Lecture 16

Bottom Up and Top Down Proof Procedure

(Ch 5.2.2)

Lecture Overview

- Recap Lecture 15
- Bottom-Up Proof Procedure
- Soundness
- Completeness
- Top-Down (TD) Proof Procedure

- TD as Search
- Datalog (time permitting)

Where Are We?

Representation

Environment

Problem Type Deterministic $\boxed{\text{Arc}}$ Stochastic Reasoning Technique

$\boxed{\text{Consistency}}$

Constraint Satisfaction ***VarsConstraints+*** $\boxed{\text{Search}}$

Logic: a framework for representation & reasoning

Static

Query

Logics

Search

Belief Nets

Variable Elimination

Sequential

Planning

STRIPS

Search

Decision Nets

Variable Elimination

Markov Processes

Value Iteration

Back to static problems, but with richer representation

- When we represent a domain about which we have only partial (but certain) information, we need to represent….
- Objects, properties, sets, groups, actions, events, time, space, …
- All these can be represented as
- Objects
- Relationships between objects
- Logic is the language to express knowledge about the world this way

We will start with a simple logic

Primitive elements are propositions: Boolean variables that can be {true, false}

Two kinds of statements:

that a proposition is true

that a proposition is true if one or more other propositions are true

To Define a Logic, We Need

- Syntax: specifies the symbols used, and how they can be combined to form legal sentences
- Knowledge base is a set of sentences in the language

- Semantics: specifies the meaning of symbols and sentences
- Reasoning theory or proof procedure: a specification of how an answer can be produced.
- Sound: only generates correct answers with respect to the semantics
- Complete: Guaranteed to find an answer if it exists

Propositional Definite Clauses: Syntax

Definition (atom)

An **atom** is a symbol starting with a lower case letter

Examples: p_1; $live\_l_1$

Definition (body)

A **body** is an atom or is of the form $b_1 \land b_2$ where b_1 and b_2 are bodies.

Examples: $p_1 \land p_2$; $ok\_w_1 \land live\_w_0$

Definition (definite clause)

A **definite clause** is

- an atom or
- a rule of the form $h \leftarrow b$ where h is an atom ("head") and b is a body. (Read this as "h if b".)

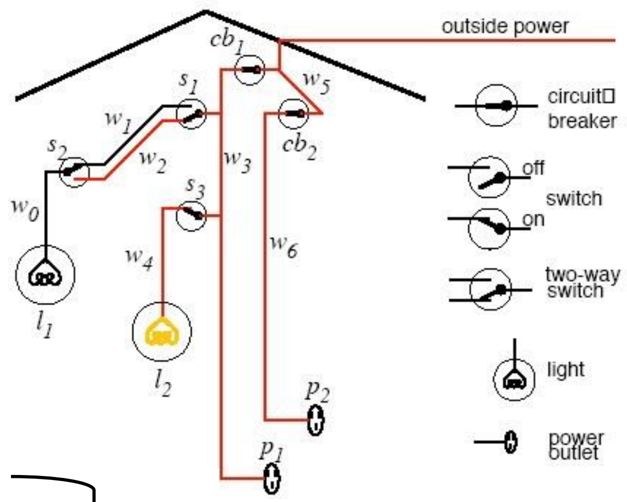
Examples: $p_1 \leftarrow p_2$; $live\_w_0 \leftarrow live\_w_1 \land up\_s_2$

Definition (KB)

A **knowledge base (KB)** is a set of definite clauses

$light\_l_1.$
$light\_l_2.$
$ok\_l_1.$
$ok\_l_2.$
$ok\_cb_1.$
$ok\_cb_2.$
$live\_outside.$

atoms



definite clauses, KB

$live\_l_1 \leftarrow live\_w_0.$
$live\_w_0 \leftarrow live\_w_1 \wedge up\_s_2.$
$live\_w_0 \leftarrow live\_w_2 \wedge down\_s_2.$
$live\_w_1 \leftarrow live\_w_3 \wedge up\_s_1.$
$live\_w_2 \leftarrow live\_w_3 \wedge down\_s_1.$
$live\_l_2 \leftarrow live\_w_4.$
$live\_w_4 \leftarrow live\_w_3 \wedge up\_s_3.$
$live\_p_1 \leftarrow live\_w_3.$
$live\_w_3 \leftarrow live\_w_5 \wedge ok\_cb_1.$
$live\_p_2 \leftarrow live\_w_6.$
$live\_w_6 \leftarrow live\_w_5 \wedge ok\_cb_2.$
$live\_w_5 \leftarrow live\_outside.$
$lit\_l_1 \leftarrow light\_l_1 \wedge live\_l_1 \wedge ok\_l_1.$
$lit\_l_2 \leftarrow light\_l_2 \wedge live\_l_2 \wedge ok\_l_2.$

rules

Propositional Definite Clauses: Semantics

Definition (interpretation)

An **interpretation I** assigns a truth value to each atom.

Definition (truth values of statements)

- A **body $b_1 \wedge b_2$** is true in I if and only if b_1 is true in I and b_2 is true in I.
- A **rule $h \leftarrow b$** is false in I if and only if b is true in I and h is false in I.

- A **knowledge base KB** is true in I if and only if every clause in KB is true in I.

PDC Semantics: Knowledge Base (KB)

- A **knowledge base KB** is true in I if and only if every clause in KB is true in I.

	p	q	r	s
I_1	false	true	true	false

KB_1 KB_2 KB_3

KB_1:
p
r
$s \leftarrow q \wedge p$

KB_2:
r
$s \leftarrow p\ q$
$\leftarrow p \wedge s$

KB_3:
$r\ q\ s$
$\leftarrow q$

Only KB_2 above is True in I_1

Propositional Definite Clauses: Semantics

Definition (interpretation)

An **interpretation I** assigns a truth value to each atom.

Definition (truth values of statements)

- A **body $b_1 \wedge b_2$** is true in I if and only if b_1 is true in I and b_2 is true in I.
- A **rule $h \leftarrow b$** is false in I if and only if b is true in I and h is false in I.
- A **knowledge base KB** is true in I if and only if every clause in KB is true in I.

Definition (model)

A **model** of a knowledge base KB is an interpretation in which KB is true.

Similar to CSPs: a model of a set of clauses is an interpretation that makes all of the clauses true

PDC Semantics: Example for models

Definition (model)
A **model** of a knowledge base KB is an interpretation in which every clause in KB is true.

$$KB = \begin{cases} p \leftarrow q \\ q \\ r \leftarrow s \end{cases}$$

Which of the interpretations below are models of KB?

All interpretations where KB is true: I_1, I_3, and I_4

	p	q	r	s	$p \leftarrow q$	q	$r \leftarrow s$	KB
I_1	T	T	T	T	T	T	T	T
I_2	F	F	F	F	T	F	T	F
I_3	T	T	F	F	T	T	T	T
I_4	T	T	T	F	T	T	T	T
I_5	F	T	F	T	F	T	F	F

What We Want to Do with Logic

1) Tell the system knowledge about a task domain.

- This is your KB
- which expresses true statements about the world

2) Ask the system whether new statements about the domain are true or false.

