UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Artificial Intelligence

Propositional Logic: Inference rules

LESSON 12

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

Inference: General concepts

- Two sentences α and β are logically equivalent ($\alpha \equiv \beta$), if they are *true* under the same models, i.e., *if and only if*
  - $\alpha \models \beta$ and $\beta \models \alpha$
  - For instance $(P \land Q) \equiv (Q \land P)$
- A sentence is valid if it is *true* in all models
  - It is also called a tautology
    - $P \lor \neg P$
- A sentence is satisfiable if it is *true* only in some model
  - $P \land Q$

Inference: General concepts

- Two useful properties related to the above concepts
  - Deduction theorem
    - For any α and β, $\alpha \vDash \beta$ if and only if $\alpha \Rightarrow \beta$ is valid
      - Hence, given a set KB of premises and a possible conclusion, the model-checking inference algorithm works by checking whether $KB \Rightarrow \alpha$ is valid

  - satisfiability is related to the standard mathematical proof technique of *reductio ad absurdum* (proof by refutation or by contradiction):

$$\alpha \vDash \beta \text{ if and only if } (\alpha \wedge \neg \beta) \text{ is unsatisfiable}$$

Inference Rules

- Practical inference algorithms are based on inference rules to avoid the exponential computational complexity of model checking
- An inference rule represents a standard pattern of inference:
  - it implements a simple reasoning step whose soundness can be easily proven and applied to a set of premises with a specific structure to derive a conclusion
- Inference rules are represented as follows:

$$\frac{\text{premises}}{\text{conclusion}}$$

Examples of Inference Rules

- The first five rules easily generalize to any set of sentences $\alpha_1, \ldots, \alpha_n$

And Elimination	$\frac{\alpha_1 \wedge \alpha_2}{\alpha_i}$, $i = 1, 2$
And Introduction	$\frac{\alpha_1, \alpha_2}{\alpha_1 \wedge \alpha_2}$
Or Introduction	$\frac{\alpha_1}{\alpha_1 \vee \alpha_2}$ (α_2 can be **any** sentence)
First De Morgan's law	$\frac{\neg(\alpha_1 \wedge \alpha_2)}{\neg\alpha_1 \vee \neg\alpha_2}$
Second De Morgan's law	$\frac{\neg(\alpha_1 \vee \alpha_2)}{\neg\alpha_1 \ \wedge \ \neg\alpha_2}$
Double Negation	$\frac{\neg(\neg\alpha)}{\alpha}$
Modus Ponens	$\frac{\alpha \Rightarrow \beta, \ \alpha}{\beta}$

Soundness of Inference Rules

- Since inference rules usually involve a few sentences, their soundness can be easily proven using model checking

- Example: Modus Ponens $$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

premise α	conclusion β	premise $\alpha \Rightarrow \beta$
false	false	true
false	true	true
true	false	false
true	true	true

Inference Rules

- Modus Ponens

If it is raining, then Harry is inside
It is raining

Harry is inside

Modus Ponens

$$\frac{\alpha \rightarrow \beta \qquad \alpha}{\beta}$$

Inference Rules

- And Elimination

Harry is friends with Ron and Hermione

Harry is friend with Hermione

And Elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

Inference Rules

- Double Negation Elimination

It is not true that Harry did not pass the test

Harry passed the test

Double Negation Elimination

$$\frac{\neg(\neg\alpha)}{\alpha}$$

Inference Rules

- Implication Elimination

If it is raining, then Harry is inside

It is not raining or Harry is inside

Implication Elimination

$$\frac{\alpha \rightarrow \beta}{\neg\alpha \vee \beta}$$

Inference Rules

- Biconditional elimination

It is raining if and only if Harry is inside

If it is raining, then Harry is inside, and if Harry is inside, then it is raining

Biconditional Elimination

$$\frac{\alpha \leftrightarrow \beta}{(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)}$$

De Morgan's Law

It is not true that both Harry and Ron passed the test

Harry did not pass the test or Ron did not pass the test

De Morgan's Law

$$\frac{\neg(\alpha \land \beta)}{\neg\alpha \lor \neg\beta}$$

De Morgan's Law

It is not true that Harry or Ron passed the test

Harry did not pass the test and Ron did not pass the test

De Morgan's Law

$$\frac{\neg(\alpha \vee \beta)}{\neg\alpha \wedge \neg\beta}$$

Distributive Property

$$\frac{(\alpha \land (\beta \lor \gamma))}{(\alpha \land \beta) \lor (\alpha \land \gamma)}$$

