Logic: Top-down proof procedure and Datalog

CPSC 322 – Logic 4

Textbook §5.2

March 11, 2011

Lecture Overview

- Recap: Bottom-up proof procedure is sound and complete
- Top-down Proof Procedure
- Datalog

Logical consequence and BU proofs

Definition (logical consequence)
If KB is a set of clauses and g is a conjunction of atoms,
g is a logical consequence of KB, written KB ⊧ g,
if g is true in every model of KB

Example: KB = {h ← a, a, a ← c}. Then KB ⊧ ?

	a	c	h	h ← a	a	a ← c	Model of KB
I_1	F	F	F	T	F	T	no
I_2	F	F	T	T	F	T	no
I_3	F	T	F	T	F	F	no
I_4	F	T	T	T	F	F	no
I_5	T	F	F	F	T	T	no
I_6	T	F	T	T	T	T	yes
I_7	T	T	F	F	T	T	no
I_8	T	T	T	T	T	T	yes

Which atoms are entailed?

Logical consequence and BU proofs

Definition (logical consequence)
If KB is a set of clauses and g is a conjunction of atoms,
g is a logical consequence of KB, written KB ⊧ g,
if g is true in every model of KB

Example: KB = {h ← a, a, a ← c}. Then KB ⊧ ?

	a	c	h	h ← a	a	a ← c	Model of KB
I_1	F	F	F	T	F	T	no
I_2	F	F	T	T	F	T	no
I_3	F	T	F	T	F	F	no
I_4	F	T	T	T	F	F	no
I_5	T	F	F	F	T	T	no
I_6	T	F	T	T	T	T	yes
I_7	T	T	F	F	T	T	no
I_8	T	T	T	T	T	T	yes

Which atoms are entailed?

KB ⊧ a and
KB ⊧ h

Logical consequence and BU proofs

Definition (logical consequence)
If KB is a set of clauses and g is a conjunction of atoms,
g is a logical consequence of KB, written KB $\models$ g,
if g is true in every model of KB

Example: KB = $\{h \leftarrow a,\ a,\ a \leftarrow c\}$. Then KB $\models$ a and KB $\models$ h.

BU 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 and only if g $\in$ C

What does BU derive for the KB above?

Logical consequence and BU proofs

Definition (logical consequence)
If KB is a set of clauses and g is a conjunction of atoms,
g is a logical consequence of KB, written KB $\models$ g,
if g is true in every model of KB

Example: KB = $\{h \leftarrow a, a, a \leftarrow c\}$. Then KB $\models$ a and KB $\models$ h.

BU 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 and only if g $\in$ C

What does BU derive for the KB above?
Trace: {a}, {a,h}. Thus KB $\vdash_{BU}$ a and KB $\vdash_{BU}$ h.
Exactly the logical consequences!

Summary for bottom-up proof procedure BU

- Proved last time
  - 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 !
  - That's why the results for $\models$ and $\vdash_{BU}$ matched for the example above
- And, it is quite efficient!
  - Linear in the number of clauses in KB
    - Each clause is used maximally once by BU

Learning Goals Up To Here

- 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 knowledge base

- 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

Lecture Overview

- Recap: Bottom-up proof procedure is sound and complete
- Top-down Proof Procedure
- Datalog

Bottom-up vs. Top-down

Bottom-up

g is proved if g $\in$ C

When does BU look at the query g?

- In every loop iteration
- Never
- At the end
- At the beginning

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*.

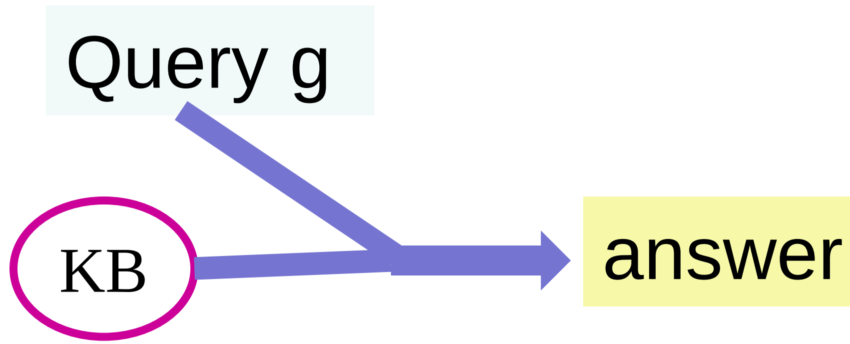
Bottom-up

g is proved if $g \in C$

When does BU look at the query g?