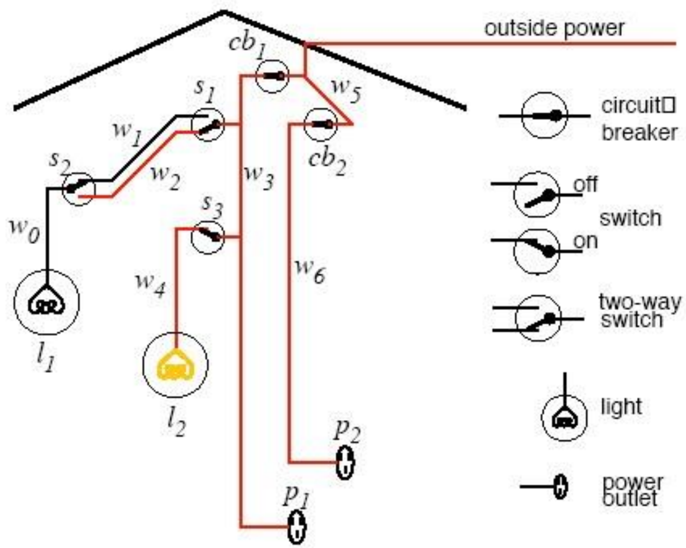
- We want the system responses to be
  - Sound: only generates correct answers with respect to the semantics
  - Complete: Guaranteed to find an answer if it exists

For Instance

1) Tell the system knowledge about a task domain.

$light\_l_1.$
$light\_l_2.$
$ok\_l_1.$
$ok\_l_2.$
$ok\_cb_1.$
$ok\_cb_2.$
$live\_outsic$

$live\_l_1 \leftarrow live\_w_0.$
$live\_w_0 \leftarrow live\_w_1 \land up\_s_2.$
$live\_w_0 \leftarrow live\_w_2 \land down\_s_2.$
$live\_w_1 \leftarrow live\_w_3 \land up\_s_1.$
$live\_w_2 \leftarrow live\_w_3 \land down\_s_1.$
$live\_l_2 \leftarrow live\_w_4.$
$live\_w_4 \leftarrow live\_w_3 \land up\_s_3.$
$live\_p_1 \leftarrow live\_w_3.$
$live\_w_3 \leftarrow live\_w_5 \land ok\_cb_1.$
$live\_p_2 \leftarrow live\_w_6.$
$live\_w_6 \leftarrow live\_w_5 \land ok\_cb_2.$
$live\_w_5 \leftarrow live\_outside.$
$lit\_l_1 \leftarrow light\_l_1 \land live\_l_1 \land ok\_l_1.$
$lit\_l_2 \leftarrow light\_l_2 \land live\_l_2 \land ok\_l_2.$



2) Ask the system whether new statements about the domain are true or false

- $live\_w_4$?

lit_l_2?

PDCL Semantics: Logical Consequence

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

- In other words, KB ⊧ g if there is no interpretation in which KB is true and g is false

- We want a **reasoning procedure** that can find all and only the logical consequences of a knowledge base

- **mechanically derivable** demonstration that a formula logically follows from a knowledge base.

- Must be sound and complete

Recap: proofs, soundness, completeness

- A proof is a mechanically derivable demonstration that a formula logically follows from a knowledge base.

Definition (derivability with a proof procedure)

Given a proof procedure P, KB $\vdash_P$ g means g can be derived from knowledge base KB with proof procedure P.

Definition (soundness)

A proof procedure P is sound if KB $\vdash_P$ g implies KB $\models$ g.

sound: every atom g that P derives follows logically from KB

Definition (completeness)

Simple proof procedure S

- Enumerate all interpretations
- For each interpretation I, check whether it is a model of KB
  - ✓i.e., check whether all clauses in KB are true in I
- KB $\vdash_S$ g if g holds in all such models

A proof procedure P is complete if KB $\models$ g implies KB $\vdash_P$ g.

complete: every atom g that logically follows from KB is derived by P

Simple Proof Procedure

problem with this approach?

- If there are n propositions in the KB, must check all the 2^n interpretations!

Goal of proof theory

- find sound and complete proof procedures that allow us to prove that a logical formula follows from a KB avoiding to do the above

Bottom-up proof procedure

- One rule of derivation, a generalized form of modus ponens:

- If "$h \leftarrow \mathbf{b_1} \wedge \ldots \wedge \mathbf{b_m}$" is a clause in the knowledge base, and

  each $\mathbf{b}_i$ has been derived,

  then h can be derived.

- This rule also covers the case when m = 0.

Bottom-up (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 $G \subseteq C$ at the end of this procedure

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

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

Bottom-up proof procedure: example

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 ∪ {h} until no more clauses can be selected.

$a \leftarrow b \wedge c$

{}

$a \leftarrow e \wedge f$

$b \leftarrow f \wedge k$

$c \leftarrow e$

$d \leftarrow k$

$e.$

$f \leftarrow j \wedge e$

$f \leftarrow c$

$j \leftarrow c$

$\{e\}$

$\{c,e\}$

$\{c,e,f\}$

$\{c,e,f,j\}$

$\{a,c,e,f,j\}$

Done.

Bottom-up proof procedure: Example

KB $z \leftarrow f \wedge e$

$q \leftarrow r \wedge g \wedge e$

$e \leftarrow a \wedge b$ a b

r f

> 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.

Which of the following is true?

A. $KB \vdash_{BU} \{z , q , a\}$

B. $KB \vdash_{BU} \{r , z , b\}$

C. $KB \vdash_{BU} \{q,a\}$

Bottom-up proof procedure: Example

KB $z \leftarrow f \wedge e$

$q \leftarrow r \wedge g \wedge e$

$e \leftarrow a \wedge b$ a b

r

f

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}

A. $KB \vdash_{BU} \{z, q, a\}$

{a,b}

{a, b, r}

{a, b, r, f}

B. $KB \vdash_{BU} \{r, z, b\}$

{a, b, r, f, e} {a, b, r, f, e, z}

C. $KB \vdash_{BU} \{q,a\}$

Done.

Lecture Overview

- Recap
- Bottom-Up Proof Procedure
- Soundness
- Completeness
- Top-Down Proof Procedure

- TD as Search
- Datalog (time permitting)

Soundness of bottom-up proof procedure BU

Definition (soundness)

A proof procedure P is sound if KB $\vdash_P$ g implies KB $\models$ g.

sound: every atom g that P derives follows logically from KB

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.

What do we need to prove to show that BU is sound ?

Definition (soundness)

A proof procedure P is sound if KB $\vdash_P$ g implies KB $\models$ g.

Soundness of bottom-up proof procedure BU

sound: every atom g that P derives follows logically from KB

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 ∪ {h} until no more

clauses can be selected.

What do we need to prove to show that BU is sound ?

If $g \in C$ at the end of BU procedure, then g is true in all models of KB ($KB \models g$) If $g \in C$ at the end of BU procedure, then g is true in all models of KB ($KB \models g$)

Soundness of bottom-up proof procedure BU

Proof by contradiction: Suppose there is a hsuch that KB

$\vdash_{BU}$ h but not KB $\models$h.

1. Let h be the first atom added to C that is not true in every model of KB.
   - In particular, suppose I is a model of KB in which h isn't true
2. Since hwas added to C, there must be a clause in KB of form

   $$h \leftarrow b_1 \wedge \ldots \wedge b_n$$

   where each $b_i$is already in C and thus true in every model of KB, including I.
3. Because h is false in I, $h \leftarrow b_1 \wedge \ldots \wedge b_n$ is false in I.

Soundness of bottom-up proof procedure BU

4. Therefore I is not a model of KB=> Contradiction with

Lecture Overview

- Recap
- Bottom-Up Proof Procedure
- Soundness
- Completeness
- Top-Down Proof Procedure
- TD as Search

Completeness of BU: general idea

- Datalog (time permitting)

Completeness of BU: general idea

- Generic completeness of proof procedure:

  If g is logically entailed by the KB ($KB \models g$) then g can be proved by the procedure ($KB \vdash_{BU} g$)

Sketch of our proof for BU:

1. Suppose $KB \models g$. Then g is true in all models of KB.

2. Thus g is true in any particular model of KB
3. We will define a model (called minimal model) so that if g is true in that model, g is proved by the bottom up algorithm.
4. Thus KB $\vdash_{BU}$ g.

