graph **KB**

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

Prove: $? \leftarrow a \wedge d.$



Possible Heuristic?

Number of atoms in the answer clause

Admissible?

A. Yes

It takes at least that many steps to reduce all Atoms in the body of the answer clause

Lecture Overview

- Recap of Lecture 16
- TD: soundness and completeness
- SLD Resolution in Datalog

- Intro to Reasoning Under Uncertainty
- Introduction to Probability
    - ✓Random Variables and Possible World Semantics
    - ✓Probability Distributions

Is Top Down Sound and Complete?

- When you have derived an answer with TD, you can find a corresponding BU a proof in the opposite direction.
- try this one γ_0: yes ← a γ_1: yes ← e ∧ f γ_2: yes ← e ∧ c

  γ_3: yes ← c γ_4: yes ← e γ_5: yes ←

- When you have derived an answer with TD, you can find a corresponding BU a proof in the opposite direction.

Is Top Down Sound and Complete?

- try this one

γ_0: yes ← a γ_1: yes ← e ∧ f γ_2: yes ← e ∧ c γ_3: yes ← c

γ_4: yes ← e γ_5: yes ←

a ← e ∧ f

f ← c

c

←

e

e

Definite clauses that generated this derivation

BU applied

Is Top Down Sound and Complete?

to
the
se
cla
us

es give
the BU
derivatio
n for a

- When you have derived an answer, you can find a corresponding BU proof in the opposite direction.

Is Top Down Sound and Complete

- Every top-down derivation corresponds to a bottom up proof and every bottom up proof has a top-down derivation
- try with example in next slide

Bottom-up vs TD proof

KB $z \leftarrow f \wedge e$

$q \leftarrow r \wedge \quad \wedge g$

e $e \leftarrow a \wedge b$ a

b Is q a logical consequence?

Find a corresponding TD r proof

g

{a}

{a,b}

{a, b, r}

C := {};

repeat

select clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ in KB

such that $b_i \in C$ for all i, and $h \notin C$;

$C := C \cup \{h\}$ until no more clauses can be selected.

$\{a, b, r, g\}$

$\{a, b, r, g, e\}$

$\{a, b, r, g, e, q\}$

Bottom-up vs TD proof

KB $z \leftarrow f \wedge e$

$q \leftarrow r \wedge \quad \wedge g$

e $e \leftarrow a \wedge b$

a

> C := {};
>
> repeat
>
> select clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ in KB
>
> such that $b_i \in C$ for all i, and $h \notin C$;
>
> $C := C \cup \{h\}$ until no more
>
> clauses can be selected.

Is q a logical consequence? Find a corresponding TD b

r

g

proof γ_0: yes $\leftarrow$ q

{a} γ_1: yes $\leftarrow$ r $\wedge$ g $\wedge$ e

{a,b}

{a, b, r} γ_2: yes $\leftarrow$ g $\wedge$ e

{a, b, r, g} γ_3: yes $\leftarrow$ e

{a, b, r, g, e} γ_4: yes $\leftarrow$ a $\wedge$ b

{a, b, r, g, e, q}

γ_5: yes $\leftarrow$ b

γ_6: yes $\leftarrow$

Is Top Down Sound and Complete?

- Every top-down derivation corresponds to a bottom up proof and every bottom up proof has a top-down derivation
- This equivalence can be used to prove the soundness and completeness of the derivation procedure.

Try to Show

- That TD is sounds $\quad$ If KB $\vdash_{TD}$ G then KB |=G
- That TD is complete $\quad$ If KB |=G then KB $\vdash_{TD}$ G

Using the fact that every top-down derivation corresponds to a bottom up proof and every bottom up proof has a top-down derivation

Sound

If KB $\vdash_{TD}$ G then KB |=G

- If KB $\vdash_{TD}$ G there is a top down derivation for G
- Therefore there is a bottom up derivation of G.
- Therefore G is in C and KB |=G (because BU is sound)

Complete

If KB $\models$ G then KB $\vdash_{TD}$ G

- If KB $\models$ G then we can find a bottom-up derivation for G (because BU is complete)

- Therefore there is a top-down derivation as well, which we can find because the search space is finite.