- Never
- It derives the same C regardless of the query

Top-down



We'll see how g is proved

TD performs a backward search starting at g

Top-down Ground Proof Procedure

Idea: search backward from a query

An answer clause is of the form: $yes \leftarrow a_1 \wedge \ldots \wedge a_m$

where $a_1, \ldots, a_m$ are atoms

We express the query as an answer clause

– E.g. query $q_1 \wedge \ldots \wedge q_k$ is expressed as $yes \leftarrow q_1 \wedge \ldots \wedge q_k$

Basic operation: SLD Resolution of an answer clause

$$yes \leftarrow c_1 \wedge \ldots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \ldots \wedge c_m$$

on an atom c_i with another clause

$$c_i \leftarrow b_1 \wedge \ldots \wedge b_p$$

yields the clause

$$yes \leftarrow c_1 \wedge \ldots \wedge c_{i-1} \wedge b_1 \wedge \ldots \wedge b_p \wedge c_{i+1} \ldots \wedge c_m$$

Rules of derivation in top-down and bottom-up

Top-down:

SLD Resolution

$$\frac{yes \leftarrow c_1 \wedge \ldots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \ldots \wedge c_m \qquad c_i \leftarrow b_1 \wedge \ldots \wedge b_p}{yes \leftarrow c_1 \wedge \ldots \wedge c_{i-1} \wedge b_1 \wedge \ldots \wedge b_p \wedge c_{i+1} \ldots \wedge c_m}$$

Bottom-up:

Generalized modus ponens

$$\frac{h \leftarrow b_1 \wedge \ldots \wedge b_m \qquad b_1 \wedge \ldots \wedge b_m}{h}$$

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. The question was "Can we derive a?"

The answer is "Yes, we can"

SLD Derivations

- An answer is an answer clause with m = 0.

  yes $\leftarrow$.

- A successful derivation from KB of query $?q_1 \wedge \ldots \wedge q_k$ is a sequence of answer clauses $\gamma_0, \gamma_1, \ldots, \gamma_n$ such that
  - γ_0 is the answer clause yes $\leftarrow q_1 \wedge \ldots \wedge q_k$.
  - γ_i is obtained by resolving γ_{i-1} with a clause in KB, and
  - γ_n is an answer yes $\leftarrow$

- An unsuccessful derivation from KB of query $?q_1 \wedge \ldots \wedge q_k$
  - We get to something like yes $\leftarrow b_1 \wedge \ldots \wedge b_k$, where there is no clause in KB with any of the b_i as its head

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 ∧ ... ∧ q_k
repeat
    select q_i ∈ body;
    choose clause C ∈ KB, C is q_i ← 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: non-deterministic, have to pick the right one

Example for failing 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 $\boxed{e.}$

2 $\boxed{f \leftarrow k.}$

$f\,.$

$j \leftarrow c.$

Query: ?a

γ_0: yes $\leftarrow$ a

γ_1: yes $\leftarrow$ e $\wedge$ f

γ_2: yes $\leftarrow$ e $\wedge$ k

γ_3: yes $\leftarrow$ k

"Can we derive a?"
"This time we failed"

There is no rule
with k as its head,
thus … fail

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	BU derivation
yes $\leftarrow$ a.	{}
yes $\leftarrow$ b $\wedge$ f.	
yes $\leftarrow$ b $\wedge$ g $\wedge$ h.	
yes $\leftarrow$ c $\wedge$ d $\wedge$ g $\wedge$ h.	
yes $\leftarrow$ d $\wedge$ g $\wedge$ h.	
yes $\leftarrow$ g $\wedge$ h.	
yes $\leftarrow$ h.	
yes $\leftarrow$.	

