

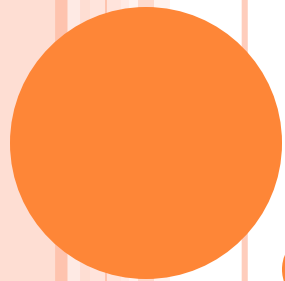
INTRODUCTION TO ARTIFICIAL INTELLIGENCE

כללי

מפגש רביעי

מבוא	פרק 1 -
סוכנים אינטיליגנטיים	פרק 2 -
פתרון בעיות באמצעות חיפוש	פרק 3 -
אלגוריתמים לחיפוש מקומי ובעיות אופטימיזציה (רק סעיף 4.1)	פרק 4 -
חיפוש בתנאי יריבות (סעיפים 5.1-5.4)	פרק 5 -
בעיות סיפוק אילוצים (סעיפים 6.1-6.4)	פרק 6 -
סוכנים לוגיים	פרק 7 -
לוגיקה מסדר ראשון (סעיפים 8.1-8.3 בלי 8.3.3)	פרק 8 -
היסק בלוגיקה מסדר ראשון (בלי 9.3.3, בסעיף 9.4 רק את 9.4.1)	פרק 9 -
תכנון (10.1-10.3, 10.4.1, 10.4.4)	פרק 10 -
אי-וודאות	פרק 13 -
הנמקה הסתברותית (14.1-14.2)	פרק 14 -
קבלת החלטות מורכבות (בלי 17.4)	פרק 17 -
למידה מדוגמאות (18.1-18.5)	פרק 18 -





LOGICAL AGENTS

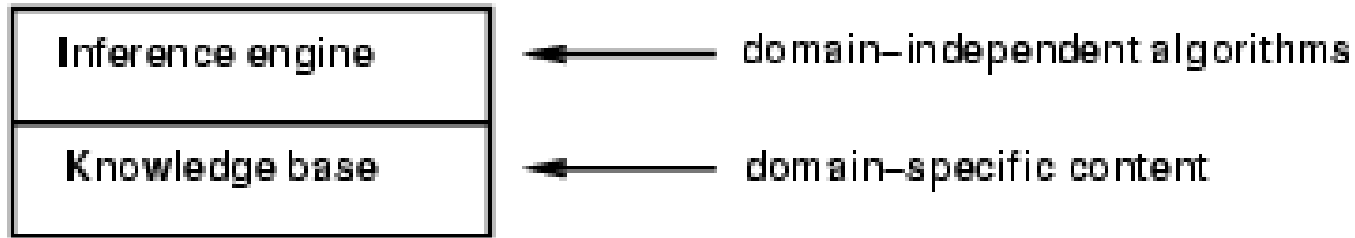
Chapter 7

OUTLINE

- Knowledge-based agents
- Wumpus world
- Logic in general - models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
 - forward chaining
 - backward chaining
 - resolution
 -



KNOWLEDGE BASES



- Knowledge base = set of **sentences** in a **formal** language
-
- **Declarative** approach to building an agent (or other system):
 - Tell it what it needs to know
 -
- Then it can Ask itself what to do - answers should follow from the KB
-
- Agents can be viewed at the **knowledge level**
i.e., what they know, regardless of how implemented
- Or at the **implementation level**
 - i.e., data structures in KB and algorithms that manipulate them
 -



A SIMPLE KNOWLEDGE-BASED AGENT

```
function KB-AGENT(percept) returns an action  
  static: KB, a knowledge base  
           t, a counter, initially 0, indicating time  
  
  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))  
  action  $\leftarrow$  ASK(KB, MAKE-ACTION-QUERY(t))  
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))  
  t  $\leftarrow$  t + 1  
  return action
```

- The agent must be able to:
 - - Represent states, actions, etc.
 - Incorporate new percepts
 - Update internal representations of the world
 - Deduce hidden properties of the world
 - Deduce appropriate actions



WUMPUS WORLD PEAS DESCRIPTION

- Performance measure

- gold +1000, death -1000
- -1 per step, -10 for using the arrow

- Environment

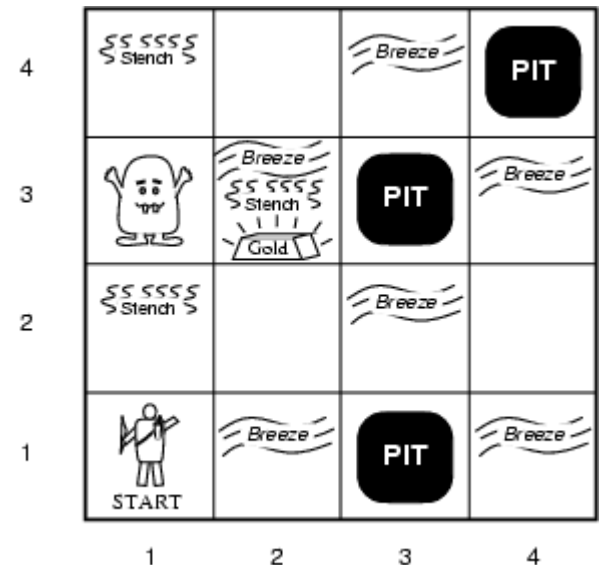
- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square

- Sensors: Stench, Breeze, Glitter, Bump, Scream

-

- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

-




WUMPUS WORLD CHARACTERIZATION

- Fully_Observable No – only local perception
-
- Deterministic Yes – outcomes exactly specified
-
- Episodic No – sequential at the level of actions
-
- Static Yes – Wumpus and Pits do not move
-
- Discrete Yes
-
- Single-agent? Yes – Wumpus is essentially a natural feature
-

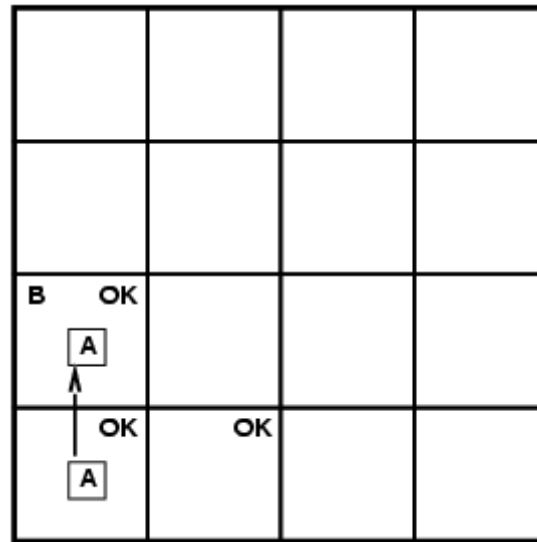


EXPLORING A WUMPUS WORLD

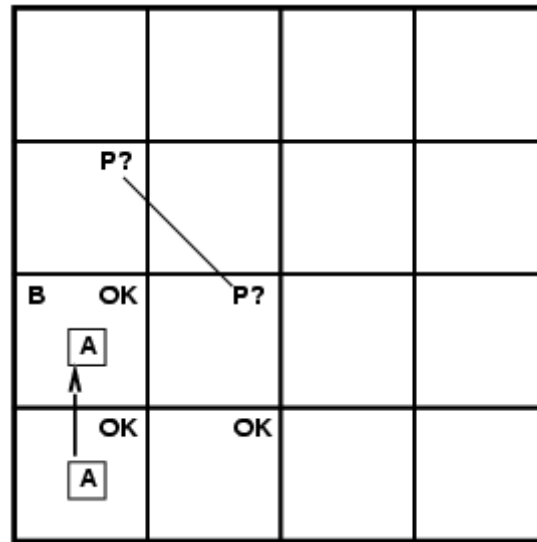
OK			
OK 	OK		



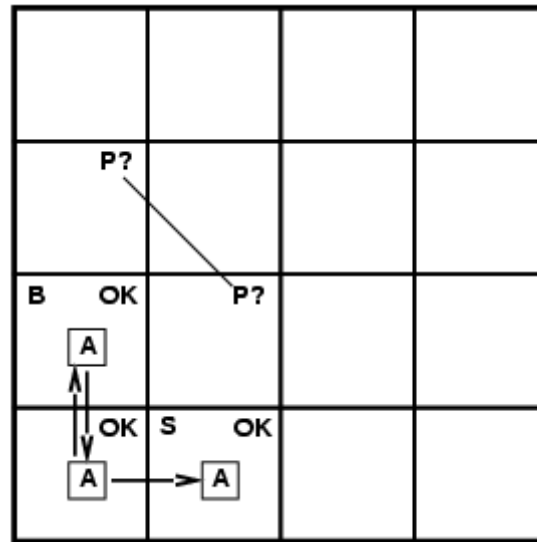
EXPLORING A WUMPUS WORLD



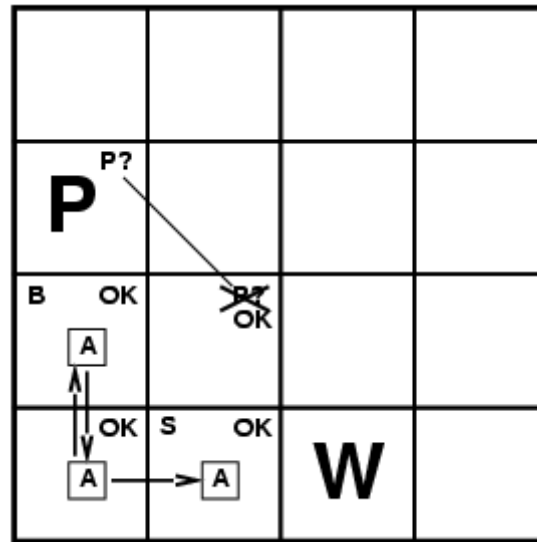
EXPLORING A WUMPUS WORLD



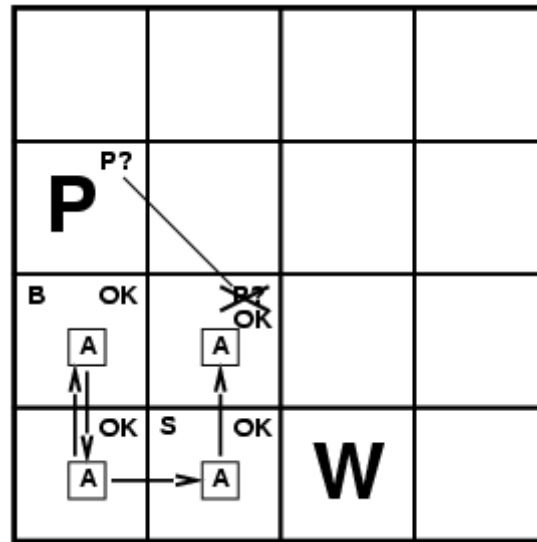
EXPLORING A WUMPUS WORLD



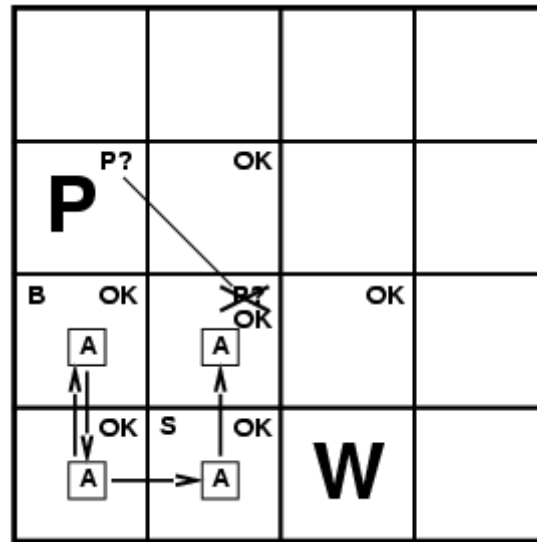
EXPLORING A WUMPUS WORLD



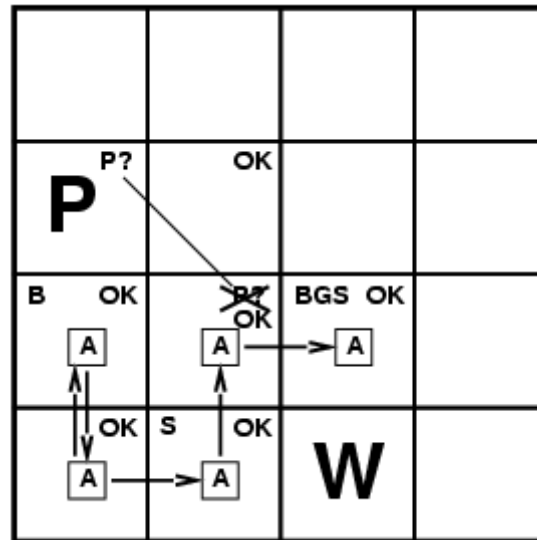
EXPLORING A WUMPUS WORLD



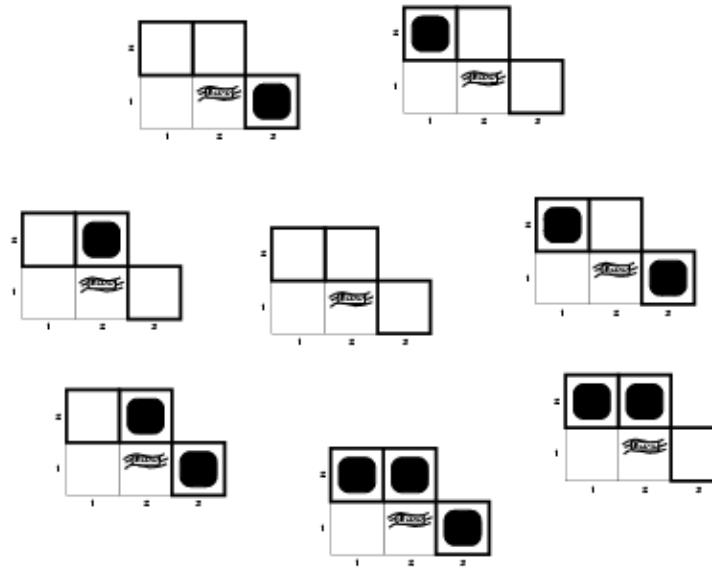
EXPLORING A WUMPUS WORLD



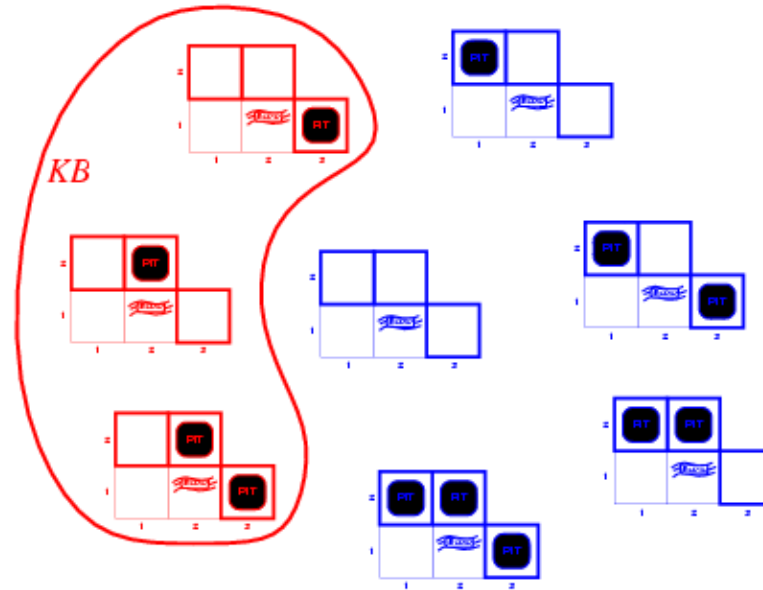
EXPLORING A WUMPUS WORLD



WUMPUS MODELS



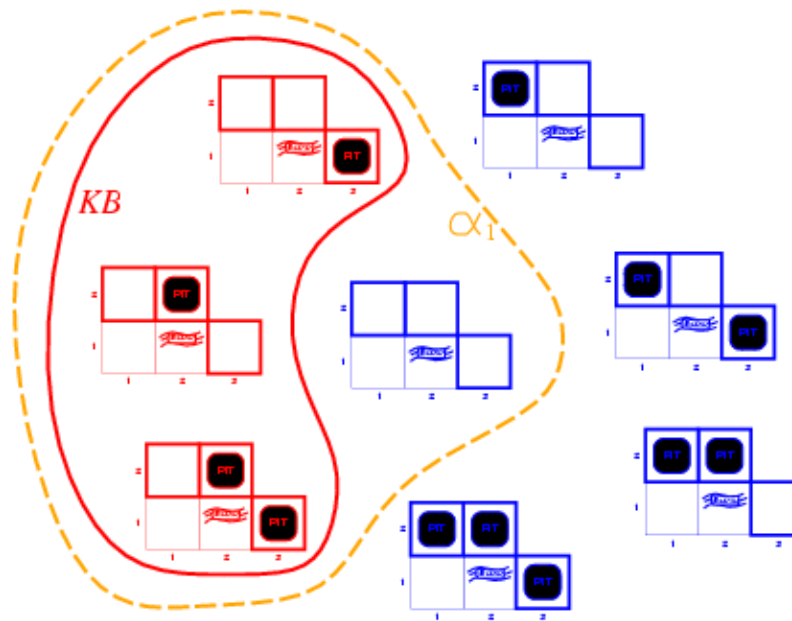
WUMPUS MODELS



- KB = wumpus-world rules + observations
-



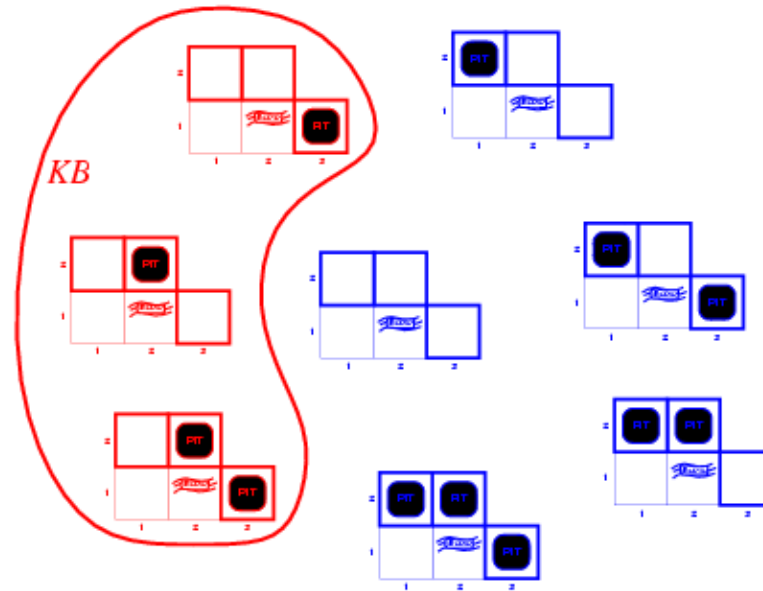
WUMPUS MODELS



- KB = wumpus-world rules + observations
- α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by [model checking](#)
-
-



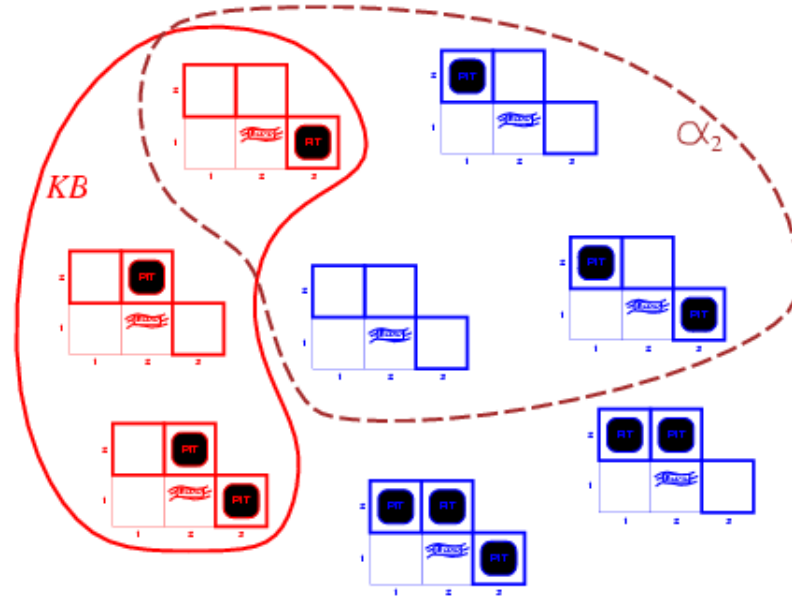
WUMPUS MODELS



- *KB* = wumpus-world rules + observations



WUMPUS MODELS



- KB = wumpus-world rules + observations
- α_2 = "[2,2] is safe", $KB \models \alpha_2$
-



LOGIC IN GENERAL

- Logics are formal languages for representing information such that conclusions can be drawn
-
- Syntax defines the sentences in the language
-
- Semantics define the "meaning" of sentences;
- - i.e., define truth of a sentence in a world
 -
- E.g., the language of arithmetic
- - $x+2 \geq y$ is a sentence; $x^2+y > \{$ is not a sentence
 -
 - $x+2 \geq y$ is true iff the number $x+2$ is no less than the number y
 -
 - $x+2 \geq y$ is true in a world where $x = 7, y = 1$
 - $x+2 \geq y$ is false in a world where $x = 0, y = 6$
 -



ENTAILMENT

- Entailment means that one thing follows from another:

-

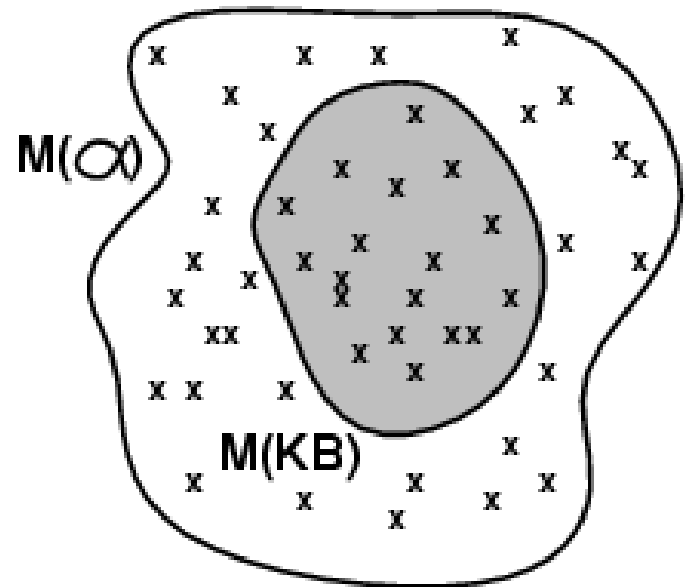
$$KB \models \alpha$$

- Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true
 - E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”
 -
 - E.g., $x+y = 4$ entails $4 = x+y$
 -
 - Entailment is a relationship between sentences (i.e., **syntax**) that is based on **semantics**



MODELS

- Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated
-
- We say m is a **model** of a sentence α if α is true in m
- $M(\alpha)$ is the set of all models of α
-
- Then $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$
- - E.g. $KB = \text{Giants won and Reds won}$ $\alpha = \text{Giants won}$
 -



MODELS

- הסמנטיקה מגדירה את הכללים לקביעת הנכונות של כל פסוק במודל מסוים. בתחשיב פסוקים, מודל קובע את ערך האמת (אמת או שקר) של כל סימבול (proposition symbol).
- מספר המודלים האפשריים מעל n פסוקים אטומיים הוא 2^n .
- הסמנטיקה קובעת כיצד לחשב את ערך האמת של כל פסוק בהינתן מודל. תהליך החישוב הוא רקורסיבי מפסוקים מורכבים לפסוקים פשוטים יותר עד שמגיעים לפסוקים אטומיים או לערכים אמת ושקר.
- הכללים ניתנים לביטוי גם בעזרת טבלאות אמת המציינות את ערך האמת של פסוק מורכב לכל השמה אפשרית של ערכי אמת לרכיביו.



KNIGHTS AND KNAVES

- באי מסויים חיים Knights (דוברי אמת תמיד) ו-Knaves (תמיד משקרים). נתייחס לשניים מתושבי האי, A ו-B.
- A אומר: "לפחות אחד משנינו הוא Knave"
- מי הם A ו-B?

A	B
0	0
0	1
1	0
1	1

בלתי אפשרי

בלתי אפשרי

אפשרי

בלתי אפשרי

בדוגמאות שלהלן נניח כי Knights ערכם 0 ו-Knaves ערכם 1.



