Introduction to Artificial Intelligence

9: Predicate Logic

Luca Doria, KPH Mainz

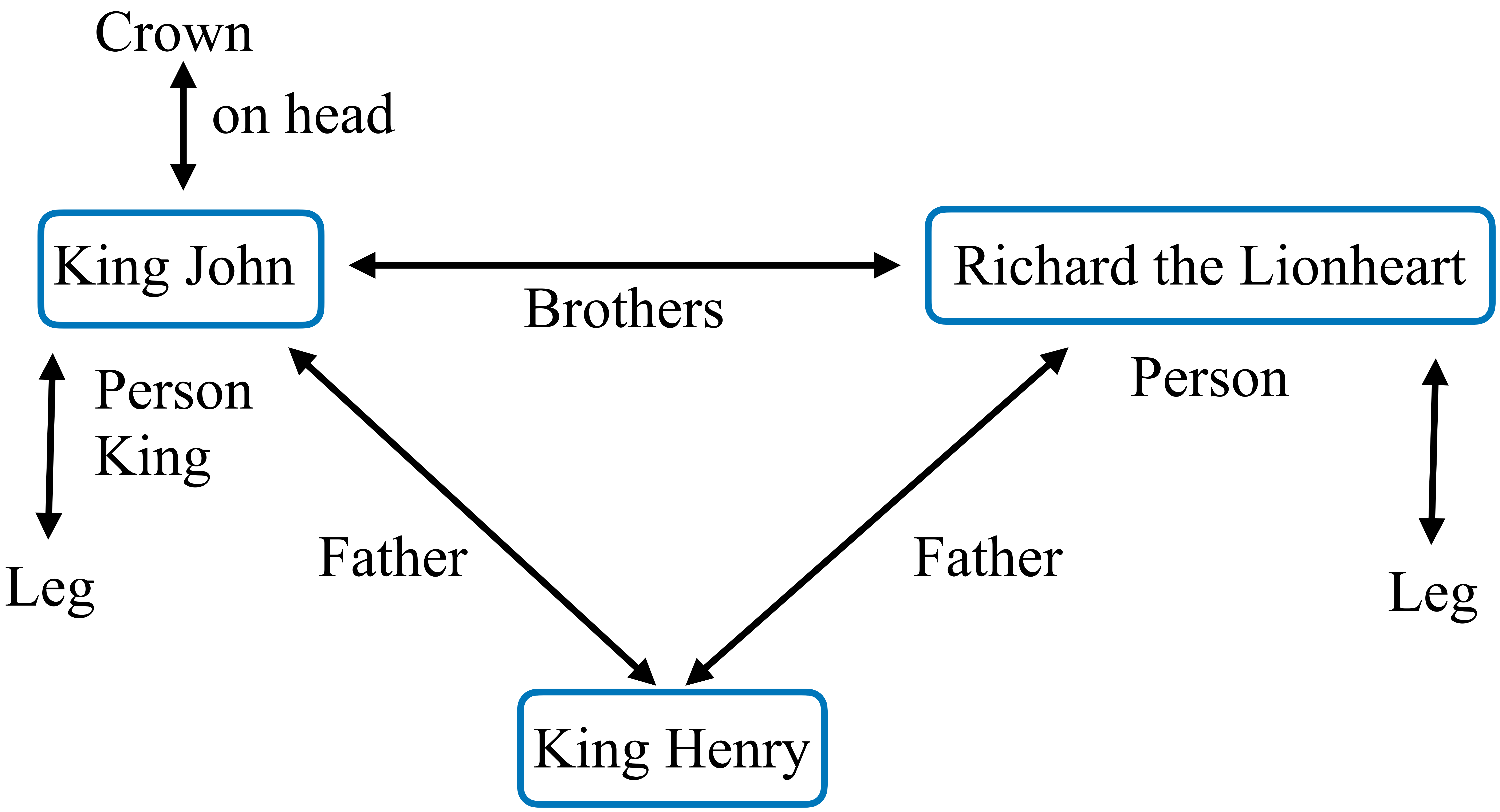
Motivation for extending Propositional Logic

- Propositional logic is quite expressive, but it has limitations.
- For example, what about sentences like:

  - "All crows are black"
  - "There is a crow A"
  - It should follow: "A is black"

- Propositional logic cannot describe this situation.
- To this aim we introduce variables, predicates, and functions.
- This leads to Predicate Logic (or first order logic, PL1)

Base Example used in the following



Syntax of PL1

Symbols of PL1 are divided in three classes:

- Constant Symbols (objects)
- Predicate Symbols (relations)
- Function Symbols

Predicates and Functions have an arity (number of arguments)

A model must provide information for determining if any given sentence is true or false. Besides objects, relations and functions, a model includes an interpretation.

