Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - Inference Rules. – Generalised Modus Ponens.
  - Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - Frames and Semantic / Conceptual Networks.
  - Managing Retractions, Assumptions, and Explanations.

2ANIM

Example of Resolution Proof (CNF)

¬P(w) ∨ Q(w)　　¬Q(y) ∨ S(y)

{y/w}

¬P(w) ∨ S(w)　　P(x) ∨ R(x)

{w/x}

S(x) ∨ R(x)　　¬R(z) ∨ S(z)

{x/A, z/A}

S(A)

KB:

¬P(w) ∨ Q(w)

P(x) ∨ R(x)

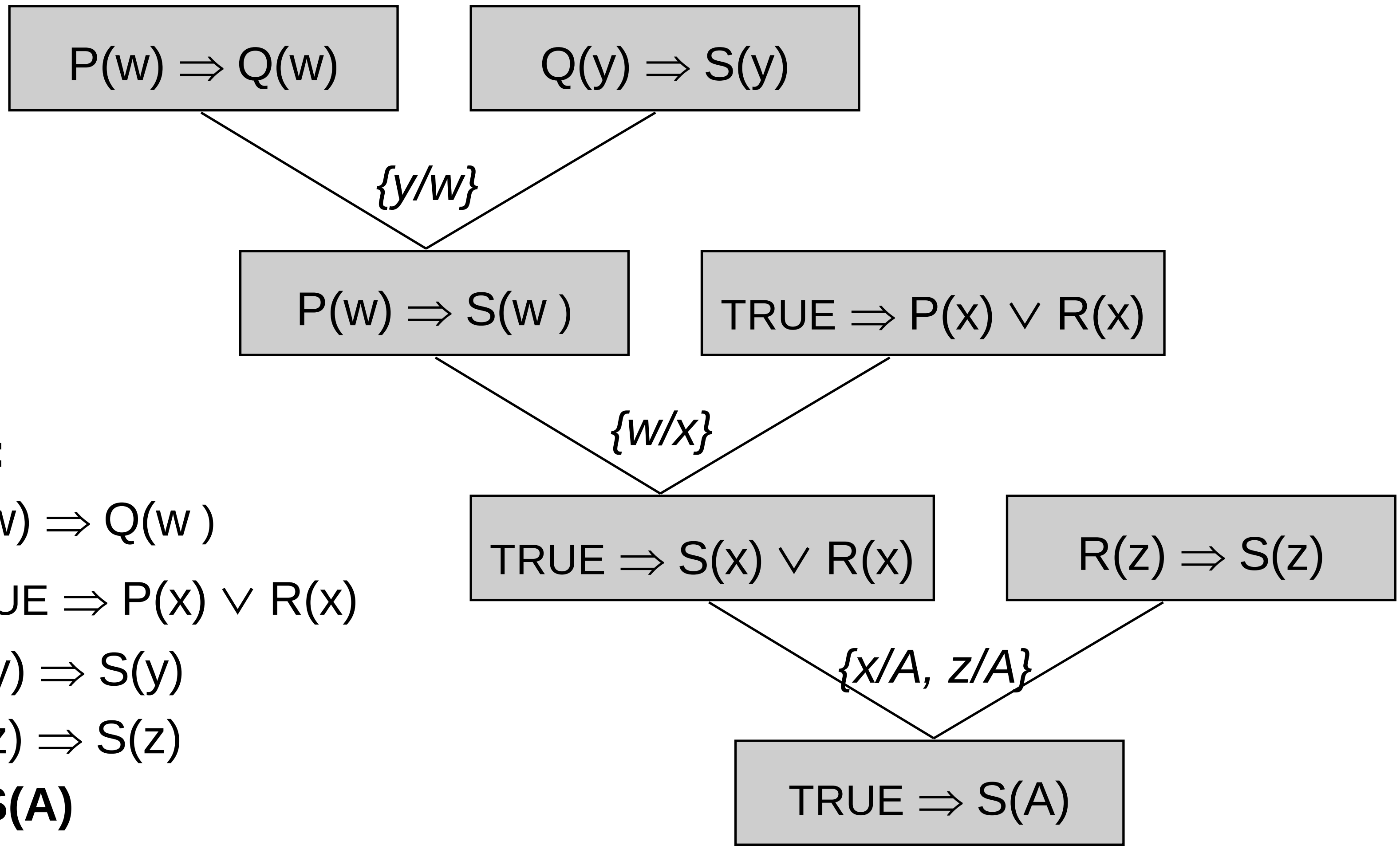
¬Q(y) ∨ S(y)