KNIGHTS AND KNAVES 2

- שלושה מתושבי האי, A B ו-C עומדים בגינה.
- אדם זר שואל את A: "כמה מכם הם Knights?" A עונה בצורה לא מובנת.
- הזר שואל את B: "מה A אמר?" B עונה: "A אמר שיש בדיוק Knight אחד בינינו".
- ואז C אומר: "אל תאמין ל-B, הוא משקר".
- מי הם A, B ו-C?



KNIGHTS AND KNAVES 2

○ נפתור בעזרת בדיקת מודלים. B הוא Knave, C הוא Knight.

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

After B's statement	After C's statement
אפשרי	בלתי אפשרי
אפשרי	אפשרי
בלתי אפשרי	בלתי אפשרי
אפשרי	בלתי אפשרי
בלתי אפשרי	בלתי אפשרי
אפשרי	אפשרי
בלתי אפשרי	בלתי אפשרי
בלתי אפשרי	בלתי אפשרי

KNIGHTS AND KNAVES 3

- שלושה מתושבי האי, A B ו-C, כל אחד מהם הוא Knight או Knave.
- A אומר: "כולנו Knaves".
- B אומר: "בדיוק אחד מאיתנו הוא Knight".
- מי הם A, B ו-C?



KNIGHTS AND KNAVES 3

<i>A</i>	<i>B</i>	<i>C</i>
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

After <i>A</i> 's statement	After <i>B</i> 's statement
בלתי אפשרי	
אפשרי	בלתי אפשרי
אפשרי	אפשרי
אפשרי	בלתי אפשרי
בלתי אפשרי	
בלתי אפשרי	
בלתי אפשרי	
בלתי אפשרי	

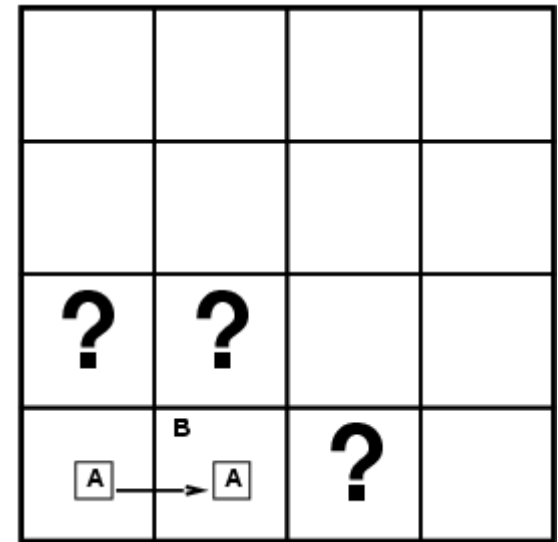


ENTAILMENT IN THE WUMPUS WORLD

Situation after detecting
nothing in [1,1], moving
right, breeze in [2,1]

Consider possible models for
KB assuming only pits

3 Boolean choices \Rightarrow 8
possible models



INFERENCE

- $KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i
-
- **Soundness**: i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
-
- **Completeness**: i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
-
- Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.
-
- That is, the procedure will answer any question whose answer follows from what is known by the KB .
-



עוד כמה מושגים

○ עד כה בדקנו נביעה לוגית בעזרת בדיקת מודלים: עברנו על כל המודלים והראנו שהפסוק אמיתי בכל המודלים. בסעיף זה נראה דרך יעילה יותר (מבדיקת מודלים) לבדיקת נביעה לוגית, על-ידי הוכחת פסוקים (Theorem Proving) – נפעיל כללי היסק על פסוקים בבסיס הידע כדי לבנות הוכחה שהפסוק המסויים אמיתי. דרך זו מתאימה למקרים בהם מספר המודלים רב ואילו ההוכחה קצרה.

○
○ נכיר כעת את המושגים הבאים:

○ שקילות לוגית

○ משפט הדדוקציה

○ ספיקות



שקילות לוגית

○ שקילות לוגית: שני פסוקים הם שקולים לוגית אם הם אמיתיים באותם מודלים, או לחילופין, אם כל אחד מהם נובע לוגית מן השני.



תקפות

○ תקפות: פסוק הוא תקף אם הוא אמיתי בכל המודלים.
כל פסוק תקף שקול לוגית ל-*True*.

○ דוגמאות:

A A



משפט הדדוקציה

○ משפט הדדוקציה: לכל שני פסוקים α ו- β , α גורר את β אם ורק אם הפסוק $\beta \Rightarrow \alpha$ תקף.



ספיקות

- **ספיקות:** פסוק הוא ספיק אם הוא אמיתי באיזשהו מודל.
- אם פסוק α אמיתי במודל m , אומרים ש- m מספק את α או ש- m הוא מודל של α .
- אפשר לבדוק ספיקות בעזרת מעבר על המודלים עד שמוצאים אחד שמספק את α .
- בדיקת הספיקות של פסוק בתחשיב הפסוקים היא הבעיה הראשונה שהוכחה להיות NP-שלמה.
- בעיות רבות במדעי המחשב הן למעשה בעיות ספיקות.
- למשל, כל בעיות ה-CSP מפרק 6 שואלות למעשה האם האילוצים ספיקים על-ידי השמה כלשהי.

○

○ **תקפות וספיקות קשורות זו לזו:**

- פסוק α הוא תקף אם ורק אם $\neg\alpha$ איננו ספיק.
- פסוק α הוא ספיק אם ורק אם $\neg\alpha$ איננו תקף.

○



שפות מסדר ראשון

- כל שפה מסדר ראשון כוללת את המרכיבים הבאים:

- אוסף (אינסופי) של משתנים: x, y, z

- קבוצה של קשרים לוגיים: \neg (שלילה), \wedge (קוניונקציה "וגם"),

- \vee (דיסיונקציה "או"), \rightarrow (גרירה), \leftrightarrow (גרירה כפולה).

- קבוצת כמתים: \forall ("לכל"), \exists ("קיים")

- סימני פיסוק: סוגר שמאלי (וסוגר ימני).

- שפות שונות נבדלות זו מזו במרכיבים הבאים:

- רשימה (סופית) של קבועים: a, b, c .

- רשימה (סופית) של סמני פונקציה: f, g, h + מס' ארגומנטים (arity).

- רשימה (סופית) של סמני יחס: R, P, Q + מס' ארגומנטים (arity).

תחשיב הפרדיקטים - תחביר (Syntax)

- בהינתן שפה מסדר ראשון L , ניתן תאור הבעיה יעשה ע"י נוסחאות:
- אוסף שמות העצם (terms) ב- L מורכב מ-
 - קבוצת המשתנים של L ,
 - קבוצת הקבועים ב- L ,
 - אם t_1, t_2, \dots, t_n הם n שמות עצם, ו- f היא פונקציה n -מקומית, אזי גם $f(t_1, t_2, \dots, t_n)$ הוא שם עצם.
- אוסף הנוסחאות החוקיות (well-formed formulae) ב- L מורכב מ-
 - אם t_1, t_2, \dots, t_n הם n שמות עצם, ו- R הוא סימן יחס n -מקומי, אזי $R(t_1, t_2, \dots, t_n)$ היא נוסחה (אטומית) ב- L .
 - אם Ψ ו- Φ הן נוסחאות חוקיות ב- L , ו- x הוא משתנה, אזי גם $\neg\Psi, \Phi\wedge\Psi, \Phi\vee\Psi, \Phi\rightarrow\Psi, \Phi\leftrightarrow\Psi, \exists x\Psi, \forall x\Psi$ הן נוסחאות ב- L .

תחשיב הפרדיקטים – תחביר

תאור BNF (Backus-Naur Form)

General

Formula:	Atomic_formula Formula Binary_Connective Formula Quantifier Variable Formula \neg Formula (Formula)
Atomic_formula:	Predicate(Term,...) Term = Term
Term:	Variable Constant Function(Term,...)
Binary_Connective:	\wedge \vee \rightarrow \leftrightarrow
Quantifier:	\forall \exists
Variable:	x y z x_1 ...

Language dependent

Constant:	John 0 \emptyset ...
Function:	f Mother_of + First_Element max ...
Predicate:	R Before \subset $<$ Has_Color ...

ייצוג ידע ע"י שפות מסדר ראשון

בדר"כ תהליך הייצוג הוא דו שלבי:

— קונספטואליזציה: בחירת שפה מתאימה לתיאור בעיה נתונה.

— הצרנה: תאור נתוני הבעיה ע"י נוסחאות בשפה שנבחרה.

בתהליך הקונספטואליזציה יובאו בחשבון שיקולים שונים, למשל:

— רמות הייצוג (הפרוט).

א- $\text{Prime}(x)$ (Prime – סימן יחס חד מקומי)

ב- $\forall x (\exists y (p = x*y) \rightarrow (p=x \vee x=1))$

— בחירת הקבועים ושילובם עם סמני היחס.

א- $\text{Black}(x)$ (Black – סימן יחס חד מקומי)

ב- $\text{Color}(x, \text{black})$ (Color – סימן יחס דו-מקומי, black – קבוע).

• במקרה א' לא צריך לציין ש- $\text{On}(\text{black}, \text{gray})$ איננו פסוק חוקי.

• במקרה ב' נחסוך כללים כ- $\text{Black}(x) \rightarrow \neg \text{Gray}(x) \wedge \neg \text{White}(x)$

ייצוג ידע ע"י שפות מסדר ראשון

דוגמאות:

(א) שפות לייצוג תורת המספרים:

למשל: $L = (\{0\}, \{+, *, S\}, \{=, <\})$.

דוגמאות לטענות ב-L:

$$\forall x (x = + (x, 0)),$$

$$\forall x \forall y (+(x, y) = +(y, x))$$

$$\forall x \forall y \forall z (+(x, +(y, z)) = +(+(x, y), z))$$

$$\forall x \exists y (y = *(x, x)),$$

$$\forall x (\exists y (x = S(y, y)) \rightarrow < (0, x)).$$



ייצוג ידע ע"י שפות מסדר ראשון

דוגמאות:

(א) שפות לייצוג תורת המספרים:

למשל: $L = (\{0\}, \{+, *, S\}, \{=, <\})$

דוגמאות לטענות ב-L:

$$\forall x (x = + (x, 0)),$$

$$\forall x \forall y (+(x, y) = +(y, x))$$

$$\forall x \forall y \forall z (+(x, +(y, z)) = +(+(x, y), z))$$

$$\forall x \exists y (y = *(x, x)),$$

$$\forall x (\exists y (x = S(y, y)) \rightarrow < (0, x)).$$

שפות לייצוג מעגלים בוליאניים:

קבועים: $X1, X2, A1, A2, O1, 0, 1, 2$

סימני פונקציה: $\text{In}(i,x)$ = ערך הכניסה ה- i לשער x ,

$\text{Out}(x)$ = ערך הפלט של שער x .

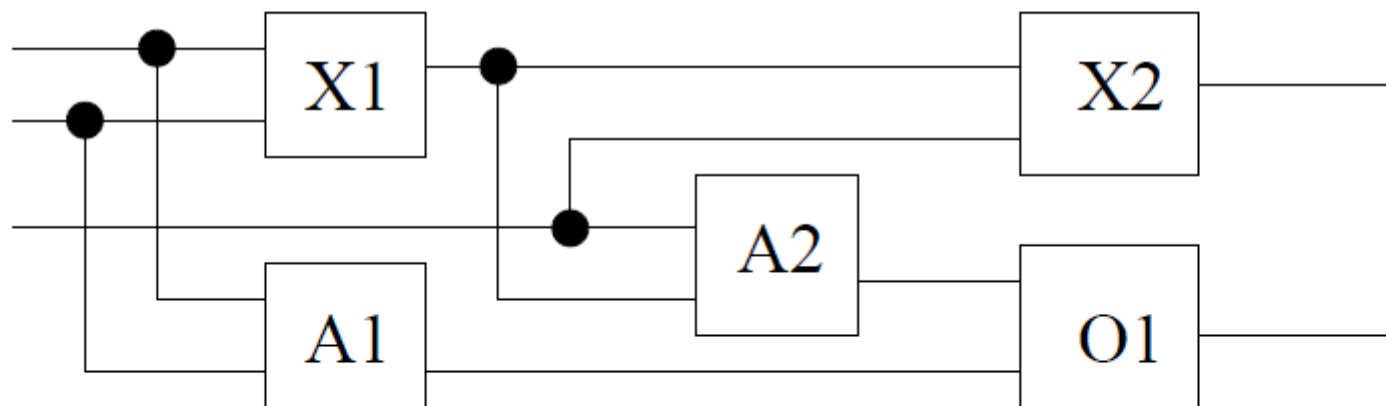
סמני יחס: $\text{Xorg}(x), \text{Andg}(x), \text{Org}(x)$ = תאור השערים,

$\text{Conn}(x,y)$ = אובייקט x מחובר לאובייקט y ,

$\text{Val}(x,y)$ = הערך של אובייקט x הוא y .



נתאר בשפה זו מחבר מלא (Full Adder):



תאור (חלקי) של המעגל:

- תאור השערים:

$\text{AndG}(A1), \text{Andg}(A2), \text{Xorg}(X1), \text{Xorg}(X2), \text{Org}(O1)$

- תאור התנהגות השערים:

$\forall x \{ [\text{AngG}(x) \wedge \exists i \text{Val}(\text{In}(i,x), 0)] \rightarrow \text{Out}(x) = 0 \}, \dots$

- תאור החיבורים:

$\text{Conn}(\text{Out}(X1), \text{In}(1,X2)), \text{In}(1,X1) = \text{In}(1,A1), \dots$

- אקסיומות כלליות לגבי התנהגות מעגלים בוליאניים:

$\forall x \forall y \forall z \{ [\text{Conn}(x,y) \wedge \text{Val}(x,z)] \rightarrow \text{Val}(y,z) \},$

$\forall x \forall y \{ \text{Val}(x,y) \rightarrow (y = 0 \vee y = 1) \}, \dots$

- תאור קלט\פלט:

$\text{Val}(\text{In}(1,X1),1), \text{Val}(\text{Out}(X2),0), \dots$



PROPOSITIONAL LOGIC: SYNTAX

- Propositional logic is the simplest logic – illustrates basic ideas
-
- The proposition symbols P_1, P_2 etc are sentences
 - If S is a sentence, $\neg S$ is a sentence (negation)
 -
 - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
 -
 - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
 -
 - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
 -
 - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)
 -



PROPOSITIONAL LOGIC: SEMANTICS

Each model specifies true/false for each proposition symbol

E.g. $P_{1,2}$ false $P_{2,2}$ true $P_{3,1}$ false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model m :

$\neg S$	is true iff	S is false
$S_1 \wedge S_2$	is true iff	S_1 is true and S_2 is true
$S_1 \vee S_2$	is true iff	S_1 is true or S_2 is true
$S_1 \Rightarrow S_2$	is true iff	S_1 is false or S_2 is true
true		
i.e.,	is false iff	S_1 is true and S_2 is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true
true		

Simple recursive process evaluates an arbitrary sentence, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{true} \wedge (\text{true} \vee \text{false}) = \text{true} \wedge \text{true} = \text{true}$$



TRUTH TABLES FOR CONNECTIVES

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>



תחשיב הפרדיקטים - משמעות (Semantics)

- פרוש לשפה מסדר ראשון: תחום רלבנטי ("עולם הפרוש") + מובן לכל סימן לא לוגי (קבועים, סמני פונקציה, וסימני יחס) במסגרת התחום הנ"ל.

דוגמא: פרוש $M1$ לשפת תורת המספרים $L = (\{0\}, \{+, *, S\}, \{=, <\})$:

- עולם הפרוש, $|M1|$, יהיה המספרים הטבעיים N .
- פרוש הקבוע 0 ב- $M1$ הוא המספר (הטבעי) 0 .
- הפרושים לסימני הפונקציה $+, *, S$ ב- $M1$ הם פונקציית העוקב, כפל וחיבור (בהתאמה) של מספרים טבעיים.
- סימני היחס $<$ ו- $=$ מתפרשים ע"י $M1$ כיחס "קטן מ-" ושוויון (בהתאמה) בין זוגות מספרים טבעיים.

פרוש לשפה הוא כמובן לא יחיד. פרושים אפשריים אחרים ל- L :
 $|M2| = R$ (הרציונליים + המובן הרגיל), $|M3| =$ עולם מחרוזות מעל א"ב Σ .

תחשיב הפרדיקטים - משמעות (Semantics)

- **השמות:** פונקציות שנותנות לכל משתנה ערך (איבר) בעולם הפרוש.
- בהינתן פרוש M והשמה v ב- M נוכל לאמר מתי נוסחה Ψ **נכונה** ביחס אליהם.

דוגמאות:

- הנוסחה $\forall y (x \leq y)$ נכונה בטבעיים ($|M| = \mathbb{N}$) רק עבור ההשמה $v(x) = 0$.
נוסחה זו איננה נכונה בממשיים ($|M| = \mathbb{R}$) (עבור אף השמה).
- הנוסחה $\forall y \exists x (x > y)$ נכונה בטבעיים ובממשיים (לכל השמה).
- הנוסחה $\forall y [(0 < y) \rightarrow \exists x (y = *(x, x))]$ נכונה בממשיים (לכל השמה)
אבל איננה נכונה בטבעיים.



תחשיב הפרדיקטים - משמעות (Semantics)

- M הוא מודל של נוסחא Ψ אם Ψ נכונה בפרוש M עבור כל השמה v ב- M .
 M הוא מודל של קבוצת נוסחאות Γ , אם הוא מודל של כל נוסחה $\Psi \in \Gamma$.

דוגמאות:

- M כאשר $|M| = N$ והמובן הרגיל לסימן $<$, הוא מודל של $\forall y \exists x (x > y)$
- M כאשר $|M| = R$ והמובן הרגיל של סימני היחס והפונקציה, הוא מודל של $\forall y [(0 < y) \rightarrow \exists x (y = *(x, x))]$

- נוסחה Ψ נובעת לוגית מקבוצת נוסחאות Γ (סימון: $\Gamma \models \Psi$), אם כל מודל של Γ הוא גם מודל של Ψ .

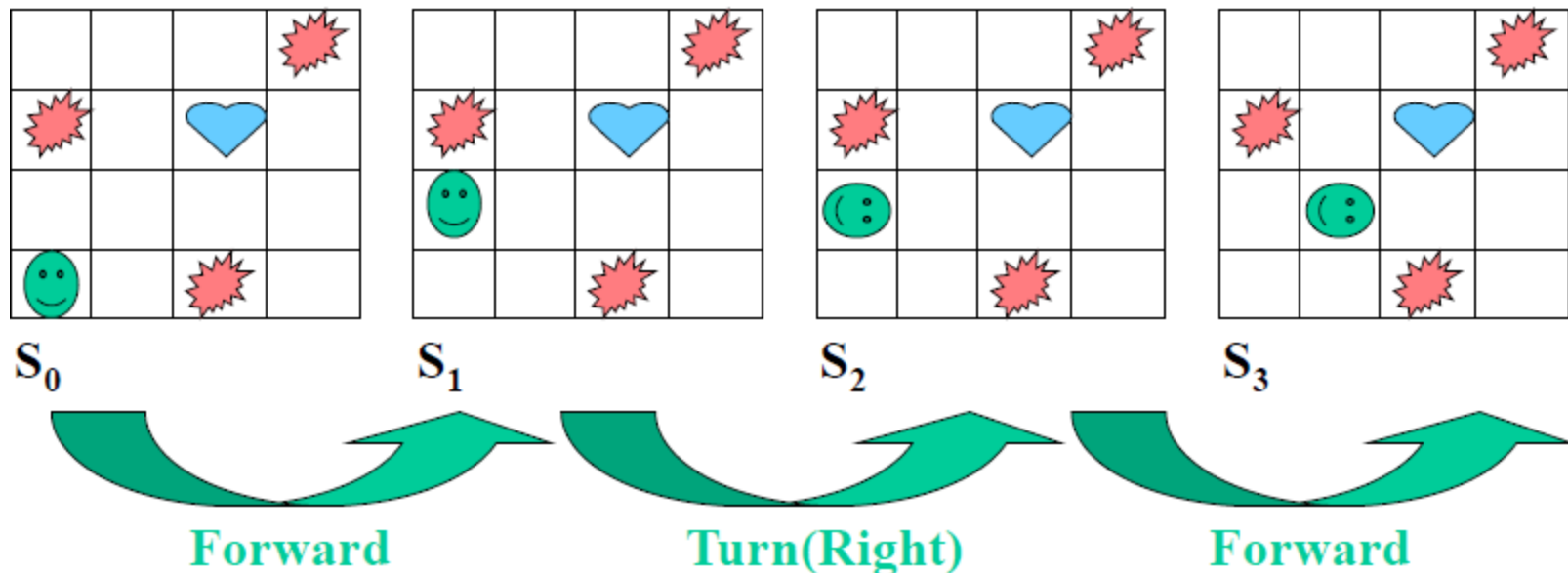
דוגמא:

$$\{\forall x \forall y \forall z (Above(x, y) \wedge On(y, z) \rightarrow Above(x, z)), \\ \forall x \forall y (On(x, y) \rightarrow Above(x, y)), On(A, B), On(B, C)\} \models Above(A, C)$$



דוגמא: תחשיב המצבים (Situation Calculus)

- מקרה פרטי של לוגיקה מסדר ראשון לתיאור עולם משתנה.
(מענה לבעיית המסגרת; תאור השנויים בלבד).
- הרעיון: תאור סדרת מצבים (situations) של העולם. כל מצב נגזר מקודמו ע"י פעולות (actions).



INFERENCE BY ENUMERATION

- Depth-first enumeration of all models is sound and complete

- ```
function TT-ENTAILS?(KB, α) returns true or false
 symbols \leftarrow a list of the proposition symbols in KB and α
 return TT-CHECK-ALL(KB, α , symbols, [])

function TT-CHECK-ALL(KB, α , symbols, model) returns true or false
 if EMPTY?(symbols) then
 if PL-TRUE?(KB, model) then return PL-TRUE?(α , model)
 else return true
 else do
 P \leftarrow FIRST(symbols); rest \leftarrow REST(symbols)
 return TT-CHECK-ALL(KB, α , rest, EXTEND(P, true, model)) and
 TT-CHECK-ALL(KB, α , rest, EXTEND(P, false, model))
```

- For  $n$  symbols, time complexity is  $O(2^n)$ , space complexity is  $O(n)$



# LOGICAL EQUIVALENCE

- Two sentences are **logically equivalent** iff true in same models:  $\alpha \equiv \beta$  iff  $\alpha \models \beta$  and  $\beta \models \alpha$
- $(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$  commutativity of  $\wedge$
- $(\alpha \vee \beta) \equiv (\beta \vee \alpha)$  commutativity of  $\vee$
- $((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$  associativity of  $\wedge$
- $((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$  associativity of  $\vee$
- $\neg(\neg\alpha) \equiv \alpha$  double-negation elimination
- $(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$  contraposition
- $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$  implication elimination
- $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$  biconditional elimination
- $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$  de Morgan
- $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$  de Morgan
- $(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$  distributivity of  $\wedge$  over  $\vee$
- $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$  distributivity of  $\vee$  over  $\wedge$

# VALIDITY AND SATISFIABILITY

A sentence is **valid** if it is true in **all** models,  
e.g., *True*,  $A \vee \neg A$ ,  $A \Rightarrow A$ ,  $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the **Deduction Theorem**:  
 $KB \models \alpha$  if and only if  $(KB \Rightarrow \alpha)$  is valid

A sentence is **satisfiable** if it is true in **some** model  
e.g.,  $A \vee B$ ,  $C$

A sentence is **unsatisfiable** if it is true in **no** models  
e.g.,  $A \wedge \neg A$

Satisfiability is connected to inference via the following:  
 $KB \models \alpha$  if and only if  $(KB \wedge \neg \alpha)$  is unsatisfiable





# PROOF METHODS

- Proof methods divide into (roughly) two kinds:
  - Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - 
    - **Proof** = a sequence of inference rule applications  
Can use inference rules as operators in a standard search algorithm
    - 
    - Typically require transformation of sentences into a **normal form**
  - Model checking
    - truth table enumeration (always exponential in  $n$ )
    - 
    - improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
    - 
    - heuristic search in model space (sound but incomplete)  
e.g., min-conflicts-like hill-climbing algorithms



# RESOLUTION

- בסעיף זה נציג כלל היסק הנקרא רזולוציה המניב אלגוריתם היסק שלם כאשר מצרפים אליו אלגוריתם חיפוש שלם.
- בהמשך הסעיף נראה את כלל רזולוצית היחידה ואת הרחבתו לכלל הרזולוציה המלאה, המקבל שתי פסוקיות ויוצר פסוקית חדשה המכילה את כל הליטרלים משתי הפסוקיות המקוריות פרט לשני ליטרלים משלימים. פסוקית התוצאה חייבת להכיל רק עותק אחד של כל ליטרל.
- 
- כלל הרזולוציה הוא נאות, והוא מהווה בסיס למשפחת אלגוריתמי היסק שלמים.
- 
- דוגמה:
- "קנדה היא באמריקה הצפונית, או קנדה היא באיחוד האירופי".
- "קנדה אינה באמריקה הצפונית, או קנדה היא ב-British Commonwealth".
- ניתן להסיק מהנ"ל: "קנדה היא באיחוד האירופי, או קנדה היא ב-British Commonwealth".



