Logic: Datalog wrap-up

CPSC 322 – Logic 5

Textbook §12.3

March 14, 2011

Lecture Overview

- Invited Presentation:
  - Chris Fawcett on scheduling UBC's exams using SLS
- Recap: Top-down Proof Procedure
- Datalog
- Logics: big picture

Lecture Overview

- Invited Presentation:
  - Chris Fawcett on scheduling UBC's exams using SLS
- Recap: Top-down Proof Procedure
- Datalog
- Logics: big picture

Bottom-up vs. Top-down

- **Key Idea of top-down:** search backward from a query g to determine if it can be derived from *KB*.

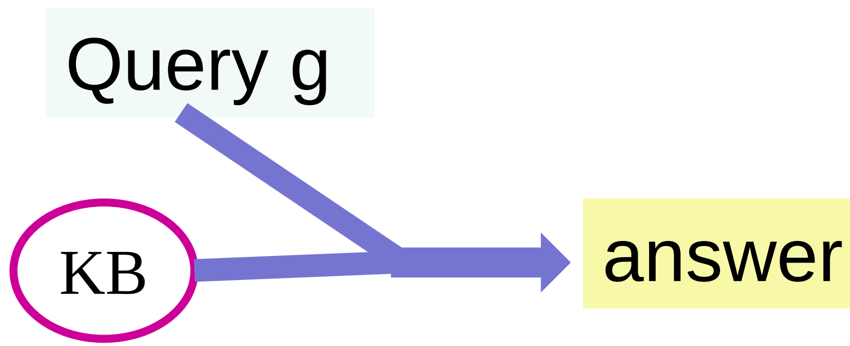
Bottom-up

g is proved if $g \in C$

BU never looks at the query g

- It derives the same C regardless of the query

Top-down



TD performs a backward search starting at g

Example for (successful) SLD derivation

$a \leftarrow b \wedge c.$	1 $\boxed{a \leftarrow e \wedge f.}$	$b \leftarrow f \wedge k.$
$c \leftarrow e.$	$d \leftarrow k$	3 $\boxed{e.}$
$f \leftarrow j \wedge e.$	2 $\boxed{f.}$	$j \leftarrow c.$

Query: ?a

γ_0: yes $\leftarrow$ a

γ_1: yes $\leftarrow$ e $\wedge$ f

γ_2: yes $\leftarrow$ e

γ_3: yes $\leftarrow$

Done. “Can we derive a?”

- Answer:“Yes, we can”

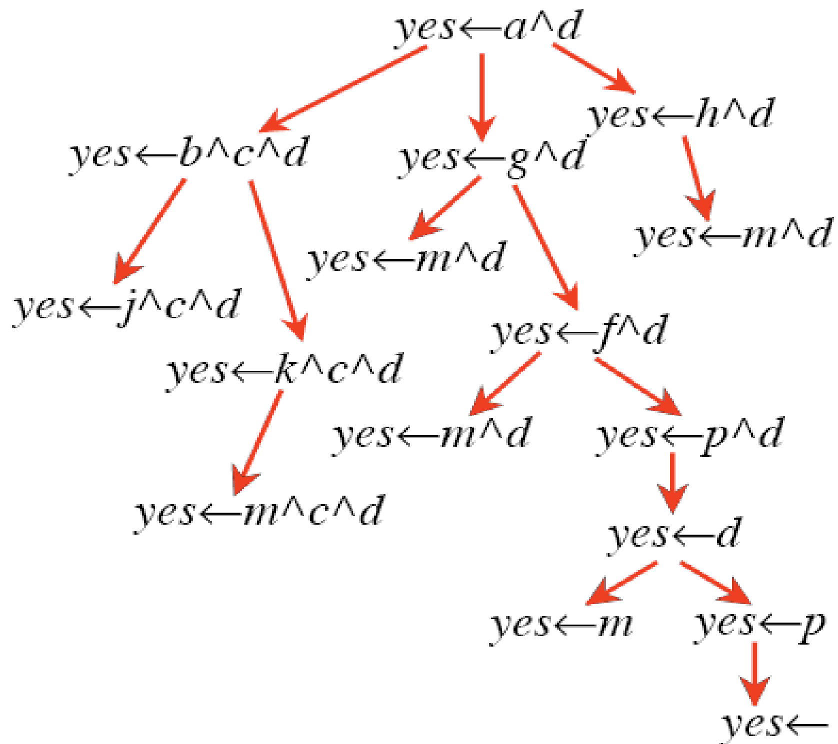
Correspondence between BU and TD proofs

If the following is a top-down (TD) derivation in a given KB, what would be the bottom-up (BU) derivation of the same query?