The interpretation specifies precisely which objects, relations, and functions are referred to constant, predicate, and function symbols.

Interpretation Example

The symbol Richard —> Refers to Richard the Lionhaeart
The symbol John —> Refers to the King John
Brother —> Refers to the brotherhood relation, which is the set of tuples of objects:

{<Richard the Lionheart, King John> , <King John, Richard the Lionheart>}

Note: a tuple is a set of objects in fixed order between <,> brackets.

OnHead —> Refers to the relation between the crown and John
Person, King, Crown are unary relations identifying persons, kings and crowns.
LeftLeg —> Function defined by <King John>→John’s left leg … and so on..

Many interpretations are possible, e.g. Richard could be related to the crown and be king,…

Terms

A term is a logical expression which refers to an object.

- Constant symbols are terms (John, Richard, …)
- A constant with a predicate is a term: King(John)
- A function with its argument is a term: f(a, b, c,…)
- Variables are terms (see next)

Atomic and Complex Sentences

Atomic sentences (or atoms) are predicate symbols (optionally) followed by terms in parentheses.
Example:

- Brother(Richard, John)
- Married(Father(Richard), Mother(John))

Complex sentences combine atoms with logical connectives:

Examples (which are true in the model of slide 3):

$\neg$Brother(LeftLeg(Richard), John)
Brother(Richard, John) $\vee$ King(John)
$\neg$King(Richard) $\Rightarrow$ King(John)

Quantifiers

Having defined objects, we might be willing to express properties of groups of them. Quantifiers do exactly that.

Universal quantification $\forall$

Existential quantification $\exists$

Universal Quantifier

With this, we can finally compactly express concepts like:
"All the squares around a pit have a breeze"
"All squares around the Wumpus are smelly" …

Introducing the new concept of variable (x for example), we can state:

$\forall x$ King(x) $\Rightarrow$ Person(x)

Terms without variables are called ground terms.

Note: if we take x→John's left leg, the previous statement might sound strange.
<u>Since the implication is true even if its premise is false</u>, still the statement holds.
So, the universally quantified sentence is true in our model under each extended interpretation.(what you consider as "x").

Existential Quantifier

While the previous quantifier referred to every object, the existential quantifier refers to some of them.

Example:

$\exists x\ Crown(x) \wedge OnHead(x,John)$

Which can be instantiated in our model as ($x \rightarrow crown$)

- The crown is a crown AND the crown is on Richard's head

But also as ($x \rightarrow$ King John)

- King John is a crown AND King John is on John's head (?)

Note that the connective AND seems the right one to use with ∃, while the implication ⇒ matches well with ∀ .

Connections between Quantifiers and Equality

Extension of the De Morgan's Rules:

$$
\begin{aligned}
\forall x \ \neg P &\equiv \neg\exists x \ P & \qquad \neg(P \lor Q) &\equiv \neg P \land \neg Q \\
\neg\forall x \ P &\equiv \exists x \ \neg P & \qquad \neg(P \land Q) &\equiv \neg P \lor \neg Q \\
\forall x \ P &\equiv \neg\exists x \ \neg P & \qquad P \land Q &\equiv \neg(\neg P \lor \neg Q) \\
\exists x \ P &\equiv \neg\forall x \ \neg P & \qquad P \lor Q &\equiv \neg(\neg P \land \neg Q)
\end{aligned}
$$

Note that: $\forall$ works similarly to a conjunction, while $\exists$ to a disjunction.