# מערכות הוכחה (Proof Systems)

- המטרה: לתאר תהליך שמאפשר ליצור (באופן אוטומטי) הוכחה במערכת R של טענה  $\Psi$  מאוסף הנחות (יתכן ריקן)  $\Gamma$ .
- הוכחה של  $\Psi$  במערכת M המבוססת על הנחות ב- $\Gamma$  היא רשימה סופית של נוסחאות  $\{A_1, A_2, \dots, A_n\}$  כך ש-  
$$A_n = \Psi \quad \blacksquare$$
  
כל  $A_i$  ( $1 \leq i \leq n-1$ ) הוא הנחה ( $A_i \in \Gamma$ ), או אקסיומה ב-M, או מתקבל מנוסחאות קודמות (מתוך  $A_1, A_2, \dots, A_{i-1}$ ) ע"י הפעלת כלל היסק ב-M.
- סימון:  $\Gamma \vdash_M \Psi$  אם קיימת הוכחה של  $\Psi$  במערכת M המבוססת על הנחות ב- $\Gamma$ .

# מערכות הוכחה (Proof Systems)

• דוגמא: נתונה המערכת M הבאה:

- אקסיומות:  $Ax.1$  :  $A \rightarrow (B \rightarrow A)$   
 $Ax.2$  :  $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$   
כלל היסק:  $M.P$  : אם ידוע ש-A וש-  $A \rightarrow B$ , אזי נסיק B.

• נראה ש-  $\{A \rightarrow B, B \rightarrow C\} \vdash_M A \rightarrow C$

- |                                                                                                      |           |
|------------------------------------------------------------------------------------------------------|-----------|
| 1. $(B \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$                                 | אקסיומה   |
| 2. $B \rightarrow C$                                                                                 | הנחה      |
| 3. $A \rightarrow (B \rightarrow C)$                                                                 | M.P (1,2) |
| 4. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$ | אקסיומה   |
| 5. $(A \rightarrow B) \rightarrow (A \rightarrow C)$                                                 | M.P (3,4) |
| 6. $A \rightarrow B$                                                                                 | הנחה      |
| 7. $A \rightarrow C$                                                                                 | M.P (5,6) |

# מערכות הוכחה (Proof Systems)

תכונות שנרצה שמערכת הוכחה M תקיים:

– **נאותות (soundness):** כל מה שיכיה ב-M ע"ס  $\Gamma$  נובע לוגית מ- $\Gamma$ :

$$\Gamma \vdash_M \Psi \Rightarrow \Gamma \models \Psi$$

– **שלמות (completeness):** קיימת הוכחה ב-M (ע"ס  $\Gamma$ ) לכל מה שנובע לוגית מ- $\Gamma$ :

$$\Gamma \models \Psi \Rightarrow \Gamma \vdash_M \Psi$$



# הוכחה ברזולוציה (Resolution)

- שיטת הוכחה המבוססת על **הפרכה** (refutation): על מנת להראות טענה מסוימת מניחים את שלילתה ומגיעים לסתירה עם הנחות קיימות.
- נגדיר מערכת הוכחה  $R$  (רזולוציה) ובכדי להוכיח את  $\Psi$  מקבוצת הנחות  $\Gamma$ , נוכיח ב- $R$  שאוסף ההנחות  $\Gamma \cup \{\neg\Psi\}$  מוביל לסתירה.
- תהליך זה שלבי:
  - המרת הנתונים ושלילתה של הטענה המוכחת לצורה של **פסוקיות** (clause form).
  - הפעלת תהליך הרזולוציה להפרכת המטרה, כלומר: הוכחת סתירה במערכת  $R$  מאוסף הפסוקיות שנוצרו בשלב הקודם.

# שלב א': המרת נוסחאות לפסוקיות

- (ייצוג אחיד שבו כל הכמתים מופיעים בראש בנוסחה, ושרמת הכינון של מרכיבי הנוסחה אחידה).

## • הגדרה:

— נוסחה בצורה קוניונקטיבית נורמאלית

(CNF - conjunctive normal form) היא מהצורה הבאה:

$$(L_{1,1} \vee L_{1,2} \vee \dots \vee L_{1,n_1}) \wedge \dots \wedge (L_{m,1} \vee L_{m,2} \vee \dots \vee L_{m,n_m})$$

כאשר כל  $L_{i,j}$  הוא ליטרל, כלומר: נוסחה אטומית או שלילתה.

— נוסחה פסוקית (clause) היא נוסחה קוניונקטיבית נורמאלית ללא

קוניונקציות, כלומר:  $L_1 \vee L_2 \vee \dots \vee L_n$ .

- התהליך: תרגם נוסחה לצורה קוניונקטיבית נורמאלית מכומתת אוניברסלית, וצור את קבוצת הפסוקיות המורכבת מ(כימות אוניברסלי של) הקוניונקטים שהתקבלו.

# אלגוריתם המרה לפסוקיות

- שלב 1: הורדת גרירות. ע"י שימוש בשקילות  $p \rightarrow q = \neg p \vee q$
- שלב 2: הקטנת טווח הפעולה של השלילות לביטוי יחיד. ע"י שימוש בכללים:  
$$\neg \neg p = p, \quad \neg(p \wedge q) = \neg p \vee \neg q, \quad \neg(p \vee q) = \neg p \wedge \neg q$$
$$\neg \forall x P(x) = \exists x \neg P(x), \quad \neg \exists x P(x) = \forall x \neg P(x)$$
  
(בסוף שלב זה הנוסחה בצורת **NNF – Negation Normal Form**)
- שלב 3: שינוי שמות משתנים קשורים, כך שכל כמת יקשור משתנה שונה.  
$$\forall x P(x) \vee \forall x Q(x) \Rightarrow \forall x P(x) \vee \forall y Q(y), \quad \dots$$
- שלב 4: הזזת כל הכמתים לראשית הנוסחה ללא שנוי הסדר שלהם.  
(ניתן לעשות זאת כי ששנינו שמות משתנים קשורים ואין גרירות בנוסחה).  
קבלנו נוסחה בצורה קידומית נורמאלית (prenex normal form).



# אלגוריתם המרה לפסוקיות, המשך

- שלב 5: הורדת כמתים ישיים ע"י הוספת **Skolem** קבועי **Skolem** ופונקציות

$$\exists x \text{ President}(x) \Rightarrow \text{President}(a),$$

$$\forall x \exists y \text{ FatherOf}(y, x) \Rightarrow \forall x \text{ FatherOf}(f(x), x)$$

$$\forall x \forall y \forall z \exists v \forall w R(x, y, z, v, w) \Rightarrow \forall x \forall y \forall z \forall w R(x, y, z, f(x, y, z), w)$$

(הנוסחאות המתקבלות בשלב 5 אינן שקולות לאלו של שלב 4. למשל,

$$P(a) \vee P(b) \models \exists x P(x) \quad \text{בעוד ש-} P(a) \vee P(b) \not\models P(c). \text{ אבל תכונת}$$

הספיקות נשמרת:  $\Psi$  ספיקה אמ"מ  $\text{Skolem}(\Psi)$  ספיקה, וזה מה שחשוב)

- שלב 6: סילוק כמתים מראשית הנוסחאות (כל המשתנים מכומתים אוניברסלית).

- שלב 7: הפיכת הנוסחאות לצורת **CNF** (קוניונקציות של דיסיונקציות) ע"י

הפעלת חוקים קומוטטיביים ודיסטריבוטיביים:

$$p \wedge (q \wedge r) = (p \wedge q) \wedge r, \quad p \vee (q \vee r) = (p \vee q) \vee r,$$

$$p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$$

# אלגוריתם המרה לפסוקיות, המשך

- שלב 8: ביטול קוניונקציות ע"י "שבירת" נוסחאות CNF לקוניונקטים.

- שלב 9: שינוי שמות משתנים כך שפסוקיות שונות ישתמשו בשמות משתנים שונים. (ניתן לעשות זאת כיון ש-  $\forall x(P(x) \wedge Q(x)) \equiv \forall xP(x) \wedge \forall xQ(x)$  ולכן אין קשר בין המשתנים גם אם מקורם מאותה נוסחה).

קבלנו אלגוריתם שבהינתן קבוצת נוסחאות  $\Gamma$  מחשב קבוצת פסוקיות  $\text{Clause}(\Gamma)$  כך ש-  $\Gamma$  ספיקה אם"מ  $\text{Clause}(\Gamma)$  ספיקה

תרגיל: המירו לקבוצת פסוקיות כל אחת מהנוסחאות הבאות:

$$\forall x \exists y \forall z (P(x,y) \rightarrow (Q(z) \vee \neg R(x,y,z)))$$

$$\forall x (P(x) \wedge Q(x)) \rightarrow \neg \forall y (P(y) \wedge \neg \exists z R(y,z))$$

# אלגוריתם המרה לפסוקיות

## דוגמא

$$\Psi = \forall x R(x) \vee \forall y \exists x \neg (P(y) \rightarrow Q(y,x))$$

1.  $\forall x R(x) \vee \forall y \exists x \neg (\neg P(y) \vee Q(y,x))$
2.  $\forall x R(x) \vee \forall y \exists x (P(y) \wedge \neg Q(y,x))$
3.  $\forall x R(x) \vee \forall y \exists z (P(y) \wedge \neg Q(y,z))$
4.  $\forall x \forall y \exists z (R(x) \vee (P(y) \wedge \neg Q(y,z)))$
5.  $\forall x \forall y (R(x) \vee (P(y) \wedge \neg Q(y,f(x,y))))$
6.  $R(x) \vee (P(y) \wedge \neg Q(y,f(x,y)))$
7.  $(R(x) \vee P(y)) \wedge (R(x) \vee \neg Q(y,f(x,y)))$
8.  $\{ R(x) \vee P(y), R(x) \vee \neg Q(y,f(x,y)) \}$
9. **Clause( $\Psi$ ) =  $\{ R(x) \vee P(y), R(u) \vee \neg Q(v,f(u,v)) \}$**

# שלב ב': תהליך הרזולוציה

- נניח תחילה שבנוסחאות אין משתנים וסמני פונקציה.

• הרעיון:  $winter \vee summer, \neg winter \vee cold \Rightarrow summer \vee cold$

• כלל הרזולוציה:  $D1 \vee p \vee D2, D3 \vee \neg p \vee D4 \Rightarrow D1 \vee D2 \vee D3 \vee D4$

• האלגוריתם: (הוכחה ברזולוציה של  $\Psi$  מקבוצת הנחות  $\Gamma$ ).

– תרגם את כל הנוסחאות ב-  $\Gamma \cup \{\neg \Psi\}$  לפסוקיות

– בצע עד להגעה לסתירה (= פסוקית ריקה), או שלא ניתן להתקדם עוד:

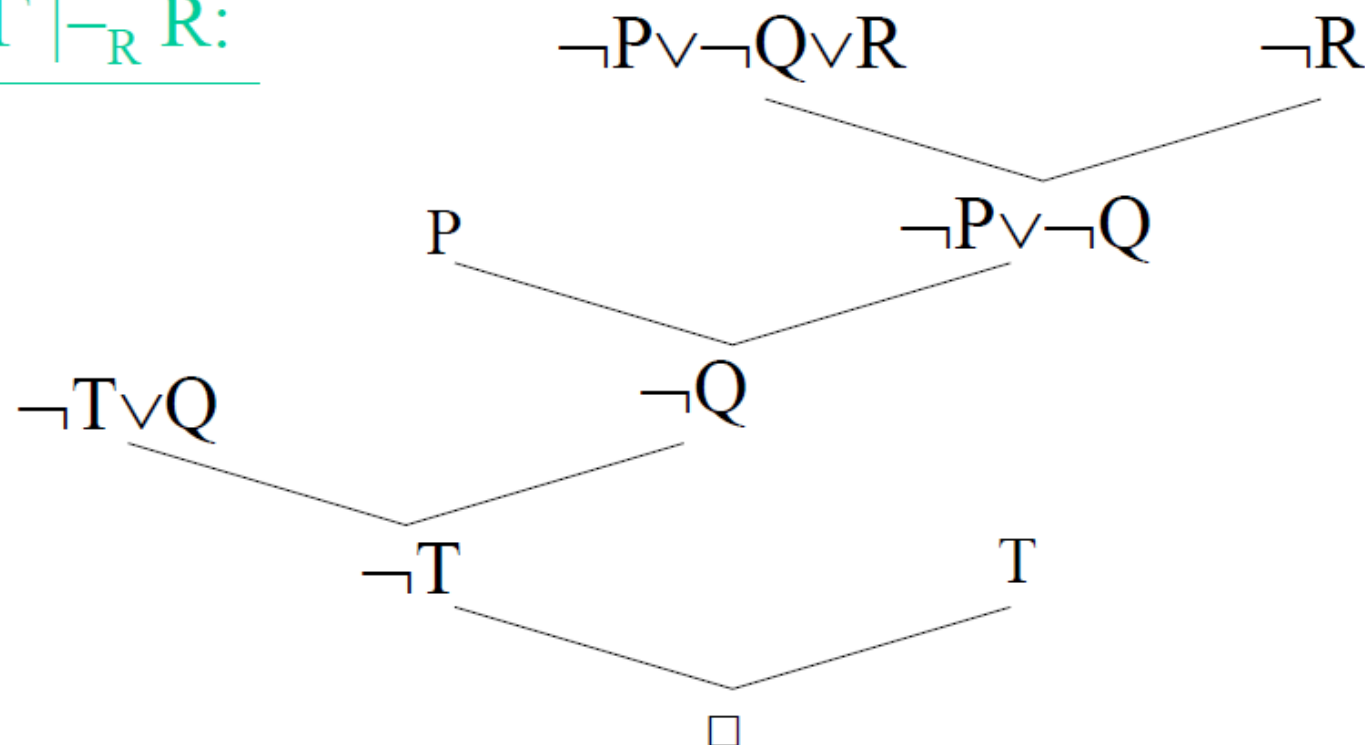
- בחר שתי פסוקיות (= פסוקיות אב) והפעל את כלל הרזולוציה עליהן ליצירת פסוקית בן. (= דיסיונקצית הליטרלים בפסוקיות האב בהורדת כל הליטרלים שיש להם הפכי).
- אם פסוקית הבן ריקה - נמצאה סתירה. אחרת - הוסף פסוקית זו למאגר.

# תהליך הרזולוציה - דוגמא

$$\Gamma = \{ P, (P \wedge Q) \rightarrow R, (S \vee T) \rightarrow Q, T \}$$

$$\text{Clause}(\Gamma) = \{ P, \neg P \vee \neg Q \vee R, \neg S \vee Q, \neg T \vee Q, T \}$$

$\Gamma \vdash_R R$ :



# RESOLUTION ALGORITHM

- Proof by contradiction, i.e., show  $KB \wedge \neg \alpha$  unsatisfiable

```
function PL-RESOLUTION(KB, α) returns true or false
 $clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg \alpha$
 $new \leftarrow \{ \}$
 loop do
 for each C_i, C_j in $clauses$ do
 $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
 if $resolvents$ contains the empty clause then return true
 $new \leftarrow new \cup resolvents$
 if $new \subseteq clauses$ then return false
 $clauses \leftarrow clauses \cup new$
```



# RESOLUTION

Conjunctive Normal Form (CNF)  
 conjunction of disjunctions of literals  
 clauses

E.g.,  $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- Resolution inference rule (for CNF):

- 

$$\frac{\ell_i \vee \dots \vee \ell_k, \quad m_1 \vee \dots \vee m_n}{\ell_i \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

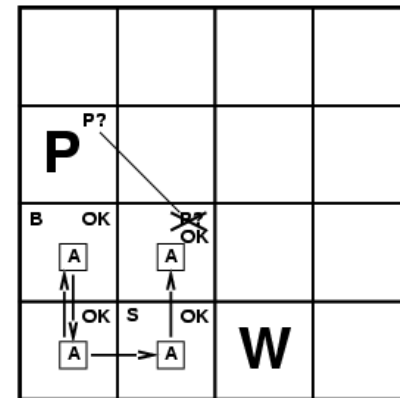
where  $\ell_i$  and  $m_j$  are complementary literals.

E.g.,  $P_{1,3} \vee P_{2,2}, \quad \neg P_{2,2}$

$P_{1,3}$

- Resolution is sound and complete for propositional logic

- 



# RESOLUTION

Soundness of resolution inference rule:

$$\frac{\begin{array}{l} \neg(\ell_1 \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k) \Rightarrow \ell_i \\ \neg m_j \Rightarrow (m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \\ \vee \dots \vee m_n) \end{array}}{\neg(\ell_1 \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k) \Rightarrow (m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)}$$





# CONVERSION TO CNF

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})\beta$$

1. Eliminate  $\Leftrightarrow$ , replacing  $\alpha \Leftrightarrow \beta$  with  $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ .

2.

$$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

2. Eliminate  $\Rightarrow$ , replacing  $\alpha \Rightarrow \beta$  with  $\neg\alpha \vee \beta$ .

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$$

3. Move  $\neg$  inwards using de Morgan's rules and double-negation:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \vee \neg P_{2,1}) \vee B_{1,1})$$

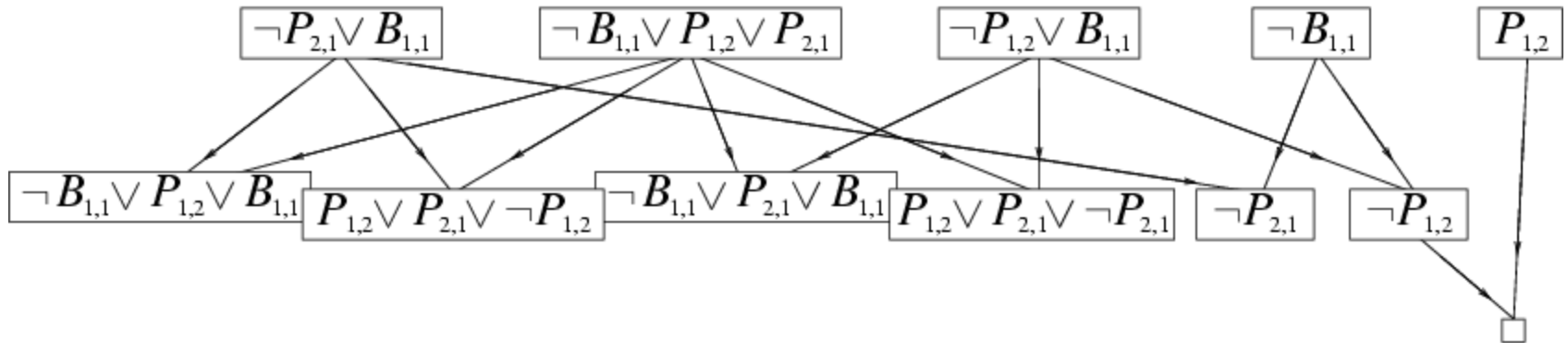
4. Apply distributivity law ( $\wedge$  over  $\vee$ ) and flatten:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$



# RESOLUTION EXAMPLE

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \quad \alpha = \neg P_{1,2}$
- 



# פסוקית HORN

- בסיסי ידע בעולם האמיתי מכילים לעתים קרובות רק פסוקיות מצורה מוגבלת הנקראת פסוקיות Horn, המאפשרת להשתמש באלגוריתם הסקה מוגבל ויעיל יותר מאלגוריתם הרזולוציה.
- פסוקית Horn היא דיסיונקציה של ליטרלים שאחד מהם לכל היותר הוא ליטרל חיובי.



# FORWARD AND BACKWARD CHAINING

- **Horn Form** (restricted)

KB = **conjunction** of **Horn clauses**

- Horn clause =
  - proposition symbol; or
  - (conjunction of symbols)  $\Rightarrow$  symbol
- E.g.,  $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

- **Modus Ponens** (for Horn Form): complete for Horn KBs

---

- 

$\alpha_1, \dots, \alpha_n,$

$\alpha_1 \wedge \dots \wedge \alpha_n \Rightarrow \beta$

$\beta$

- Can be used with **forward chaining** or **backward chaining**.
- These algorithms are very natural and run in **linear** time
- 



# FORWARD AND BACKWARD CHAINING

- אלגוריתם שרשור קדימה (FC) עבור תחשיב הפסוקים מופיע באיור 7.15, והוא רץ בזמן לינארי. האלגוריתם קובע האם סימבול יחיד (שאלתא) נובע מבסיס ידע של פסוקיות מוגדרות. אלגוריתם זה הוא **נאות ושלם**.
- אלגוריתם FC הוא דוגמה להנמקה (reasoning) מבוססת מידע, כלומר מתחיל במידע שידוע.
- סוכן יכול להשתמש באלגוריתם כדי להסיק מסקנות מקלטים שהוא מקבל מהעולם האמיתי (תחושות).



