

Artificial Intelligence

Inference in First-Order Logic

Readings: Chapter 9 of Russell &
Norvig.

A brief history of reasoning

450 _{B.C.}	Stoics	propositional logic, inference (maybe)
322 _{B.C.}	Aristotle	“syllogisms” (inference rules), quantification
1565	Cardano	probability theory (propositional logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	\exists complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	“practical” algorithm for propositional logic
1965	Robinson	“practical” algorithm for FOL— 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., $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \implies \text{Evil}(x)$ yields

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

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

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

⋮

Existential instantiation (EI)

For any sentence α , variable v , and constant symbol k that does not appear elsewhere in the knowledge base:

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

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.

UI versus EI

From $\forall x \forall y (y + x = y)$ we obtain

$$y + a = y$$

From $\exists x \forall y (y + x = y)$ we obtain

$$y + e = y$$

provided e is a new constant symbol.

- UI can be applied several times to *add* new sentences; the new KB is logically equivalent to the old.
- EI can be applied once to *replace* the existential sentence; the new KB is *not* equivalent to the old, but is satisfiable iff the old KB was satisfiable.

Reduction to Propositional Inference

Suppose the KB contains just the following:

$$\forall x \text{ King}(x) \wedge \text{Greedy}(x) \implies \text{Evil}(x)$$

King(John)

Greedy(John)

Brother(Richard, John)

Instantiating the universal sentence in *all possible* ways, we have

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

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

King(John)

Greedy(John)

Brother(Richard, John)

The new KB is propositionalized: proposition symbols are

Reduction to Propositional Inference

- Claim: a ground sentence is entailed by new KB iff entailed by original KB.
- Claim: every FOL KB can be propositionalized so as to preserve entailment.
- Idea: propositionalize KB and query, apply resolution, return result.
- Problem: with function symbols, ground terms are infinitely many, e.g., $Father(Father(Father(John)))$.
- Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a *finite* subset of the propositional KB.
- Idea: For $n = 0$ to ∞ , create a propositional KB by instantiating with depth- n terms see if α is entailed by this KB.
- Problem: works if α is entailed, loops if α is not entailed

Problems with Propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.
- E.g., from

$$\forall x \text{ King}(x) \wedge \text{Greedy}(x) \implies \text{Evil}(x)$$

$$\text{King}(\text{John})$$

$$\forall y \text{ Greedy}(y)$$

$$\text{Brother}(\text{Richard}, \text{John})$$

it seems obvious that $\text{Evil}(\text{John})$, but propositionalization produces lots of facts such as $\text{Greedy}(\text{Richard})$ that are irrelevant.

- In general, for one k -ary predicate and n constants, there are n^k instantiations!

Unification

- We can get the inference immediately if we can find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. That is, $\theta = \{x/John, y/John\}$ works
- $UNIFY(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	

Unification

- We can get the inference immediately if we can find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. That is, $\theta = \{x/John, y/John\}$ works
- $UNIFY(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

Unification

- We can get the inference immediately if we can find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. That is, $\theta = \{x/John, y/John\}$ works
- $UNIFY(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

Unification

- We can get the inference immediately if we can find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. That is, $\theta = \{x/John, y/John\}$ works
- $UNIFY(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

Unification

- We can get the inference immediately if we can find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. That is, $\theta = \{x/John, y/John\}$ works.
- $UNIFY(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	<i>fail</i>

Standardizing apart eliminates overlap of variables, e.g.,

$Knows(z_{17}, OJ)$

Conversion to CNF

Everyone who loves all animals is loved by someone:

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

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: $\neg \forall x, p \equiv \exists x \neg p$, $\neg \exists x, p \equiv \forall x \neg p$:

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

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

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

Conversion to CNF contd.

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

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

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

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

Conversion to CNF contd.

5. Drop universal quantifiers:

$$[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$$

6. Distribute \wedge over \vee :

$$[Animal(F(x)) \vee Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$$

Generalized Modus Ponens (GMP)

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

p_1' is *King(John)*

p_1 is *King(x)*

p_2' is *Greedy(y)*

p_2 is *Greedy(x)*

θ is $\{x/\text{John}, y/\text{John}\}$ q is *Evil(x)*

$q\theta$ is *Evil(John)*

- GMP used with KB of definite clauses (those clauses having *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 definite clause p , we have $p \models p\theta$ by UI

1. $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \models (p_1 \wedge \dots \wedge p_n \Rightarrow q)\theta$ and
 $(p_1 \wedge \dots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \dots \wedge p_n\theta \Rightarrow q\theta)$
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

Resolution

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \quad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

where $\text{UNIFY}(\ell_i, \neg m_j) = \theta$.