The equality symbol "=" signify that two terms refer to the same object.
Example: Father(John)=Henry

PL1 Summary

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{Predicate} \mid \mathit{Predicate}(\mathit{Term}, \ldots) \mid \mathit{Term} = \mathit{Term} \\
\mathit{ComplexSentence} & \rightarrow & (\ \mathit{Sentence}\ ) \mid [\ \mathit{Sentence}\ ] \\
 & \mid & \neg\ \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \wedge \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \vee \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Rightarrow \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Leftrightarrow \mathit{Sentence} \\
 & \mid & \mathit{Quantifier}\ \mathit{Variable}, \ldots\ \mathit{Sentence} \\
 & & \\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
 & \mid & \mathit{Constant} \\
 & \mid & \mathit{Variable}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{After} \mid \mathit{Loves} \mid \mathit{Raining} \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \cdots \\
\text{OPERATOR PRECEDENCE} & : & \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

PL1 Summary

PL1 (or first-order logic) is a representation language which is much more expressive than propositional logic.

<u>Propositional logic commits only to the existence of facts.</u>
PL1 commits to the existence of objects (ontological commitment) and to relations among them (epistemological commitment).

Why we consider these languages? Because we would like to express knowledge in an expressive way (better if also declarative and compositional).

A possible world or model for PL1 is:

1) a set of objects
2) an interpretation that maps constant symbols to objects and predicate symbols to functions on objects.

An extended interpretation maps quantifier variables to objects in the model and thus define the truth of quantified sentences.

Resolution

Is it possible to extend the resolution technique of propositional logic to PL1 as inference procedure.

The idea is similar: reduce a sentence to conjunctive normal form (CNF) and then apply a version of the resolution procedure. The reduction to CNF will be more complex (we need to deal with the quantifiers!).

Reminder: the idea is to reduce our KB to CNF and then prove

$KB \models \phi$

by proving (by contradiction) that

$KB \wedge \neg \phi$ is unsatisfiable.

Step 1: Eliminate implications

Example:
"*Everyone who loves all animals is loved by someone*":

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

Implication elimination: $P \Rightarrow Q$ is equivalent to $\neg P \lor Q$:

First implication: $\forall x\ \neg[\forall y\ Animal(y) \Rightarrow Loves(x,y)] \lor [\exists\ y\ Loves(y,x)]$

Second implication: $\forall x\ \neg[\forall y\ \neg Animal(y) \lor Loves(x,y)] \lor [\exists\ y\ Loves(y,x)]$

Step 2: Move negations inwards

The negation rules with quantifiers are:

R1: $\neg\forall x\ P$ is equivalent to $\exists x\ \neg P$

R2: $\neg\exists x\ P$ is equivalent to $\forall x\ \neg P$

Previous step: $\forall x\ \neg[\forall y\ \neg Animal(y) \lor Loves(x,y)] \lor [\exists\ y\ Loves(y,x)]$

Apply R1: $\forall x\ [\exists y\ \neg(\neg Animal(y) \lor Loves(x,y))] \lor [\exists\ y\ Loves(y,x)]$

De Morgan: $\forall x\ [\exists y\ \neg\neg Animal(y) \land \neg Loves(x,y))] \lor [\exists\ y\ Loves(y,x)]$

Double negation: $\forall x\ [\exists y\ Animal(y) \land \neg Loves(x,y))] \lor [\exists\ y\ Loves(y,x)]$

Now the sentence reads: "*Either there is some animal that x doesn't love, or if this is not the case, someone loves x*", which is equivalent to the original one (check it!)

Step 3: Variables Standardization

For each sentence of the form: $(\exists x P(x)) \vee (\exists x Q(x))$ where we use the same variable twice, we change the name of one variable.
This will prevent confusion later on, where the quantifiers will be dropped.

In our example case:

Last Step: $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y))] \vee [\exists\ y\ Loves(y,x)]$

After Standardization: $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y))] \vee [\exists\ z\ Loves(z,x)]$

Step 4: "Skolemization"

The "skolemization" procedure consists in the removal of existential quantifiers applying instead Skolem Functions.

T. A. Skolem (1887 -1963)

Previous step: $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y))] \vee [\exists\ z\ Loves(z,x)]$

After Skolemization: $\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

The Skolem function "picks" the right element (realising $\exists$) among $\forall$.