TD derivation	Part of KB:	BU derivation
$yes \leftarrow a.$	$a \leftarrow b \wedge f$	{}
$yes \leftarrow b \wedge f.$	$f \leftarrow g \wedge h$	{h}
$yes \leftarrow b \wedge g \wedge h.$	$b \leftarrow c \wedge d$	{g,h}
$yes \leftarrow c \wedge d \wedge g \wedge h.$	c.	{d,g,h}
$yes \leftarrow d \wedge g \wedge h.$	d.	{c,d,g,h}
$yes \leftarrow g \wedge h.$	h.	{b,c,d,g,h}
$yes \leftarrow h.$	g.	{b,c,d,f,g,h}
$yes \leftarrow .$		{a,b,c,d,f,g,h}

Inference as Standard Search

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



- Inference (Top-down/SLD resolution)
  - State: answer clause of the form $yes \leftarrow q_1 \wedge \ldots \wedge q_k$
  - Successor function: all states resulting from substituting first atom a with $b_1 \wedge \ldots \wedge b_m$ if there is a clause $a \leftarrow b_1 \wedge \ldots \wedge b_m$
  - Goal test: is the answer clause empty (i.e. $yes \leftarrow$) ?
  - Solution: the proof, i.e. the sequence of SLD resolutions
  - Heuristic function: number of atoms in the query clause

Lecture Overview

- Invited Presentation:
  - Chris Fawcett on scheduling UBC's exams using SLS
- Recap: Top-down Proof Procedure
- Datalog
- Logics: big picture

Datalog

- An extension of propositional definite clause (PDC) logic
  - We now have variables
  - We now have relationships between variables

  - We can write more powerful clauses, such as

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

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

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

Datalog syntax

Datalog expands the syntax of PDCL….

A variable is a symbol starting with an upper case letter

Examples: X, W_1

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 (continued)

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

Examples: sunny, in(alan,X)

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

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

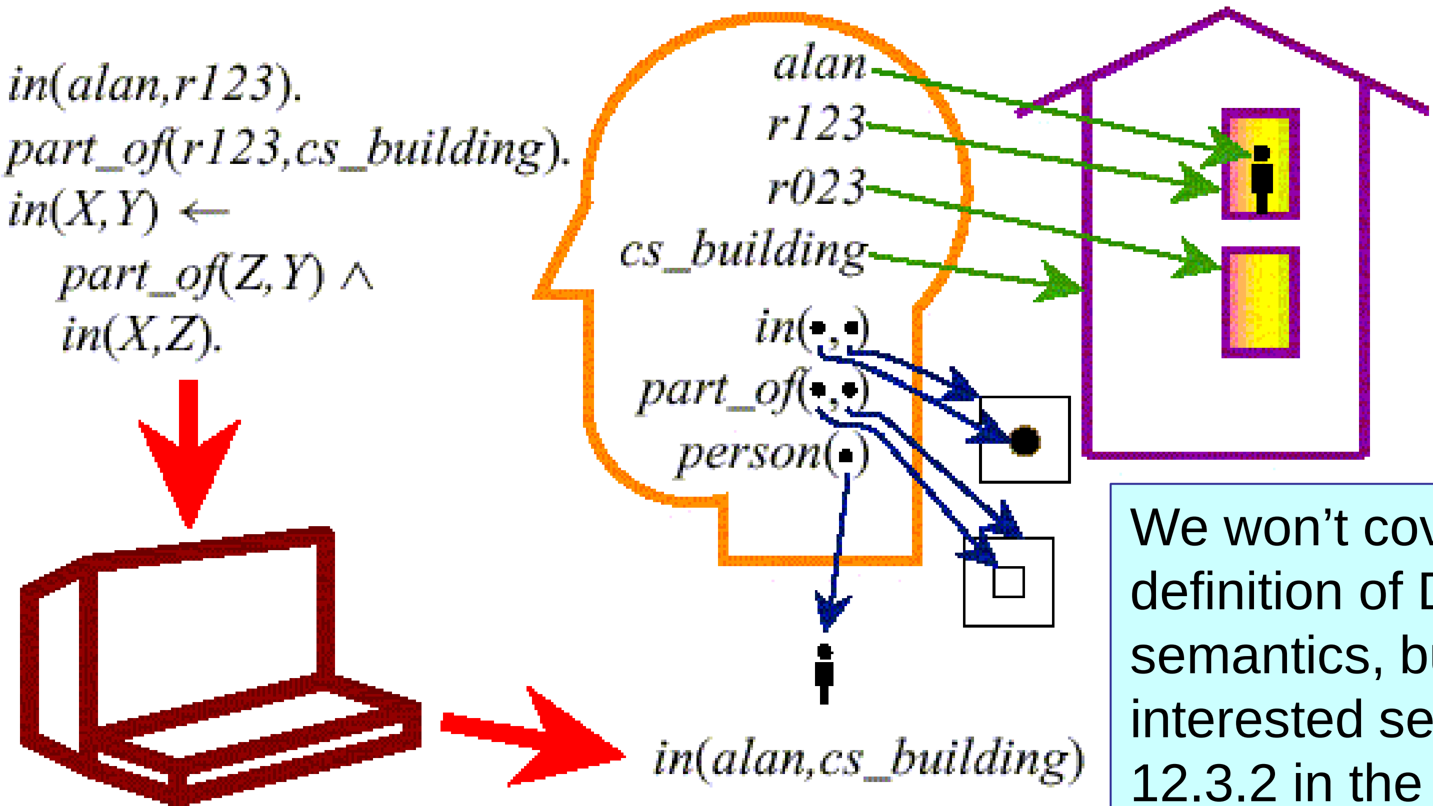
where h and the b_i are atoms (Read this as ``h if b.")