# FORWARD CHAINING

- Idea: fire any rule whose premises are satisfied in the *KB*,
  - add its conclusion to the *KB*, until query is found

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

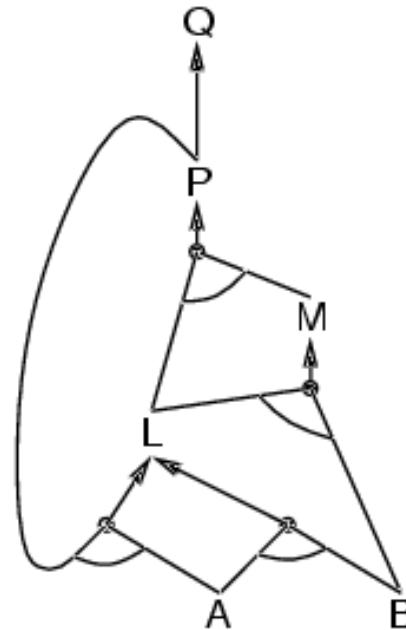
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

*A*

*B*



# FORWARD CHAINING ALGORITHM

```
function PL-FC-ENTAILS?(KB, q) returns true or false
 local variables: count, a table, indexed by clause, initially the number of premises
 inferred, a table, indexed by symbol, each entry initially false
 agenda, a list of symbols, initially the symbols known to be true

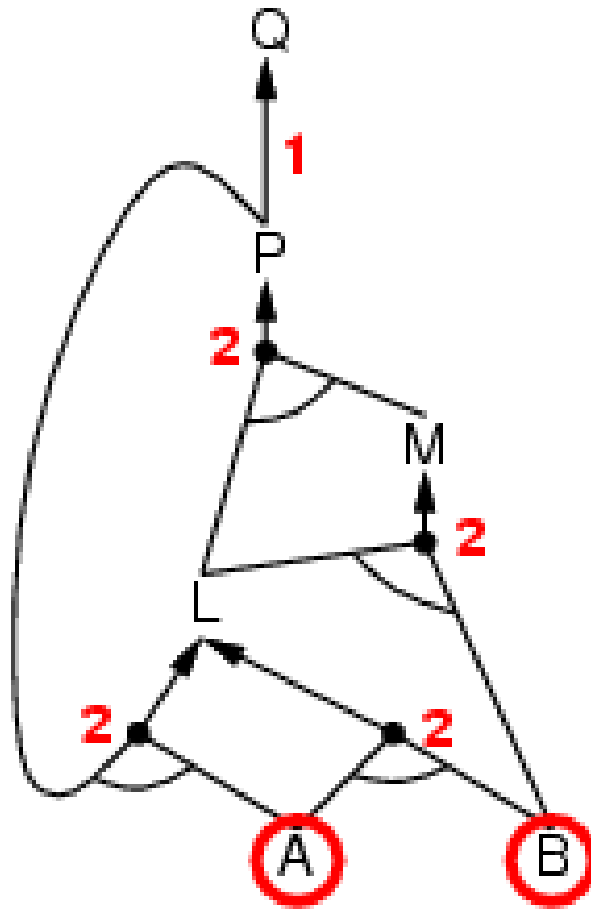
 while agenda is not empty do
 p ← POP(agenda)
 unless inferred[p] do
 inferred[p] ← true
 for each Horn clause c in whose premise p appears do
 decrement count[c]
 if count[c] = 0 then do
 if HEAD[c] = q then return true
 PUSH(HEAD[c], agenda)

 return false
```

- Forward chaining is sound and complete for Horn KB
- 

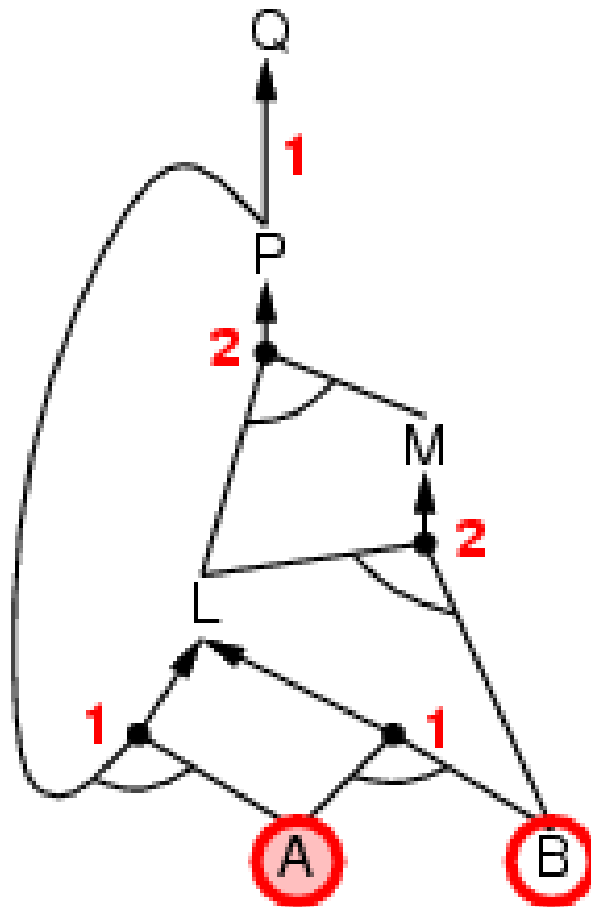


# FORWARD CHAINING EXAMPLE

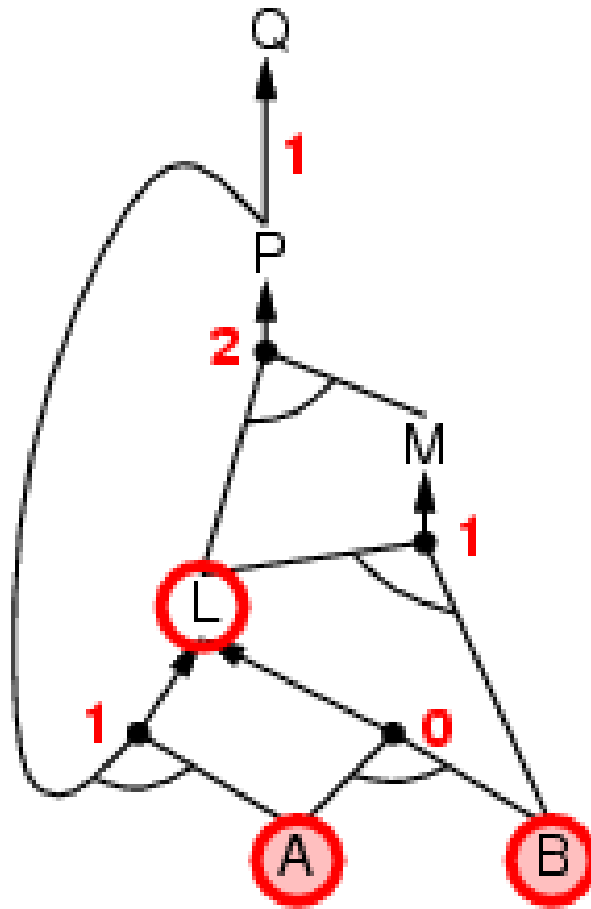




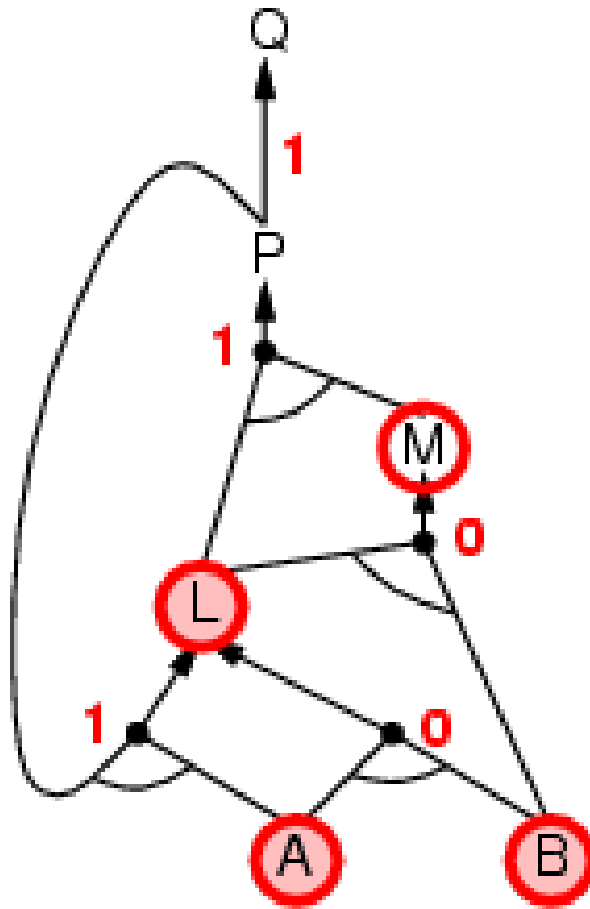
# FORWARD CHAINING EXAMPLE



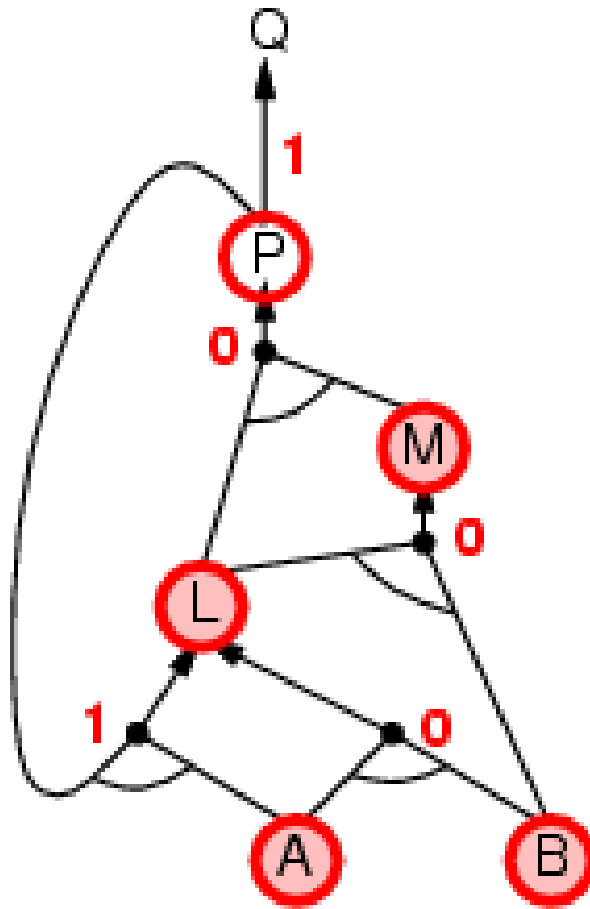
# FORWARD CHAINING EXAMPLE



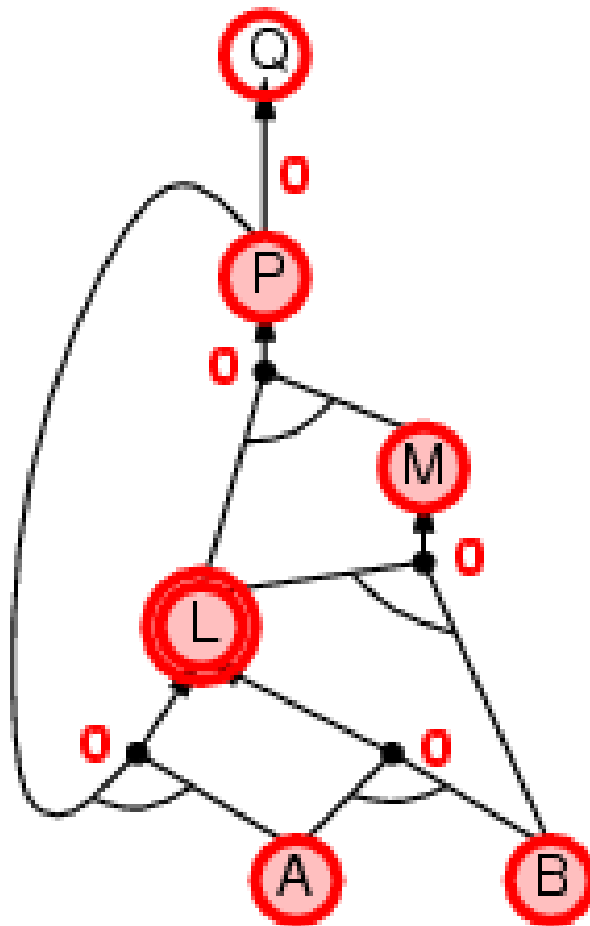
# FORWARD CHAINING EXAMPLE



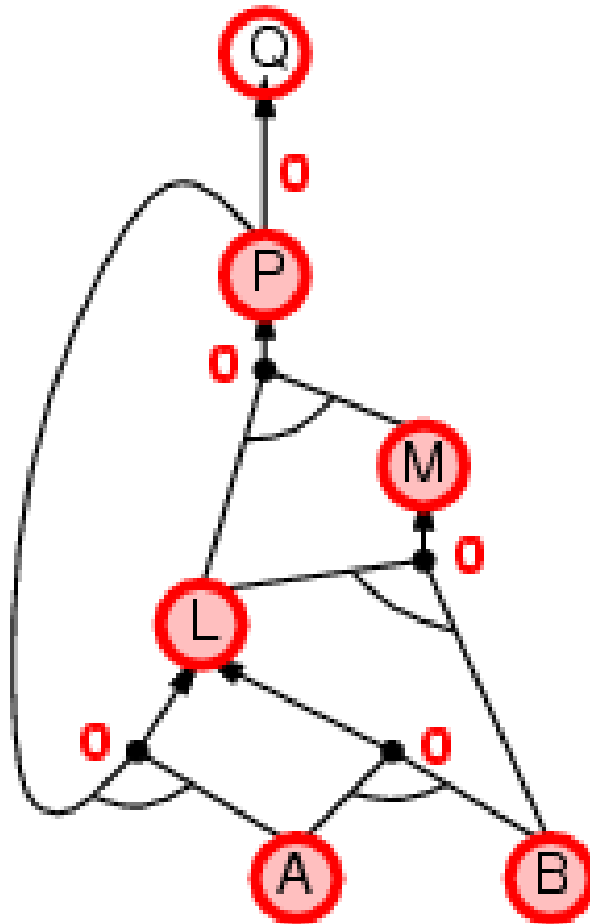
# FORWARD CHAINING EXAMPLE



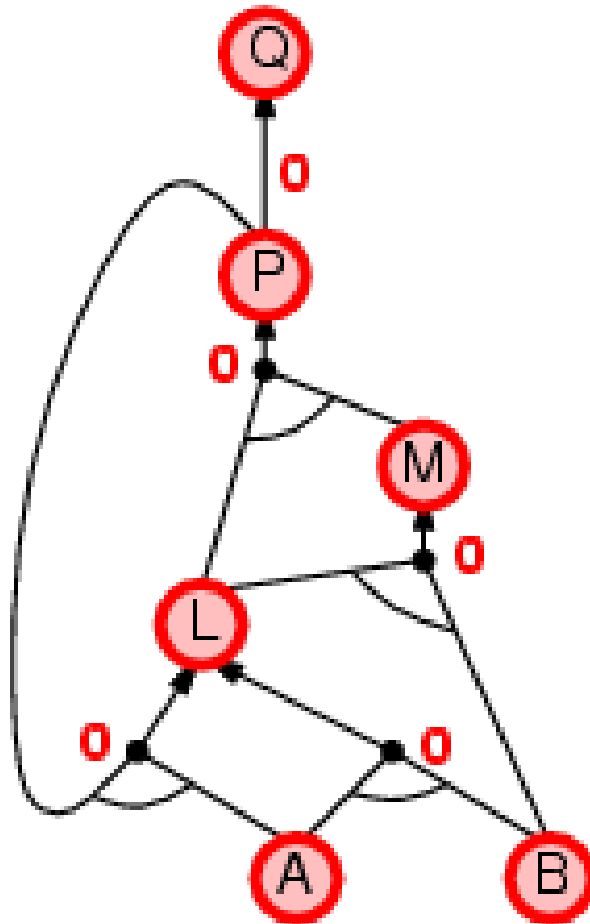
# FORWARD CHAINING EXAMPLE



# FORWARD CHAINING EXAMPLE



# FORWARD CHAINING EXAMPLE



# PROOF OF COMPLETENESS

- FC derives every atomic sentence that is entailed by  $KB$
- 1. FC reaches a **fixed point** where no new atomic sentences are derived
  - 2.
  2. Consider the final state as a model  $m$ , assigning true/false to symbols
  - 3.
  3. Every clause in the original  $KB$  is true in  $m$
  - 4.
  - $$a_1 \wedge \dots \wedge a_k \Rightarrow b$$
  4. Hence  $m$  is a model of  $KB$
  - 5.
  5. If  $KB \models q$ ,  $q$  is true in **every** model of  $KB$ , including  $m$
  - 6.





# BACKWARD CHAINING

Idea: work backwards from the query  $q$ :

to prove  $q$  by BC,  
    check if  $q$  is known already, or  
    prove by BC all premises of some rule concluding  $q$

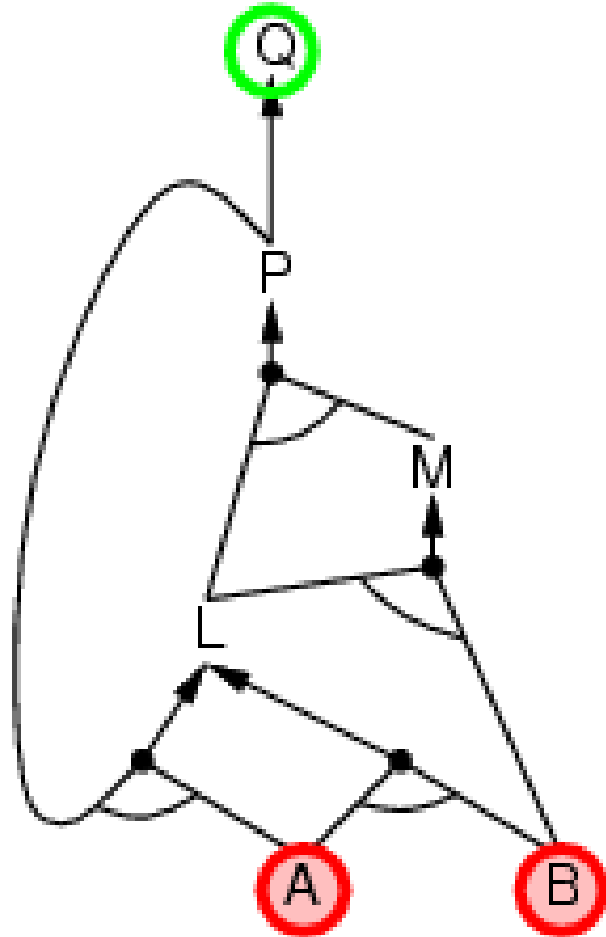
Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal

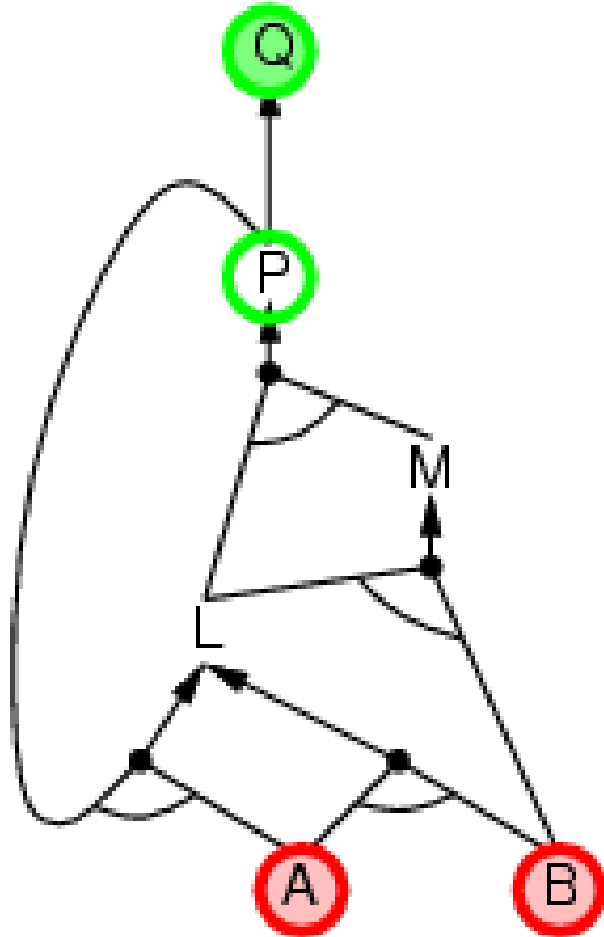
1. has already been proved true, or
2. has already failed
- 3.