Step 5: Drop universal quantifier

After Skolemization: $\forall x\ [\mathrm{Animal}(F(x)) \land \neg \mathrm{Loves}(x,F(x))] \lor \mathrm{Loves}(G(x),x)$

Note that the only variable left is “x” and without possibility of confusion, we can drop the universal quantifier :

$$[\mathrm{Animal}(F(x)) \land \neg \mathrm{Loves}(x,F(x))] \lor \mathrm{Loves}(G(x),x)$$

Step 6: Apply Distributive Law

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

Distributing $\vee$ over $\wedge$:

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

Which is finally in CNF: we can apply Resolution

Resolution Example: Statement of the problem

- Everyone who loves animals is loved by someone
- Anyone who kills an animal is loved by no-one
- Jack loves all animals.
- Either Jack or Curiosity killed the cat, who is named Tuna.
- Did curiosity kill the cat?

A. $\forall x \; [\forall y \; Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y \; Loves(y,x)]$

B. $\forall x \; [\exists z \; Animal(z) \wedge Kills(x,z)] \Rightarrow [\forall y \; \neg Loves(y,x)]$

C. $\forall x \; Animal(x) \Rightarrow Loves(Jack, x)$

D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

F. $\forall x \; Cat(x) \Rightarrow Animal(x)$

Knowledge Base

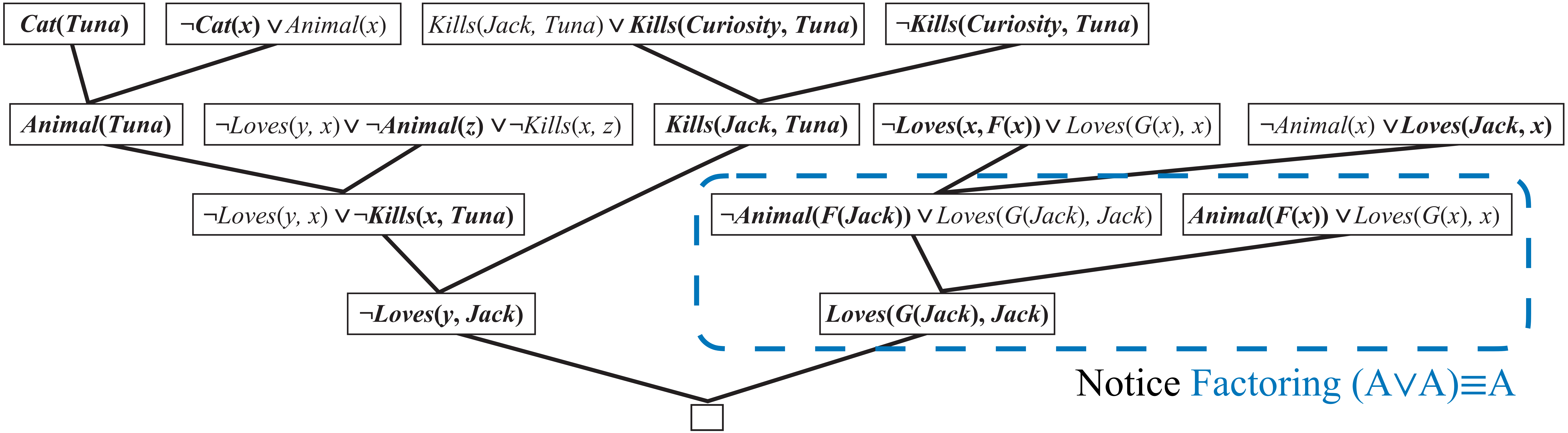
¬G. $\neg Kills(Curiosity, Tuna)$

Negated consequence

Resolution Example : Reduction to CNF

A1.	$Animal(F(x)) \lor Loves(G(x), x)$
A2.	$\neg Loves(x, F(x)) \lor Loves(G(x), x)$
B.	$\neg Loves(y, x) \lor \neg Animal(z) \lor \neg Kills(x, z)$
C.	$\neg Animal(x) \lor Loves(Jack, x)$
D.	$Kills(Jack, Tuna) \lor Kills(Curiosity, Tuna)$
E.	$Cat(Tuna)$
F.	$\neg Cat(x) \lor Animal(x)$
¬G.	$\neg Kills(Curiosity, Tuna)$

Resolution Example : Proof by Resolution



The Alphabet of PL1

Symbols

Operators: $\neg$, $\wedge$, $\vee$, $\forall$, $\exists$, $=$, $\Rightarrow$, $\Leftrightarrow$

Variables: $x_1, x_2, \ldots, x', x'', w, y, z, \ldots$ (lower case letters)

Brackets: (), [] , ..