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

A knowledge base is a set of definite clauses

Datalog Semantics

- Semantics still 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 the textbook

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

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).

yes ←.

yes ← part_of(Z, r123), in(alan, Z).

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

fail

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 clauses with heads that match the query
    - Substitute variable in the clause with the matching constant
  - Example:

Query: yes ← in(alan, cs_building).

in(X,Y) with Y = cs_building

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

- We will not cover the formal details of this process (called ***unification***)

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

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 Z = alan

yes ←.

yes ← part_of(Z, r123), in(alan, Z).

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

fail

One important Datalog detail

- In its SLD resolution proof, Datalog always chooses the first clause with a matching head it finds in KB
- What does that mean for recursive function definitions?

You cannot have recursive definitions

You need tail recursion

The clause(s) defining your base case(s) have to appear first in KB

One important Datalog detail

- In its SLD resolution proof, Datalog always chooses the first clause with a matching head it finds in KB
- What does that mean for recursive function definitions?
  - The clause(s) defining your base case(s) have to appear first in KB
  - Otherwise, you can get infinite recursions
  - This is similar to recursion in imperative programming languages

Tracing Datalog proofs in AIspace

- You can trace the example from the last slide in the AIspace Deduction Applet, using file http://cs.ubc.ca/~hutter/teaching/cpsc322/in-part-of.pl

- Question 4 of assignment 3 asks you to use this applet

Datalog: queries with variables

in(alan, r123).

part_of(r123,cs_building).

in(X,Y) $\leftarrow$ part_of(Z,Y) & in(X,Z).

Query: in(alan, X1).

Yes(X1) $\leftarrow$ 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).

You can trace the SLD derivation for this query in the AIspace Deduction Applet, using file http://cs.ubc.ca/~hutter/teaching/cpsc322/in-part-of.pl