Distributive Property

$$\frac{(\alpha \vee (\beta \wedge \gamma))}{(\alpha \vee \beta) \wedge (\alpha \vee \gamma)}$$

Inference Algorithms

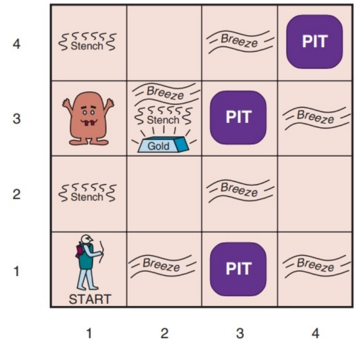
- Given a set of premises KB and a hypothetical conclusion α, the goal of an inference algorithm A is to find a proof KB $\vdash_A \alpha$ (if any)
  - A sequence of applications of inference rules that leads from KB to α

Inference Algorithms: Example

- In the initial configuration of the Wumpus game shown in the figure below, the agent's KB includes

(a) $\neg B_{1,1}$ (current percept)

(b) $\neg B_{1,1} \Rightarrow \neg P_{1,2} \wedge \neg P_{2,1}$
(one of the rules of the game)



- The agent can be interested in knowing whether room (1,2) contains a pit, i.e., whether KB $\models P_{1,2}$:
  - Applying modus ponens to (a) and (b) it derives (c) $\neg P_{1,2} \wedge \neg P_{2,1}$
  - Applying And elimination to (c), it derives $\neg P_{1,2}$
- Hence, it can conclude that room (1,2) does not contain a pit

Search Problems

- Initial state
- Actions
- Transition model
- Goal test
- Path cost function

Theorem Proving as a Search Problem

- Initial state: starting knowledge base
- Actions: inference rules
- Transition model: new knowledge base after inference
- Goal test: check statement we're trying to prove
- Path cost function: number of steps in proof

Proof by Resolution

- What about the completeness of the inference algorithm?
  - If the search algorithm that uses the inference rule is complete and the rules are adequate the inference algorithm is complete
  - However, if the inference rule is not adequate, for instance, the goal is unreachable
  - Therefore, we turn on a single inference rule, the resolution, that yields a complete inference algorithm when coupled with any complete search algorithm

Resolution

- Resolution is based on another inference rule that let us prove anything that can be proven about a KB

(Ron is in the Great Hall) v (Hermione is in the library)

Ron is not in the Great Hall

Hermione is in the library

Resolution: Unit Resolution Rule

$$\frac{P \lor Q \qquad \neg P}{Q}$$

Resolution

$$\frac{\begin{array}{c} P \lor Q_1 \lor Q_2 \lor \ldots \lor Q_n \\ \neg P \end{array}}{Q_1 \lor Q_2 \lor \ldots \lor Q_n}$$

- The two statements resolve to produce a new statement
  - that is, $Q_1 \lor \ldots \lor Q_n$

Resolution

(Ron is in the Great Hall) v (Hermione is in the library)

(Ron is not in the Great Hall) v (Harry is sleeping)

(Hermione is in the library) v (Harry is sleeping)

Resolution

$$\frac{\begin{array}{c} P \vee Q \\ \neg P \vee R \end{array}}{Q \vee R}$$

Resolution

$$\frac{\begin{array}{c} P \lor Q_1 \lor Q_2 \lor \ldots \lor Q_n \\ \neg P \lor R_1 \lor R_2 \lor \ldots \lor R_m \end{array}}{Q_1 \lor Q_2 \lor \ldots \lor Q_n \lor R_1 \lor R_2 \lor \ldots \lor R_m}$$

Clause and Conjunctive Normal Form

- A disjunction of literals
  - e.g. $P \vee Q \vee R$
- Disjunction means literals connected with *or*
- Conjunction means literals connected with *and*
- Literal is either a propositional symbol or the opposite of a propositional symbol
- Any logical sentence can be turned into a conjunctive normal form (CNF)
  - That is, a logical sentence that is a conjunction of clauses