We define a specific interpretation of our KB, in which

- every atom in C at the end of BU is true
- every other atom is false

This is called minimal model

Completeness of BU: general idea

EXAMPLE

All atoms = {a, b, c, d,e, f, g}

C = ?

KB $a \leftarrow e$
$\wedge g.$ $b \leftarrow f \wedge$
$g.$ c
$\leftarrow e.$ $f \leftarrow c$
$e.$
$d.$

Completeness of BU: general idea

We define a specific interpretationof our KB, in which

- every atom in C at the end of BU is true
- every other atom is false

Completeness of BU: general idea

This is called minimal model

All atoms = {a, b, c, d,e, f, g}

C = {e,d, c, f,}

Minimal Model =

We define a specific interpretationof our KB, in which

- every atom in C at the end of BU is true
- every other atom is false

KB $a \leftarrow e \wedge g$.
$b \leftarrow f \wedge g$.
$c \leftarrow e$.
$f \leftarrow c$
e.
d.

Completeness of BU: general idea

This is called minimal model

All atoms = {a, b, c, d,e, f, g}

C = {e,d, c, f,}

Minimal Model =

a=F, b=F, c=T, d = T, e=T, f=T, g=F

We define a specific interpretationof our KB, in which

- every atom in C at the end of BU is true
- every other atom is false

KB $a \leftarrow e \wedge g$.
$b \leftarrow f \wedge g$.
$c \leftarrow e$.
$f \leftarrow c$.
e.
d.

Completeness of BU: general idea

This is called minimal model

- Using this interpretation, we'll then show that, if

KB $\models$ G, then G must be in C, that is

If g is true in all models of KB (KB $\models$ g) then g $\in$ C at the end of BU procedure (KB $\vdash_{BU}$ g)

Definition

The minimal model MM is the interpretation in which

- every element of BU's fixed point C is true
- every other atom is false.

First, we prove that: MM is a model of KB

Proof by contradiction: assume that MM is not a model of KB.

- Then there must exist some clause in KB which is false in MM
  - ✓Like every clause in KB, it is of the form $h \leftarrow b_1 \wedge \ldots \wedge b_m$ (with $m \geq 0$).
- $h \leftarrow b_1 \wedge \ldots \wedge b_m$ can only be false in MM if each b_i is true in MM and h is false in MM.
  - ✓Since each b_i is true in MM, each b_i must be in C as well.

✓BU would add h to C, so h would be true in MM. Contradiction!

- Thus, MM is a model of KB

Completeness of bottom-up procedure

If g is true in all models of KB (KB ⊧ g) **then** g ∈ C at the end of BU procedure (KB $\vdash_{BU}$ g)

Direct proof based on minimal model:

- Suppose KB ⊧ g. Then g is true in all models of KB.
- Thus g is true in the minimal model.
- Thus g ∈ C at the end of BU procedure.
- Thus KB $\vdash_{BU}$ g. Done. KB ⊧ g implies KB $\vdash_{BU}$ g

Summary for bottom-up proof procedure BU

- 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

Let's consider these two alternative proof procedures for PDCL….

KB $a \leftarrow e \wedge g$.
$b \leftarrow f \wedge g$. $c \leftarrow e$
$\leftarrow c$
e.
d.

X. C_X = {All clauses in KB with empty bodies}

Y. C_Y = {All atoms in the knowledge base}

A. Both X and Y are sound and complete

B. Both Y and X are neither sound nor complete

C X is sound only and Y is complete only

D. X is complete only and Y is sound only

Let's consider these two alternative proof procedures for PDCL….

KB $a \leftarrow e$
$\wedge g$. $b \leftarrow f \wedge$
g. $c \leftarrow$
e. $f \leftarrow c$
e.
d.

X. C_X = {All clauses in KB with empty bodies}

Returns atoms that are indeed logical consequences (sound), but misses all those derived from the application of rules with nonempty bodies (not complete)

Y. C_Y = {All atoms in the knowledge base}

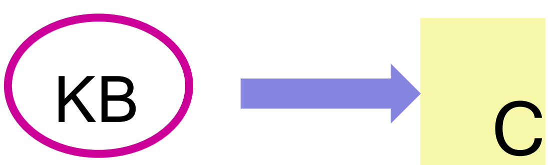
Returns all the logical consequences (complete), but also returns atoms that are not (not sound),

X is sound only and Y is complete only

Bottom-up vs. Top-down

Let g be the query

Bottom-up



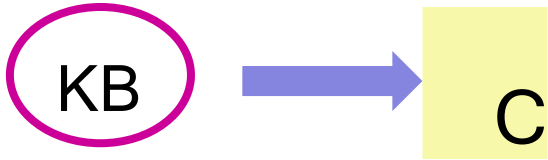
g is proved if $g \in C$

When does BU use the information that
G is the query?

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



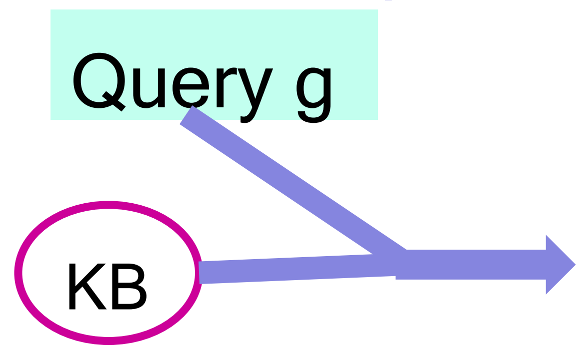
Query g is proven if $g \in C$

answer

When does BU uses the info that g is the query?

- Only at the end
- It derives the same C regardless of the query

Top-down



TD performs a backward search starting at g

Lecture Overview

- Recap
- Bottom-Up Proof Procedure
- Soundness
- Completeness
- Top-Down (TD) Proof Procedure
- TD as Search
- Datalog (time permitting)

Top-down Proof Procedure for PDCL

- Idea: search backward from a query to determine if it is a logical consequence of KB.
- An answer clause is of the form

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

- We express a query $q_1 \wedge q_2 \ldots \wedge q_m$ as an answer clause

  $$yes \leftarrow q_1 \wedge \ldots \wedge q_k$$

  Selective Linear Definite claus

- Basic operation: SLD Resolution of an answer 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:

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

yields the clause yes $\leftarrow a_1 \wedge \ldots \wedge a_{i-1} \wedge b_1 \wedge \ldots \wedge b_p$

$\wedge a_{i+1} \ldots \wedge a_m$

Example

- Rule of derivation: the SLD Resolutionof clause

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

on atom a_i with the clause:

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

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

$\wedge a_{i+1} \ldots \wedge a_m$

$yes \leftarrow b \wedge c.$

$b \leftarrow k \wedge f.$

SLD resolution

$yes \leftarrow e \wedge f.$

$e.$

Example

- 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

$yes \leftarrow e \wedge f.$
$e.$

$yes \leftarrow f$

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 $\leftarrow b_1 \wedge \ldots \wedge b_k$.
  - 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 $q_1 \wedge \ldots \wedge$

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

Example: successful derivation

a← b ∧ c. 1 a ← e ∧ f. b← f ∧ k.

4 c ← e. d ← k 3 5 $\boxed{\text{e.}}$

f ← j ∧ e. 2 f ← c. j ← c.

Query: ?a 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 ←

Done. The question was

“Can we derive a?”