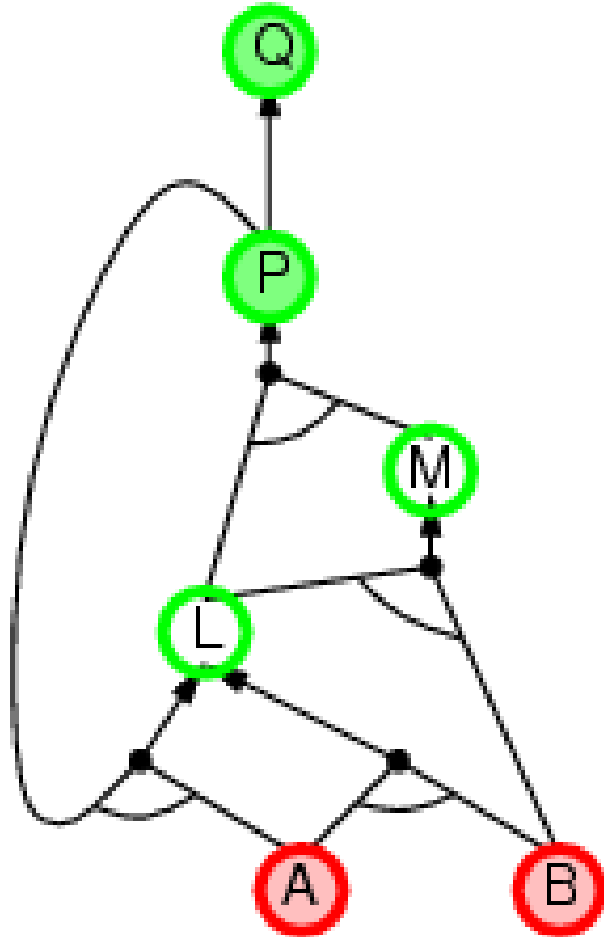
# BACKWARD CHAINING EXAMPLE



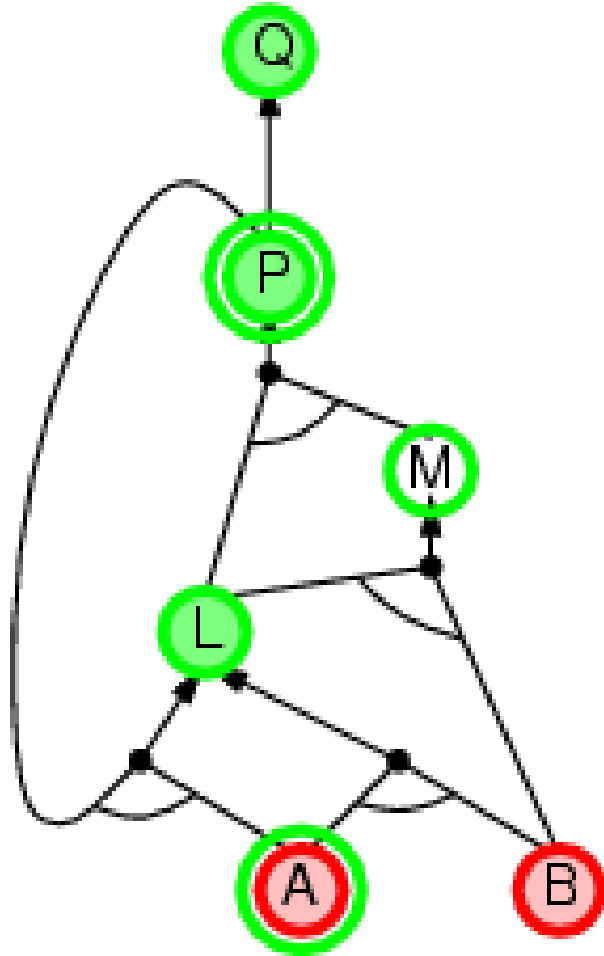
# BACKWARD CHAINING EXAMPLE



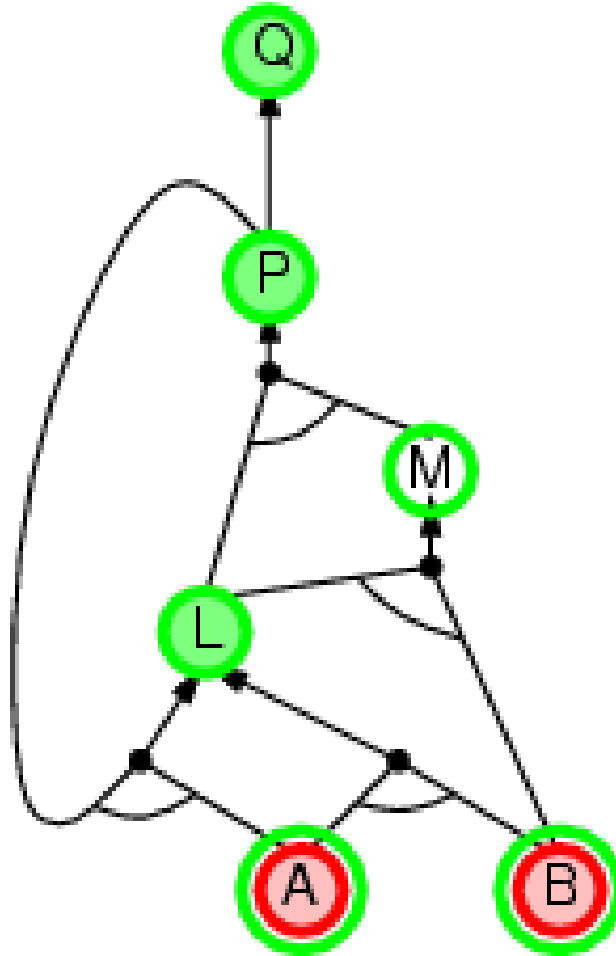
# BACKWARD CHAINING EXAMPLE



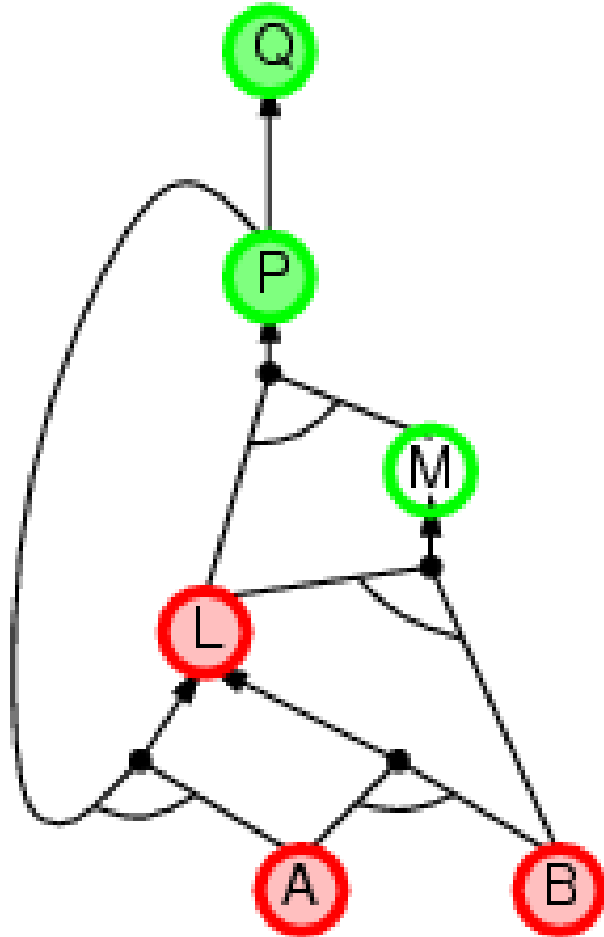
# BACKWARD CHAINING EXAMPLE



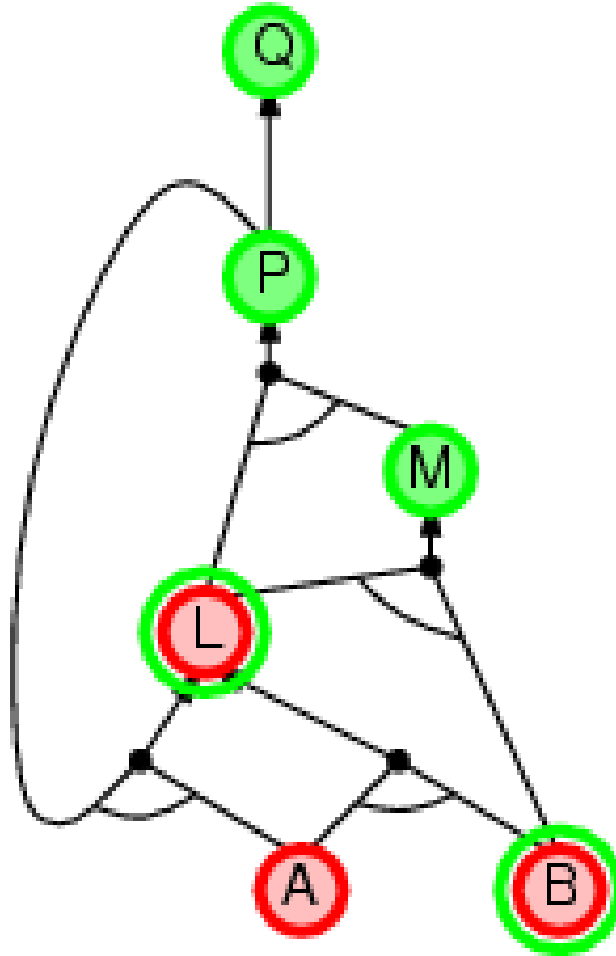
# BACKWARD CHAINING EXAMPLE



# BACKWARD CHAINING EXAMPLE

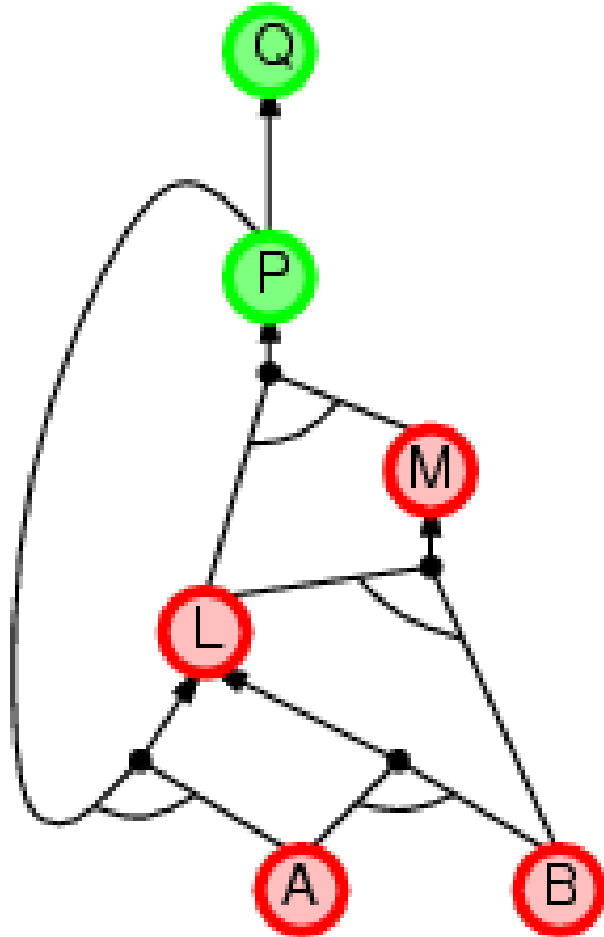


# BACKWARD CHAINING EXAMPLE

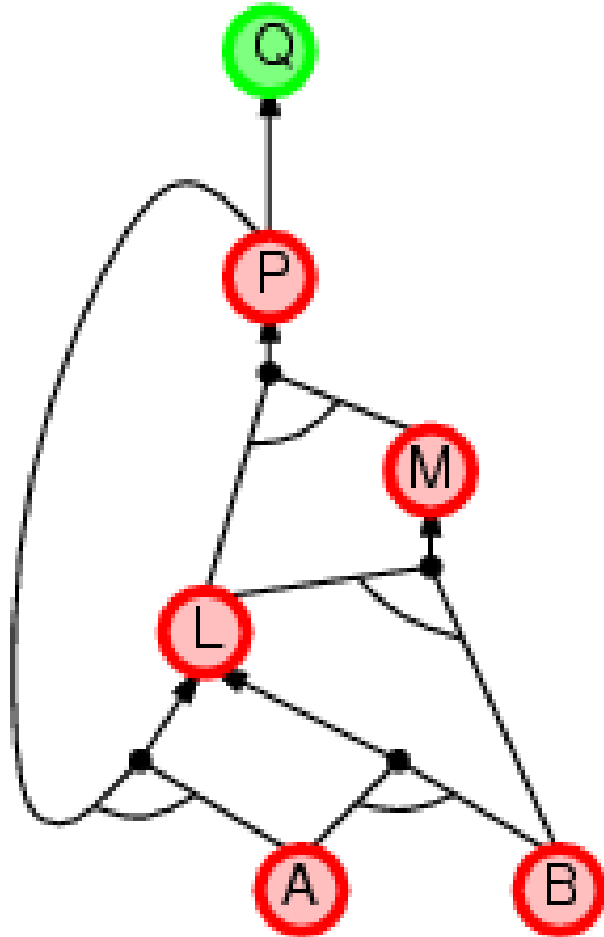




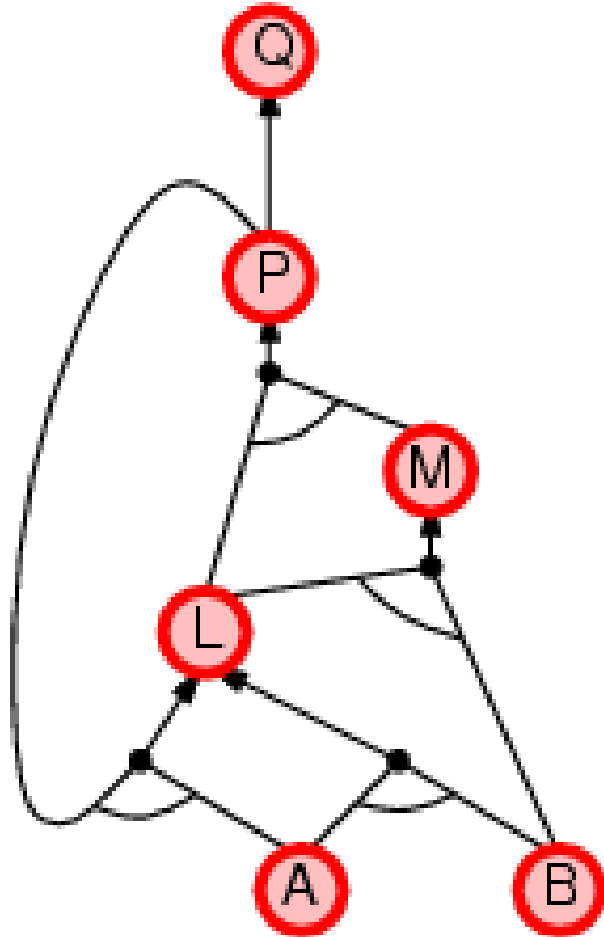
# BACKWARD CHAINING EXAMPLE



# BACKWARD CHAINING EXAMPLE



# BACKWARD CHAINING EXAMPLE



# FORWARD VS. BACKWARD CHAINING

- FC is **data-driven**, automatic, unconscious processing,
  - e.g., object recognition, routine decisions
  -
- May do lots of work that is irrelevant to the goal
- BC is **goal-driven**, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?
- Complexity of BC can be **much less** than linear in size of KB
- 



# EFFICIENT PROPOSITIONAL INFERENCE

Two families of efficient algorithms for propositional inference:

Complete backtracking search algorithms

- DPLL algorithm (Davis, Putnam, Logemann, Loveland)

- 

- Incomplete local search algorithms

- WalkSAT algorithm

- 



# THE DPLL ALGORITHM

Determine if an input propositional logic sentence (in CNF) is satisfiable.

Improvements over truth table enumeration:

1. Early termination

A clause is true if any literal is true.

A sentence is false if any clause is false.

2. Pure symbol heuristic

Pure symbol: always appears with the same "sign" in all clauses.

e.g., In the three clauses  $(A \vee \neg B)$ ,  $(\neg B \vee \neg C)$ ,  $(C \vee A)$ , A and B are pure, C is impure.

Make a pure symbol literal true.

3. Unit clause heuristic

Unit clause: only one literal in the clause

The only literal in a unit clause must be true.



# THE DPLL ALGORITHM

**function** DPLL-SATISFIABLE?(*s*) **returns** *true* or *false*

**inputs:** *s*, a sentence in propositional logic

*clauses*  $\leftarrow$  the set of clauses in the CNF representation of *s*

*symbols*  $\leftarrow$  a list of the proposition symbols in *s*

**return** DPLL(*clauses*, *symbols*, [])

---

**function** DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

**if** every clause in *clauses* is true in *model* **then return** *true*

**if** some clause in *clauses* is false in *model* **then return** *false*

*P*, *value*  $\leftarrow$  FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)

**if** *P* is non-null **then return** DPLL(*clauses*, *symbols* - *P*, [*P* = *value* | *model*])

*P*, *value*  $\leftarrow$  FIND-UNIT-CLAUSE(*clauses*, *model*)

**if** *P* is non-null **then return** DPLL(*clauses*, *symbols* - *P*, [*P* = *value* | *model*])

*P*  $\leftarrow$  FIRST(*symbols*); *rest*  $\leftarrow$  REST(*symbols*)

**return** DPLL(*clauses*, *rest*, [*P* = *true* | *model*]) **or**

DPLL(*clauses*, *rest*, [*P* = *false* | *model*])



# THE WALKSAT ALGORITHM

- Incomplete, local search algorithm
- 
- Evaluation function: The min-conflict heuristic of minimizing the number of unsatisfied clauses
- 
- Balance between greediness and randomness
- 





# THE WALKSAT ALGORITHM

**function** WALKSAT(*clauses*, *p*, *max-flips*) **returns** a satisfying model or *failure*

**inputs:** *clauses*, a set of clauses in propositional logic

*p*, the probability of choosing to do a “random walk” move

*max-flips*, number of flips allowed before giving up

*model*  $\leftarrow$  a random assignment of *true/false* to the symbols in *clauses*

**for** *i* = 1 **to** *max-flips* **do**

**if** *model* satisfies *clauses* **then return** *model*

*clause*  $\leftarrow$  a randomly selected clause from *clauses* that is false in *model*

**with probability** *p* flip the value in *model* of a randomly selected symbol  
        from *clause*

**else** flip whichever symbol in *clause* maximizes the number of satisfied clauses

**return** *failure*



# HARD SATISFIABILITY PROBLEMS

- Consider random 3-CNF sentences. e.g.,

- $$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \\ \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

$m$  = number of clauses

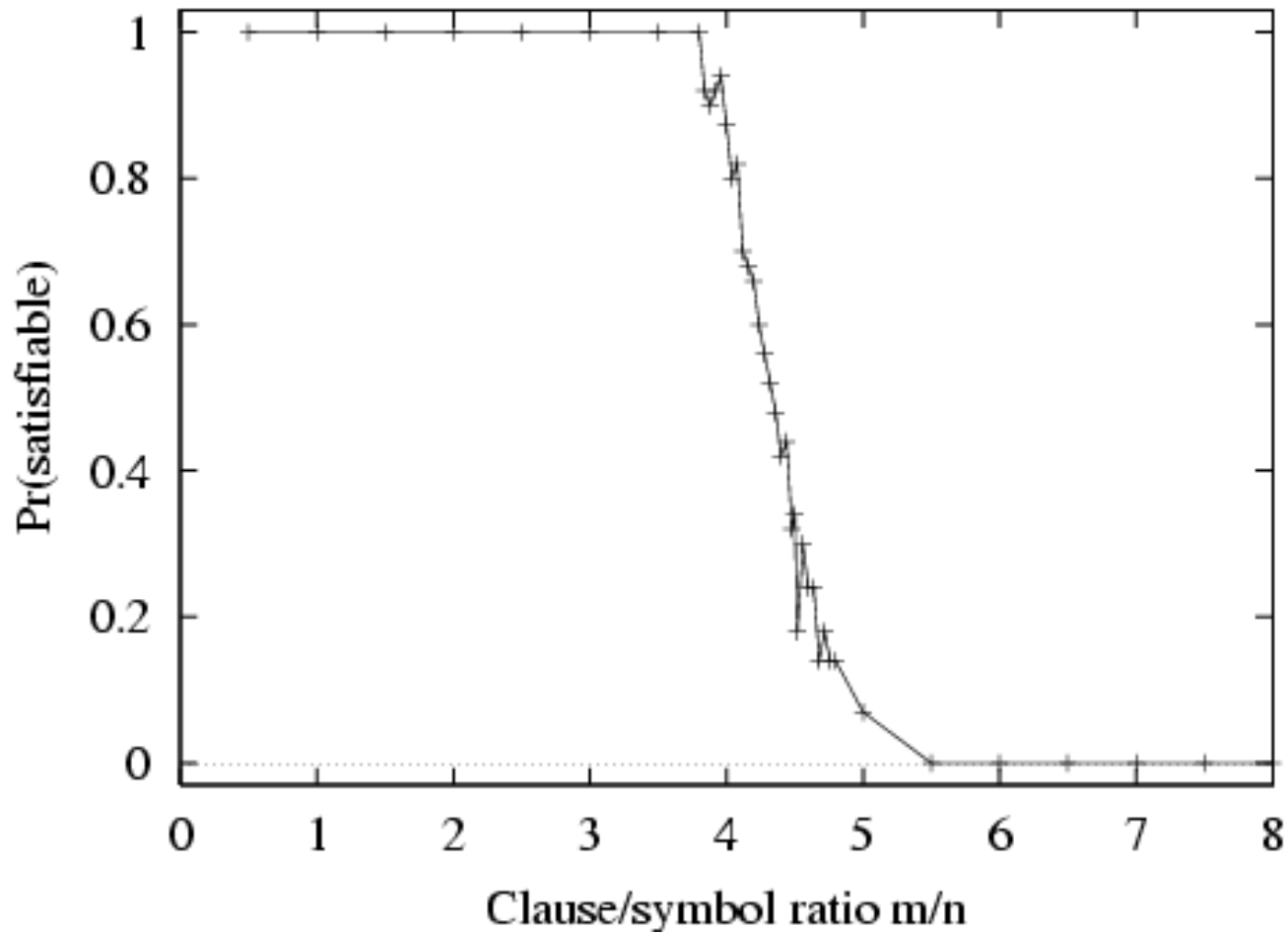
$n$  = number of symbols

- Hard problems seem to cluster near  $m/n = 4.3$  (critical point)

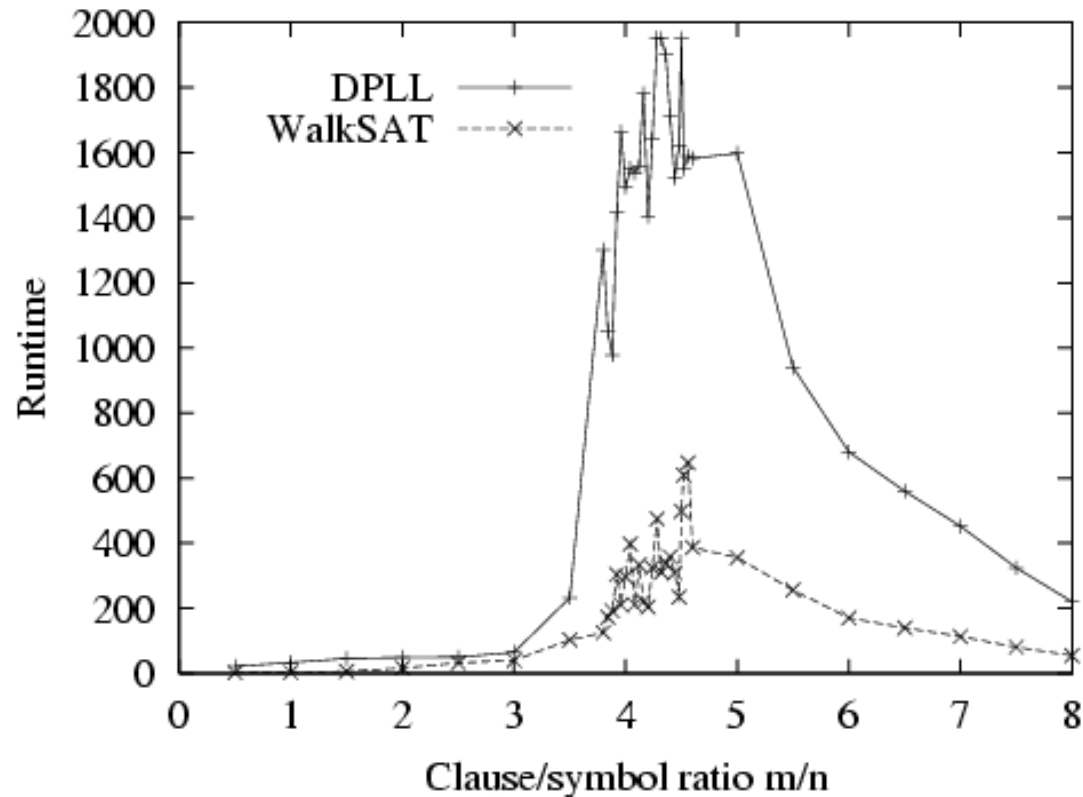
- 



# HARD SATISFIABILITY PROBLEMS



# HARD SATISFIABILITY PROBLEMS



- Median runtime for 100 **satisfiable** random 3-CNF sentences,  $n = 50$



# INFERENCE-BASED AGENTS IN THE WUMPUS WORLD

A wumpus-world agent using propositional logic:

$$\begin{aligned}& \neg P_{1,1} \\& \neg W_{1,1} \\& B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y}) \\& S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y}) \\& W_{1,1} \vee W_{1,2} \vee \dots \vee W_{4,4} \\& \neg W_{1,1} \vee \neg W_{1,2} \\& \neg W_{1,1} \vee \neg W_{1,3} \\& \dots\end{aligned}$$

$\Rightarrow$  64 distinct proposition symbols, 155 sentences



```

function PL-WUMPUS-AGENT(percept) returns an action
 inputs: percept, a list, [stench, breeze, glitter]
 static: KB, initially containing the “physics” of the wumpus world
 x, y, orientation, the agent’s position (init. [1,1]) and orient. (init. right)
 visited, an array indicating which squares have been visited, initially false
 action, the agent’s most recent action, initially null
 plan, an action sequence, initially empty

 update x, y, orientation, visited based on action
 if stench then TELL(KB, $S_{x,y}$) else TELL(KB, $\neg S_{x,y}$)
 if breeze then TELL(KB, $B_{x,y}$) else TELL(KB, $\neg B_{x,y}$)
 if glitter then action \leftarrow grab
 else if plan is nonempty then action \leftarrow POP(plan)
 else if for some fringe square $[i,j]$, ASK(KB, $(\neg P_{i,j} \wedge \neg W_{i,j})$) is true or
 for some fringe square $[i,j]$, ASK(KB, $(P_{i,j} \vee W_{i,j})$) is false then do
 plan \leftarrow A*-GRAPH-SEARCH(ROUTE-PB($[x,y]$, orientation, $[i,j]$, visited))
 action \leftarrow POP(plan)
 else action \leftarrow a randomly chosen move
 return action

```

# EXPRESSIVENESS LIMITATION OF PROPOSITIONAL LOGIC

- KB contains "physics" sentences for every single square



- <sup>t</sup> For every time  $t$  and every location  $[x,y]$ ,



$$L_{x,y} \wedge FacingRight^t \wedge Forward^t \Rightarrow L_{x+1,y}$$

- Rapid proliferation of clauses

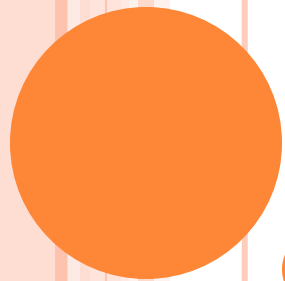


# SUMMARY

- Logical agents apply **inference** to a **knowledge base** to derive new information and make decisions
- 
- Basic concepts of logic:
  - 
  - **syntax**: formal structure of **sentences**
  - 
  - **semantics**: **truth** of sentences wrt **models**
  - 
  - **entailment**: necessary truth of one sentence given another
  - 
  - **inference**: deriving sentences from other sentences
  - 
  - **soundness**: derivations produce only entailed sentences
  - 
  - **completeness**: derivations can produce all entailed sentences
  -
- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- 
- Resolution is complete for propositional logic  
Forward, backward chaining are linear-time, complete for Horn clauses
- 
- Propositional logic lacks expressive power







# FIRST-ORDER LOGIC

## Chapter 8

# OUTLINE

- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Wumpus world in FOL
- Knowledge engineering in FOL



# PROS AND CONS OF PROPOSITIONAL LOGIC

- ☺ Propositional logic is **declarative**
- ☺ Propositional logic allows partial/disjunctive/negated information
  - (unlike most data structures and databases)
  -
- ☺ Propositional logic is **compositional**:
  - ☺
    - meaning of  $B_{1,1} \wedge P_{1,2}$  is derived from meaning of  $B_{1,1}$  and of  $P_{1,2}$
    -
- ☺ Meaning in propositional logic is **context-independent**
  - (unlike natural language, where meaning depends on context)
  -
- ☹ Propositional logic has very limited expressive power
  - (unlike natural language)
  - E.g., cannot say "pits cause breezes in adjacent squares"
    - except by writing one sentence for each square
    -



# FIRST-ORDER LOGIC

- Whereas propositional logic assumes the world contains **facts**,
- first-order logic (like natural language) assumes the world contains
  - - **Objects**: people, houses, numbers, colors, baseball games, wars, ...
    - 
    - **Relations**: red, round, prime, brother of, bigger than, part of, comes between, ...
    - **Functions**: father of, best friend, one more than, plus, ...
    -



# SYNTAX OF FOL: BASIC ELEMENTS

- Constants KingJohn, 2, NUS,...
- Predicates Brother,  $>$ ,...
- Functions Sqrt, LeftLegOf,...
- Variables  $x$ ,  $y$ ,  $a$ ,  $b$ ,...
- Connectives  $\neg$ ,  $\Rightarrow$ ,  $\wedge$ ,  $\vee$ ,  $\Leftrightarrow$
- Equality  $=$
- Quantifiers  $\forall$ ,  $\exists$



# ATOMIC SENTENCES

Atomic sentence =  $\text{predicate}(\text{term}_1, \dots, \text{term}_n)$   
or  $\text{term}_1 = \text{term}_2$

Term =  $\text{function}(\text{term}_1, \dots, \text{term}_n)$   
or *constant* or *variable*