- Therefore KB $\vdash_{TD}$ G

- Expressing knowledge with

- What we need is a more natural way 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)

propositions can be quite limiting

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

E.g. there is no notion that

Now there is a notion that

Representation and Reasoning in complex domains

w_1 is the same in live_w_1
and in connected_w_1_w_2

and in connected(w_1, w_2)

up is the same in up(s_1)
and up(s_3)

up_s_1 and up_s_3 are about the
same property

w_1 is the same in live(w_1)

Lecture Overview

- Recap of Lecture 16
- TD: soundness and completeness

- SLD Resolution in Datalog
- Intro to Reasoning Under Uncertainty
- Introduction to Probability
  - ✓Random Variables and Possible World Semantics
  - ✓Probability Distributions

Datalog

- An extension of propositional definite clause (PDC) logic
- We now have constants and variables
- We now have relationships between those
- We can express knowledge that holds for a set of individuals, writing more powerful clauses by introducing variables, such as:

$$live(W) \leftarrow wire(W) \wedge connected\_to(W,W_1) \wedge wire(W_1) \wedge live(W_1).$$

- We can ask generic queries,
  - ✓E.g. “which wires are connected to w_1?“

$$? \; connected\_to(W, w_1)$$

Datalog: a relational rule language

Datalog expands the syntax of PDCL….

A variable is a symbol starting with an upper case letter

Examples: X, Y

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

Examples: alan, w1

A term is either a variable or a constant.

Examples: X, Y, alan, w1

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

Examples: live, connected, part-of, in

Datalog Syntax (cont'd)

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

Examples: sunny, in(alan,X)

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

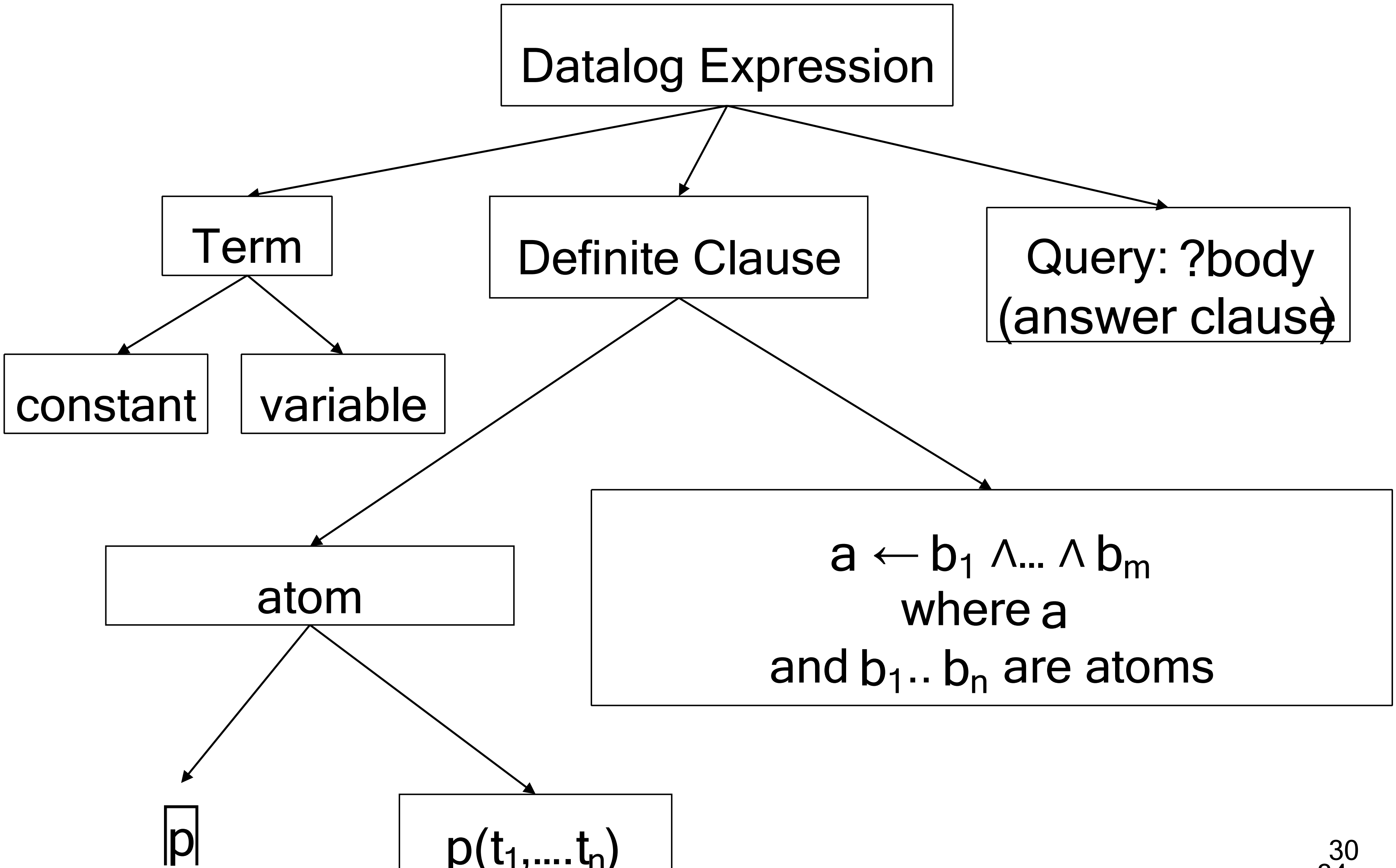
$$h \leftarrow b_1 \wedge \ldots \wedge b_m$$