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	BU derivation
yes $\leftarrow$ a.	{}
yes $\leftarrow$ b $\wedge$ f.	{h}
yes $\leftarrow$ b $\wedge$ g $\wedge$ h.	{g,h}
yes $\leftarrow$ c $\wedge$ d $\wedge$ g $\wedge$ h.	{d,g,h}
yes $\leftarrow$ d $\wedge$ g $\wedge$ h.	{c,d,g,h}
yes $\leftarrow$ g $\wedge$ h.	{b,c,d,g,h}
yes $\leftarrow$ h.	{b,c,d,f,g,h}
yes $\leftarrow$.	{a,b,c,d,f,g,h}

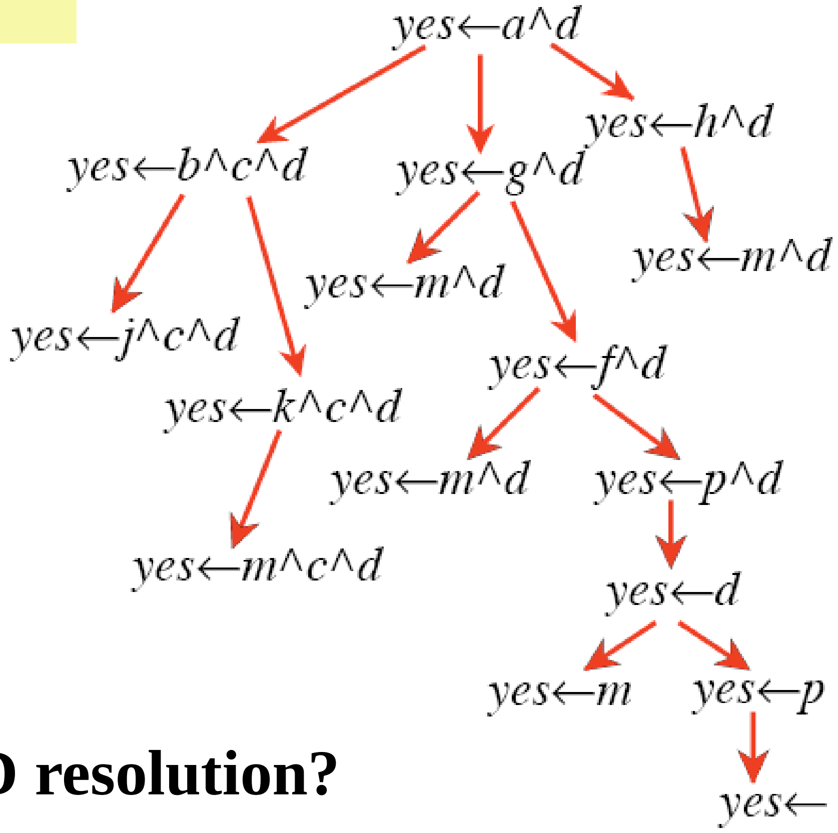
Is the Top-down procedure sound and complete?

- Yes, since there is a 1:1 correspondence between top-down and bottom-up proofs
  - The two methods derive exactly the same atoms (if the SLD resolution picks the successful derivations)

Search Graph for Top-down proofs

Query: ?$a \wedge d$.

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



What kind of search is SLD resolution?