Function Symbols e.g.: weight() , color(), …

Predicate Symbols e.g.: Crow(), Black(), …

Predicates and Symbols have an arity (number of arguments).
0-ary predicate = propositional logic atoms (P, Q, …)
0-ary function = constants: a, b, c …
We assume a <u>countable</u> set of predicates and functions of any arity.
<u>Note</u>: "=" is not considered a predicate but a logical symbol.

The Grammar of PL1

Terms (represent objects)

Every variable is a term.
If $t_1, t_2, \ldots, t_n$ are terms and f an n-ary function, then $f(t_1, t_2, \ldots, t_n)$ is also a term.
Variables: $x_1, x_2, \ldots, x', x'', w, y, z, ..$
Brackets: (), [] , ..

Function without variables (ground terms) e.g.: f() , f(g(), h(), …), …

Atomic formulae (statements about objects)

If $t_1, t_2, \ldots, t_n$ are terms and P is an n-ary predicate, then $P(t_1, t_2, \ldots, t_n)$ is an atomic formula.
If t_1, t_2 are terms $t_1=t_2$ is an atomic formula.
Atomic formulae without variables are <u>ground atoms</u>.

The Grammar of PL1

Atomic formulae (statements about objects)

- Every atomic formula is a formula
- If ϕ and ψ are formula and x is a variable, then

$$\neg\phi \wedge \psi, \phi \vee \psi, \phi \Rightarrow \psi, \phi \Leftrightarrow \psi, \exists x\phi \text{ and } \forall x\phi$$

are also a formulae.

- $\forall$, $\exists$ are as strongly binding as $\neg$.

Propositional logic is part of the PL1 language

- Atomic formulae: only 0-ary predicates
- Neither variables nor quantifiers

Meaning of PL1 Formulae

$\forall x[Crow(x) \Rightarrow Black(x)]$, $Crow(a)$

Means: for all objects x: if x is a Crow, then x is black and it is a crow.

In general:

- Terms are interpreted as objects
- Universally-quantified variables denote all objects in the universe made true by the quantified expression.
- Predicates represent subsets of the universe (e.g. some objects of the universe are crows).

Analogously to propositional logic, we define interpretations, satisfiability, models, validity, …

Summary of Reduction to Clausal Form

The clausal form is a standardization for PL1 formulas useful for automated reasoning.

The conversion process consists in the following steps:

1. Removing implications and biconditionals.
2. Moving negations inward.
3. Standardizing variables.
4. Moving quantifiers to the front.
5. Skolemization to eliminate existential quantifiers.
6. Dropping universal quantifiers.
7. Converting to conjunctive normal form (CNF).
8. Extracting clauses.

Clausal form allows efficient application of logical inference techniques like resolution.

Limitations of Logic Systems

	Propositional logic	First Order Logic
A logical system is **sound** if every theorem that can be derived using the system's inference rules is logically valid (i.e., true in all models). $\vdash \Rightarrow \models$	Propositional logic is sound. If a formula can be derived using the inference rules of propositional logic, then it is true in all possible interpretations (truth assignments).	First-order logic is sound. If a formula can be derived using the inference rules of first-order logic, then it is true in all models.
A logical system is complete if every logically valid formula (i.e., true in all models) can be derived using the system's inference rules. $\models \Rightarrow \vdash$	Propositional logic is complete. If a formula is true in all possible interpretations, it can be derived using the inference rules of propositional logic.	First-order logic is complete, as proven by Gödel's Completeness Theorem. This theorem states that if a formula is true in all models, it can be derived using the axioms and inference rules of first-order logic.

Gödel's Theorems

First Incompleteness Theorem:

In any consistent formal system F that is capable of expressing elementary arithmetic (allowing induction, includes Turing machines..), there exist statements that are true but not provable within the system.

This means that no sufficiently powerful and consistent formal system can be both complete and sound. There will always be true arithmetic statements that the system cannot prove.

Second Incompleteness Theorem

A sufficiently powerful and consistent formal system F cannot prove its own consistency (i.e. you cannot prove P and ¬P).

This means that the consistency of a formal system capable of arithmetic cannot be established by the system itself.