The answer is "Yes, we can"

Example: failing derivation

1 $a \leftarrow b \wedge c.$ $a \leftarrow e \wedge f.$ 2 $b \leftarrow f \wedge k.$

$\boxed{c \leftarrow e.}$ 46 $d \leftarrow k$ 57 $f \leftarrow j$ $\boxed{e.}$

$\wedge e.$ 3 $f \leftarrow c.$ $j \leftarrow c.$

Query: ?a γ_0: yes $\leftarrow$ a

γ_1: yes ← b ∧ c γ_2:
yes ← f ∧ k ∧ c
γ_3: yes ← c ∧ k ∧
c γ_4: yes ← e ∧ k
∧ c

γ_5: yes ← k ∧ c

There is no rule

γ_6: yes ← k ∧ e

with k as its head, γ_7:

yes ← k thus … fail

Rules of derivation in top-down and bottom-up

Top-down:

SLD Resolution

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

Bottom-up:

Generalized modus ponens

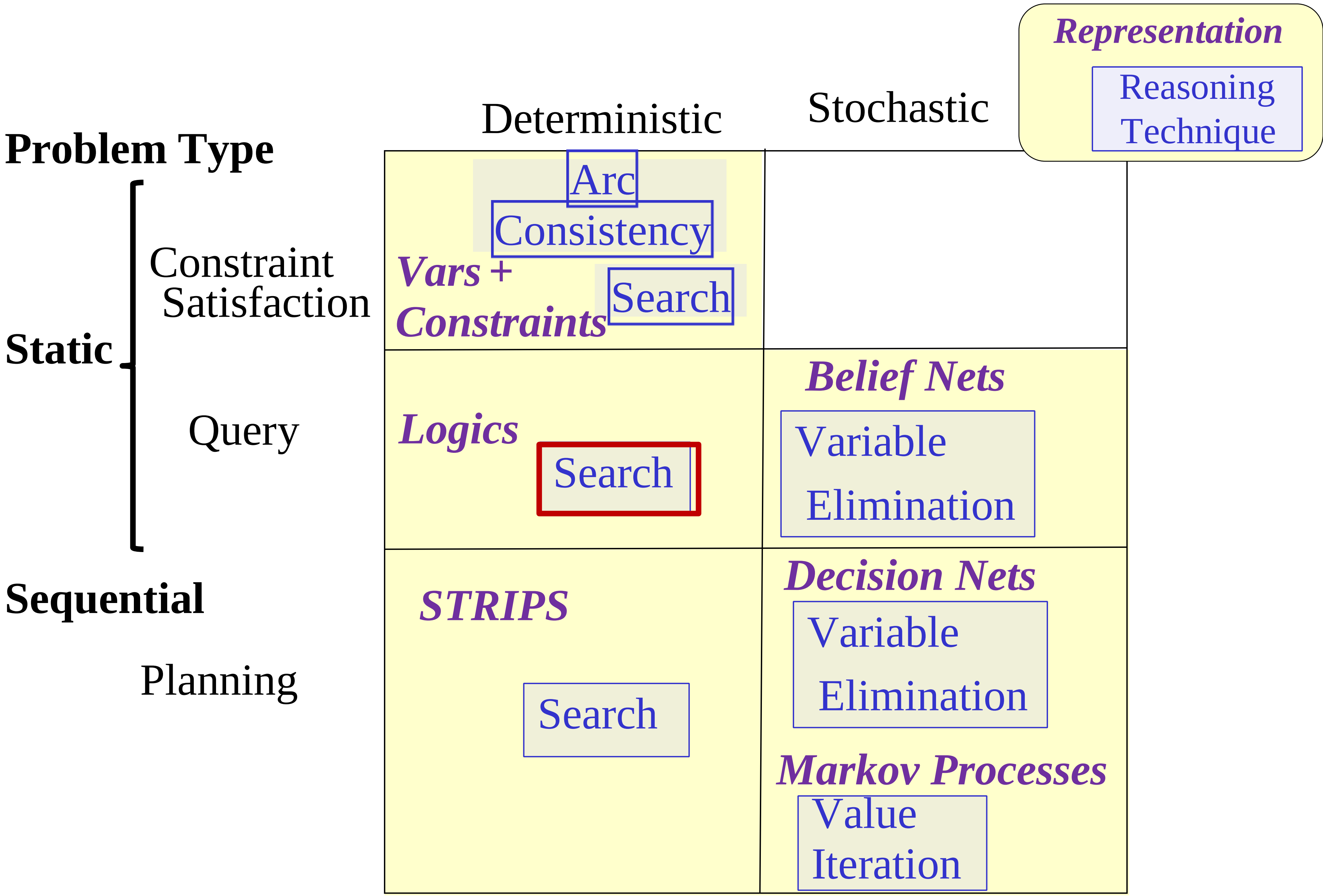
$$\frac{h \leftarrow b_1 \wedge \ldots \wedge b_m \qquad b_1 \wedge \ldots \wedge b_m}{h}$$

Lecture Overview

- Recap
- Bottom-Up Proof Procedure
- Soundness
- Completeness
- Top-Down (TD) Proof Procedure
- TD as Search
- Datalog (time permitting)

SLD resolution as search

- SLD resolution can be seen as a search
- from a query stated as an answer clause
- to an answer
- Through the space of all possible answer clauses


Representation
Reasoning Technique
Deterministic
Stochastic
Problem Type
Static
Constraint Satisfaction
Query
Sequential
Planning
Arc Consistency
Vars + Constraints
Search
Logics
Search
STRIPS
Search
Belief Nets
Variable Elimination
Decision Nets
Variable Elimination
Markov Processes
Value Iteration

Environment

Inference as Standard Search

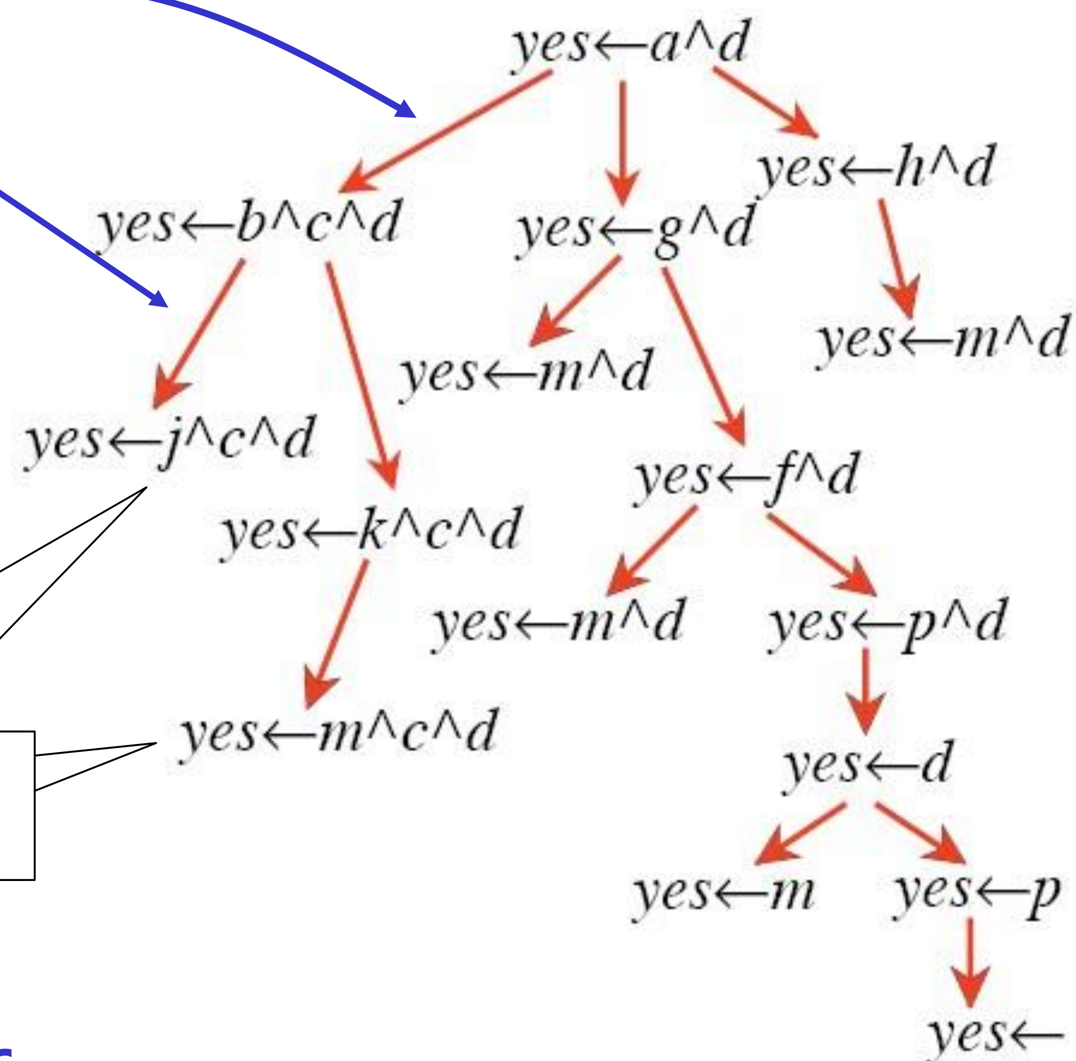
- Constraint Satisfaction (Problems):
- State: assignments of values to a subset of the variables
- Successor function: assign values to a “free” variable
- Goal test: set of constraints
- Solution: possible world that satisfies the constraints
- Heuristic function: none (all solutions at the same distance 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”