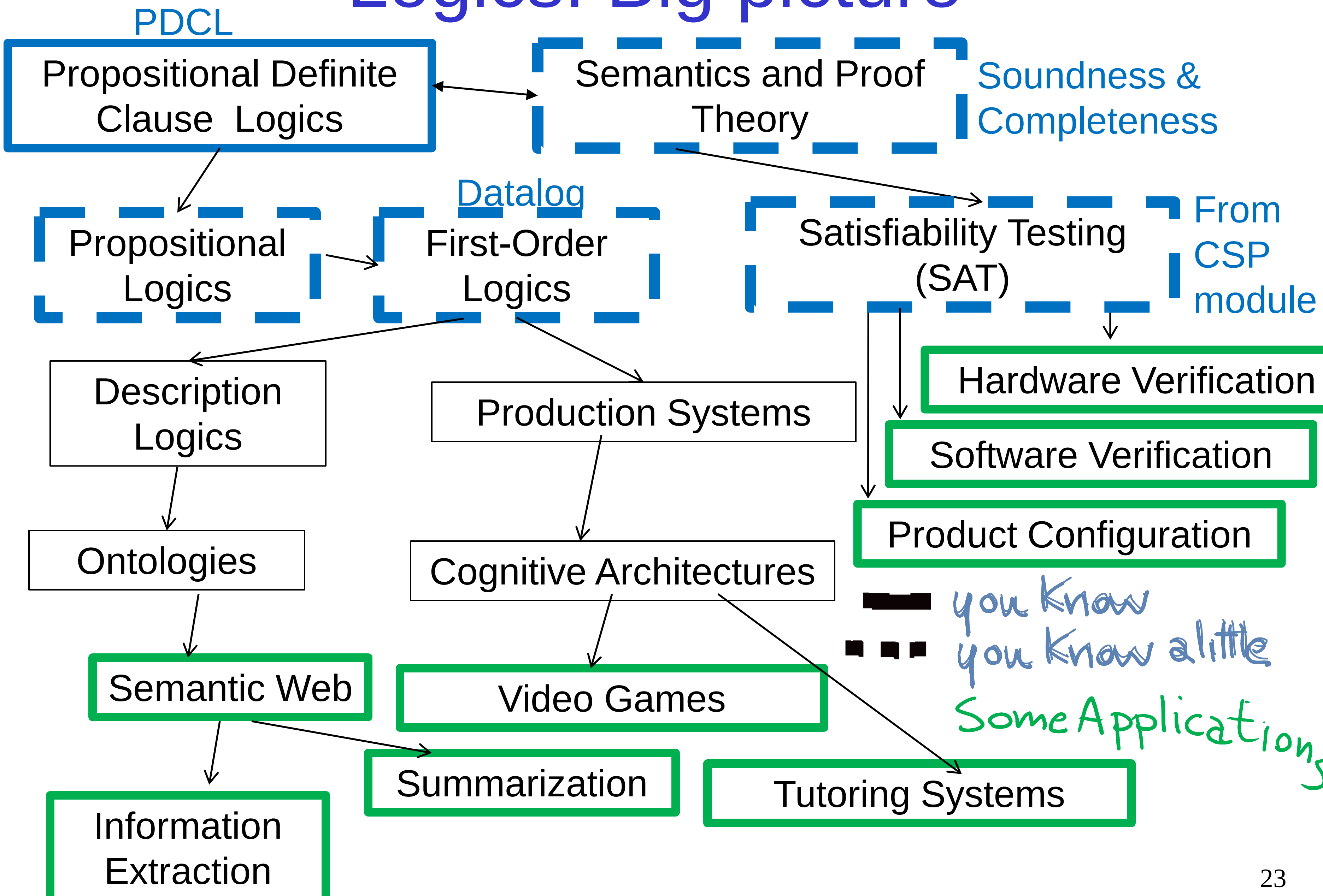
Learning Goals For Logic

- 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 (as a search problem)
- Datalog
  - Represent simple domains in Datalog
  - Apply the Top-down proof procedure in Datalog

Lecture Overview

- Invited Presentation:
  - Chris Fawcett on scheduling UBC's exams using SLS
- Recap: Top-down Proof Procedure
- Datalog
- Logics: big picture

Logics: Big picture



Logics: Big picture

- We only covered rather simple logics
  - There are much more powerful representation and reasoning systems based on 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

Example problem: automated travel agent

- Examples for typical 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 can be powerful (see Watson!)
    - But some information may be in pictures (beach), or implicit in the text, so simple approaches like Watson still don’t get

More structured representation: the Semantic Web

- Beyond HTML pages only made for humans
- Languages and formalisms based on logics that allow websites to include information in a more structured format
  - Goal: software agents that can roam the web and carry out sophisticated tasks on our behalf.
  - This is different than searching content for keywords and popularity!

- For further references, see, e.g. tutorial given at 2009 Semantic Technology Conference:

  http://www.w3.org/2009/Talks/0615-SanJose-tutorial-IH

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
    - annotation terminology and data
- OpenCyc project: a 150,000-concept ontology including
  - Top-level ontology
    - describes general concepts such as numbers, time, space, etc
  - Hierarchical composition: superclasses and subclasses
  - Many specific concepts such as "OLED display", "iPhone"