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

Conversion to CNF

- Eliminate biconditionals
  - turn $(\alpha \leftrightarrow \beta)$ into $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
- Eliminate implications
  - turn $(\alpha \rightarrow \beta)$ into $\neg\alpha \vee \beta$
- Move ¬ inwards using De Morgan's Laws
  - e.g. turn $\neg(\alpha \wedge \beta)$ into $\neg\alpha \vee \neg\beta$
- Use distributive law to distribute ∨ wherever possible

Conversion to CNF

$$(P \lor Q) \rightarrow R$$

$$\neg(P \lor Q) \lor R$$ eliminate implication

$$(\neg P \land \neg Q) \lor R$$ De Morgan's Law

$$(\neg P \lor R) \land (\neg Q \lor R)$$ distributive law

- Converting into a CNF is useful in order to apply the resolution
  - Inference by resolution

Inference by Resolution

$$\frac{P \vee Q \qquad \neg P \vee R}{(Q \vee R)}$$

Inference by Resolution

$$P \vee Q \vee S$$

$$\neg P \vee R \vee S$$

$$(Q \vee S \vee R \vee S)$$

Factoring -> eliminates all redundant variables

Inference by Resolution

$$P$$

$$\neg P$$

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$(\;)$$

Empty clause

- The empty clause is always false
- This is the base of the inference by resolution algorithm

Inference by Resolution

- To determine if KB $\models \alpha$:
  - Check if (KB $\land \neg\alpha$) is a contradiction?
    - If so, then KB $\models \alpha$
    - Otherwise, no entailment
- In practice
  - Convert if (KB $\land \neg\alpha$) to Conjunctive Normal Form
  - Keep checking to see if we can use the resolution to produce a new clause
    - If ever we produce the empty clause (equivalent to False), we have a contradiction, and KB $\models \alpha$
    - Otherwise, if we can't add new clauses, no entailment

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$(A \vee B)$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$(A \vee B)$ $\quad$ $\underline{(\neg B \vee C)}$ $\quad$ $\underline{(\neg C)}$ $\quad$ $(\neg A)$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$(A \vee B)$ $\quad$ $\underline{(\neg B \vee C)}$ $\quad$ $\underline{(\neg C)}$ $\quad$ $(\neg A)$ $\quad$ $(\neg B)$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$(A \vee B)$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$ $(\neg B)$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$\underline{(A \vee B)}$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$ $\underline{(\neg B)}$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$\underline{(A \vee B)}$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$ $\underline{(\neg B)}$ (A)

Inference by Resolution

Does $(A \lor B) \land (\neg B \lor C) \land (\neg C)$ entail A?

$$(A \lor B) \land (\neg B \lor C) \land (\neg C) \land (\neg A)$$

$(A \lor B)$ $(\neg B \lor C)$ $(\neg C)$ $(\neg A)$ $(\neg B)$ (A)

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$(A \vee B)$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$ $(\neg B)$ (A)

Inference by Resolution

Does $(A \lor B) \land (\neg B \lor C) \land (\neg C)$ entail A?

$$(A \lor B) \land (\neg B \lor C) \land (\neg C) \land (\neg A)$$

$(A \lor B)$ $(\neg B \lor C)$ $(\neg C)$ $\underline{(\neg A)}$ $(\neg B)$ $\underline{(A)}$ $(\,)$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$(A \vee B)$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$ $(\neg B)$ (A) $(\;)$

Horn Clauses

- In many domains of practical interest, the whole KB can be encoded as Horn clauses
  - Disjunction of literals of which at most one is positive
    - For instance, $(\neg P \vee Q \vee \neg S)$ is a Horn clause
- Horn clauses can be expressed as implications in which
  - The antecedent is a conjunction ($\wedge$) of atomic sentences (non-negated propositional symbols)
  - The consequent is a single atomic sentence

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q$$

  - As a special case, also atomic sentences (i.e., proposition symbols) and their negation can be rewritten as Horn clauses
    - Since $(P \Rightarrow Q) \Leftrightarrow (\neg P \vee Q)$:

$$\begin{array}{ccccc} P & \Leftrightarrow & \neg \mathit{True} \vee P & \Leftrightarrow & \mathit{True} \Rightarrow P \\ \neg P & \Leftrightarrow & \neg P \vee \mathit{False} & \Leftrightarrow & P \Rightarrow \mathit{False} \end{array}$$