- Query (Top-down/SLD resolution)
  - 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
  - Heuristic function: ?????

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

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

Prove: ?← a ∧ d.



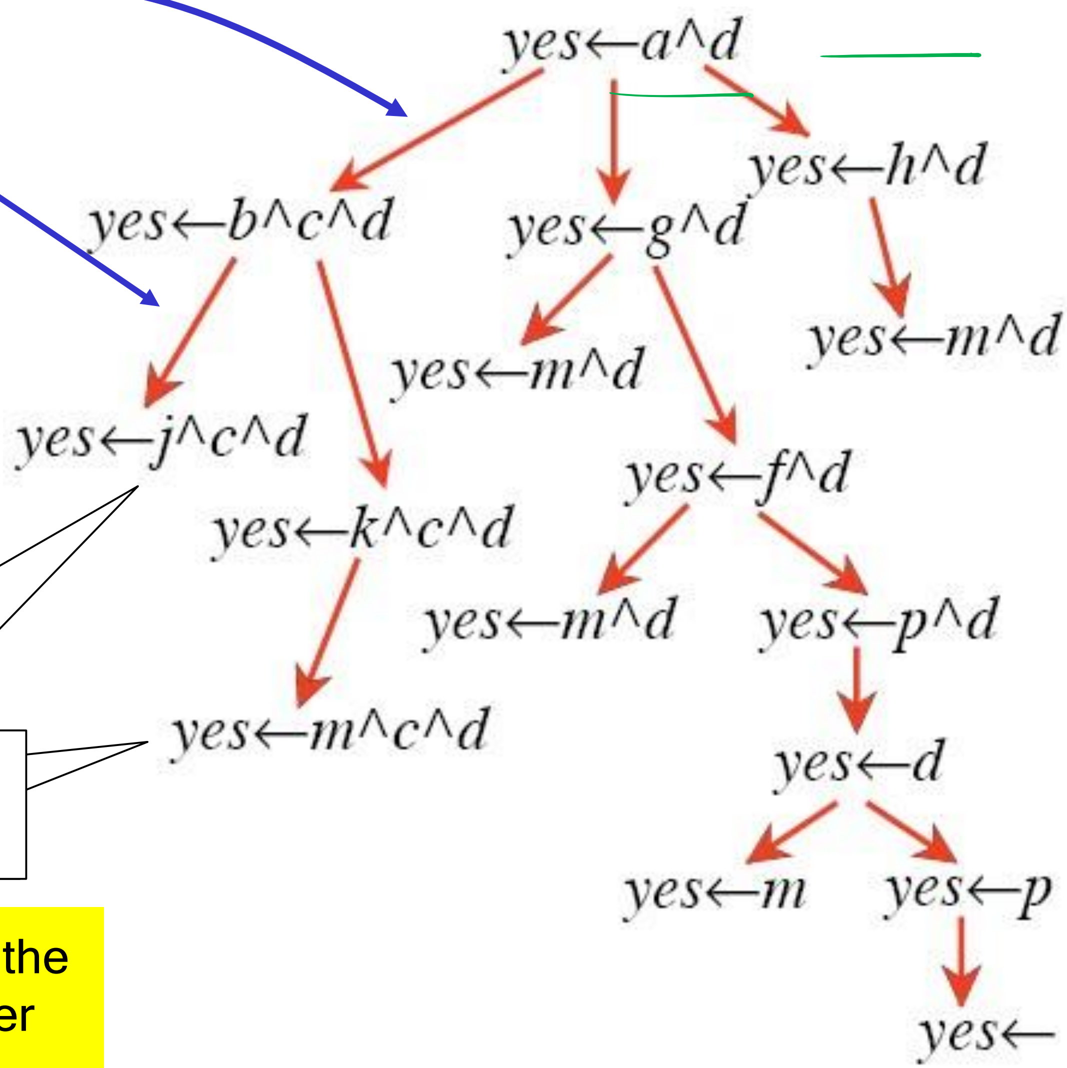
Why are these dead ends in the search space?

Search Graph

KB

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

Prove: ?← a ∧ d.



Why are these dead ends in the search space?

- the successor function resolves the first atom in the body of the answer clause
- But j and m cannot be resolved

