Inference in first-order logic

Chapter 9
Reducing first-order inference to propositional inference

Unification

Generalized Modus Ponens

Forward and backward chaining

Logic programming

Resolution
A brief history of reasoning

450 B.C.  Stoics  propositional logic, inference (maybe)
322 B.C.  Aristotle "syllogisms" (inference rules), quantifiers
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 (reduce to propositional)
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:

\[
\forall v \alpha \\
\text{Subst}\{v/g\}, \alpha
\]

for any variable \( v \) and ground term \( g \)

E.g., \( \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \) yields

\[
King(John) \land Greedy(John) \Rightarrow Evil(John) \\
King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard) \\
King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John)) \\
\vdots
\]
Existential instantiation (EI)

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

$$\exists v \, \alpha \Rightarrow \text{Subst}\{\{v/k\}, \alpha\}$$

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

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

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

Another example: from $\exists x \, \frac{d(x^y)}{dy} = x^y$ we obtain

$$\frac{d(e^y)}{dy} = e^y$$

provided $e$ is a new constant symbol
Existential instantiation contd.

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 if the old KB was satisfiable.
Reduction to propositional inference

Suppose the KB contains just the following:

\[ \forall x \quad \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]
\[ \text{King}(\text{John}) \]
\[ \text{Greedy}(\text{John}) \]
\[ \text{Brother}(\text{Richard}, \text{John}) \]

Instantiating the universal sentence in all possible ways, we have

\[ \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \]
\[ \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \]
\[ \text{King}(\text{John}) \]
\[ \text{Greedy}(\text{John}) \]
\[ \text{Brother}(\text{Richard}, \text{John}) \]

The new KB is propositionalized: proposition symbols are

\[ \text{King}(\text{John}), \quad \text{Greedy}(\text{John}), \quad \text{Evil}(\text{John}), \quad \text{King}(\text{Richard}) \quad \text{etc.} \]
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, there are infinitely many ground terms, e.g., \( \text{Father(Father(Father(John)))} \)

Theorem: Herbrand (1930). If a sentence \( \alpha \) is entailed by an FOL KB, it is entailed by a finite subset of the propositional 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 in FOL is semidecidable
Problems with propositionalization

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

\[ \forall x \; \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]
\[ \text{King}(\text{John}) \]
\[ \forall y \; \text{Greedy}(y) \]
\[ \text{Brother}(\text{Richard}, \text{John}) \]

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

With \( p \) \( k \)-ary predicates and \( n \) constants, there are \( p \cdot n^k \) instantiations.

With function symbols, it gets much much worse!
We can get the inference immediately if we can find a substitution \( \theta \) such that \( \text{King}(x) \) and \( \text{Greedy}(x) \) match \( \text{King}(\text{John}) \) and \( \text{Greedy}(y) \)

\[ \theta = \{ x/\text{John}, y/\text{John} \} \text{ works} \]

\( \text{UNIFY}(\alpha, \beta) = \theta \) if \( \alpha \theta = \beta \theta \)

\begin{array}{ccc}
\text{p} & \text{q} & \theta \\
\text{Knows}(\text{John}, x) & \text{Knows}(\text{John}, \text{Jane}) & \\
\text{Knows}(\text{John}, x) & \text{Knows}(y, \text{OJ}) & \\
\text{Knows}(\text{John}, x) & \text{Knows}(y, \text{Mother}(y)) & \\
\text{Knows}(\text{John}, x) & \text{Knows}(x, \text{OJ}) & \\
\end{array}
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 $Greedy(y)$

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

$\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(John, Jane)$</td>
<td>${x/Jane}$</td>
</tr>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(y, OJ)$</td>
<td></td>
</tr>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(y, Mother(y))$</td>
<td></td>
</tr>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(x, OJ)$</td>
<td></td>
</tr>
</tbody>
</table>
We can get the inference immediately if we can find a substitution $\theta$ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$.