Forward and Backward Chaining

- Two practical inference algorithms exist when
  - The KB can be expressed as a set of Horn clauses
  - The conclusion is an atomic and non-negated sentence
- These algorithms, named Forward and Backward Chaining, exhibit the following characteristics
  - A single inference rule: Modus Ponens
  - Completeness
  - Linear computational complexity in the size of the KB

Forward Chaining

- Given a Horn clauses-formed KB, Forward Chaining (FC) derives all the entailed (non-negated) sentences

```
function FORWARD-CHAINING (KB)
    repeat
        apply MP in all possible ways to sentences in KB
        add to KB the derived sentences not already present (if any)
    until some sentences not yet present in KB have been derived
    return KB
```

Forward Chaining

- FC is an example of data-driven reasoning
  - it starts from known data and derives its consequences
- For instance, in the *Wumpus game*
  - FC could be used to update the agent's knowledge about the environment (the presence of pits in each room, etc.), based on the new percepts after each move
- The inference engine of expert systems is inspired by the FC inference algorithm

Forward Chaining Example

- Consider a KB made up of Horn clauses

1. $P \Longrightarrow Q$
2. $L \land M \Longrightarrow P$
3. $B \land L \Longrightarrow M$
4. $A \land P \Longrightarrow L$
5. $A \land B \Longrightarrow L$
6. A
7. B

$\ldots$

Forward Chaining Example

- Through FC we get
  8. The only implication whose premises (individual symbols) are in the KB is 5:
     - MP derives L and adds it to the current KB
  9. Now the premises of 3 are all true:
     - MP derives M and adds it to the KB
  10. The premises of 2 have become all true:
      - MP derives P and adds it to the KB
  11. The premises of 1 and 4 are now all true:
      - MP derives Q from 1 and adds it to the KB, but disregards 4 since its consequent (L) is already in the KB
  12. No new sentences can be derived from 1-11:
      - FC ends and returns the updated KB containing the original sentences 1-7 and the ones derived in the above steps: *{L, M, P, Q}*

Backward Chaining

- For a given KB made up of Horn clauses, and a given atomic, non-negated sentence α, FC can be used to prove whether KB $\vDash \alpha$
  - So, one must check if α is present or not among the derived sentences
- In this case, Backward Chaining (BC) is more effective
  - It recursively applies MP backward, since KB $\vDash \alpha$ iff
    - Either α in KB or
    - KB contains some implication $\beta_1, \ldots, \beta_n \Rightarrow \alpha$ and (recursively) KB $\vDash \beta_1, \ldots,$ KB $\vDash \beta_n$
- The sentence α to be proved is called a query

Backward Chaining

function BACKWARD-CHAINING (KB, α)
    if $\alpha \in KB$ **then return** True
    let B be the set of sentences of KB having α as the consequent
    for each $\beta \in B$
        let $\beta_1, \beta_2, \ldots$ be the propositional symbols in the antecedent of β
        if BACKWARD-CHAINING $(KB, \beta_i) =$ True for all β_i's
        then return True
    return False

Backward Chaining

- BC is a form of goal-directed reasoning
  - in the *Wumpus game*, it could be used to answer queries like given the current agent's knowledge, is *moving upward* the best action?
- The computational complexity of BC is even lower than FC since BC focuses only on relevant sentences
- The Prolog inductive logic programming language is based on the predicate logic version of the BC inference algorithm

Backward Chaining Example

- Consider a KB representing the rules followed by a financial institution for deciding whether to grant a loan to an individual
- The following proposition symbols are used
  - *OK* -> the loan should be approved
  - *COLLAT* -> the collateral for the loan is satisfactory
  - *PYMT* -> the applicant is able to repay the loan
  - *REP* -> the applicant has a good financial reputation
  - *APP* -> the appraisal on the collateral is sufficiently greater than the loan amount
  - *RATING* -> the applicant has a good credit rating
  - *INC* -> the applicant has a good, steady income
  - *BAL* -> the applicant has an excellent balance sheet

Backward Chaining Example

- The KB is made up of the five rules (implications) on the left, and of the data about a specific applicant encoded by the four sentences on the right (all Horn clauses)