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

Example: in(X,Z) ← in(X,Y) ∧ part-of(Y,Z)

A knowledge base is a set of definite clauses

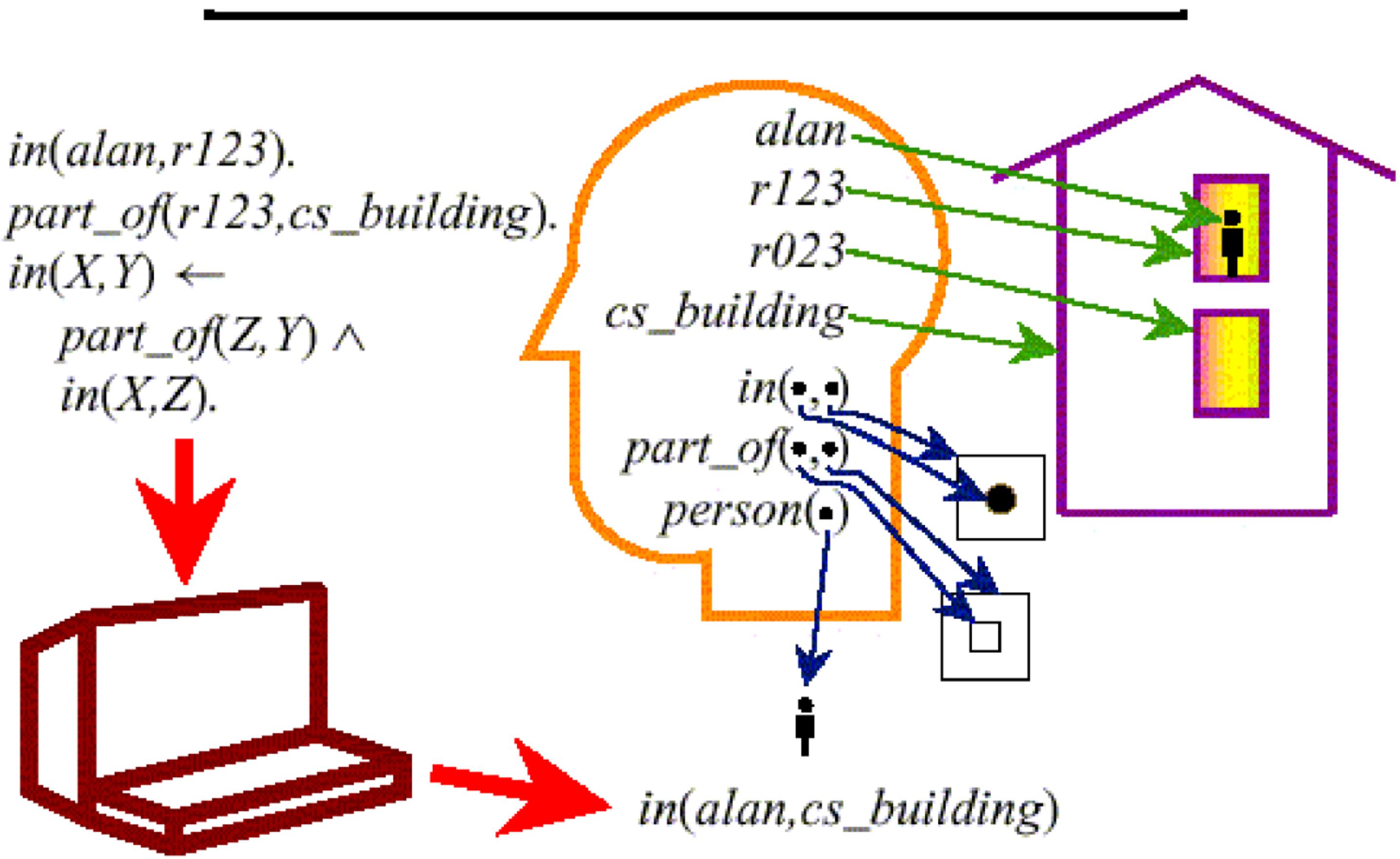
Summary of Datalog Syntax



Datalog Expression
Term
Definite Clause
Query: ?body
(answer clause)
constant
variable
atom
a ← b1 ∧… ∧ bm
where a
and b1.. bn are atoms
p
p(t1,….tn)