- E.g.,  $\text{Brother}(\text{KingJohn}, \text{RichardTheLionheart}) >$   
 $(\text{Length}(\text{LeftLegOf}(\text{Richard})),$   
 $\text{Length}(\text{LeftLegOf}(\text{KingJohn})))$



# COMPLEX SENTENCES

- Complex sentences are made from atomic sentences using connectives



$$\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2,$$

E.g. *Sibling(KingJohn, Richard)  $\Rightarrow$   
Sibling(Richard, KingJohn)*

$$>(1,2) \vee \leq(1,2)$$

$$>(1,2) \wedge \neg >(1,2)$$



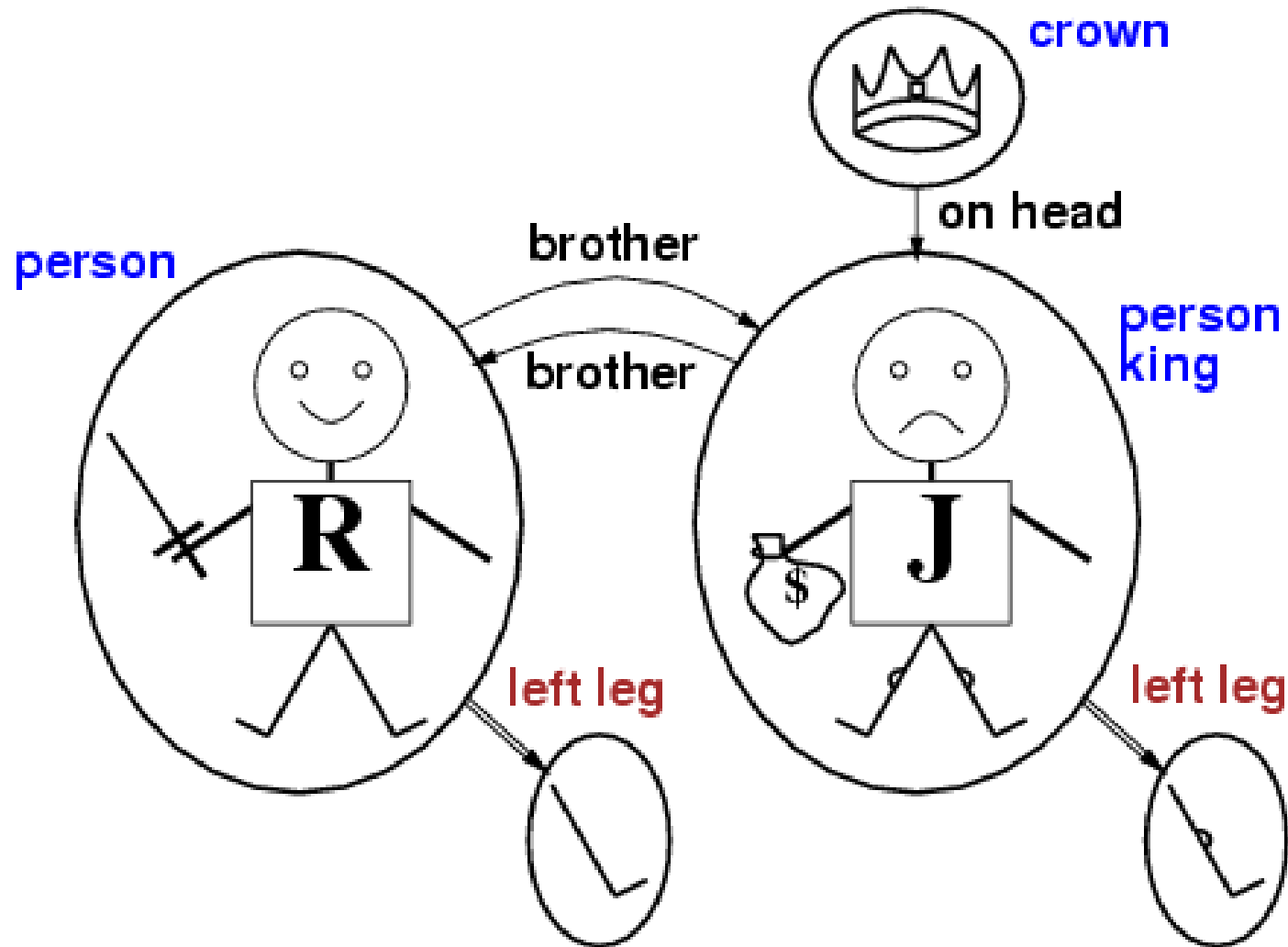
# TRUTH IN FIRST-ORDER LOGIC

- Sentences are true with respect to a **model** and an **interpretation**
- Model contains objects (**domain elements**) and relations among them
- 
- Interpretation specifies referents for
  - constant symbols** → **objects**
  - predicate symbols** → **relations**
  - function symbols** → **functional relations**
- An atomic sentence  $predicate(term_1, \dots, term_n)$  is true iff the **objects** referred to by  $term_1, \dots, term_n$  are in the **relation** referred to by  $predicate$





# MODELS FOR FOL: EXAMPLE



# UNIVERSAL QUANTIFICATION

- $\forall \langle \text{variables} \rangle \langle \text{sentence} \rangle$

- 

Everyone at NUS is smart:

$$\forall x \text{ At}(x, \text{NUS}) \Rightarrow \text{Smart}(x)$$

- $\forall x P$  is true in a model  $m$  iff  $P$  is true with  $x$  being each possible object in the model

- 

- Roughly speaking, equivalent to the conjunction of instantiations of  $P$

- 

$$\begin{aligned} & \text{At}(\text{KingJohn}, \text{NUS}) \Rightarrow \text{Smart}(\text{KingJohn}) \\ \wedge & \text{At}(\text{Richard}, \text{NUS}) \Rightarrow \text{Smart}(\text{Richard}) \\ \wedge & \text{At}(\text{NUS}, \text{NUS}) \Rightarrow \text{Smart}(\text{NUS}) \\ \wedge & \dots \end{aligned}$$



## A COMMON MISTAKE TO AVOID

- Typically,  $\Rightarrow$  is the main connective with  $\forall$



- Common mistake: using  $\wedge$  as the main connective with  $\forall$ :

$$\forall x \text{ At}(x, \text{NUS}) \wedge \text{Smart}(x)$$

means “Everyone is at NUS and everyone is smart”



# EXISTENTIAL QUANTIFICATION

- $\exists \langle \text{variables} \rangle \langle \text{sentence} \rangle$
- Someone at NUS is smart:
- $\exists x \text{ At}(x, \text{NUS}) \wedge \text{Smart}(x)$
- 
- $\exists x P$  is true in a model  $m$  iff  $P$  is true with  $x$  being some possible object in the model
- 
- Roughly speaking, equivalent to the disjunction of instantiations of  $P$
- - $\text{At}(\text{KingJohn}, \text{NUS}) \wedge \text{Smart}(\text{KingJohn})$
  - $\vee \text{ At}(\text{Richard}, \text{NUS}) \wedge \text{Smart}(\text{Richard})$
  - $\vee \text{ At}(\text{NUS}, \text{NUS}) \wedge \text{Smart}(\text{NUS})$
  - $\vee \dots$



## ANOTHER COMMON MISTAKE TO AVOID

- Typically,  $\wedge$  is the main connective with  $\exists$
- Common mistake: using  $\Rightarrow$  as the main connective with  $\exists$ :



$$\exists x \text{ At}(x, \text{NUS}) \Rightarrow \text{Smart}(x)$$

is true if there is anyone who is not at NUS!



# PROPERTIES OF QUANTIFIERS

- $\forall x \forall y$  is the same as  $\forall y \forall x$
- 
- $\exists x \exists y$  is the same as  $\exists y \exists x$
- 
- $\exists x \forall y$  is **not** the same as  $\forall y \exists x$
- 
- $\exists x \forall y \text{ Loves}(x,y)$ 
  - “There is a person who loves everyone in the world”
  -
- $\forall y \exists x \text{ Loves}(x,y)$ 
  - “Everyone in the world is loved by at least one person”
  -
- **Quantifier duality**: each can be expressed using the other
- 
- $\forall x \text{ Likes}(x, \text{IceCream}) \rightarrow \neg \exists x \neg \text{Likes}(x, \text{IceCream})$
- 
- $\exists x \text{ Likes}(x, \text{Broccoli}) \quad \neg \forall x \neg \text{Likes}(x, \text{Broccoli})$
- 



# EQUALITY

- $term_1 = term_2$  is true under a given interpretation if and only if  $term_1$  and  $term_2$  refer to the same object
- 
- E.g., definition of *Sibling* in terms of *Parent*:
- $$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow [\neg(x = y) \wedge \exists m, f \neg (m = f) \wedge \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y)]$$



# USING FOL

The kinship domain:

- Brothers are siblings



$$\forall x,y \text{ Brother}(x,y) \Leftrightarrow \text{Sibling}(x,y)$$

- One's mother is one's female parent



$$\forall m,c \text{ Mother}(c) = m \Leftrightarrow (\text{Female}(m) \wedge \text{Parent}(m,c))$$

- “Sibling” is symmetric



$$\forall x,y \text{ Sibling}(x,y) \Leftrightarrow \text{Sibling}(y,x)$$





# USING FOL

The set domain:

- $\forall s \text{ Set}(s) \Leftrightarrow (s = \{\} ) \vee (\exists x, s_2 \text{ Set}(s_2) \wedge s = \{x \mid s_2\})$
- 
- $\neg \exists x, s \{x \mid s\} = \{\}$
- 
- $\forall x, s x \in s \Leftrightarrow s = \{x \mid s\}$
- 
- $\forall x, s x \in s \Leftrightarrow [ \exists y, s_2 \{ (s = \{y \mid s_2\} \wedge (x = y \vee x \in s_2)) ) ]$
- 
- $\forall s_1, s_2 s_1 \subseteq s_2 \Leftrightarrow (\forall x x \in s_1 \Rightarrow x \in s_2)$
- $\forall s_1, s_2 (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- 
- $\forall x, s_1, s_2 x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
- 
- $\forall x, s_1, s_2 x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$



# INTERACTING WITH FOL KBs

- Suppose a wumpus-world agent is using an FOL KB and perceives a smell and a breeze (but no glitter) at  $t=5$ :

```
Tell(KB, Percept([Smell, Breeze, None], 5))
Ask(KB, $\exists a$ BestAction(a, 5))
```

- I.e., does the KB entail some best action at  $t=5$ ?
- 
- Answer: *Yes*,  $\{a/Shoot\}$   $\leftarrow$  substitution (binding list)
- Given a sentence  $S$  and a substitution  $\sigma$ ,
- $S\sigma$  denotes the result of plugging  $\sigma$  into  $S$ ; e.g.,  
 $S = \text{Smarter}(x, y)$   
 $\sigma = \{x/Hillary, y/Bill\}$   
 $S\sigma = \text{Smarter}(Hillary, Bill)$
- $\text{Ask}(\text{KB}, S)$  returns some/all  $\sigma$  such that  $\text{KB} \models \sigma$
- 



# KNOWLEDGE BASE FOR THE WUMPUS WORLD

## ○ Perception

- $\forall t, s, b \text{ Percept}([s, b, \text{Glitter}], t) \Rightarrow \text{Glitter}(t)$
- 

## ○ Reflex

- $\forall t \text{ Glitter}(t) \Rightarrow \text{BestAction}(\text{Grab}, t)$



# DEDUCING HIDDEN PROPERTIES

- $\forall x,y,a,b \text{ Adjacent}([x,y],[a,b]) \Leftrightarrow [a,b] \in \{[x+1,y], [x-1,y], [x,y+1], [x,y-1]\}$

Properties of squares:

- $\forall s,t \text{ At}(\text{Agent},s,t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(s)$

Squares are breezy near a pit:

- **Diagnostic** rule---infer cause from effect  
 $\forall s \text{ Breezy}(s) \Rightarrow \exists r \text{ Adjacent}(r,s) \wedge \text{Pit}(r)$
- **Causal** rule---infer effect from cause  
 $\forall r \text{ Pit}(r) \Rightarrow [\forall s \text{ Adjacent}(r,s) \Rightarrow \text{Breezy}(s)]$



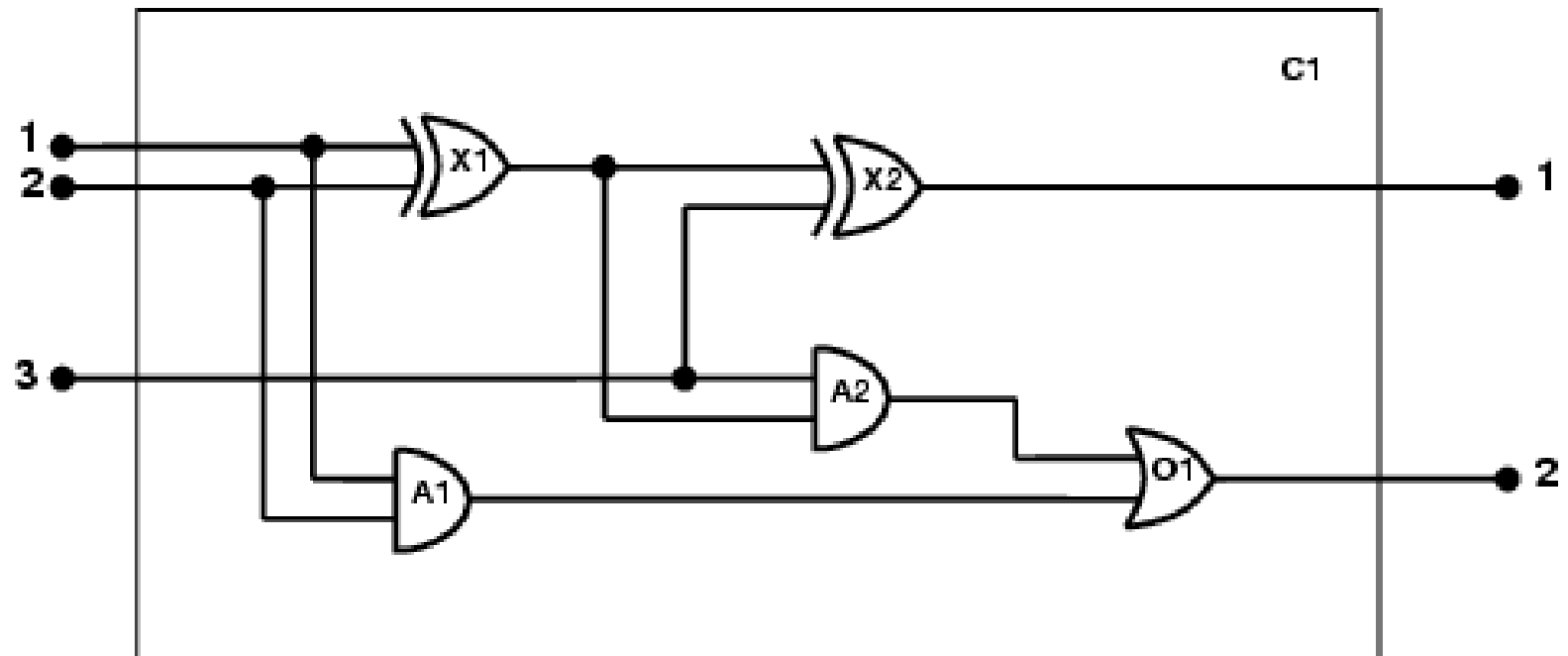
# KNOWLEDGE ENGINEERING IN FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base
- 8.



# THE ELECTRONIC CIRCUITS DOMAIN

## One-bit full adder



# THE ELECTRONIC CIRCUITS DOMAIN

1. Identify the task
2. Does the circuit actually add properly? (circuit verification)
2. Assemble the relevant knowledge
3. Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
- Irrelevant: size, shape, color, cost of gates
3. Decide on a vocabulary
4. Alternatives:  
 $\text{Type}(X_1) = \text{XOR}$   
 $\text{Type}(X_1, \text{XOR})$   
 $\text{XOR}(X_1)$



# THE ELECTRONIC CIRCUITS DOMAIN

## 4. Encode general knowledge of the domain

- 5.
- $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$
  - $\forall t \text{ Signal}(t) = 1 \vee \text{Signal}(t) = 0$
  - 
  - $1 \neq 0$
  - 
  - $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$
  - 
  - $\forall g \text{ Type}(g) = \text{OR} \Rightarrow \text{Signal}(\text{Out}(1, g)) = 1 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n, g)) = 1$
  - 
  - $\forall g \text{ Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1, g)) = 0 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n, g)) = 0$
  - 
  - $\forall g \text{ Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1, g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1, g)) \neq \text{Signal}(\text{In}(2, g))$
  - 
  - $\forall g \text{ Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1, g)) \neq \text{Signal}(\text{In}(1, g))$
  -





# THE ELECTRONIC CIRCUITS DOMAIN

5. Encode the specific problem instance

6.

Type( $X_1$ ) = XOR

Type( $A_1$ ) = AND

Type( $O_1$ ) = OR

Type( $X_2$ ) = XOR

Type( $A_2$ ) = AND

Connected(Out(1, $X_1$ ),In(1, $X_2$ ))    Connected(In(1, $C_1$ ),In(1, $X_1$ ))

Connected(Out(1, $X_1$ ),In(2, $A_2$ ))    Connected(In(1, $C_1$ ),In(1, $A_1$ ))

Connected(Out(1, $A_2$ ),In(1, $O_1$ ))    Connected(In(2, $C_1$ ),In(2, $X_1$ ))

Connected(Out(1, $A_1$ ),In(2, $O_1$ ))    Connected(In(2, $C_1$ ),In(2, $A_1$ ))

Connected(Out(1, $X_2$ ),Out(1, $C_1$ ))    Connected(In(3, $C_1$ ),In(2, $X_2$ ))

Connected(Out(1, $O_1$ ),Out(2, $C_1$ ))

Connected(In(3, $C_1$ ),In(1, $A_2$ ))



# THE ELECTRONIC CIRCUITS DOMAIN

6. Pose queries to the inference procedure

7. What are the possible sets of values of all the terminals for the adder circuit?

$$\begin{aligned} \exists i_1, i_2, i_3, o_1, o_2 \text{ Signal(In(1, C}_1\text{))} = i_1 \wedge \\ \text{Signal(In(2, C}_1\text{))} = i_2 \wedge \text{Signal(In(3, C}_1\text{))} = i_3 \wedge \\ \text{Signal(Out(1, C}_1\text{))} = o_1 \wedge \text{Signal(Out(2, C}_1\text{))} = o_2 \end{aligned}$$

7. Debug the knowledge base

8. May have omitted assertions like  $1 \neq 0$



# SUMMARY

- First-order logic:
  - - objects and relations are semantic primitives
    - syntax: constants, functions, predicates, equality, quantifiers
    -
- Increased expressive power: sufficient to define wumpus world
- 





# **INFERENCE IN FIRST-ORDER LOGIC**

## **Chapter 9**

# OUTLINE

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution



# UNIVERSAL INSTANTIATION (UI)

- Every instantiation of a universally quantified sentence is entailed by it:

- 

$$\frac{\forall v \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

for any variable  $v$  and ground term  $g$

- E.g.,  $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$  yields:

- 

- 

$\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$

$\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$

$\text{King}(\text{Father}(\text{John})) \wedge \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))$

·

·

·



## EXISTENTIAL INSTANTIATION (EI)

- For any sentence  $\alpha$ , variable  $v$ , and constant symbol  $k$  that does **not** appear elsewhere in the knowledge base:

○

$$\frac{\exists v \alpha}{\text{Subst}(\{v/k\}, \alpha)} \text{---}$$

- E.g.,  $\exists x \text{Crown}(x) \wedge \text{OnHead}(x, \text{John})$  yields:

$$\text{Crown}(C_1) \wedge \text{OnHead}(C_1, \text{John})$$

provided  $C_1$  is a new constant symbol, called a **Skolem constant**



# REDUCTION TO PROPOSITIONAL INFERENCE

Suppose the KB contains just the following:

$\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$   
 $\text{King}(\text{John})$   
 $\text{Greedy}(\text{John})$   
 $\text{Brother}(\text{Richard}, \text{John})$

- Instantiating the universal sentence in **all possible** ways, we have:  
 $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$   
 $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$   
 $\text{King}(\text{John})$   
 $\text{Greedy}(\text{John})$   
 $\text{Brother}(\text{Richard}, \text{John})$
- The new KB is **propositionalized**: proposition symbols are
- 

$\text{King}(\text{John})$ ,  $\text{Greedy}(\text{John})$ ,  $\text{Evil}(\text{John})$ ,  $\text{King}(\text{Richard})$ , etc.





## REDUCTION CONTD.

- Every FOL KB can be propositionalized so as to preserve entailment
- 
- (A ground sentence is entailed by new KB iff entailed by original KB)
- 
- Idea: propositionalize KB and query, apply resolution, return result
- 
- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., *Father(Father(Father(John)))*
  -



# REDUCTION CONTD.

Theorem: Herbrand (1930). If a sentence  $\alpha$  is entailed by an FOL KB, it is entailed by a **finite** subset of the propositionalized KB

Idea: For  $n = 0$  to  $\infty$  do  
    create a propositional KB by instantiating with depth- $n$  terms  
    see if  $\alpha$  is entailed by this KB

Problem: works if  $\alpha$  is entailed, loops if  $\alpha$  is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is **semidecidable** (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)



# PROBLEMS WITH PROPOSITIONALIZATION

- Propositionalization seems to generate lots of irrelevant sentences.
- E.g., from:
  - 
  - $$\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$$
  - $$\text{King}(\text{John})$$
  - $$\forall y \text{ Greedy}(y)$$
  - $$\text{Brother}(\text{Richard}, \text{John})$$
- it seems obvious that *Evil(John)*, but propositionalization produces lots of facts such as *Greedy(Richard)* that are irrelevant
- 
- With  $p$   $k$ -ary predicates and  $n$  constants, there are  $p \cdot n^k$  instantiations.
- 



# UNIFICATION

- We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(John)$  and  $Greedy(y)$

○

$\theta = \{x/John, y/John\}$  works

- $Unify(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

○

| p             | q                  | $\theta$ |
|---------------|--------------------|----------|
| Knows(John,x) | Knows(John,Jane)   |          |
| Knows(John,x) | Knows(y,OJ)        |          |
| Knows(John,x) | Knows(y,Mother(y)) |          |
| Knows(John,x) | Knows(x,OJ)        |          |

- **Standardizing apart** eliminates overlap of variables, e.g., Knows( $z_{17}$ , OJ)

○



# UNIFICATION

- We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(John)$  and  $Greedy(y)$

○

$\theta = \{x/John, y/John\}$  works

- $Unify(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

○

| p             | q                             | $\theta$ |
|---------------|-------------------------------|----------|
| Knows(John,x) | Knows(John,Jane) $\{x/Jane\}$ |          |
| Knows(John,x) | Knows(y,OJ)                   |          |
| Knows(John,x) | Knows(y,Mother(y))            |          |
| Knows(John,x) | Knows(x,OJ)                   |          |

- **Standardizing apart** eliminates overlap of variables, e.g., Knows( $z_{17}$ , OJ)

○



# UNIFICATION

- We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(John)$  and  $Greedy(y)$

○

$\theta = \{x/John, y/John\}$  works

- $Unify(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

○

| p             | q                  | $\theta$           |
|---------------|--------------------|--------------------|
| Knows(John,x) | Knows(John,Jane)   | $\{x/Jane\}$       |
| Knows(John,x) | Knows(y,OJ)        | $\{x/OJ, y/John\}$ |
| Knows(John,x) | Knows(y,Mother(y)) |                    |
| Knows(John,x) | Knows(x,OJ)        |                    |

- **Standardizing apart** eliminates overlap of variables, e.g.,  
Knows( $z_{17}$ , OJ)

○



# UNIFICATION

- We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(John)$  and  $Greedy(y)$

○

$\theta = \{x/John, y/John\}$  works

- $Unify(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

○

| p             | q                  | $\theta$                     |
|---------------|--------------------|------------------------------|
| Knows(John,x) | Knows(John,Jane)   | $\{x/Jane\}$                 |
| Knows(John,x) | Knows(y,OJ)        | $\{x/OJ, y/John\}$           |
| Knows(John,x) | Knows(y,Mother(y)) | $\{y/John, x/Mother(John)\}$ |
| Knows(John,x) | Knows(x,OJ)        |                              |