1. $COLLAT \wedge PYMT \wedge REP \Rightarrow OK$
2. $APP \Rightarrow COLLAT$
3. $RATING \Rightarrow REP$
4. $INC \Rightarrow PYMT$
5. $BAL \wedge REP \Rightarrow OK$
6. APP
7. $RATING$
8. INC
9. $\neg BAL$

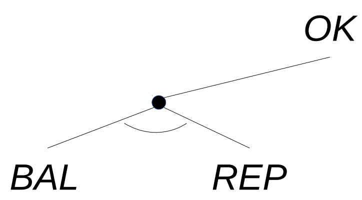
- Should the loan be approved for this specific applicant?
  - This amount to prove whether OK is entailed by the KB
    - KB ⊨ OK

Backward Chaining Example

- The BC recursive proof $KB \vdash_{BC} OK$ can be conveniently represented as an AND-OR graph, a tree-like graph in which
  - multiple links joined by an arc indicate a conjunction:
    - every linked proposition must be proven for proving the proposition in the parent node
  - multiple links without an arc indicate a disjunction:
    - any linked proposition can be proven for proving prove the proposition in the parent node

Backward Chaining Example

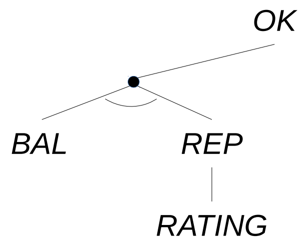
- The first call Backward-Chaining(KB,OK) is represented by the tree root, corresponding to the sentence to be proved
  - OK
- Since $OK \notin KB$, implications having OK as the consequent are searched for
- There are two such sentences: 1 and 5
  - The BC procedure tries to prove **all** the antecedents of **at least one** of them. Considering first 5, a recursive call to Backward-chaining is made for each of its two antecedents, represented by an AND-link:



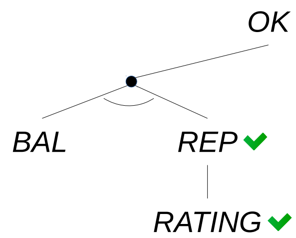
1. $COLLAT \land PYMT \land REP \Rightarrow OK$
2. $APP \Rightarrow COLLAT$
3. $RATING \Rightarrow REP$
4. $INC \Rightarrow PYMT$
5. $BAL \land REP \Rightarrow OK$
6. APP
7. $RATING$
8. INC
9. $\neg BAL$

Backward Chaining Example

- Consider the call *Backward-Chaining(KB,REP)*
  - Since REP ∉ KB, and the only implication having REP as consequent is 3, another recursive call is made for the antecedent of 3



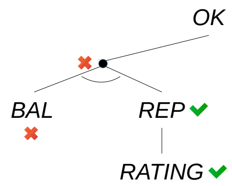
- The call *Backward-Chaining(KB,RATING)* returns *True*, since RATING ∈ KB, and thus also the call *Backward-Chaining(KB,REP)* returns True



1. $COLLAT \land PYMT \land REP \Rightarrow OK$
2. $APP \Rightarrow COLLAT$
3. $RATING \Rightarrow REP$
4. $INC \Rightarrow PYMT$
5. $BAL \land REP \Rightarrow OK$
6. APP
7. $RATING$
8. INC
9. $\neg BAL$

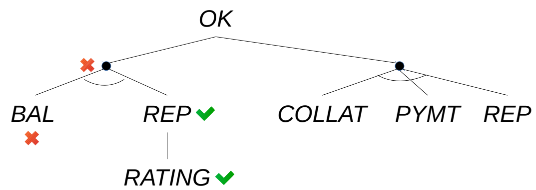
Backward Chaining Example

- However, the call *Backward-Chaining(KB,BAL)* returns *False* since BAL $\notin$ KB and there are no implications having BAL as a consequent
- Hence, the first call *Backward-Chaining(KB,OK)* is not able to prove OK through this AND-link



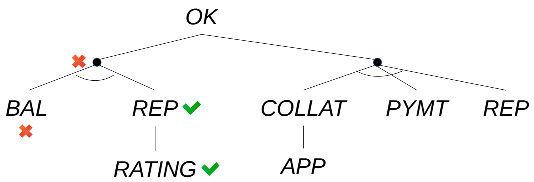
1. $COLLAT \wedge PYMT \wedge REP \Rightarrow OK$
2. $APP \Rightarrow COLLAT$
3. $RATING \Rightarrow REP$
4. $INC \Rightarrow PYMT$
5. $BAL \wedge REP \Rightarrow OK$
6. APP
7. $RATING$
8. INC
9. $\neg BAL$