¬R(z) ∨ S(z)

? S(A)

2ANIM

Example of Resolution Proof (INF)



KB:

$P(w) \Rightarrow Q(w)$

$TRUE \Rightarrow P(x) \vee R(x)$

$Q(y) \Rightarrow S(y)$

$R(z) \Rightarrow S(z)$

? S(A)

Refutation

- **Resolution is incomplete**
  – There exist entailed sentences that it cannot prove
    - e.g. Empty KB (!), KB |– $P \vee \neg P$, but resolution cannot prove it

- **Resolution by refutation**
  – Proof by contradiction (reductio ad absurdum):
  To prove P true, assume it false and prove a contradiction
  i.e.

$$(KB \wedge \neg P \Rightarrow FALSE) \quad \Leftrightarrow \quad (KB \Rightarrow P)$$

  – Simple, sound, complete
    - e.g. assume $\neg(P \vee \neg P)$, rewrite as $\neg P \wedge P$, infer contradiction.

Another Example Proof

- **Problem statement and query**
  - *"Jack owns a dog. Every dog owner is an animal lover. No animal lover kills an animal. Either Jack or curiosity killed the cat, who is named Tuna."*
  - *Did curiosity kill the cat?*
- **Translation in First Order Logic**
  - 1. $\exists x\ Dog(x) \wedge Owns(Jack,x)$
  - 2. $\forall x\ \forall y\ Owns(x,y) \wedge Dog(y) \Rightarrow AnimalLover(x)$
  - 3. $\forall x\ AnimalLover(x) \Rightarrow (\forall y\ Animal(y) \Rightarrow \neg Kills(x,y))$
  - 4. $Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$
  - 5. $Cat(Tuna)$
  - 6. $\forall x\ Cat(x) \Rightarrow Animal(x)$ [background knowledge]

Variations on Translation to FOL

- **Example**
  - Statement: "*No animal lover kills an animal.*"
  - Translation in FOL:

1. $\forall x\ L(x) \Rightarrow (\forall y\ A(y) \Rightarrow \neg K(x,y))$

   $\forall x\ \neg L(x) \vee (\forall y\ \neg A(y) \vee \neg K(x,y))$

   $\forall x\ \forall y\ \ \neg L(x) \vee \neg A(y) \vee \neg K(x,y)$ **CNF**

2.

$\forall x,y\ \neg(L(x)\ \Lambda\ A(y)) \vee \neg K(x,y)$

$\forall x,y\ \ L(x)\ \Lambda\ A(y) \Rightarrow\ \neg K(x,y)$

3.

$\neg\ (L(x)\ \Lambda\ A(y)\ \Lambda\ K(x,y))$

$L(x)\ \Lambda\ A(y)\ \Lambda\ K(x,y) \Rightarrow F$

INF

Conversion from FOL to CNF

- 1. $\exists x\ Dog(x) \wedge Owns(Jack,x)$
  - $\rightarrow$ 1a. $Dog(D)$
  - $\rightarrow$ 1b. $Owns(Jack,D)$
- 2. $\forall x\ \forall y\ Owns(x,y) \wedge Dog(y) \Rightarrow AnimalLover(x)$
  - $\rightarrow$ 2. $\neg Dog(y) \vee \neg Owns(x,y) \vee AnimalLover(x)$
- 3. $\forall x\ AnimalLover(x) \Rightarrow (\forall y\ Animal(y) \Rightarrow \neg Kills(x,y))$
  - $\rightarrow$ 3. $\neg AnimalLover(u) \vee \neg Animal(v) \vee \neg Kills(u,v)$
- 4. $Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$
  - $\rightarrow$ 4. $Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$
- 5. $Cat(Tuna)$
  - $\rightarrow$ 5. $Cat(Tuna)$
- 6. $\forall x\ Cat(x) \Rightarrow Animal(x)$
  - $\rightarrow$ 6. $\neg Cat(z) \vee Animal(z)$

Query:

? $Kills(Curiosity,Tuna)$

$\neg Kills(Curiosity,Tuna)$

Conversion from FOL to INF

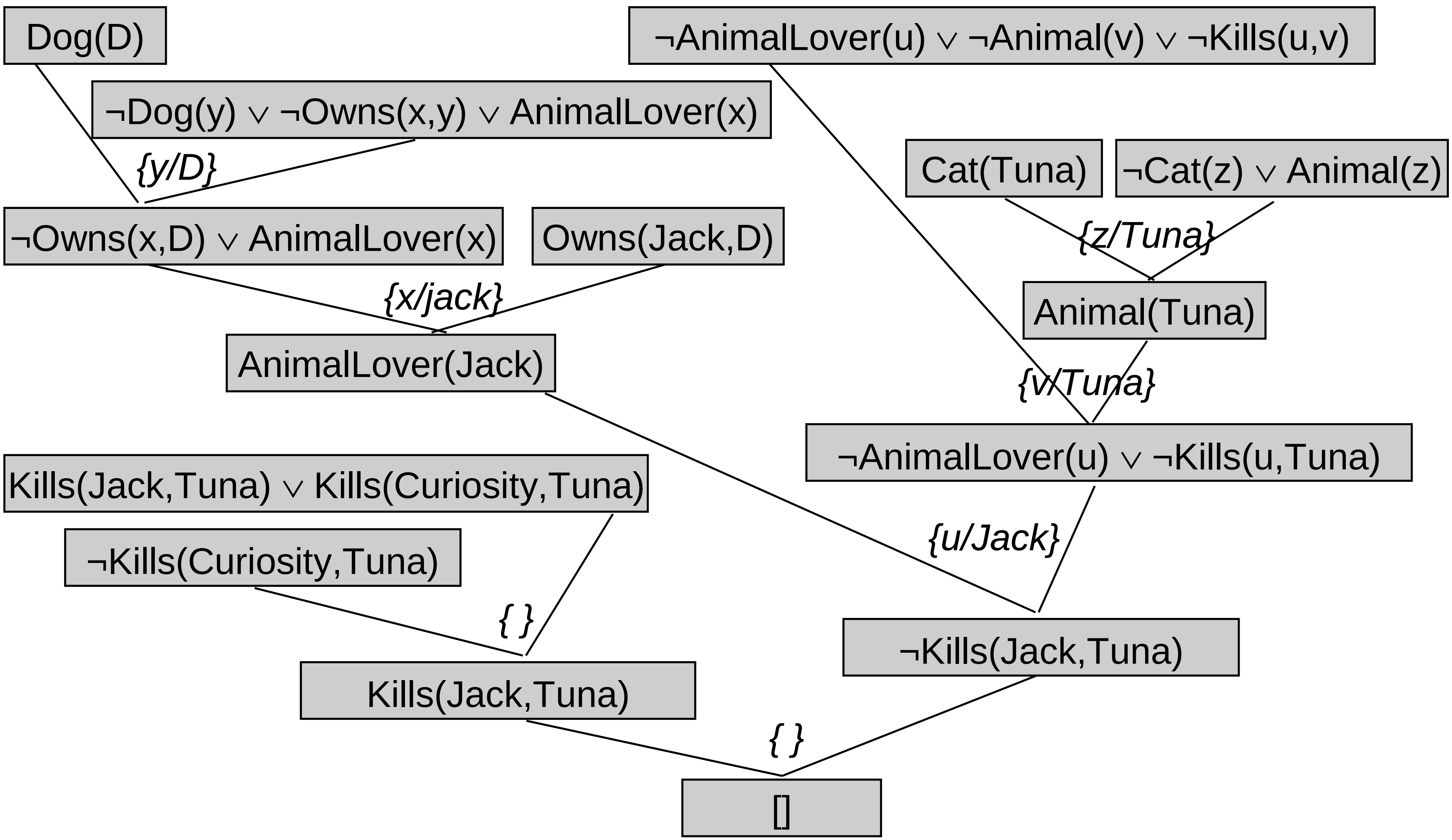
- 1. $\exists x$ Dog(x) Λ Owns(Jack,x)
  - $\rightarrow$ 1a. Dog(D)
  - $\rightarrow$ 1b. Owns(Jack,D)
- 2. $\forall x \forall y$ Owns(x,y) Λ Dog(y) $\Rightarrow$ AnimalLover(x)
  - $\rightarrow$ 2. Dog(y) Λ Owns(x,y) $\Rightarrow$ AnimalLover(x)
- 3. $\forall x$ AnimalLover(x) $\Rightarrow$ ($\forall y$ Animal(y) $\Rightarrow$ ¬Kills(x,y))
  - $\rightarrow$ 3. AnimalLover(u) Λ Animal(v) Λ Kills(u,v) $\Rightarrow$ FALSE
- 4. Kills(Jack,Tuna) $\vee$ Kills(Curiosity,Tuna)
  - $\rightarrow$ 4. Kills(Jack,Tuna) $\vee$ Kills(Curiosity,Tuna)
- 5. Cat(Tuna)
  - $\rightarrow$ 5. Cat(Tuna)
- 6. $\forall x$ Cat(x) $\Rightarrow$ Animal(x)
  - $\rightarrow$ 6. Cat(z) $\Rightarrow$ Animal(z)