DataLog Sematics

- Role of semantics is still to connect symbols and sentences in the language with the target domain. Main difference:



- need to create

correspondence both between terms and individuals, as well as between predicate symbols and relations

We won't cover the formal definition of Datalog semantics, but if you are interested see 12.3.1 and 12.3.2 in textbook 31

Datalog: Top Down Proof Procedure

in(alan, r123). part_of(r123,cs_building).

in(X,Y) ← part_of(Z,Y) & in(X,Z).

• Extension of Top-Down procedure for PDCL.

How do we deal with variables? Idea:

- Find a clause with head that matches the query
- Substitute variables in the clause with their matching constants

• Example:

Query: yes ← in(alan, cs_building).

in(X,Y) ← part_of(Z,Y) & in(X,Z).

with Y = cs_building

X = alan

$$yes \leftarrow part\_of(Z,cs\_building), in(alan, Z).$$

- We will not cover the formal details of this process, called unification.
  See textbook Section 12.4.2, p. 511 for the details.

Example proof of a Datalog query

in(alan, r123).
part_of(r123,cs_building).
in(X,Y) ← part_of(Z,Y) ∧ in(X,Z).

Query: yes ← in(alan, cs_building).

Using clause: in(X,Y) ← part_of(Z,Y) ∧ in(X,Z),
with Y = cs_building
X = alan

yes ← part_of(Z,cs_building) ∧ in(alan, Z).

Using clause: part_of(r123,cs_building) with Z = r123

??????

A. yes ← part_of(Z, r123)∧ in(alan, Z).

B. yes ← in(alan, r123).

C. yes ←.

D. None of the above

Example proof of a Datalog query

in(alan, r123).
part_of(r123,cs_building).
in(X,Y) ← part_of(Z,Y) ∧ in(X,Z).

Query: yes ← in(alan, cs_building).

Using clause: in(X,Y) ← part_of(Z,Y) ∧ in(X,Z), with Y = cs_building X = alan

yes ← part_of(Z,cs_building) ∧ in(alan, Z).

Using clause: part_of(r123,cs_building)

?????? with Z = r123 B. yes ← in(alan, r123).

in(alan, r123). part_of(r123,cs_building).
in(X,Y) ← part_of(Z,Y) & in(X,Z).

Example proof of a Datalog query

Query: yes ← in(alan, cs_building).

Using clause: in(X,Y) ← part_of(Z,Y) & in(X,Z), with Y = cs_building X = alan

yes ← part_of(Z,cs_building), in(alan, Z).

Using clause: part_of(r123,cs_building) with Z = r123

yes ← in(alan, r123).

Using clause: in(alan, r123).

Using clause: in(X,Y) ← part_of(Z,Y) & in(X,Z). With X = alan and Y = r123

yes ← part_of(Z, r123), in(alan, Z).

No clause with matching head: part_of(Z,r123).

fail

Example proof of a Datalog query

yes $\leftarrow$.

Tracing Datalog proofs in AIspace

- You can trace the example from the last slide in the AIspace Deduction Applet at http://aispace.org/deduction/ using file in-part-of available in course schedule

Datalog: queries with variables

in(alan, r123). part_of(r123,cs_building).

in(X,Y) ← part_of(Z,Y) & in(X,Z).

Query: in(alan, X1).

yes(X1) ← in(alan, X1).

What would the answer(s) be?

Datalog: queries with variables

in(alan, r123). part_of(r123,cs_building).

in(X,Y) ← part_of(Z,Y) & in(X,Z).

Query: in(alan, X1).

yes(X1) ← in(alan, X1).

What would the answer(s) be?
yes(r123). yes(cs_building).