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

\[
\text{UNIFY}(\alpha, \beta) = \theta \text{ if } \alpha\theta = \beta\theta
\]

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{John}, \text{Jane})$</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{OJ})$</td>
<td>${x/\text{OJ}, y/\text{John}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Mother}(y))$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x, \text{OJ})$</td>
<td></td>
</tr>
</tbody>
</table>
Unification

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

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

$\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{John}, \text{Jane})$</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{OJ})$</td>
<td>${x/\text{OJ}, y/\text{John}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Mother}(y))$</td>
<td>${y/\text{John}, x/\text{Mother}(\text{John})}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x, \text{OJ})$</td>
<td></td>
</tr>
</tbody>
</table>
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

$\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(John, Jane)$</td>
<td>${x/Jane}$</td>
</tr>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(y, OJ)$</td>
<td>${x/OJ, y/John}$</td>
</tr>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(y, Mother(y))$</td>
<td>${y/John, x/Mother(John)}$</td>
</tr>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(x, OJ)$</td>
<td>fail</td>
</tr>
</tbody>
</table>

Standardizing apart eliminates overlap of variables, e.g., $Knows(z_{17}, OJ)$
Generalized Modus Ponens (GMP)

\[ p_1', \ p_2', \ \ldots, \ p_n', \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \]
\[ \frac{}{q_\theta} \]

where \( p_i' \theta = p_i \theta \) for all \( i \)

- \( p_1' \) is \( King(John) \)
- \( p_1 \) is \( King(x) \)
- \( p_2' \) is \( Greedy(y) \)
- \( p_2 \) is \( Greedy(x) \)
- \( \theta \) is \( \{x/John, y/John\} \)
- \( q \) is \( Evil(x) \)
- \( q_\theta \) is \( Evil(John) \)

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

Need to show that

\[ p_1', \ldots, p_n', (p_1 \land \ldots \land p_n \Rightarrow q) \models q^\theta \]

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. \((p_1 \land \ldots \land p_n \Rightarrow q) \models (p_1 \land \ldots \land p_n \Rightarrow q)^\theta = (p_1^\theta \land \ldots \land p_n^\theta \Rightarrow q^\theta)\)
2. \( p_1', \ldots, p_n' \models p_1' \land \ldots \land p_n' \models p_1^\theta \land \ldots \land p_n^\theta \)
3. From 1 and 2, \( q^\theta \) follows by ordinary Modus Ponens
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:

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles
it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono has some missiles, i.e., \( \exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono}, M_1) \ \text{and} \ \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( \exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

Missiles are weapons:
Example knowledge base contd.

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

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

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

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

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

\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

Missiles are weapons:

\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]

An enemy of America counts as “hostile”: 

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:
\[
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \implies \text{Criminal}(x)
\]

Nono ... has some missiles, i.e., \( \exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[
\text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1)
\]

... all of its missiles were sold to it by Colonel West
\[
\forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \implies \text{Sells}(\text{West}, x, \text{Nono})
\]

Missiles are weapons:
\[
\text{Missile}(x) \implies \text{Weapon}(x)
\]

An enemy of America counts as “hostile”:
\[
\text{Enemy}(x, \text{America}) \implies \text{Hostile}(x)
\]

West, who is American ...
\[
\text{American}(\text{West})
\]

The country Nono, an enemy of America ...
\[
\text{Enemy}(\text{Nono}, \text{America})
\]
Forward chaining algorithm

function FOL-FC-Ask(\(KB, \alpha\)) returns a substitution or \textit{false}

repeat until \textit{new} is empty

\(\textit{new} \leftarrow \{\}\)

for each sentence \(r\) in \(KB\) do

\((p_1 \land \ldots \land p_n \implies q) \leftarrow \text{STANDARDIZE-APART}(r)\)

for each \(\theta\) such that \((p_1 \land \ldots \land p_n)\theta = (p'_1 \land \ldots \land p'_n)\theta\)

for some \(p'_1, \ldots, p'_n\) in \(KB\)