For example,

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

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

Soundness of Resolution

Need to show $l_1 \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_n \models c$ where c is $(l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)\theta$ and $\text{UNIFY}(l_i, \neg m_j) = \theta$.

This is true because

1. $l_1 \vee \dots \vee l_k \models (l_1 \vee \dots \vee l_k)\theta$ by UI,
2. $m_1 \vee \dots \vee m_n \models (m_1 \vee \dots \vee m_n)\theta$ by UI, and
3. $(l_1 \vee \dots \vee l_k)\theta$ and $(m_1 \vee \dots \vee m_n)\theta$ imply c by propositional resolution.

Using Resolution

1. Obtain $CNF(KB \wedge \neg\alpha)$
2. Apply resolution steps to the CNF
3. If the empty clause is generated, then $KB \models \alpha$.

Theorem: Resolution is a refutationally complete inference system for $KB \models \alpha$.

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:

Example Knowledge Base contd.

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

$$\textit{American}(x) \wedge \textit{Weapon}(y) \wedge \textit{Sells}(x, y, z) \wedge \textit{Hostile}(z) \implies \textit{Criminal}(x)$$

Nono ... has some missiles

Example Knowledge Base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \implies \text{Criminal}(x)$$

Nono ... has some missiles, i.e.,

$$\exists x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x):$$

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

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

Example Knowledge Base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \implies \text{Criminal}(x)$$

Nono ... has some missiles, i.e.,

$$\exists x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x):$$

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

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

$$\forall x \text{ Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \implies \text{Sells}(\text{West}, x, \text{Nono})$$

Missiles are weapons:

Example Knowledge Base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \implies \text{Criminal}(x)$$

Nono ... has some missiles, i.e.,

$$\exists x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x):$$

Owns(Nono, M_1) and *Missile*(M_1)

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

$$\forall x \text{ Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \implies \text{Sells}(\text{West}, x, \text{Nono})$$

Missiles are weapons:

$$\text{Missile}(x) \implies \text{Weapon}(x)$$

An enemy of America counts as “hostile”:

Example Knowledge Base contd.

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

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \implies \text{Criminal}(x)$$

Nono ... has some missiles, i.e.,

$$\exists x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x):$$

Owns(Nono, M₁) and Missile(M₁)