Course Overview

Course Module

Representation
Reasoning Technique

Environment

Problem Type

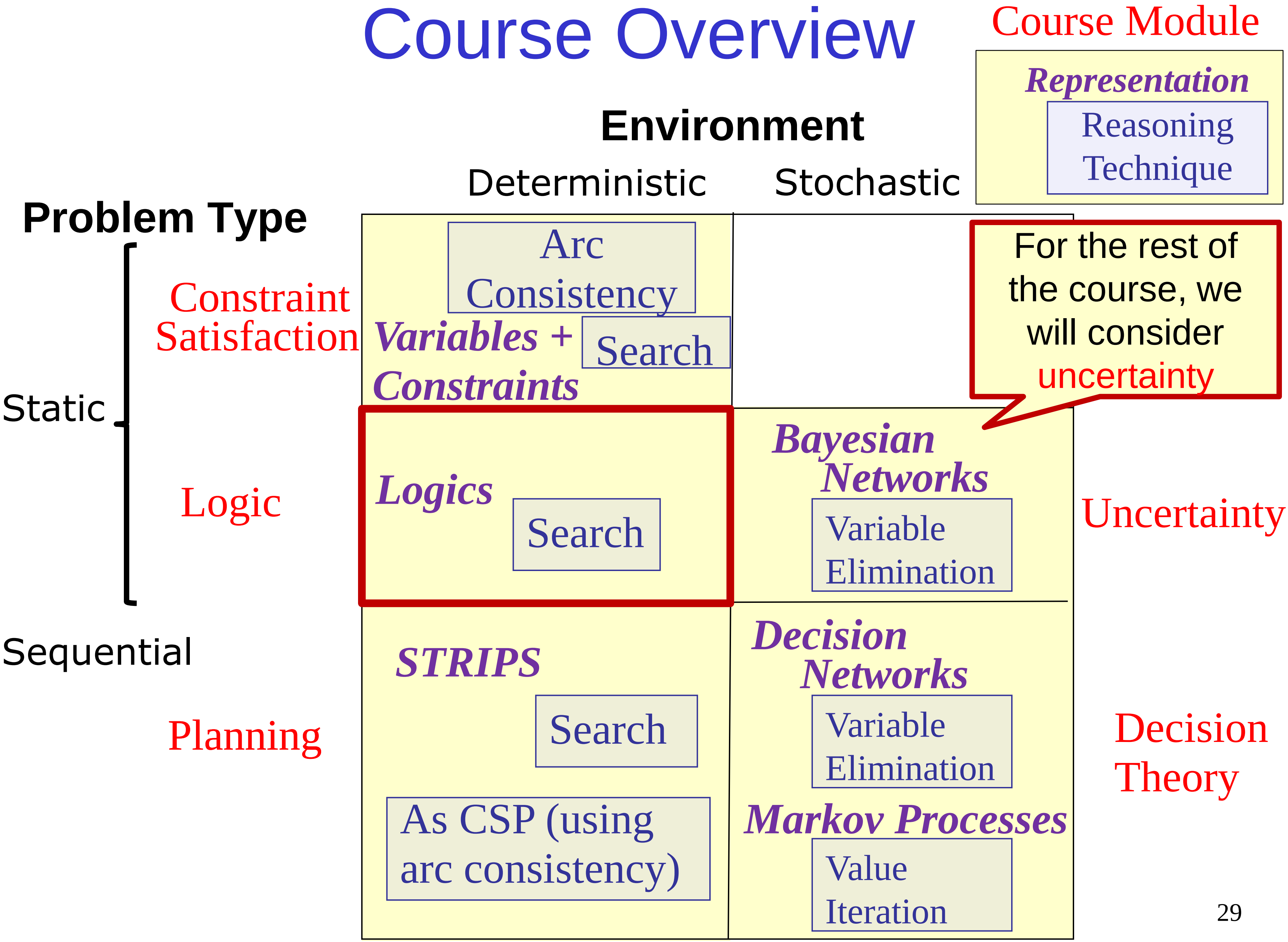
		Deterministic	Stochastic
Static	Constraint Satisfaction	***Variables + Constraints*** — Arc Consistency; Search	
Static	Logic	***Logics*** — Search	***Bayesian Networks*** — Variable Elimination
Sequential	Planning	***STRIPS*** — Search; As CSP (using arc consistency)	***Decision Networks*** — Variable Elimination; ***Markov Processes*** — Value Iteration

Uncertainty

Decision Theory

This concludes the logic module

Course Overview



Types of uncertainty (from Lecture 2)

- Sensing Uncertainty:
  - The agent cannot fully observe a state of interest
  - E.g.: Right now, how many people are in this room? In this building?
- Effect Uncertainty:
  - The agent cannot be certain about the effects of its actions
  - E.g.: If I work hard, will I get an A?
- Motivation for uncertainty: in the real world, we almost always have to handle uncertainty (both types)
  - Deterministic domains are an abstraction
    - Sometimes this abstraction enables much more powerful inference
  - Now we don't make this abstraction anymore
    - Our representations and reasoning techniques will now handle uncertainty

More motivation for uncertainty

- Interesting article: "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_tM7xPAv4pI4JslK0M1JtxI
  - "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, etc.) just by glancing at them
    - Very difficult for computer vision to perform at this level
  - "Dealing with uncertainty turned out to be more important than thinking with logical precision."
    - AI's focus shifted from Logic to Probability in the late 1980s

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 (as a search problem)
- Datalog
  - Represent simple domains in Datalog
  - Apply the Top-down proof procedure in Datalog

- Assignment 3 is due on Wednesday
- Posted short answer questions up to logic on WebCT (to be updated)