Again, you can trace the SLD derivation for this query in the AIspace Deduction Applet,

Learning Goals For Logic

- PDCL syntax & semantics - Verify whether a logical statement belongs to the language of propositional definite clauses
  - Verify whether an interpretation is a model of a PDCL KB.
  - Verify when a conjunction of atoms is a logical consequence of a KB
- Bottom-up proof procedure
  - Define/read/write/trace/debug the Bottom Up (BU) proof procedure

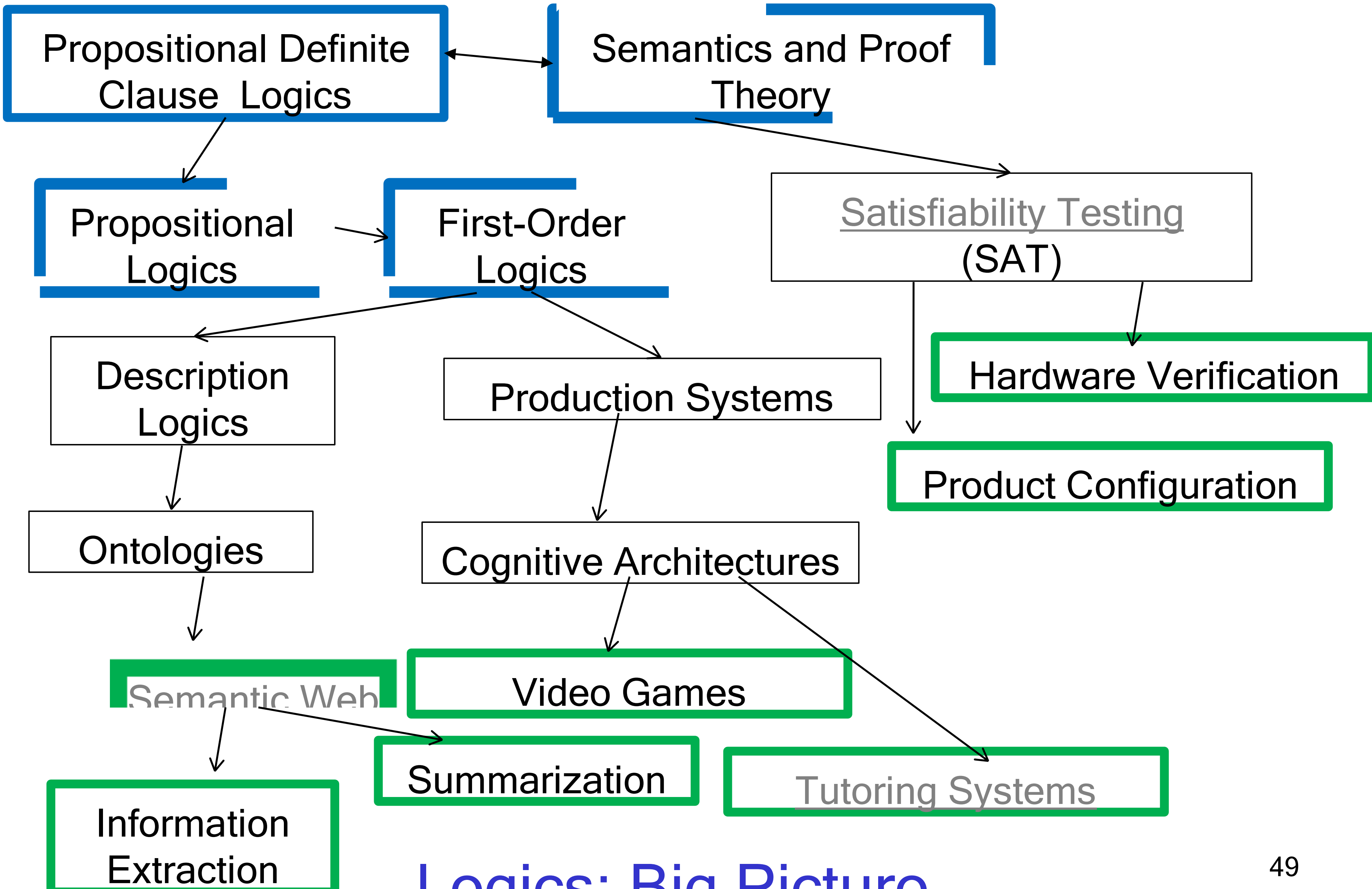
- Prove that the BU proof procedure is sound and complete

- Top-down proof procedure
  - Define/read/write/trace/debug the Top-down (SLD) proof procedure
  - Define it as a search problem
  - Prove that the TD proof procedure is sound and complete
- Datalog
  - Represent simple domains in Datalog
  - Apply the Top-down proof procedure in Datalog

Logics: Big picture

- We only covered rather simple logics • There are much more powerful representation and reasoning systems based on logics e.g.
  - ✓full first order logic (with negation, disjunction and function symbols)
  - ✓second-order logics (predicates over predicates)
  - ✓non-monotonic logics, modal logics, …
- There are many important applications of logic • For example, software agents roaming the web on our behalf

- ✓Based on a more structured representation: the semantic web
- ✓This is just one example for how logics are used



Logics: Big Picture

Semantic Web: Extracting data

- Examples for typical \Web queries
- How much is a typical flight to Mexico for a given date?
- What's the cheapest vacation package to some place in the Caribbean in a given week?
  - ✓Plus, the hotel should have a white sandy beach and scuba diving
- If webpages are based on basic HTML
- Humans need to scout for the information and integrate it • Computers are not reliable enough (yet)
  - ✓Natural language processing (NLP) can be powerful (see Watson and Siri!)

