



# Logical Inference 2

## Rule-based reasoning

### Chapter 9

# Automated inference for FOL

- Automated inference for FOL is harder than PL
  - Variables can take on an infinite number of possible values from their domains
  - Hence there are potentially an infinite number of ways to apply the Universal Elimination rule
- Godel's Completeness Theorem says that FOL entailment is only semi-decidable
  - If a sentence is **true** given a set of axioms, there is a procedure that will determine this
  - If a sentence is **false**, there's no guarantee a procedure will ever discover this — it **may never halt**

# Generalized Modus Ponens (GMP)

- Modus Ponens:  $P, P \Rightarrow Q \models Q$
- Generalized Modus Ponens extends this to rules in FOL
- Combines And-Introduction, Universal-Elimination, and Modus Ponens, e.g.
  - given  $P(c), Q(c), \forall x P(x) \wedge Q(x) \rightarrow R(x)$
  - derive  $R(c)$
- Must deal with
  - more than one condition on rule's left side
  - variables

# Often rules restricted to Horn clauses

- A Horn clause is a sentence of the form:

$$P_1(x) \wedge P_2(x) \wedge \dots \wedge P_n(x) \rightarrow Q(x)$$

where

- $\geq 0$   $P_i$ s and 0 or 1  $Q$
- $P_i$ s and  $Q$  are positive (i.e., non-negated) literals
- Prolog and most rule-based systems are limited to Horn clauses
- Horn clauses are a subset of all FOL sentences

# Horn clauses 2

- Special cases

- Typical rule:  $P_1 \wedge P_2 \wedge \dots \wedge P_n \rightarrow Q$
- Constraint:  $P_1 \wedge P_2 \wedge \dots \wedge P_n \rightarrow \text{false}$
- A fact:  $\rightarrow Q$
- A goal:  $Q \rightarrow$

- Examples

- $\text{parent}(P1,P2) \wedge \text{parent}(P2,P3) \rightarrow \text{grandparent}(P1,P3)$
- $\text{male}(X) \wedge \text{female}(X) \rightarrow \text{false}$
- $\rightarrow \text{male}(\text{john})$
- $\text{female}(\text{mary}) \rightarrow$

# Horn clauses 3

- These are not Horn clauses:
  - $\text{married}(x, y) \rightarrow \text{loves}(x, y) \vee \text{hates}(x, y)$
  - $\neg \text{likes}(\text{john}, \text{mary})$
  - $\neg \text{likes}(x, y) \rightarrow \text{hates}(x, y)$
- Can't assert/conclude disjunctions (i.e., an "or")
- Can't have "true" negation
  - Though some systems, like Prolog, allow a negation operator that means "can't prove"
- No wonder Horn clause reasoning is easier

# Horn clauses 3

- Where are the quantifiers?
  - Variables in conclusion universally quantified
  - Variables only appearing in premises existentially quantified
- Examples:
  - $\text{parentOf}(P,C) \rightarrow \text{childOf}(C,P)$   
 $\forall P \forall C \text{parentOf}(P,C) \rightarrow \text{childOf}(C,P)$
  - $\text{parentOf}(P,X) \rightarrow \text{isParent}(P)$   
 $\forall P \exists X \text{parent}(P,X) \rightarrow \text{isParent}(P)$
  - $\text{parent}(P1, X) \wedge \text{parent}(X, P2) \rightarrow \text{grandParent}(P1, P2)$   
 $\forall P1,P2 \exists X \text{parent}(P1,X) \wedge \text{parent}(X, P2)$   
 $\rightarrow \text{grandParent}(P1, P2)$

# Definite Clauses

- A **definite clause** is a horn clause with a conclusion
- What's not allowed is a horn clause w/o a conclusion, e.g.
  - $\text{male}(x), \text{female}(x) \rightarrow$
  - i.e.,  $\text{male}(x) \vee \text{female}(x)$
- Most rule-based reasoning systems, like Prolog, allow only definite clauses in the KB