Search Graph

KB

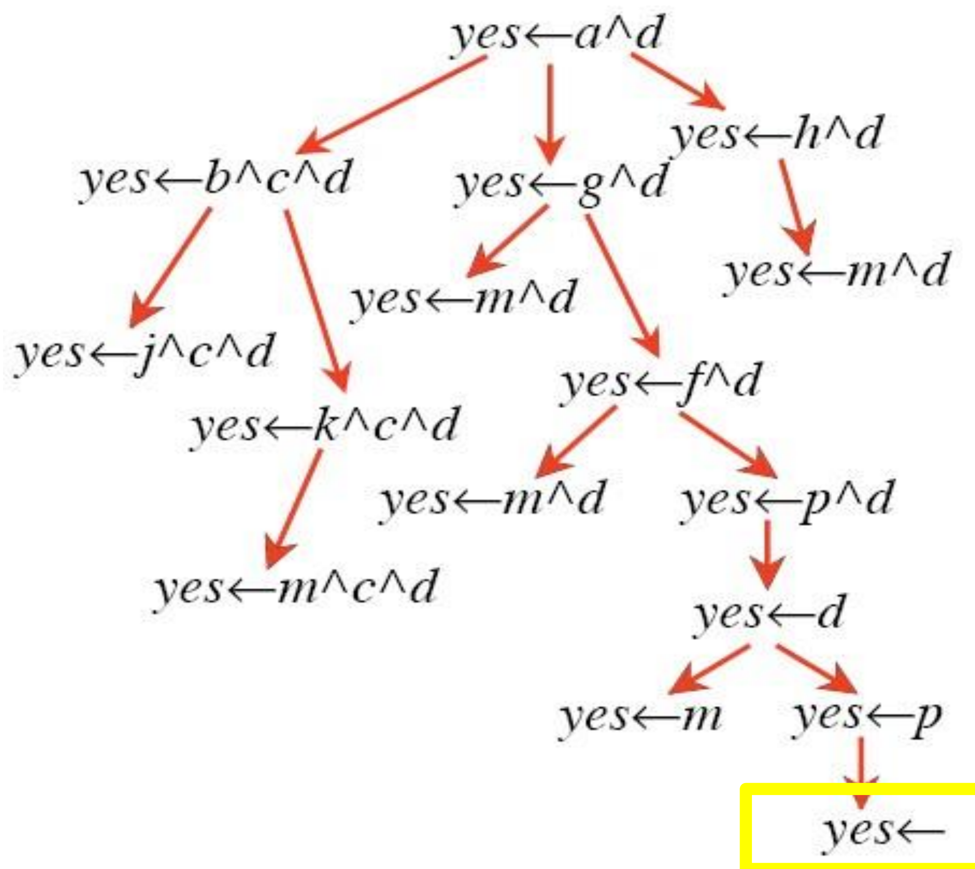
Top-down/SLD resolution as Search

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: $? \leftarrow a \wedge d$.



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

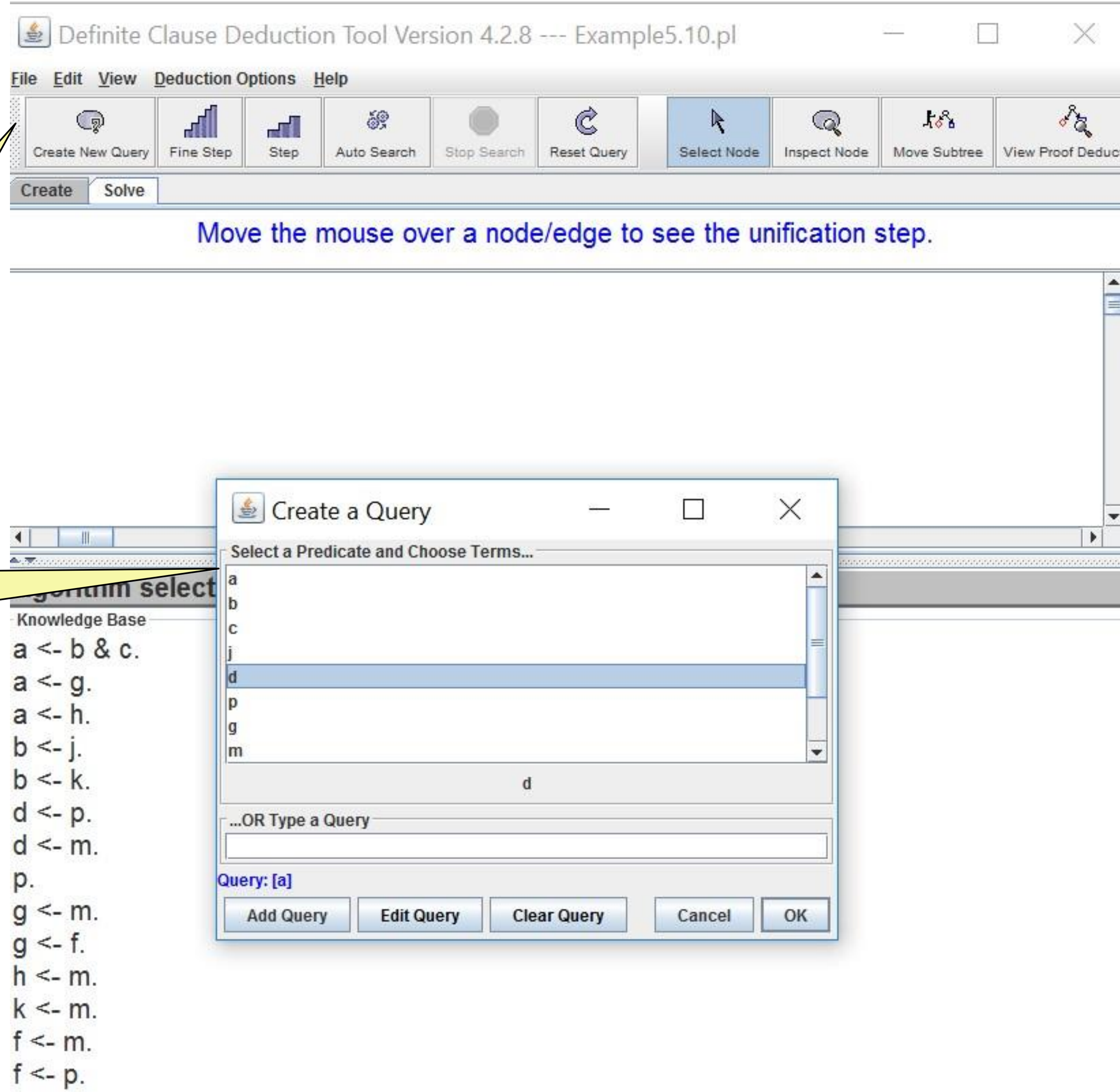
can trace the example in the Deduction Applet at http://aispace.org/deduction/ using file *kb-for-topdown-search* available in course schedule