✓But some information may be in pictures (beach), or implicit in the text, so existing NLP techniques still don't get all the info.

Semantic Web

- Languages and formalisms based on descriptionlogics that allow websites to include rich, explicit information on
- relevant concepts, individual and their relationships
- Goal: software agents that can roam the web and carry out sophisticated tasks on our behalf, based on these richer representations
- Different than searching content for keywords and popularity.
- Infer meaning from content based on metadata and assertions that have already been made.

- Automatically classify and integrate information
- For further material, P&M text, Chapter 13. Also
- the Introduction to the Semantic Web tutorial given at 2011 Semantic TechnologyConference

http://www.w3.org/People/Ivan/CorePresentations/SWTutorial/

Examples of ontologies for the Semantic Web

Ontology: logic-based representation of the world

- eClassOwl: eBusiness ontology
- for products and services
- 75,000 classes (types of individuals) and 5,500 properties
- National Cancer Institute's ontology: 58,000 classes
- Open Biomedical Ontologies Foundry: several ontologies
- including the Gene Ontology to describe
  - ✓gene and gene product attributes in any organism or protein sequence
- OpenCyc project: a 150,000-concept ontology including
- Top-level ontology

✓describes general concepts such as numbers, time, space, etc

- Many specific concepts such as “OLED display”, “iPhone”

See more examples at https://www.w3.org/2001/sw/sweo/public/UseCases/[4]

[4]

A different example of applications of logic

Cognitive Tutors (http://pact.cs.cmu.edu/)

- computer tutors for a variety of domains (math, geometry, programming, etc.)

Markov Processes

Value

Iteration

- Provide individualized support to problem solving exercises, as good human tutors do
- Rely on logic-based, detailed computational models (ACT-R) of skills and misconceptions underlying a learning domain.
- CarnegieLearning

(http://www.carnegielearning.com/):

- a company that commercializes these tutors, sold to hundreds of thousands of high schools in the USA