- **Standardizing apart** eliminates overlap of variables, e.g.,  
Knows( $z_{17}$ , OJ)

○



# UNIFICATION

- We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(John)$  and  $Greedy(y)$

○

$\theta = \{x/John, y/John\}$  works

- $Unify(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

○

| p             | q                  | $\theta$                     |
|---------------|--------------------|------------------------------|
| Knows(John,x) | Knows(John,Jane)   | $\{x/Jane\}$                 |
| Knows(John,x) | Knows(y,OJ)        | $\{x/OJ, y/John\}$           |
| Knows(John,x) | Knows(y,Mother(y)) | $\{y/John, x/Mother(John)\}$ |
| Knows(John,x) | Knows(x,OJ)        | $\{fail\}$                   |

- **Standardizing apart** eliminates overlap of variables, e.g.,  $Knows(z_{17}, OJ)$

○





# UNIFICATION

- To unify  $Knows(John, x)$  and  $Knows(y, z)$ ,
- $\theta = \{y/John, x/z\}$  or  $\theta = \{y/John, x/John, z/John\}$
- The first unifier is **more general** than the second.
- 
- There is a single **most general unifier** (MGU) that is unique up to renaming of variables.
- MGU =  $\{y/John, x/z\}$



# THE UNIFICATION ALGORITHM

**function** UNIFY( $x, y, \theta$ ) **returns** a substitution to make  $x$  and  $y$  identical

**inputs:**  $x$ , a variable, constant, list, or compound

$y$ , a variable, constant, list, or compound

$\theta$ , the substitution built up so far

**if**  $\theta = \text{failure}$  **then return failure**

**else if**  $x = y$  **then return**  $\theta$

**else if** VARIABLE?( $x$ ) **then return** UNIFY-VAR( $x, y, \theta$ )

**else if** VARIABLE?( $y$ ) **then return** UNIFY-VAR( $y, x, \theta$ )

**else if** COMPOUND?( $x$ ) **and** COMPOUND?( $y$ ) **then**

**return** UNIFY(ARGS[ $x$ ], ARGS[ $y$ ], UNIFY(OP[ $x$ ], OP[ $y$ ],  $\theta$ ))

**else if** LIST?( $x$ ) **and** LIST?( $y$ ) **then**

**return** UNIFY(REST[ $x$ ], REST[ $y$ ], UNIFY(FIRST[ $x$ ], FIRST[ $y$ ],  $\theta$ ))

**else return failure**

# THE UNIFICATION ALGORITHM

```
function UNIFY-VAR(var, x, θ) returns a substitution
 inputs: var , a variable
 x , any expression
 θ , the substitution built up so far

 if $\{var/val\} \in \theta$ then return UNIFY(val, x, θ)
 else if $\{x/val\} \in \theta$ then return UNIFY(var, val, θ)
 else if OCCUR-CHECK?(var, x) then return failure
 else return add $\{var/x\}$ to θ
```



# GENERALIZED MODUS PONENS (GMP)

$$\frac{p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{q\theta} \quad \text{where } p_i'\theta = p_i \theta \text{ for all } i$$

|                                                |                           |
|------------------------------------------------|---------------------------|
| $p_1'$ is <i>King(John)</i>                    | $p_1$ is <i>King(x)</i>   |
| $p_2'$ is <i>Greedy(y)</i>                     | $p_2$ is <i>Greedy(x)</i> |
| $\theta$ is $\{x/\text{John}, y/\text{John}\}$ | $q$ is <i>Evil(x)</i>     |
| $q \theta$ is <i>Evil(John)</i>                |                           |

- GMP used with KB of definite clauses (**exactly** one positive literal)
- All variables assumed universally quantified
- 



# SOUNDNESS OF GMP

- Need to show that

- 

$$p_1', \dots, p_n', (p_1 \wedge \dots \wedge p_n \Rightarrow q) \models q\theta$$

provided that  $p_i'\theta = p_i\theta$  for all  $i$

- Lemma: For any sentence  $p$ , we have  $p \models p\theta$  by UI

- 

1.  $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \models (p_1 \wedge \dots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \dots \wedge p_n\theta \Rightarrow q\theta)$
- 2.
2.  $p_1', \dots, p_n' \models p_1' \wedge \dots \wedge p_n' \models p_1'\theta \wedge \dots \wedge p_n'\theta$
3. From 1 and 2,  $q\theta$  follows by ordinary Modus Ponens
- 4.



## EXAMPLE KNOWLEDGE BASE

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
- 
- Prove that Col. West is a criminal
- 



## EXAMPLE KNOWLEDGE BASE CONTD.

... it is a crime for an American to sell weapons to hostile nations:

*$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$*

Nono ... has some missiles, i.e.,  $\exists x \text{ Owns(Nono},x) \wedge \text{Missile}(x)$ :

*$\text{Owns(Nono},M_1) \text{ and Missile}(M_1)$*

... all of its missiles were sold to it by Colonel West

*$\text{Missile}(x) \wedge \text{Owns(Nono},x) \Rightarrow \text{Sells(West},x,\text{Nono})$*

Missiles are weapons:

*$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$*

An enemy of America counts as "hostile":

*$\text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x)$*

West, who is American ...

*$\text{American(West)}$*

The country Nono, an enemy of America ...

*$\text{Enemy(Nono,America)}$*



# FORWARD CHAINING ALGORITHM

```
function FOL-FC-ASK(KB, α) returns a substitution or false
 repeat until new is empty
 $new \leftarrow \{\}$
 for each sentence r in KB do
 $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-APART}(r)$
 for each θ such that $(p_1 \wedge \dots \wedge p_n)\theta = (p'_1 \wedge \dots \wedge p'_n)\theta$
 for some p'_1, \dots, p'_n in KB
 $q' \leftarrow \text{SUBST}(\theta, q)$
 if q' is not a renaming of a sentence already in KB or new then do
 add q' to new
 $\phi \leftarrow \text{UNIFY}(q', \alpha)$
 if ϕ is not fail then return ϕ
 add new to KB
 return false
```



# FORWARD CHAINING PROOF

*American(West)*

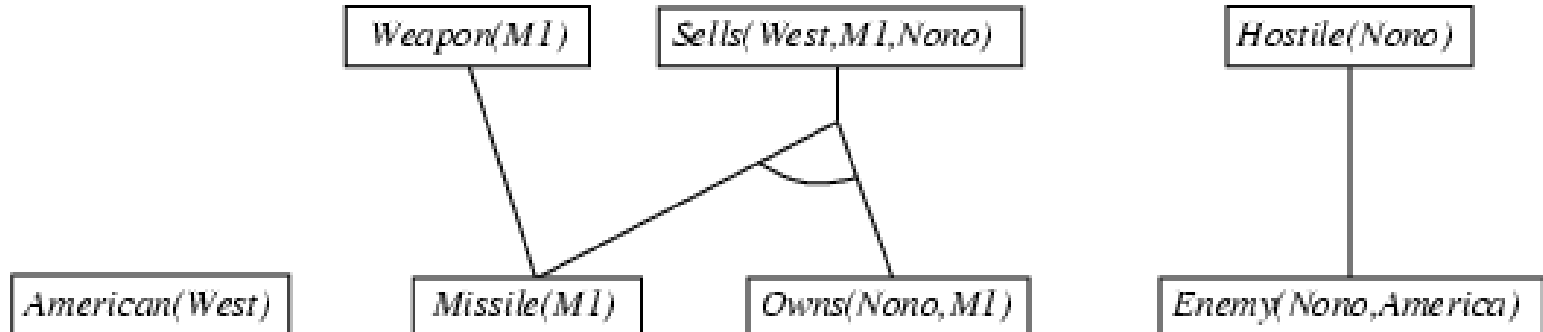
*Missile(M1)*

*Owns(Nono,M1)*

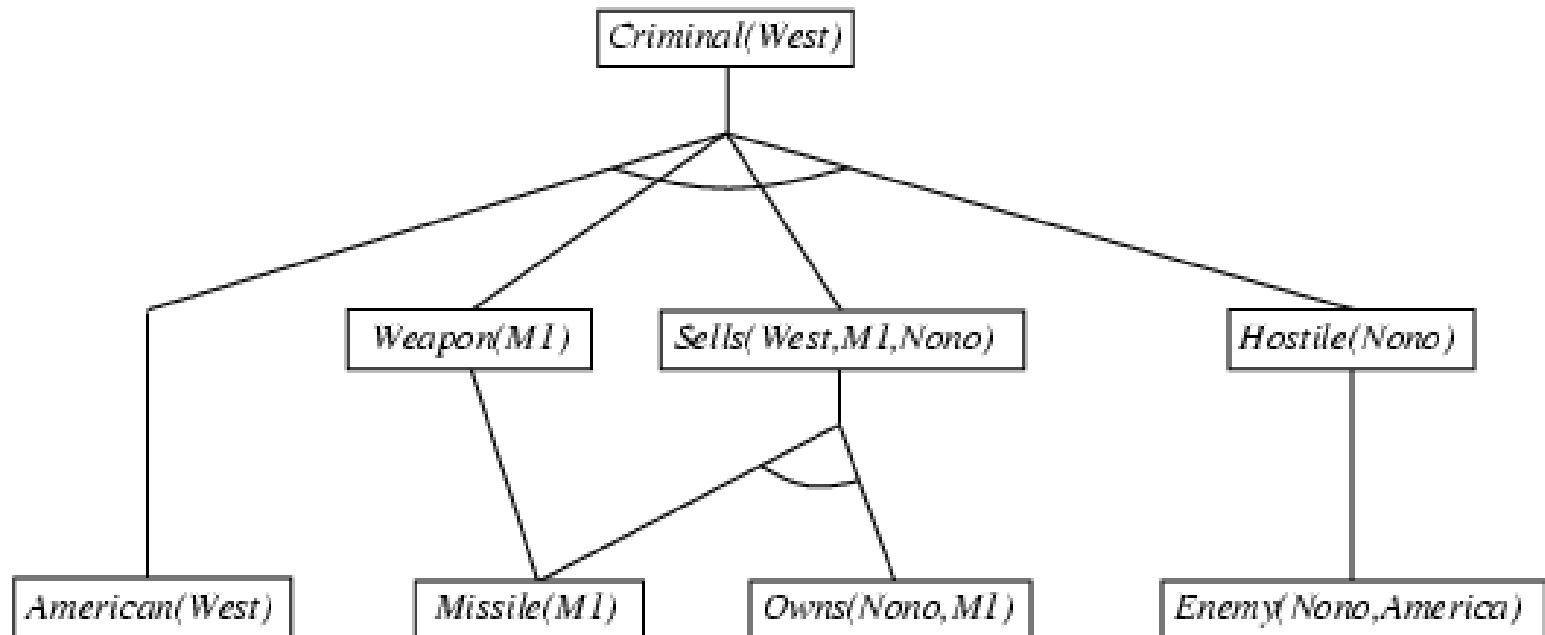
*Enemy(Nono,America)*



# FORWARD CHAINING PROOF



# FORWARD CHAINING PROOF



# PROPERTIES OF FORWARD CHAINING

- Sound and complete for first-order definite clauses
- 
- **Datalog** = first-order definite clauses + **no functions**
- FC terminates for Datalog in finite number of iterations
- 
- May not terminate in general if  $\alpha$  is not entailed
- 
- This is unavoidable: entailment with definite clauses is semidecidable
- 



# EFFICIENCY OF FORWARD CHAINING

Incremental forward chaining: no need to match a rule on iteration  $k$  if a premise wasn't added on iteration  $k-1$

⇒ match each rule whose premise contains a newly added positive literal

Matching itself can be expensive:

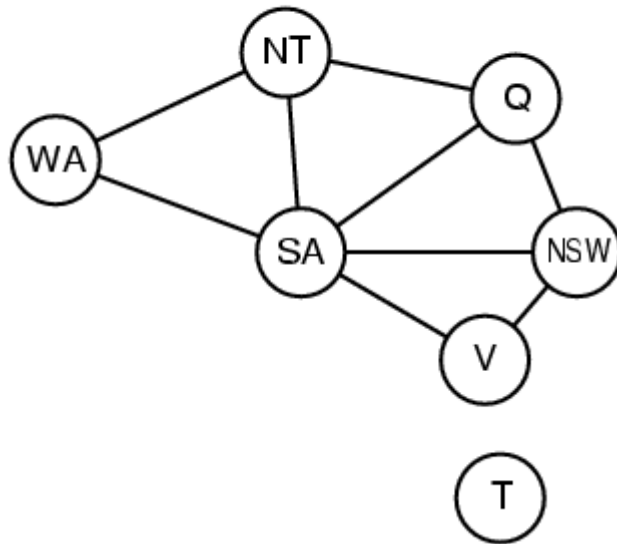
Database indexing allows  $O(1)$  retrieval of known facts

- e.g., query  $Missile(x)$  retrieves  $Missile(M_1)$
- 

Forward chaining is widely used in deductive databases



## GRAPH MATCHING EXAMPLE



$$\begin{aligned}
 &Diff(wa, nt) \wedge Diff(wa, sa) \wedge Diff(nt, q) \wedge \\
 &Diff(nt, sa) \wedge Diff(q, nsw) \wedge Diff(q, sa) \wedge \\
 &Diff(nsw, v) \wedge Diff(nsw, sa) \wedge Diff(v, sa) \\
 &\Rightarrow Colorable()
 \end{aligned}$$

$$\begin{aligned}
 &Diff(Red, Blue) \quad Diff(Red, Green) \\
 &Diff(Green, Red) \quad Diff(Green, Blue) \\
 &Diff(Blue, Red) \quad Diff(Blue, Green)
 \end{aligned}$$

- *Colorable()* is inferred iff the CSP has a solution
- CSPs include 3SAT as a special case, hence matching is NP-hard
- 



# BACKWARD CHAINING ALGORITHM

```
function FOL-BC-ASK(KB , $goals$, θ) returns a set of substitutions
 inputs: KB , a knowledge base
 $goals$, a list of conjuncts forming a query
 θ , the current substitution, initially the empty substitution $\{ \}$
 local variables: ans , a set of substitutions, initially empty

 if $goals$ is empty then return $\{ \theta \}$
 $q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(goals))$
 for each r in KB where $\text{STANDARDIZE-APART}(r) = (p_1 \wedge \dots \wedge p_n \Rightarrow q)$
 and $\theta' \leftarrow \text{UNIFY}(q, q')$ succeeds
 $ans \leftarrow \text{FOL-BC-ASK}(KB, [p_1, \dots, p_n | \text{REST}(goals)], \text{COMPOSE}(\theta, \theta')) \cup ans$
 return ans
```

$$\text{SUBST}(\text{COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p))$$