\(q' \leftarrow \text{SUBST}(\theta, q)\)

if \(q'\) is not a renaming of a sentence already in \(KB\) or \textit{new} then do

add \(q'\) to \textit{new}

\(\phi \leftarrow \text{UNIFY}(q', \alpha)\)

if \(\phi\) is not \textit{fail} then return \(\phi\)

add \textit{new} to \(KB\)

return \textit{false}
Forward chaining proof

\begin{tabular}{|c|c|c|c|}
  \hline
  American(West) & Missile(M1) & Owns(Nono,M1) & Enemy(Nono,America) \\
  \hline
\end{tabular}
Forward chaining proof

Diagram:
- Weapon(M1)
- Sells(West,M1,Nono)
- Owns(Nono,M1)
- Hostile(Nono)
- American(West)
- Missile(M1)
- Enemy(Nono,America)
Forward chaining proof

Criminal(West)

Weapon(M1)  Sells(West,M1,Nono)  Hostile(Nono)

American(West)  Missile(M1)  Owns(Nono,M1)  Enemy(Nono,America)
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 already applied)
θ, the current substitution, initially the empty substitution {}

local variables: answers, a set of substitutions, initially empty

if goals is empty then return {θ}
q′ ← Subst(θ, First(goals))
for each sentence r in KB
    where Standardize-Apart(r) = (p₁ ∧ ... ∧ pₙ ⇒ q)
    and θ′ ← Unify(q, q′) succeeds
    new_goals ← [p₁, ... , pₙ]Rest(goals)
    answers ← FOL-BC-Ask(KB, new_goals, Compose(θ′, θ)) ∪ answers

return answers
Backward chaining example

Criminal(West)
Backward chaining example

\[
\text{Criminal}(\text{West}) \quad \{x/\text{West}\}
\]

- American\((x)\)
- Weapon\((y)\)
- \(\text{Sells}(x,y,z)\)
- Hostile\((z)\)
Backward chaining example

Criminal(West)  {x/West}

American(West)  Weapon(y)  Sells(x,y,z)  Hostile(z)

{ }
Backward chaining example

\begin{itemize}
\item \textit{American(West)}
\item \textit{Weapon(y)}
\item \textit{Sells(x,y,z)}
\item \textit{Hostile(z)}
\item \textit{Missile(y)}
\end{itemize}
Backward chaining example

- Criminal(West)
- \{x/West, y/M1\}
- American(West)
- \{\}
- Weapon(y)
- Missile(y)
- Sells(x,y,z)
- Hostile(z)
- \{y/M1\}
Backward chaining example

\[
\text{Criminal}(\text{West}) \quad \{x/\text{West}, y/M1, z/\text{Nono}\}
\]

\[
\text{American}(\text{West}) \quad \{\}
\]

\[
\text{Weapon}(y) \quad \{\} \quad \text{Sells}(\text{West}, M1, z) \quad \{z/\text{Nono}\} \quad \text{Hostile}(z)
\]

\[
\text{Missile}(y) \quad \{y/M1\} \quad \text{Missile}(M1) \quad \text{Owns}(\text{Nono}, M1)
\]
Backward chaining example

Criminal(West)

\{x/West, y/M1, z/Nono\}

American(West)
\{ \}

Weapon(y)
\{ \}

Sells(West,M1,z)
\{ z/Nono \}

Missile(y)
\{ y/M1 \}

Missile(M1)
\{ \}

Owns(Nono,M1)
\{ \}

Hostile(Nono)
\{ \}

Enemy(Nono,America)
\{ \}
Resolution: brief summary

Full first-order version:

\[
\frac{\ell_1 \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_n}{(\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)\theta}
\]

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

For example,

\[
\begin{align*}
\neg \text{Rich}(x) & \lor \text{Unhappy}(x) \\
\text{Rich}(Ken) & \\
\hline
\text{Unhappy}(Ken)
\end{align*}
\]

with \( \theta = \{x/\text{Ken}\} \)

Apply resolution steps to \( \text{CNF}(KB \land \neg \alpha) \); complete for FOL
Conversion to CNF