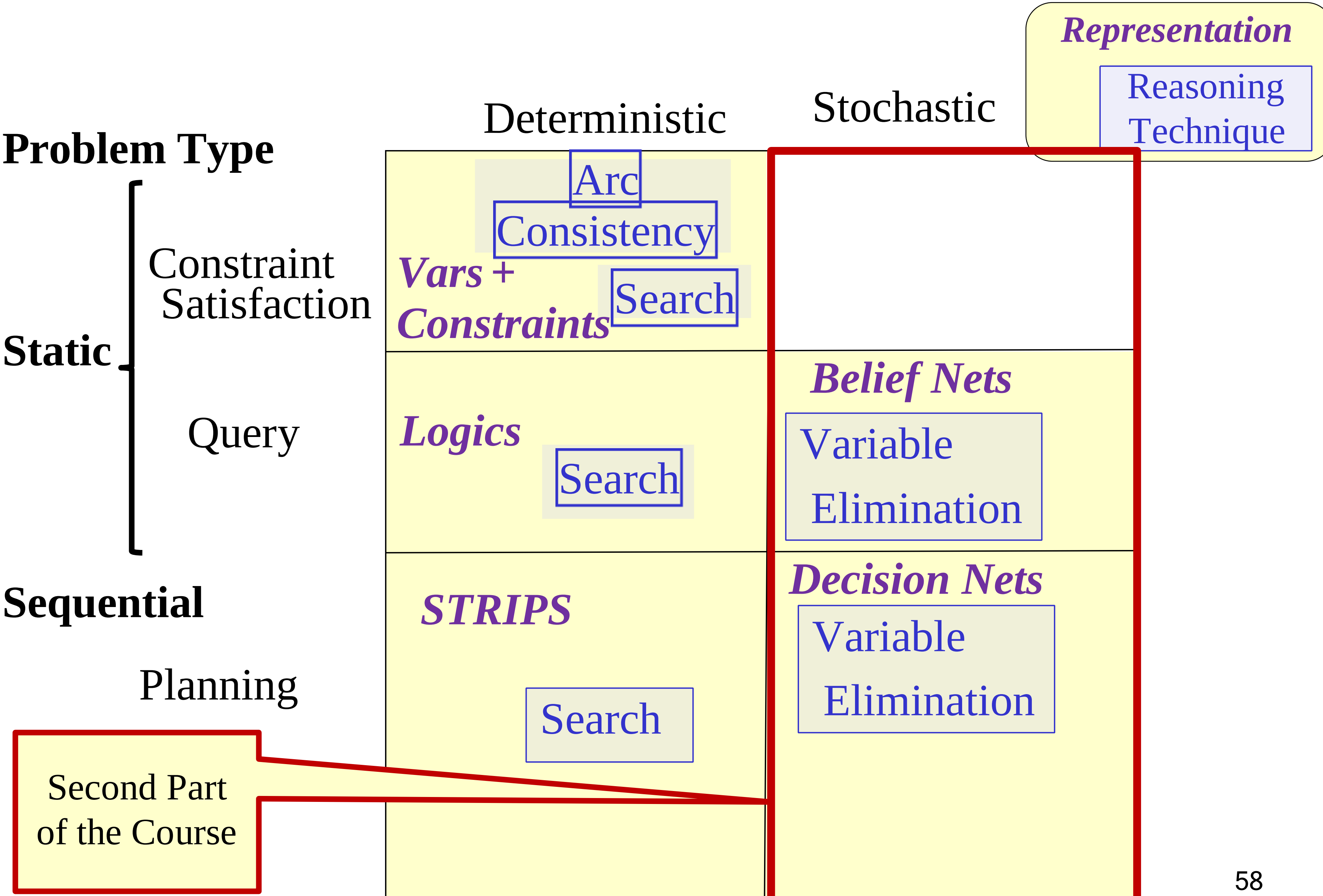
Representation

Reasoning Technique

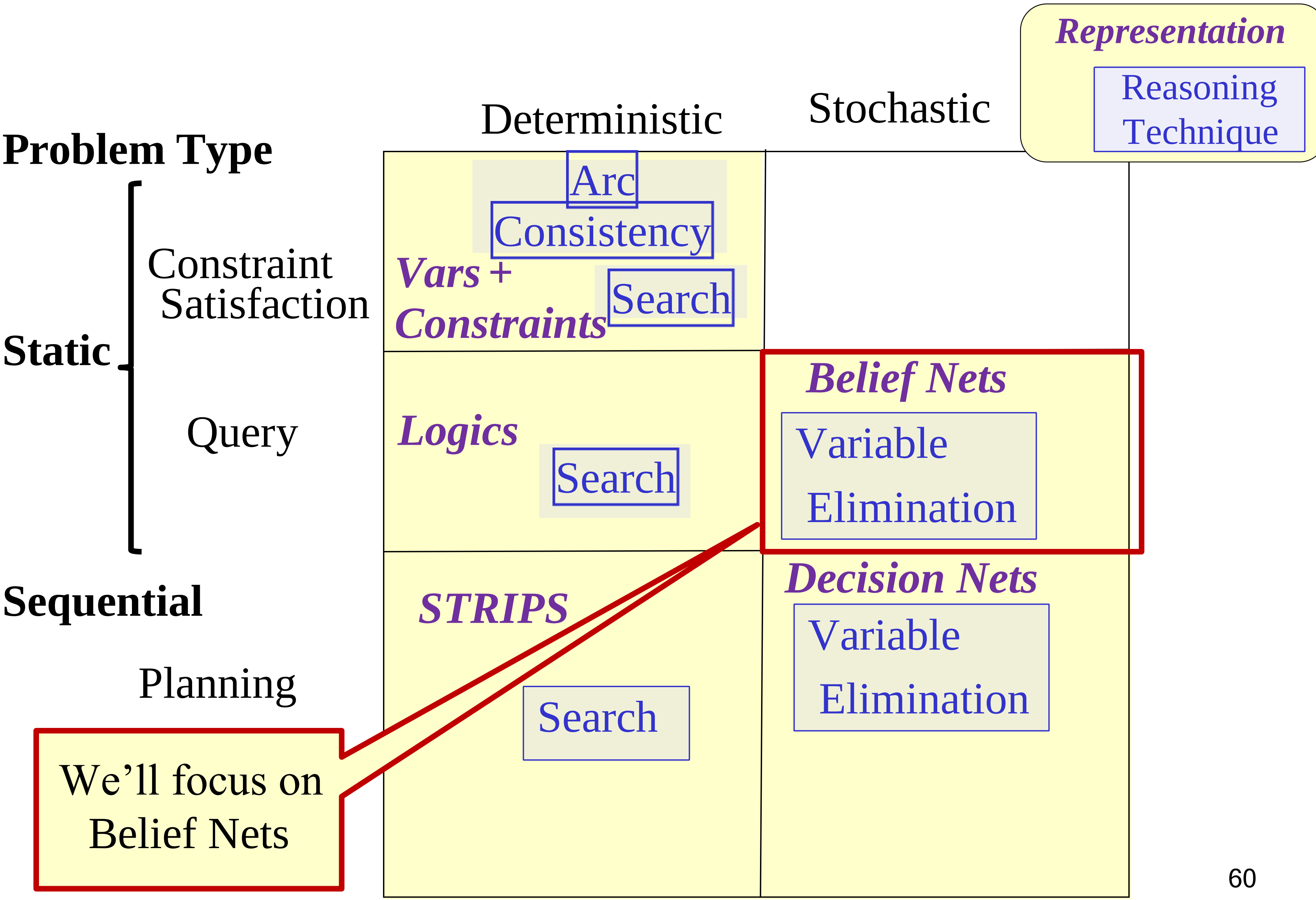
Problem Type		Deterministic	Stochastic
Static	Constraint Satisfaction	***Vars + Constraints*** Arc Consistency Search	
	Query	***Logics*** Search	***Belief Nets*** Variable Elimination
Sequential	Planning	***STRIPS*** Search	***Decision Nets*** Variable Elimination