Deduction

Button to initiatethe creation a new query

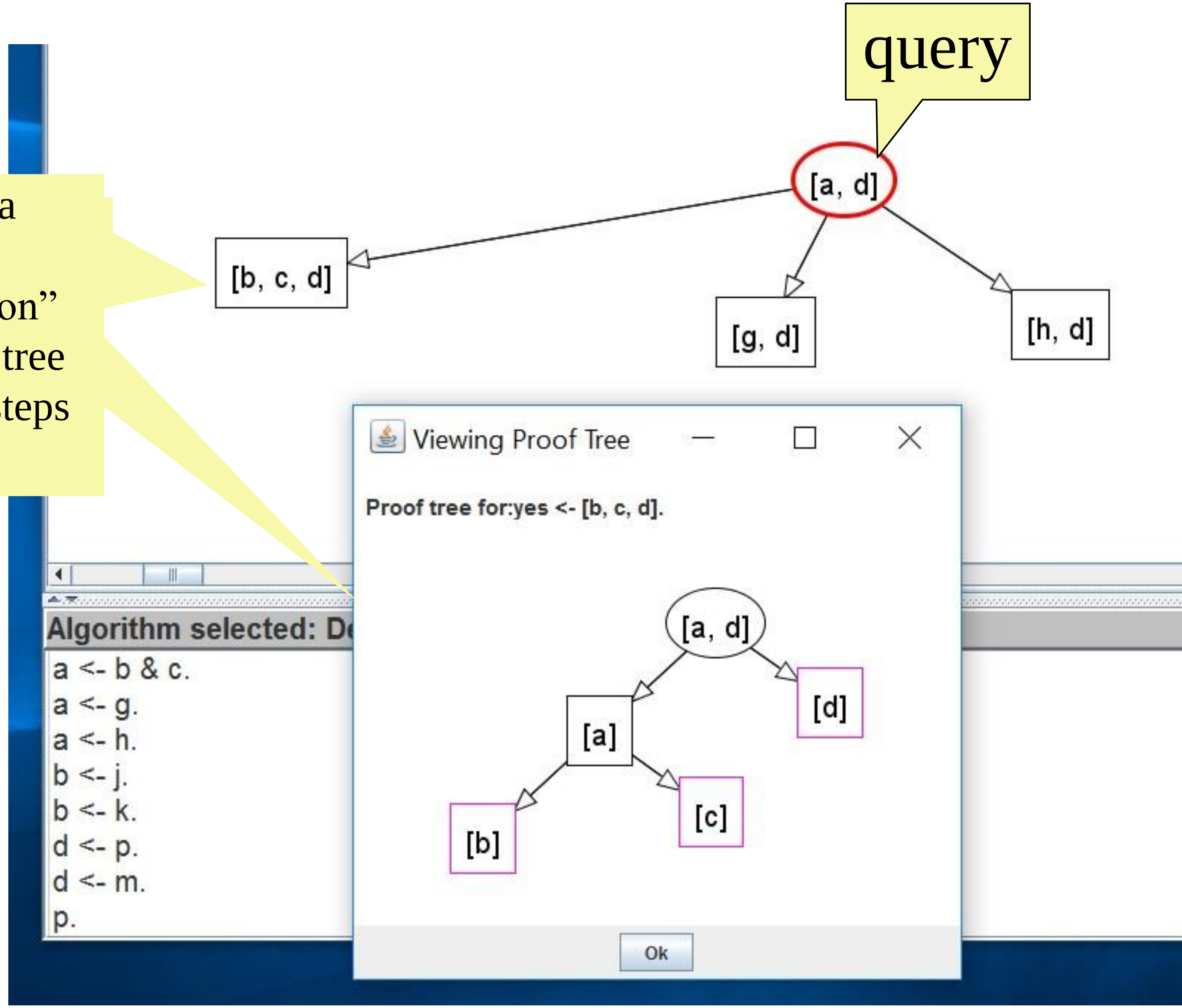
Query creationpanel

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



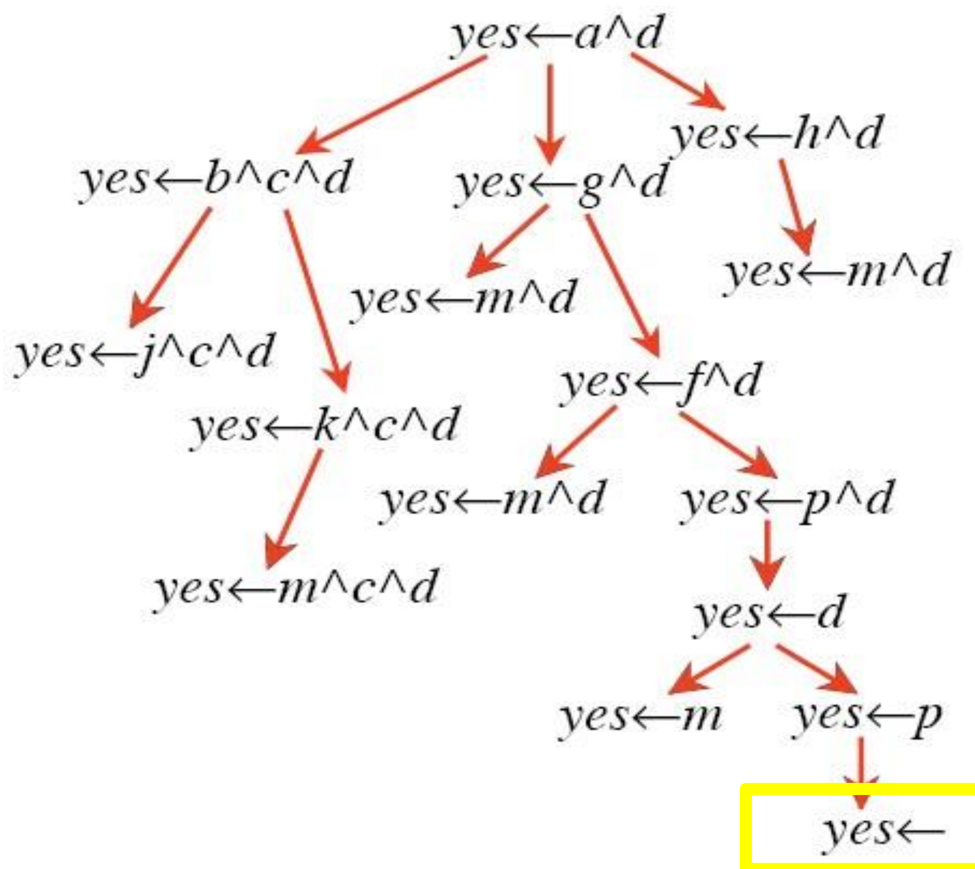
Top-down/SLD resolution as Search

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: ?$\leftarrow a \wedge d$.

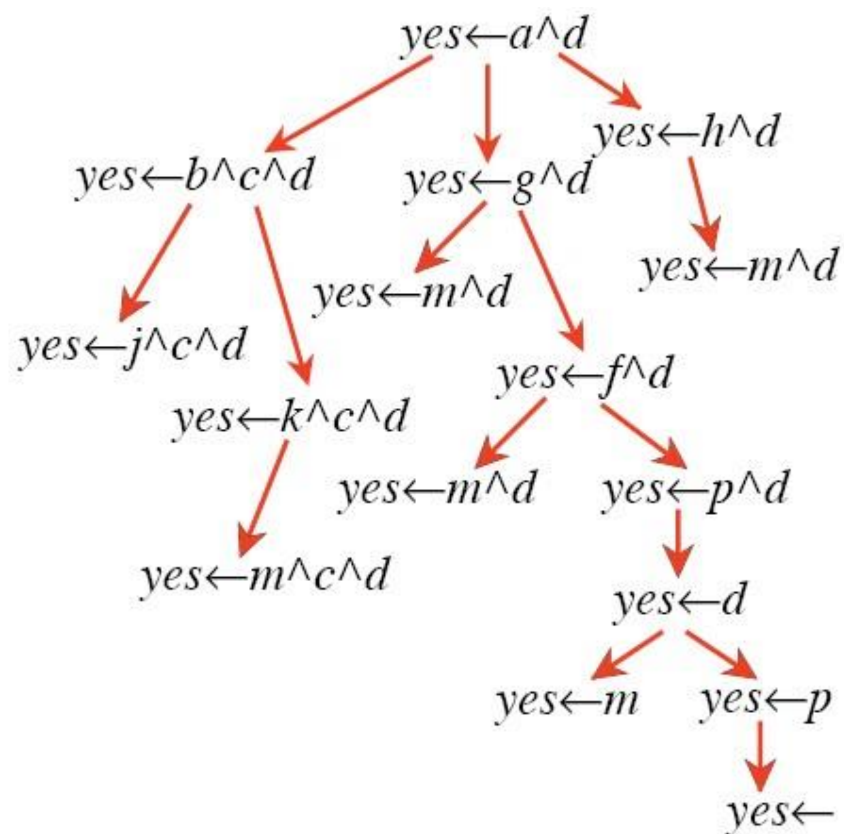


$a \leftarrow b \wedge c.$ $a \leftarrow g.$ $a \leftarrow h.$ $b \leftarrow j.$ $b \leftarrow k.$ $d \leftarrow m.$ $d \leftarrow p.$ $f \leftarrow m.$ $f \leftarrow p.$ $g \leftarrow m.$ $g \leftarrow f.$ $k \leftarrow m.$ $h \leftarrow m.$ $p.$

Possible Heuristic?

Search Graph

KB



Possible Heuristic?

Search Graph

KB

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

Prove: ?← a ∧ d.

Number of atoms in the answer clause

Admissible?