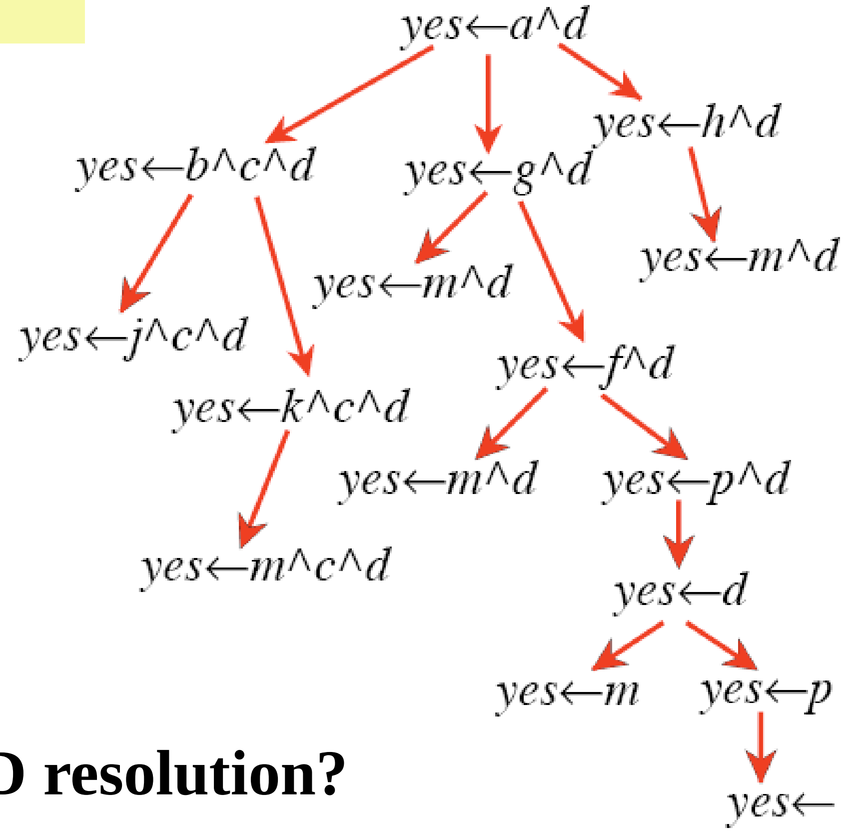
Breadth-first search

Depth-first-search

Search Graph for Top-down proofs

Query: ?$a \wedge d$.

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



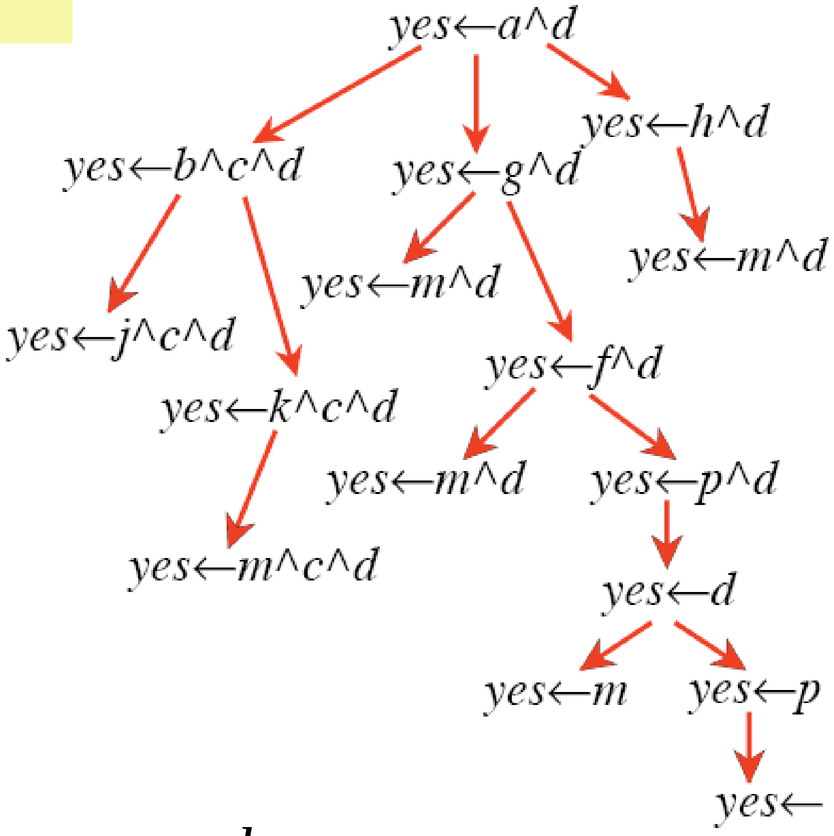
What kind of search is SLD resolution?

It’s a depth-first-search. Failing resolutions are paths where the search has to backtrack.

Search Graph for Top-down proofs

Query: ?$a \wedge d$.