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

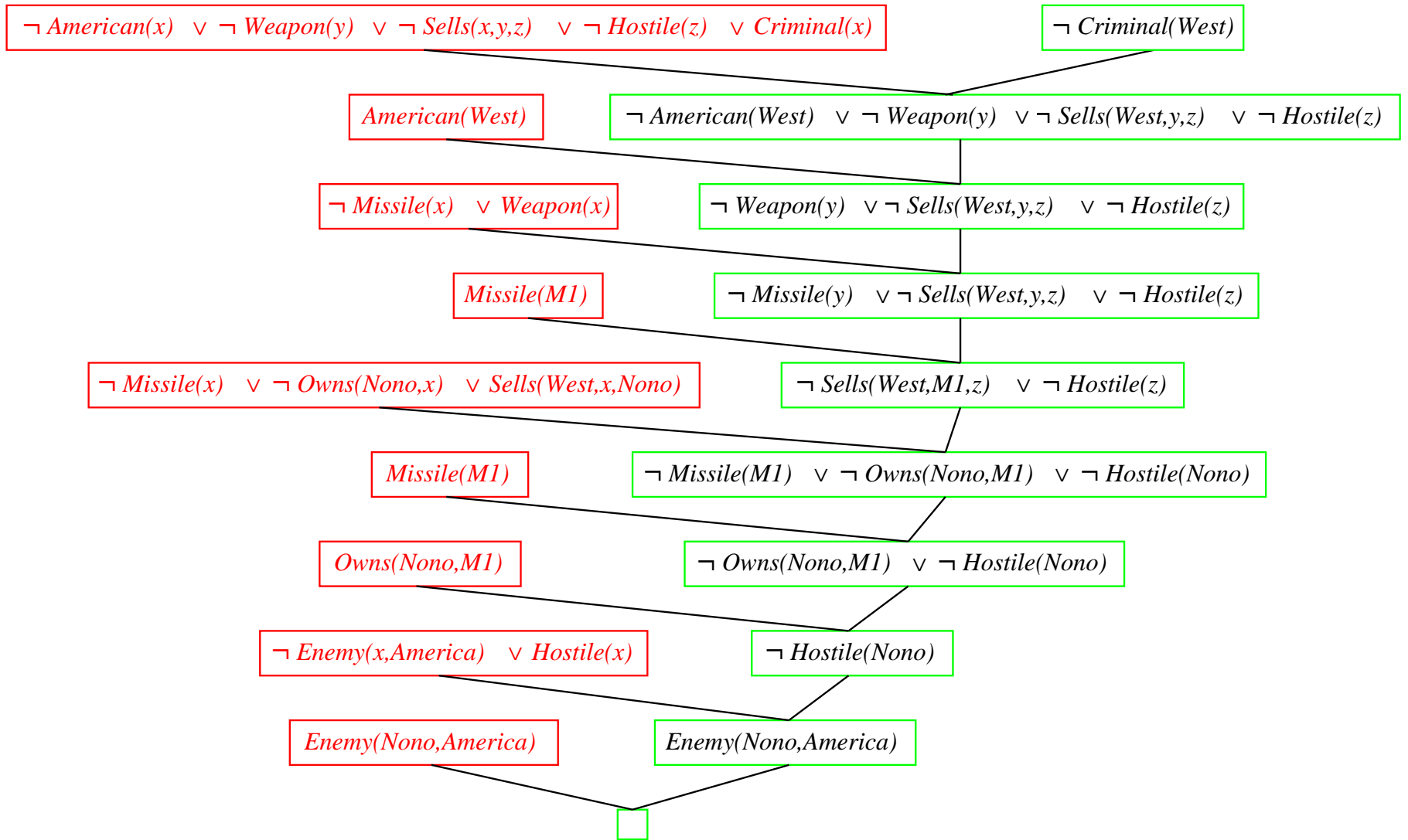
$$\forall x \text{ Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \implies \text{Sells}(\text{West}, x, \text{Nono})$$

Missiles are weapons:

$$\text{Missile}(x) \implies \text{Weapon}(x)$$

An enemy of America counts as “hostile”:

Resolution Proof: Definite Clauses



Logic programming

Sound bite: computation as inference on logical KBs

Logic programming	Ordinary programming
1. Identify problem	Identify problem
2. Assemble information	Assemble information
3. Tea break	Figure out solution
4. Encode information in KB	Program solution
5. Encode problem instance as facts	Encode problem instance as data
6. Ask queries	Apply program to data
7. Find false facts	Debug procedural errors

Should be easier to debug $Capital(NewYork, US)$ than
 $x := x + 2 !$

Prolog Systems

- A resolution inference system on Horn clauses + bells & whistles
- Widely used in Europe, Japan (basis of 5th Generation project)
Compilation techniques \Rightarrow 60 million LIPS
- Program = set of definite clauses, which are of form:
`head :- literal1, ... literaln.`

`weapon(X) :- missile(X).`

- Efficient unification by open coding
- Efficient retrieval of matching clauses by direct linking
- Depth-first, left-to-right backward chaining
- Built-in predicates for arithmetic etc., e.g., `X is Y*Z+3`
- Closed-world assumption (“negation as failure”) e.g., given
`alive(X) :- not dead(X).` `alive(joe)` succeeds if
`dead(joe)` fails.

Resolution Proof in Prolog

```
% it is a crime to sell weapons to hostile nations:
criminal(X):-american(X),weapon(Y),sells(X,Y,Z),hostile(Z)
% Nono ... has some missiles,
owns(nono,m1).
missile(m1).
% all of its missiles were sold to it by Colonel West
sells(west,X,nono) :- missile(X), owns(nono,X).
% Missiles are weapons
weapon(X) :- missile(X).
% An enemy of America counts as ``hostile``:
hostile(X) :- enemy(X,america).
% The country Nono, an enemy of America ...\al
enemy(nono,america).
% West, who is American ...\al
american(west).
?- criminal(Who).
Who = west.
```

Prolog Examples

- Depth-first search from a start state X:

```
dfs(X) :- goal(X).
```

```
dfs(X) :- successor(X,S), dfs(S).
```

No need to loop over S: successor succeeds for each

- 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=[]
```