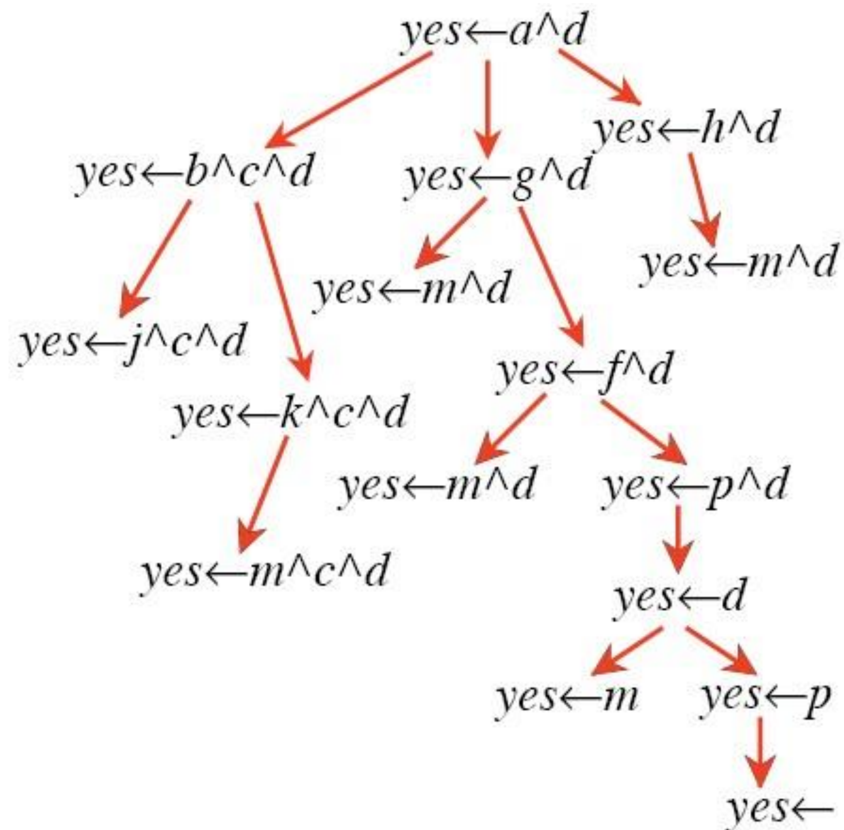
A. Yes

B. No

C. It depends

Search Graph

KB



Possible Heuristic?

Search Graph

KB

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

Number of atoms in the answer clause

Admissible?

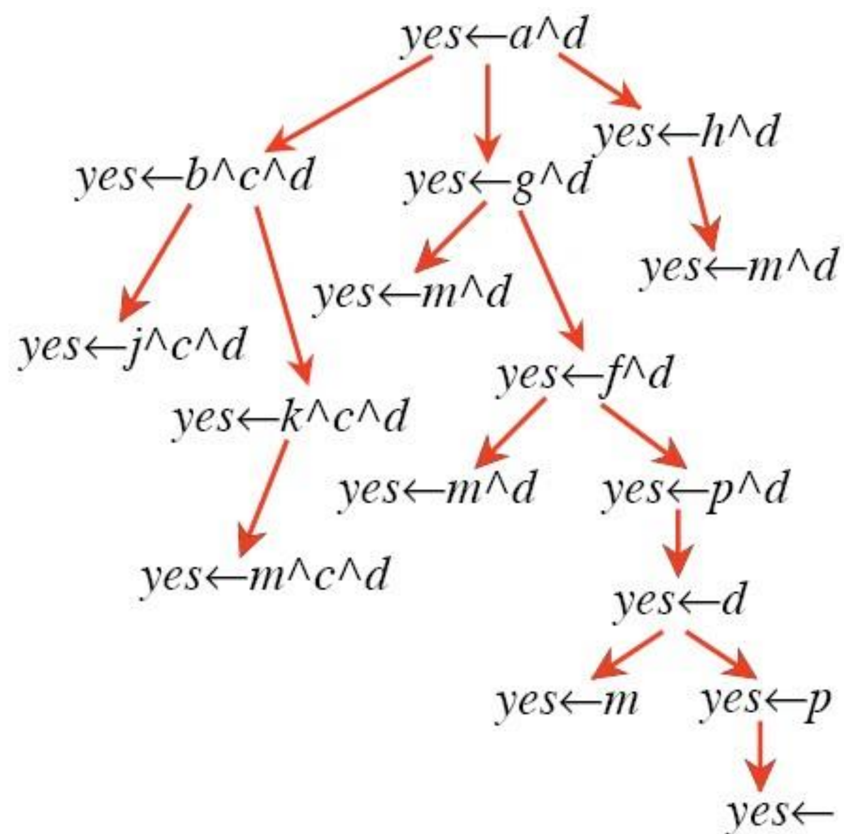
Prove: ?← a ∧d.

A. Yes

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

Search Graph

KB



Possible Heuristic?

Search Graph

KB

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

Prove: ?← a ∧ d.

Number of atoms in the answer clause

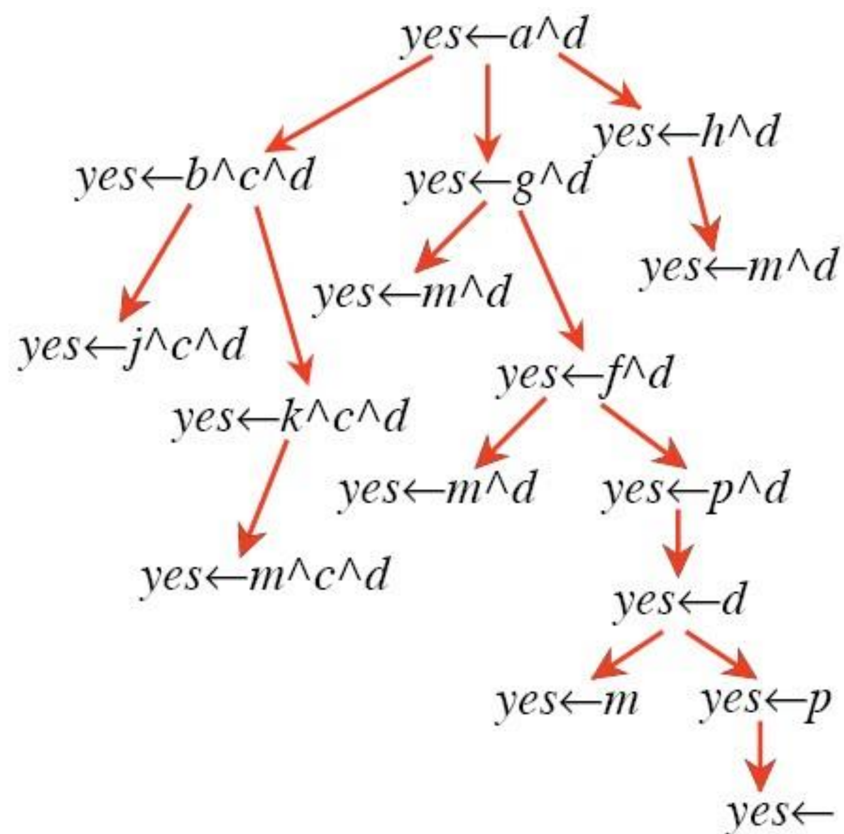
Admissible? Yes

Is this a domain-dependent or domain-independent heuristics?

A. Domain dependent B. Domain Independent C. It depends

Search Graph

KB



Possible Heuristic?

Search Graph

KB

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

Prove: ?← a ∧ d.

Number of atoms in the answer clause

Admissible? Yes

Is this a domain-dependent or domain-independent heuristics?

B. Domain Independent

Learning Goals For Logic so Far

- 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