Done with Deterministic Environments

Environment


Representation
Reasoning Technique
Deterministic
Stochastic
Problem Type
Static
Constraint Satisfaction
Query
Sequential
Planning
Vars + Constraints
Arc Consistency
Search
Logics
Search
STRIPS
Search
Belief Nets
Variable Elimination
Decision Nets
Variable Elimination
Second Part of the Course

Environment


Representation
Reasoning Technique
Deterministic
Stochastic
Problem Type
Static
Constraint Satisfaction
Vars + Constraints
Arc Consistency
Search
Query
Logics
Search
Belief Nets
Variable Elimination
Sequential
Planning
STRIPS
Search
Decision Nets
Variable Elimination
We'll focus on Belief Nets

Environment

Lecture Overview

- Recap of Lecture 16
- TD: soundness and completeness
- SLD Resolution in Datalog
- Intro to Reasoning Under Uncertainty
- Introduction to Probability

Markov Processes

Value Iteration

✓Random Variables and Possible World Semantics

✓Probability Distributions

Two main sources of uncertainty

(From Lecture 2)

- Sensing Uncertainty: The agent cannot fully observe a state of interest.

For example:

- Right now, how many people are in this building?
- What disease does this patient have?
- Where is the soccer player behind me?

- Effect Uncertainty: The agent cannot be certain about the effects of its actions.

For example:

- If I work hard, will I get an A?

- Will this drug work for this patient?
- Where will the ball go when I kick it?

Motivation for uncertainty

- To act in the real world, we almost always have to handle uncertainty (both effect and sensing uncertainty)
- Deterministic domains are an abstraction
  - ✓Sometimes this abstraction enables more powerful inference
- Now we don’t always make this abstraction anymore
- AI main focus shifted from logic to probability in the 1980s
- The language of probability is very expressive and general • New representations enable efficient reasoning

- ✓We will see some of these, in particular Bayesian networks

- Reasoning under uncertainty is part of the ‘new’ AI
  - ✓ This is not a dichotomy: framework for probability is logical!

- New frontier: combine logic and probability

Interesting article about AI and uncertainty

- “The machine age” , by Peter Norvig (head of research at Google)
  - ✓New York Post, 12 February 2011
  - ✓http://www.nypost.com/f/print/news/opinion/opedcolumnists/the_machine_age_tM7xPAv4pI4JsIK0M1JtxI

- “The things we thought were hard turned out to be easier.”
  - ✓Playing grandmaster level chess, or proving theorems in integral calculus

- “Tasks that we at first thought were easy turned out to be hard.”

- ✓A toddler (or a dog) can distinguish hundreds of objects (ball, bottle, blanket, mother, …) just by glancing at them
- ✓Still difficult for computer vision to perform at this level

- "Dealing with uncertainty turned out to be more important than thinking with logical precision."
  - ✓Reasoning under uncertainty (and lots of data) are key to progress