Everyone who loves all animals is loved by someone:
\[ \forall x \ [\forall y \ Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y \ Loves(y, x)] \]

1. Eliminate biconditionals and implications
\[ \forall x \ [\neg \forall y \ \neg Animal(y) \lor Loves(x, y)] \lor [\exists y \ Loves(y, x)] \]

2. Move \( \neg \) inwards: \( \neg \forall x, p \equiv \exists x \ \neg p \), \( \neg \exists x, p \equiv \forall x \ \neg p \):
\[ \forall x \ [\exists y \ \neg (\neg Animal(y) \lor Loves(x, y))] \lor [\exists y \ Loves(y, x)] \]
\[ \forall x \ [\exists y \ \neg Animal(y) \land \neg Loves(x, y)] \lor [\exists y \ Loves(y, x)] \]
\[ \forall x \ [\exists y \ Animal(y) \land \neg Loves(x, y)] \lor [\exists y \ Loves(y, x)] \]
Conversion to CNF contd.

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

\[ \forall x \ [\exists y \ Animal(y) \land \neg Loves(x, y)] \lor [\exists z \ Loves(z, x)] \]

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

\[ \forall x \ [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x) \]

5. Drop universal quantifiers:

\[ [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x) \]

6. Distribute \( \land \) over \( \lor \):

\[ [Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x)] \]
Resolution proof: definite clauses

\[ \neg American(x) \lor \neg Weapon(y) \lor \neg Sells(x,y,z) \lor \neg Hostile(z) \lor Criminal(x) \]

\[ \neg Criminal(West) \]

\[ American(West) \]

\[ \neg American(West) \lor \neg Weapon(y) \lor \neg Sells(West,y,z) \lor \neg Hostile(z) \]

\[ \neg Missile(x) \lor Weapon(x) \]

\[ \neg Missile(y) \lor \neg Sells(West,y,z) \lor \neg Hostile(z) \]

\[ Missile(M1) \]

\[ \neg Missile(M1) \lor \neg Owns(Nono,M1) \lor \neg Hostile(Nono) \]

\[ \neg Missile(M1) \lor \neg Owns(Nono,M1) \lor \neg Hostile(Nono) \]

\[ Missile(M1) \]

\[ \neg Sells(West,M1,z) \lor \neg Hostile(z) \]

\[ \neg Owns(Nono,M1) \lor \neg Hostile(Nono) \]

\[ Owns(Nono,M1) \]

\[ \neg Enemy(x,America) \lor Hostile(x) \]

\[ \neg Hostile(Nono) \]

\[ Enemy(Nono,America) \]

\[ Enemy(Nono,America) \]
Logic programming

Sound bite: computation as inference on logical KBs

- Logic programming
- Ordinary programming
  1. Identify problem
  2. Assemble information
  3. Tea break
  4. Encode information in KB
  5. Encode problem instance as facts
  6. Ask queries
  7. Find false facts

- Identify problem
- Assemble information
- Figure out solution
- Program solution
- Encode problem instance as data
- Apply program to data
- Debug procedural errors

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

Basis: backward chaining with Horn clauses + bells & whistles
Widely used in Europe, Japan (basis of 5th Generation project)
Compilation techniques ⇒ approaching a billion LIPS

Program = set of clauses = head :- literal₁, ... literalₙ.

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

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
Prolog examples

Depth-first search from a start state \( X \):

\[
\begin{align*}
dfs(X) & : \text{ goal}(X). \\
dfs(X) & : \text{ successor}(X,S), dfs(S).
\end{align*}
\]

No need to loop over \( S \): \text{successor} succeeds for each

Appending two lists to produce a third:

\[
\begin{align*}
\text{append}([], Y, Y). \\
\text{append}([X|L], Y, [X|Z]) & : \text{ append}(L, Y, Z).
\end{align*}
\]

query: append(A,B,[1,2]) ?
answers: A=[]{ } B=[1,2]
A=[1,2] B=[]