- The other sentence in the KB having OK as the consequent, 1, is now considered, and another AND-link is generated with three recursive calls for each of the antecedents of 1

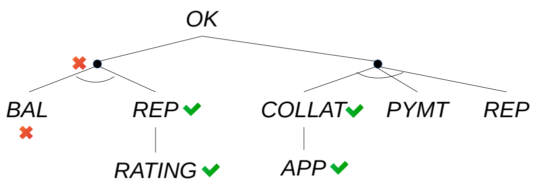


Backward Chaining Example

- The call *Backward-Chaining(KB,COLLAT)* generates in turn another recursive call to prove the antecedent of the only implication having COLLAT as the consequent, 2:



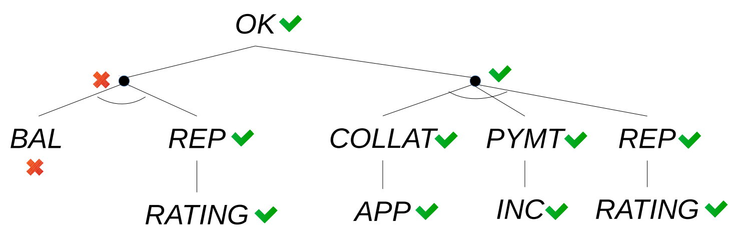
- The call *Backward-Chaining(KB,APP)* returns *True*, since APP ∈ KB, and thus also *Backward-Chaining(KB,COLLAT)* returns *True*



1. $COLLAT \wedge PYMT \wedge REP \Rightarrow OK$
2. $APP \Rightarrow COLLAT$
3. $RATING \Rightarrow REP$
4. $INC \Rightarrow PYMT$
5. $BAL \wedge REP \Rightarrow OK$
6. APP
7. $RATING$
8. INC
9. $\neg BAL$

Backward Chaining Example

- Similarly, the calls *Backward-Chaining(KB,PYMT)* and *Backward-Chaining(KB,REP)* return *True*
- The corresponding AND-link is the proved, which finally allows the first call *Backward-Chaining(KB,OK)* to return *True*



1. $COLLAT \wedge PYMT \wedge REP \Rightarrow OK$
2. $APP \Rightarrow COLLAT$
3. $RATING \Rightarrow REP$
4. $INC \Rightarrow PYMT$
5. $BAL \wedge REP \Rightarrow OK$
6. APP
7. $RATING$
8. INC
9. $\neg BAL$

- The proof $KB \vdash_{BC} OK$ is then successfully completed

Exercise

- Construct the agent's KB for the Wumpus game
- The KB should contain
  - The rules of the game
    - The agent starts in room (1,1), there is a breeze in rooms adjacent to pits, etc.
  - rules to decide the agent's move at each step of the game
- Note that the KB must be updated at each step of the game
  1. Adding percepts in the current room (from sensors)
  2. Reasoning to derive new knowledge about the position of pits and Wumpus
  3. Reasoning to decide the next move
  4. Updating the agent's position after a move

Limitations of propositional logic

- Main problems
  - Limited expressive power
    - Inferences involving the structure of atomic sentences (e.g., All men are mortal, …) cannot be made
  - Lack of conciseness
    - Even small KBs (in natural language) require many propositional symbols and sentences

From Propositional to Predicate Logic

- The description of many domains of interest for real-world applications (e.g., mathematics, philosophy, AI) involves the following elements in natural language:
  - nouns denoting objects (or persons), e.g.: Wumpus and pits; Socrates and Plato; the numbers one, two, etc.
  - predicates denoting properties of individual objects and relations between them, e.g.: Socrates is a man, five is prime, four is lower than five; the sum of two and two equals four
  - some relations between objects can be represented as functions, e.g.: "father of", "two plus two"
  - facts involving some or all objects, e.g.: all squares neighboring the Wumpus are smelly; some numbers are prime
- These elements cannot be represented in propositional logic, and require the more expressive predicate logic
  - The predicate logic version of the Resolution algorithm is used in automatic theorem provers, to assist mathematicians to develop complex proofs