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



We can use heuristics!

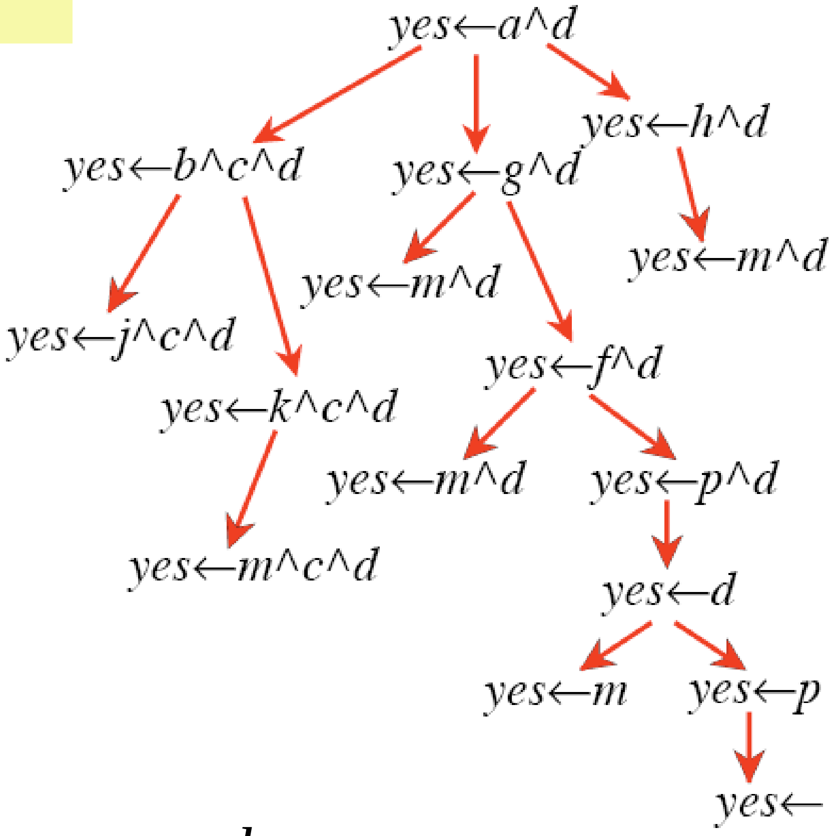
E.g.: number of atoms in the answer clause

Admissible? Yes No

Search Graph for Top-down proofs

Query: ?$a \wedge d$.

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



We can use heuristics!

E.g.: number of atoms in the answer clause

Admissible?

Yes, you need at least these many SLD steps to get an answer

Inference as Standard Search

- 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 :
  - State: full assignment of values to features
  - Successor function: states reachable by applying valid actions
  - Goal test: partial assignment of values to features
  - Solution: a sequence of actions
  - Heuristic function: relaxed problem! E.g. “ignore delete lists”
- 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

- Recap: Bottom-up proof procedure is sound and complete
- Top-down Proof Procedure
- Datalog

Representation and Reasoning in complex domains

- Expressing knowledge with **propositions** can be quite limiting

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

E.g. there is no notion that w_1 *is the same in* $live\_w_1$ and in $connected\_w_1\_w_2$

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

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

Now there is a notion that w_1 *is the same in* $live(w_1)$ and in $connected(w_1, w_2)$

What do we gain?

- Express knowledge that holds for set of individuals (by introducing variables), e.g.

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

- We can ask generic queries, such as “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')

An atom is a symbol of the form p or $p(t_1 \dots t_n)$ 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:

$$h \leftarrow b_1 \wedge \dots \wedge b_m$$

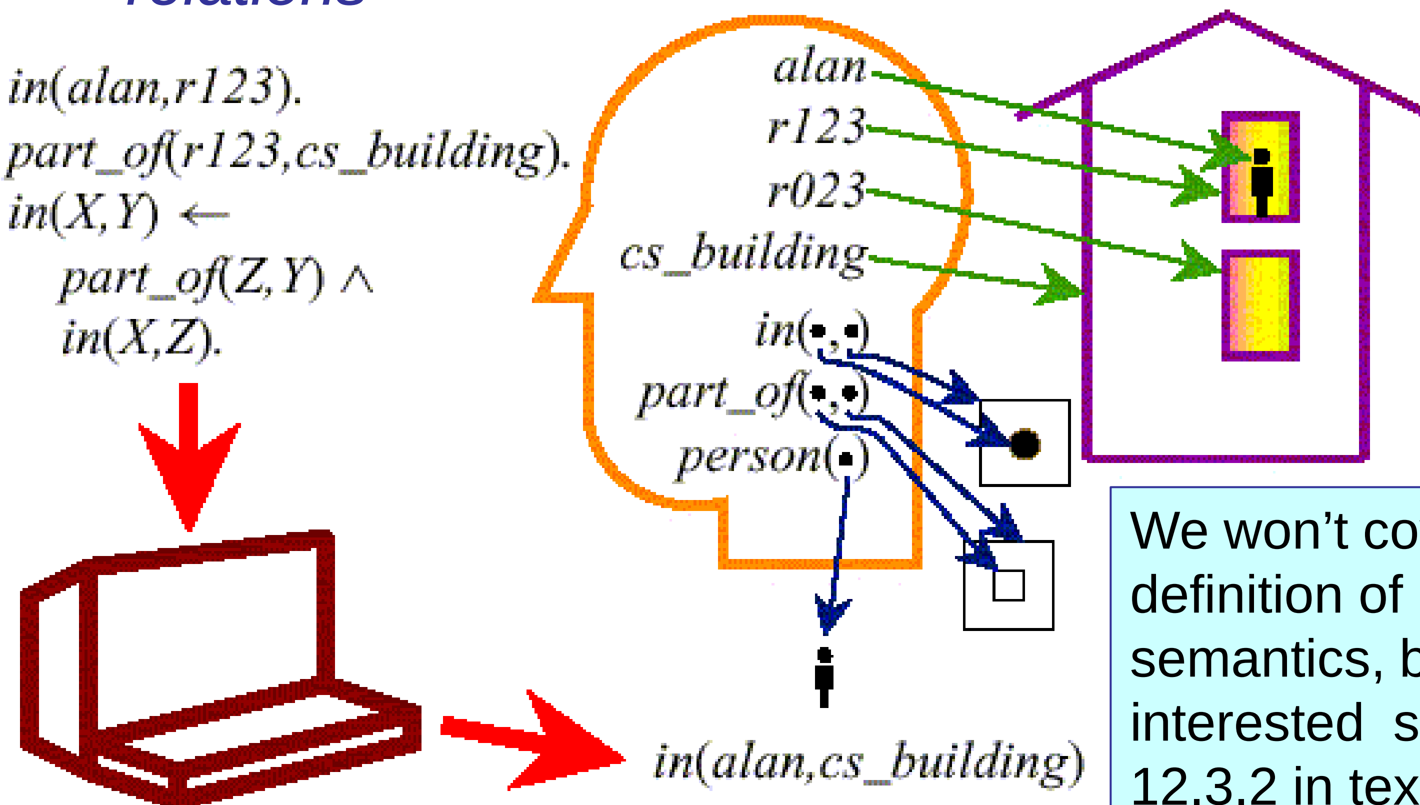
where h and the b_i are atoms (Read this as ``h if b."")

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

A knowledge base is a set of definite clauses

Datalog 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

Datalog: Top Down Proof

- Extension of TD for PDCL. How to deal with variables?
  - Idea: TD finds clauses with consequence predicates that match the query, then substitutes variables with the appropriate constants ***throughout*** the clause
    - We won't look at the details of the formal process (called ***unification***)

Example:

```
in(alan, r123).
part_of(r123,cs_building).
in(X,Y) <- part_of(Z,Y) & in(X,Z).
```

Query: yes <- in(alan, cs_building).

……………

yes <-

See trace of how the answer is found in Deduction Applet, example *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).

See outcome in Deduction Applet,
example *in-part-of* available at
http://cs.ubc.ca/~hutter/teaching/cpsc322/ /alan.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