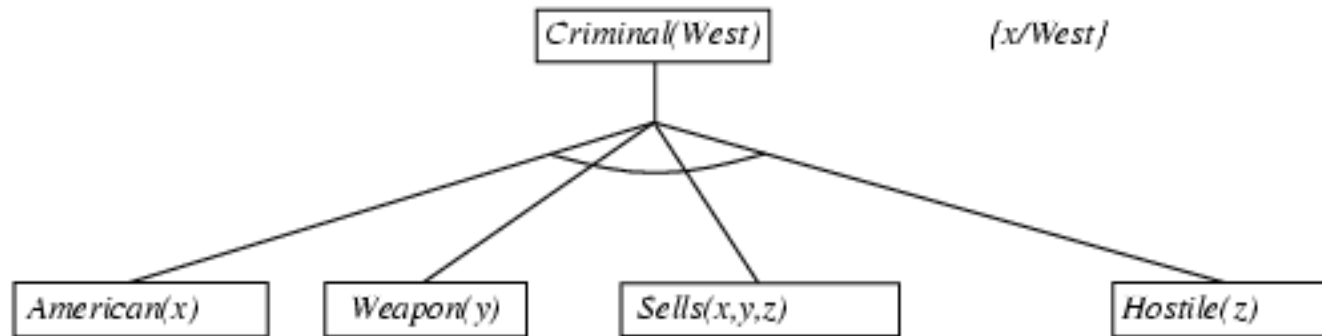
# BACKWARD CHAINING EXAMPLE

*Criminal(West)*

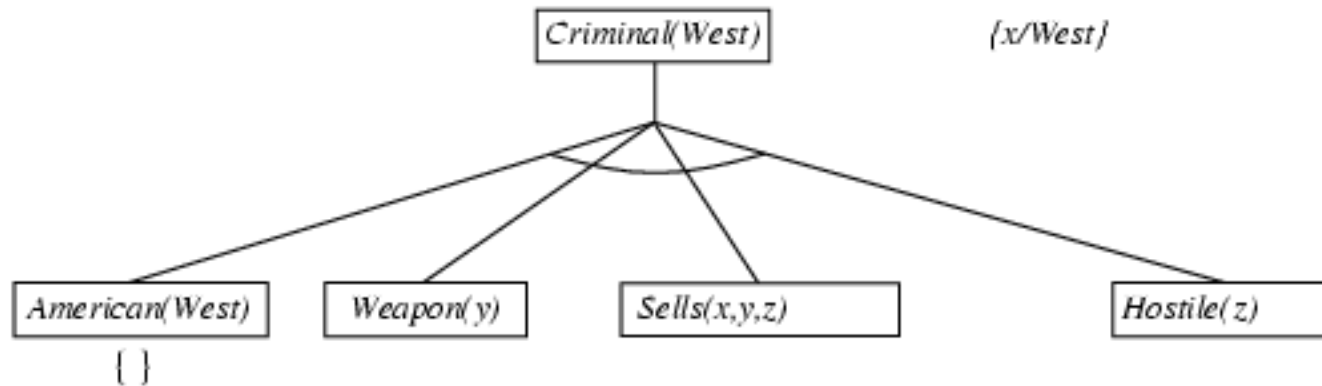




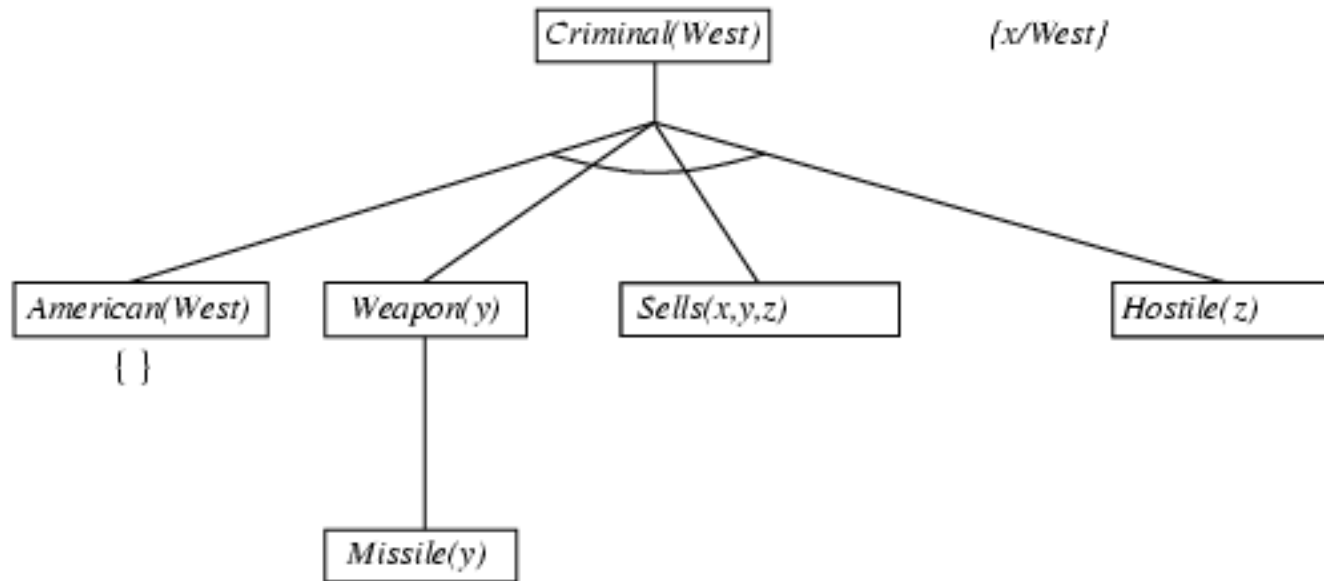
# BACKWARD CHAINING EXAMPLE



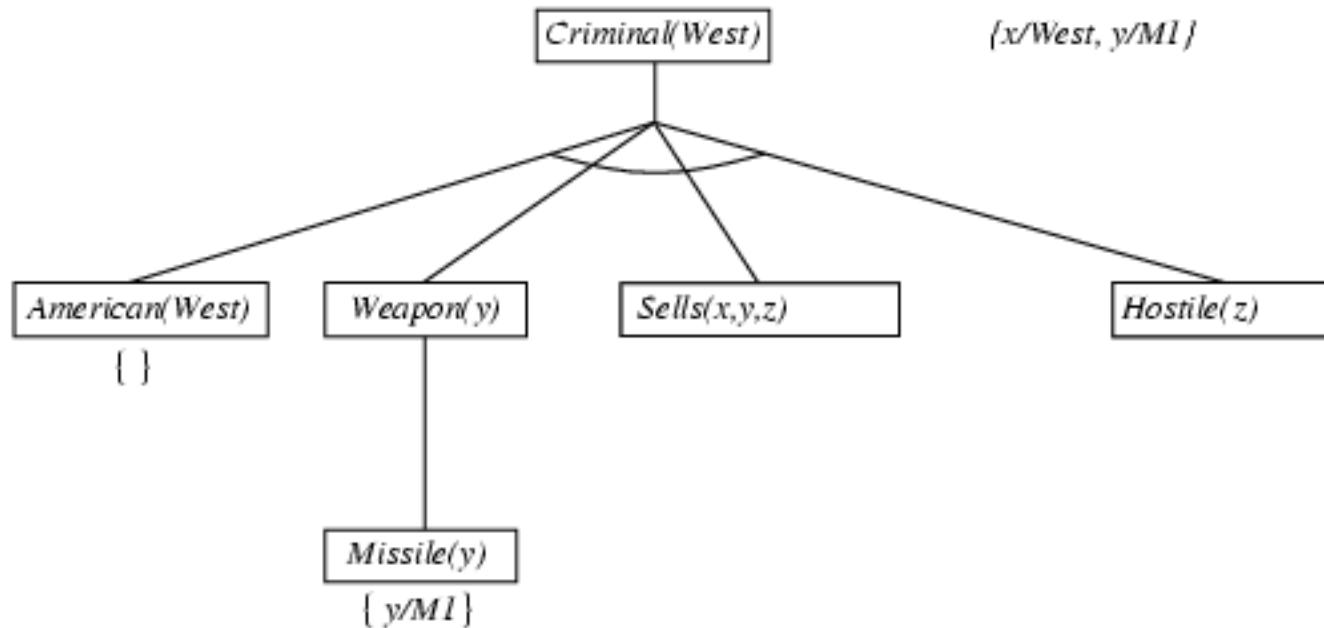
# BACKWARD CHAINING EXAMPLE



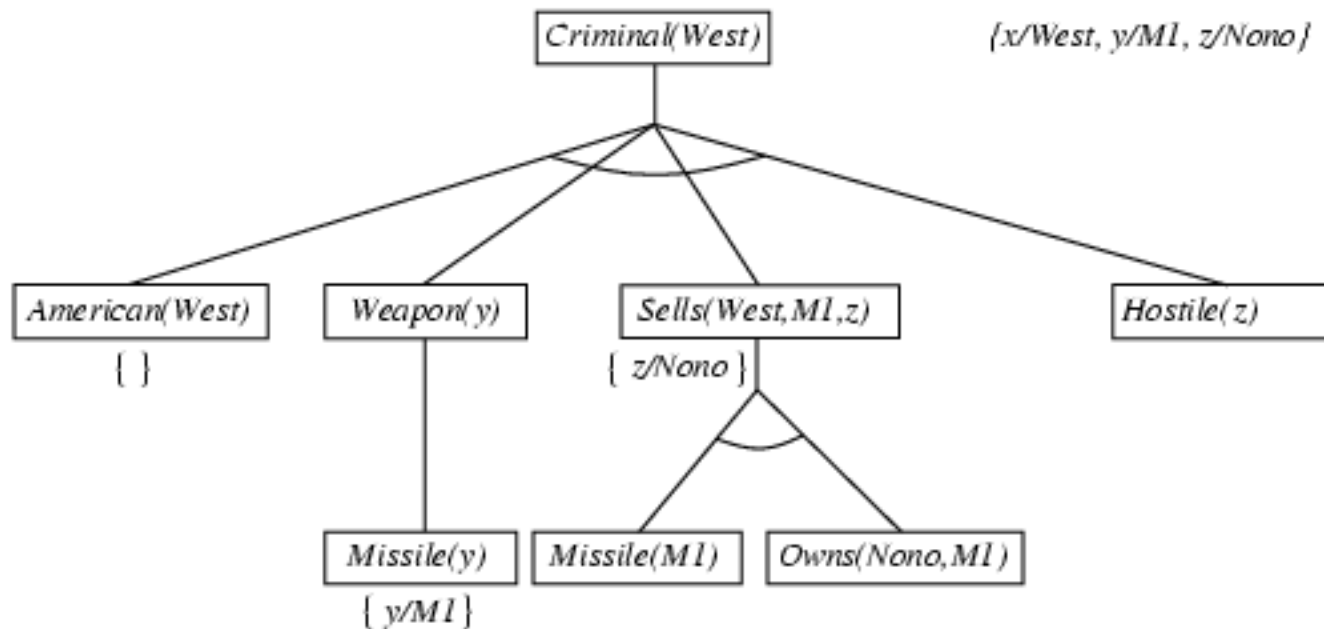
# BACKWARD CHAINING EXAMPLE



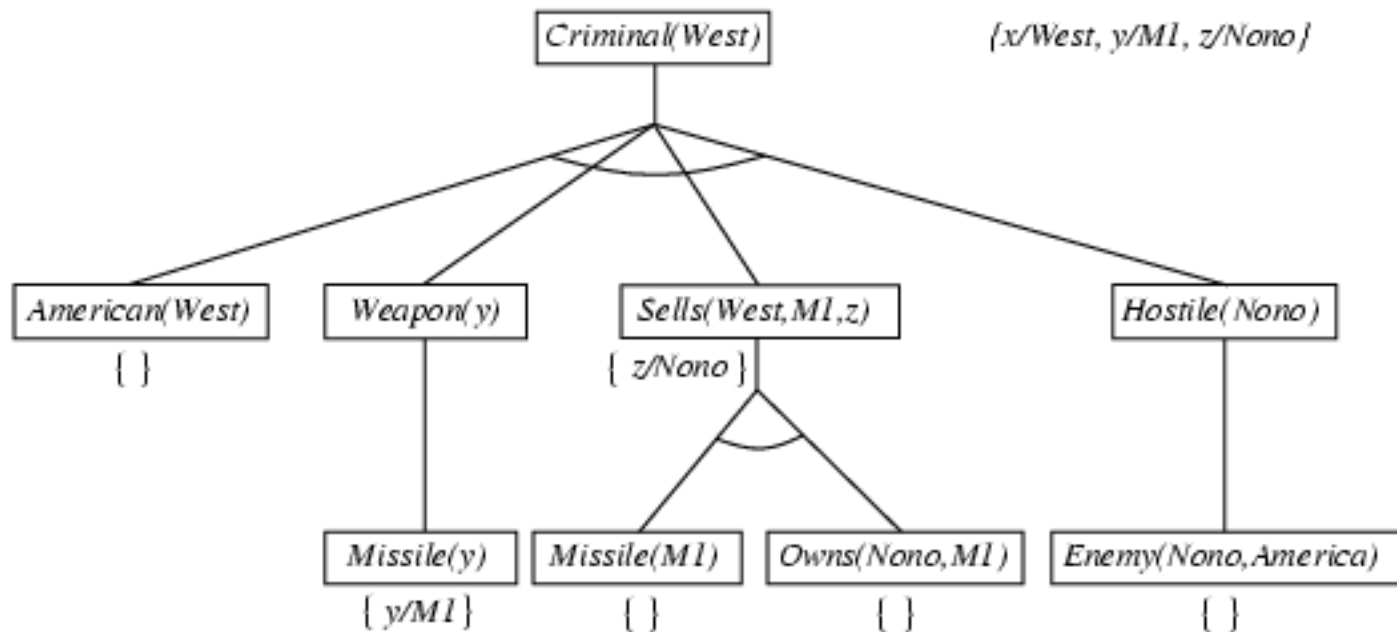
# BACKWARD CHAINING EXAMPLE



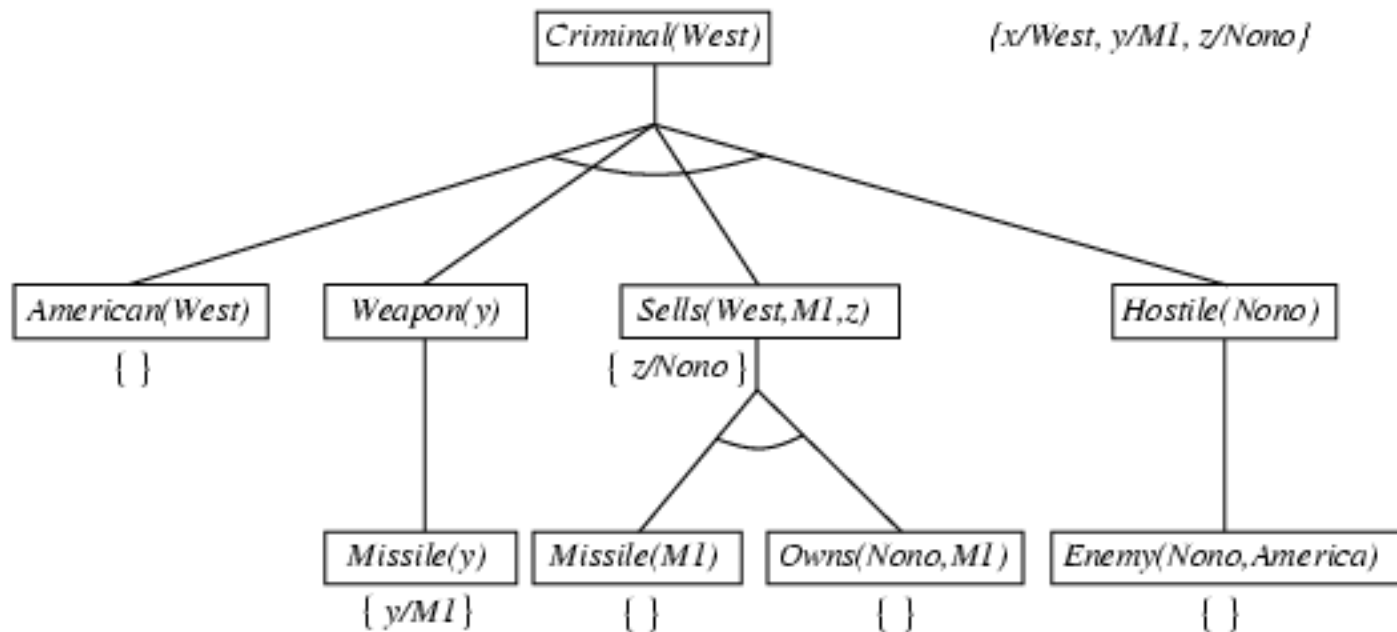
# BACKWARD CHAINING EXAMPLE



# BACKWARD CHAINING EXAMPLE



# BACKWARD CHAINING EXAMPLE



# PROPERTIES OF BACKWARD CHAINING

- Depth-first recursive proof search: space is linear in size of proof
- 
- Incomplete due to infinite loops
- - $\Rightarrow$  fix by checking current goal against every goal on stack
  -
- Inefficient due to repeated subgoals (both success and failure)
  - $\Rightarrow$  fix using caching of previous results (extra space)
  -
- Widely used for logic programming





# LOGIC PROGRAMMING: PROLOG

- Algorithm = Logic + Control

- 

- Basis: backward chaining with Horn clauses + bells & whistles  
Widely used in Europe, Japan (basis of 5th Generation project)  
Compilation techniques  $\Rightarrow$  60 million LIPS

- Program = set of clauses = head :- literal<sub>1</sub>, ... literal<sub>n</sub>.

- 

`criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).`

- Depth-first, left-to-right backward chaining
- Built-in predicates for arithmetic etc., e.g., `X is Y*Z+3`
- Built-in predicates that have side effects (e.g., input and output)
- 
- predicates, assert/retract predicates)
- Closed-world assumption ("negation as failure")
  - e.g., given `alive(X) :- not dead(X).`
  - `alive(joe)` succeeds if `dead(joe)` fails
  -



# PROLOG

- Appending two lists to produce a third:

- ```
append([ ], Y, Y) .  
append([X|L], Y, [X|Z]) :- append(L, Y, Z) .
```

- query: `append(A, B, [1, 2]) ?`

-

- answers: `A=[] B=[1, 2]`

-

`A=[1] B=[2]`

`A=[1, 2] B=[]`



RESOLUTION: BRIEF SUMMARY

- Full first-order version:

-

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \quad m_1 \vee \cdots \vee m_n}{\quad}$$

$(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta$
 where $\text{Unify}(\ell_i, \neg m_j) = \theta$.

- The two clauses are assumed to be standardized apart so that they share no variables.

-

- For example,

-

$$\frac{\neg \text{Rich}(x) \vee \text{Unhappy}(x)}{\text{Rich}(Ken) \quad \text{Unhappy}(Ken)}$$

with $\theta = \{x/Ken\}$

- Apply resolution steps to $\text{CNF}(\text{KB} \wedge \neg \alpha)$; complete for FOL

-



CONVERSION TO CNF

- Everyone who loves all animals is loved by someone:

$$\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x,y)] \Rightarrow [\exists y \text{ Loves}(y,x)]$$

- 1. Eliminate biconditionals and implications

- $$\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Loves}(x,y)] \vee [\exists y \text{ Loves}(y,x)]$$

- 2. Move \neg inwards: $\neg \forall x p \equiv \exists x \neg p$, $\neg \exists x p \equiv \forall x \neg p$

- $$\begin{aligned} & \forall x [\exists y \neg(\neg \text{Animal}(y) \vee \text{Loves}(x,y))] \vee [\exists y \text{ Loves}(y,x)] \\ & \forall x [\exists y \neg \neg \text{Animal}(y) \wedge \neg \text{Loves}(x,y)] \vee [\exists y \text{ Loves}(y,x)] \\ & \forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Loves}(x,y)] \vee [\exists y \text{ Loves}(y,x)] \end{aligned}$$



CONVERSION TO CNF CONTD.

- 3. Standardize variables: each quantifier should use a different one

- 4.
$$\forall x [\exists y \textit{Animal}(y) \wedge \neg \textit{Loves}(x,y)] \vee [\exists z \textit{Loves}(z,x)]$$

- 4. Skolemize: a more general form of existential instantiation.
Each existential variable is replaced by a **Skolem function** of the enclosing universally quantified variables:

$$\forall x [\textit{Animal}(F(x)) \wedge \neg \textit{Loves}(x,F(x))] \vee \textit{Loves}(G(x),x)$$

- 5. Drop universal quantifiers:

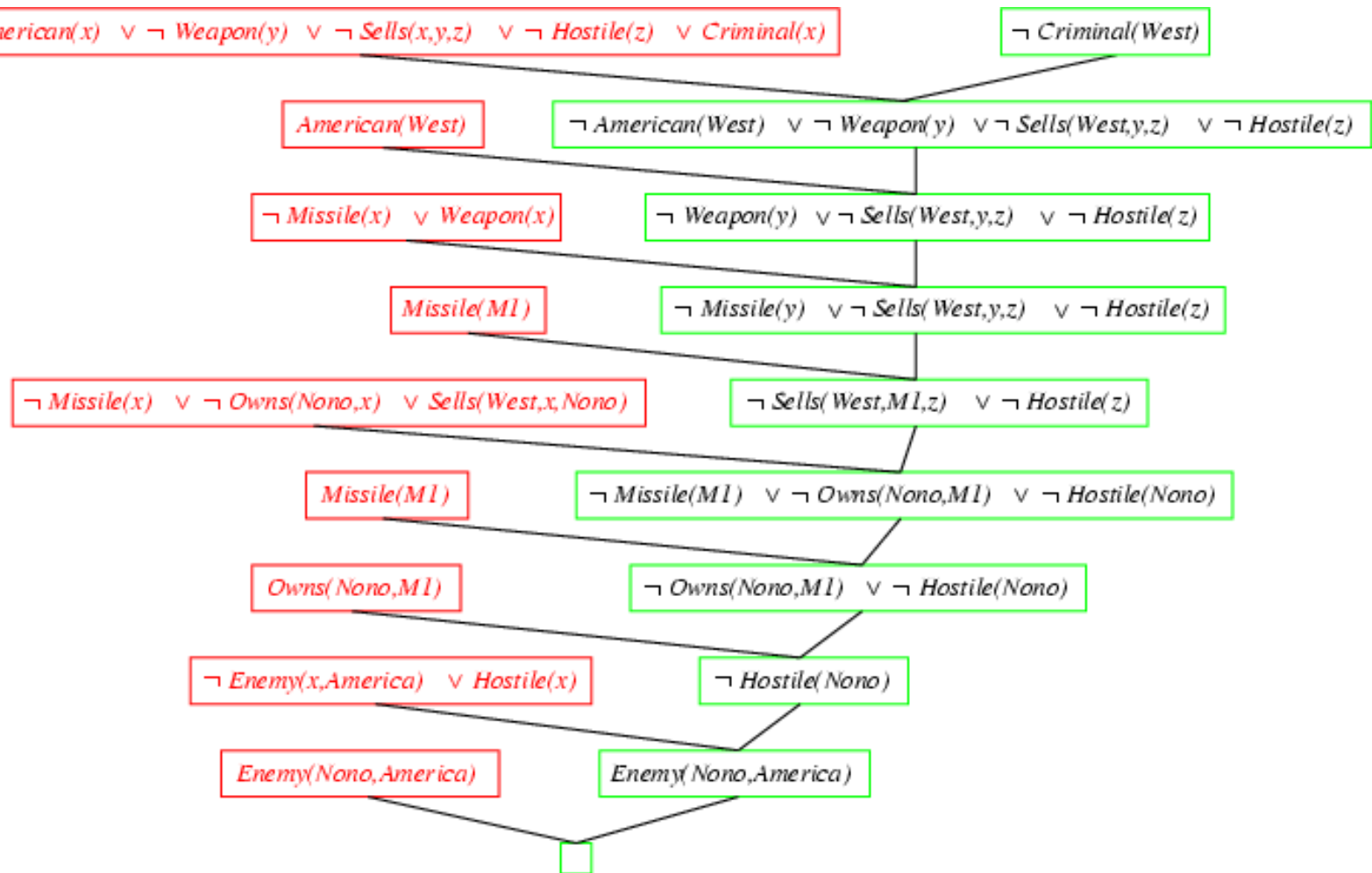
- 6.
$$[\textit{Animal}(F(x)) \wedge \neg \textit{Loves}(x,F(x))] \vee \textit{Loves}(G(x),x)$$

- 6. Distribute \vee over \wedge :

- 7.
$$[\textit{Animal}(F(x)) \vee \textit{Loves}(G(x),x)] \wedge [\neg \textit{Loves}(x,F(x)) \vee \textit{Loves}(G(x),x)]$$



RESOLUTION PROOF: DEFINITE CLAUSES



תרגילים

דוגמה להסקה בעזרת רזולוציה.

הבעיה

בחדר נמצאים שלושה אנשים A, B, C . A נשוי, C לא-נשוי. A מסתכל על B , B מסתכל על C .

תאר את הבעיה בעזרת פסוקים בצורה נורמלית.

הסק בעזרת רזולוציה שנמצא בחדר אדם נשוי שמביט על אדם לא-נשוי.

בסיס הידע

- $T \rightarrow \text{married}(A)$
- $\text{married}(C) \rightarrow F$
- $T \rightarrow \text{looks_at}(A, B)$
- $T \rightarrow \text{looks_at}(B, C)$

השאלתה

- $\exists x. \exists y. \text{looks_at}(x, y) \wedge \neg \text{married}(y) \wedge \text{married}(x)$

שלילת השאלתה

- $\neg \exists x. \exists y. \text{looks_at}(x, y) \wedge \neg \text{married}(y) \wedge \text{married}(x)$
- $\forall x. \forall y. \neg \text{looks_at}(x, y) \vee \text{married}(y) \vee \neg \text{married}(x)$

ובצורה נורמלית

- $\text{looks_at}(x, y) \wedge \text{married}(x) \rightarrow \text{married}(y)$

תהליך ההסקה

- $T \rightarrow \text{married}(A)$
- $\text{married}(C) \rightarrow F$
- $T \rightarrow \text{looks_at}(A, B)$
- $T \rightarrow \text{looks_at}(B, C)$
- $\text{looks_at}(x, y) \wedge \text{married}(x) \rightarrow \text{married}(y)$
- $(5, 3) \text{ married}(A) \rightarrow \text{married}(B)$
- $(5, 4) \text{ married}(B) \rightarrow \text{married}(C)$
- $(6, 1) \neg \text{married}(B)$
- $(7, 2) \text{ married}(B) \rightarrow F$
- $(9, 8) T \rightarrow F$

○ רזולוציה

○ בסיס ידע:

○ למשה יש מחלת גבהים

○ חוקים:

○ מי שיש לו מחלת גבהים לא מטיס מטוס.

○ לכל מטוס יש טייס.

○ במטוס יש נוסעים וטייסים.

○ משה ודוד טסים במטוס "רום".

○ מטרה:

○ דוד הוא טייס של מטוס "רום"

○

○ החלף את המשפטים הנ"ל ל First Order Logic. (יש לזכור להגדיר את כל הפרדיקטים והקבועים)

○ העבר את המשפטים שקבלת לצורת Conjunctive Normal Form.

○ הוכח או הפרך את המטרה ע"י שימוש בחוקי הרזולוציה. הראה את פעולות האיחוד (unification) הנדרשות.

○ השתמשו בפרדיקטים:

○ MS (עבור מחלת גבהים), PT(עבור טייס), IP (במטוס), AP(מטוס), Pass(נוסע)

○ השתמשו בקבועים

○ M (עבור משה), D (עבור דוד), R("רום")

○ :

פתרון

○ לוגיקה מסדר ראשון, צורה נורמלית ורזולוציה

○ תיאור המצב:

- פרד הוא קולי.
- סם הוא הבעלים של פרד.
- היום שבת.
- בשבת לא חם.
- פרד מאולף.
- ספניאלים הם כלבים טובים וגם קולי מאולפים הם כלבים טובים.
- אם כלב הוא כלב טוב ויש לו בעלים אז הוא יימצא עם בעליו.
- אם שבת וחם בשבת אז סם בפארק.
- אם שבת ולא חם בשבת אז סם במוזיאון.



פרד הוא קולי.
סם הוא הבעלים של פרד.
היום שבת.
בשבת לא חם.
פרד מאולף.
ספניאלים הם כלבים טובים וגם קולי מאולפים הם כלבים טובים.
אם כלב הוא כלב טוב ויש לו בעלים אז הוא ימצא עם בעליו.
אם שבת וחם בשבת אז סם בפארק.
אם שבת ולא חם בשבת אז סם במוזיאון.

לוגיקה מסדר ראשון:

collie(Fred)
master(Fred, Sam)
day(Sat)
 \neg warm(Sat)
trained(Fred)
 $\forall x$ [spaniel(x) \vee (collie(x) \wedge trained(x))] \rightarrow gooddog(x)
 $\forall x, y, z$ [gooddog(x) \wedge master(x, y) \wedge location(y, z)] \rightarrow location(x, z)
[day(Sat) \wedge warm(Sat)] \rightarrow location(Sam, Park)
[day(Sat) \wedge \neg warm(Sat)] \rightarrow location(Sam, Museum)



collie(Fred)
 master(Fred, Sam)
 day(Sat)
 \neg warm(Sat)
 trained(Fred)
 $\forall x$ [spaniel(x) \vee (collie(x) \wedge trained(x))] \rightarrow gooddog(x)
 $\forall x,y,z$ [gooddog(x) \wedge master(x,y) \wedge location(y,z)] \rightarrow location(x,z)
 [day(Sat) \wedge warm(Sat)] \rightarrow location(Sam,Park)
 [day(Sat) \wedge \neg warm(Sat)] \rightarrow location(Sam,Museum)

בסיס הידע בצורה נורמלית:

$T \rightarrow$ collie(Fred)
 $T \rightarrow$ master(Fred, Sam)
 $T \rightarrow$ day(Sat)
 warm(Sat) \rightarrow F
 $T \rightarrow$ trained(Fred)
 spaniel(a) \rightarrow gooddog(a)
 collie(b) \wedge trained(b) \rightarrow gooddog(b)
 gooddog(c) \wedge master(c,d) \wedge location(d,e) \rightarrow location(c,e)
 day(Sat) \wedge warm(Sat) \rightarrow location(Sam,Park)
 day(Sat) \rightarrow warm(Sat) \vee location(Sam,Museum)



○ שאילתות

- ננסה לענות על השאלתה: האם פרד במוזיאון?
- האם לכל כלב ובעליו מתקיים: אם הם באותו מקום אז היום שבת ?
- ננסה, לסיום, לענות על השאלתה: האם יש מקום שכולם בו?

פתרון



ההוכחה האונטולוגית של אנסלם לקיום האל.

○ הנחות היסוד:

○ 1. אלוהים הוא הדבר הגדול ביותר שניתן להעלות על הדעת.

○ 2. דבר קיים גדול יותר מאותו דבר בדמיון.

○ בלוגיקה מסדר ראשון:

○ $\forall x \text{ exist}(\text{real-of}(\text{God})) \rightarrow \text{bigger}(\text{real-of}(\text{God}), x)$

○ $\forall x \neg \text{exist}(\text{real-of}(\text{God})) \rightarrow \text{bigger}(\text{img-of}(\text{God}), x)$

○ $\forall x \neg \text{bigger}(\text{img-of}(x), \text{real-of}(x))$

○ בצורה נורמלית:

○ $\text{exist}(\text{real-of}(\text{God})) \rightarrow \text{bigger}(\text{real-of}(\text{God}), a)$

○ $T \rightarrow \text{exist}(\text{real-of}(\text{God})) \vee \text{bigger}(\text{img-of}(\text{God}), b)$

○ $\text{bigger}(\text{img-of}(c), \text{real-of}(c)) \rightarrow F$

○ נוסף שאילתא:

○ $\text{exists}(\text{real-of}(\text{God})) \rightarrow F$



שאלה

- התחתנתי עם אלמנה (W). לאשתי בת בוגרת (D). אבי (F), שהרבה לבקרנו, התאהב בבת ונשא אותה לאשה.
- תארו את הנ"ל בלוגיקה מסדר ראשון.
- הוסיפו לבסיס הידע הגדרות רלבנטיות לקשרי משפחה.
- הסיקו בעזרת רזולוציה שאני סבא של עצמי.



שאלה

- ברזולוצית קלט (input resolution) מרשים שימוש בכלל הרזולוציה רק אם לפחות אחת משתי הפסוקיות המשתתפות ברזולוציה שייכת לפסוק המקורי. (כלומר, לא מרשים שימוש בכלל הרזולוציה אם שתי הפסוקיות המשתתפות בגזירת הרזולוציה אינן שייכות לפסוק המקורי).
- האם רזולוצית קלט שלמה להפרכה? הוכיחו.
- רמז: התבוננו בשלב האחרון של גזירת הפסוקית הריקה מן הפסוק המקורי.
-



לכל זוג של פסוקים אטומיים שלהלן, מצאו את המאחד הכללי ביותר (MGU), אם הוא קיים:

$P(A, B, B), P(x, y, z)$

תשובה:

- $P(x, y, z) = P(A, B, B)$
- $x=A, y=B, z=B$
- MGU: $\{x/A, y/B, z/B\}$
- Result: $P(A, B, B)$
-
- $Q(y, G(A, B)), Q(G(x, x), y)$

תשובה:

- $Q(G(x, x), y) = Q(y, G(A, B))$
- $G(x, x) = y, y = G(A, B)$
- $G(x, x) = G(A, B)$
- false*

-
- $Older(Father(y), y), Older(Father(x), John)$

תשובה:

- $Older(Father(y), y) = Older(Father(x), John)$
- $Father(y) = Father(x), y = John$
- $Father(John) = Father(x), y = John$
- $x = John, y = John$
- MGU: $\{y/John, x/John\}$
- Result: $Older(Father(John), John)$
- $Knows(Father(y), y) Knows(x, x)$

תשובה:

- $Knows(Father(y), y) = Knows(x, x)$
- $Father(y) = x, y = x$
- false*