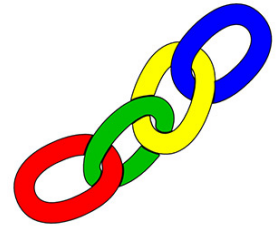
# Limitations

- Most rule-based reasoning systems use only definite horn clauses
  - Limited ability to reason about negation and disjunction
- Benefit is decidability and efficiency
- Some limitations can be overcome by
  - Adding procedural components
  - Augmenting with other reasoners

# Forward & Backward Reasoning

- We often talk about two reasoning strategies:
  - Forward chaining and
  - Backward chaining
- Both are equally powerful, but optimized for different use cases
- You can also have a mixed strategy

# Forward chaining

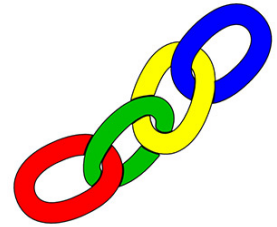


- Proofs start with given axioms/premises in KB, deriving new sentences using GMP until the goal/query sentence is derived
  - Process follows a chain of rules and facts going from the KB to the conclusion
- This defines a **forward-chaining** inference procedure because it moves “forward” from the KB to the goal [eventually]
- Inference using GMP is **sound** and **complete** for KBs containing **only Horn clauses**

# Forward chaining example

- KB:
  - $\text{allergies}(X) \rightarrow \text{sneeze}(X)$
  - $\text{cat}(Y) \wedge \text{allergicToCats}(X) \rightarrow \text{allergies}(X)$
  - $\text{cat}(\text{felix})$
  - $\text{allergicToCats}(\text{mary})$
- Goal:
  - $\text{sneeze}(\text{mary})$

# Backward chaining



- **Backward-chaining** deduction using GMP is also **complete** for KBs containing **only Horn clauses**
- Proofs start with the goal query, find rules with that conclusion, and then tries to prove each of the antecedents in the rule
- Keep going until you reach premises
- Avoid loops by checking if new subgoal is already on the goal stack
- Avoid repeated work: use a cache to check if new subgoal already proved true or failed

# Backward chaining example

- KB:
  - $\text{allergies}(X) \rightarrow \text{sneeze}(X)$
  - $\text{cat}(Y) \wedge \text{allergicToCats}(X) \rightarrow \text{allergies}(X)$
  - $\text{cat}(\text{felix})$
  - $\text{allergicToCats}(\text{mary})$
- Goal:
  - $\text{sneeze}(\text{mary})$

# Forward vs. backward chaining

- Forward chaining is data-driven
  - Automatic, unconscious processing, e.g., object recognition, routine decisions
  - May do lots of work that is irrelevant to the goal
  - Efficient when you want to compute **all conclusions**
- Backward chaining is goal-driven, better for problem-solving and query answering
  - Where are my keys? How do I get to my next class?
  - Complexity can be much less than linear wrt KB size
  - Efficient when you want **one or a few conclusions**
  - Good where the underlying facts are changing

# Mixed strategy

- Many practical reasoning systems do both forward and backward chaining
- The way you encode a rule determines how it is used, as in
  - % this is a forward chaining rule  
spouse(X,Y) => spouse(Y,X).
  - % this is a backward chaining rule  
wife(X,Y) <= spouse(X,Y), female(X).
- Given a model of the rules you have and the kind of reason you need to do, it's possible to decide which to encode as FC and which as BC rules.



# Completeness of GMP

- GMP (using forward or backward chaining) is complete for KBs that contain only Horn clauses
- **not complete** for simple KBs with **non-Horn clauses**
- What is entailed by the following sentences:
  1.  $(\forall x) P(x) \rightarrow Q(x)$
  2.  $(\forall x) \neg P(x) \rightarrow R(x)$
  3.  $(\forall x) Q(x) \rightarrow S(x)$
  4.  $(\forall x) R(x) \rightarrow S(x)$

# Completeness of GMP

- The following entail that  $S(A)$  is true:
  1.  $(\forall x) P(x) \rightarrow Q(x)$
  2.  $(\forall x) \neg P(x) \rightarrow R(x)$
  3.  $(\forall x) Q(x) \rightarrow S(x)$
  4.  $(\forall x) R(x) \rightarrow S(x)$
- If we want to conclude  $S(A)$ , with GMP we cannot, since the second one is not a Horn clause
- It is equivalent to  $P(x) \vee R(x)$