Query:

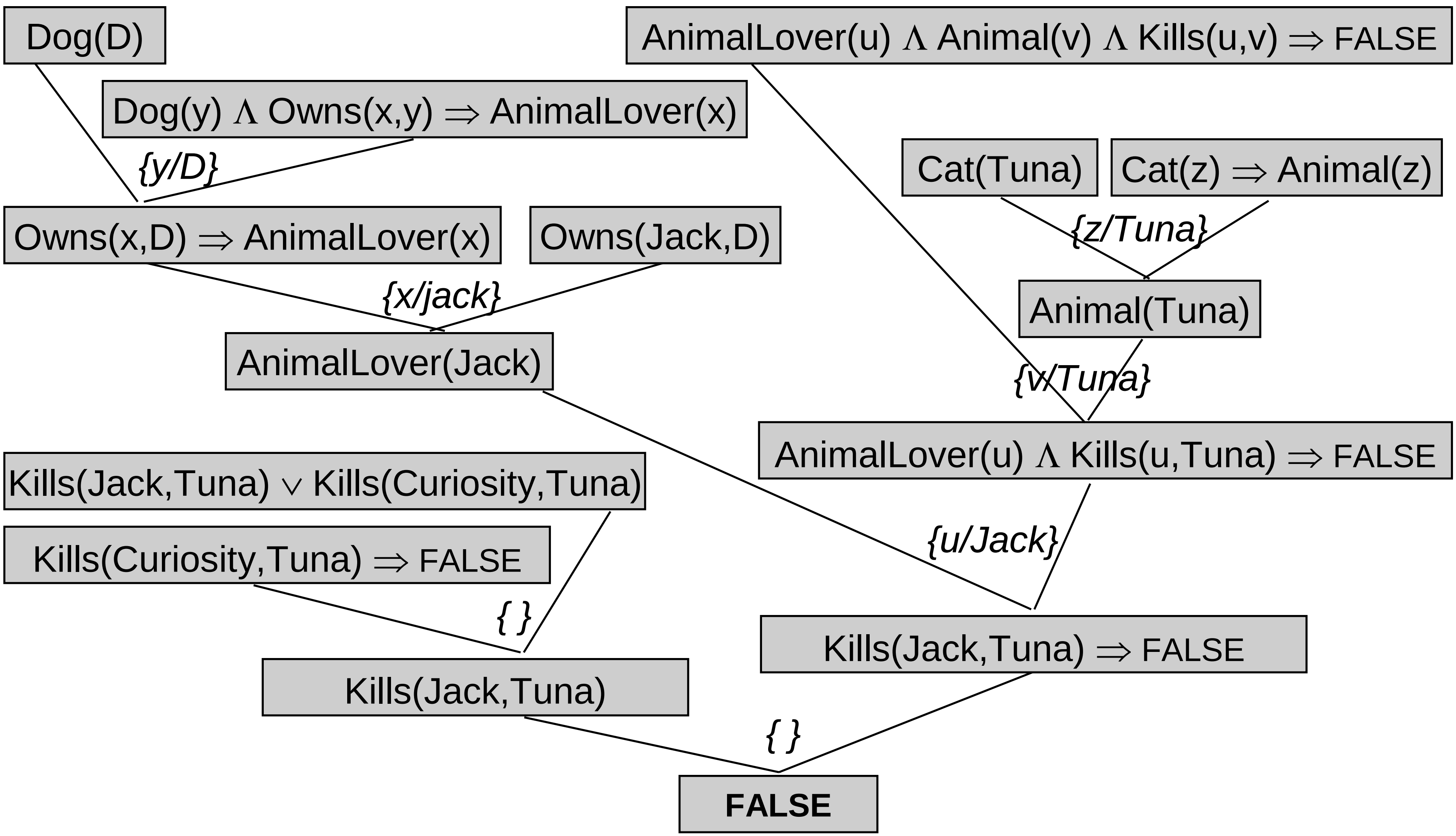
? Kills(Curiosity,Tuna)

Kills(Curiosity,Tuna) $\Rightarrow$ F

Proof by Resolution–Refutation (CNF)



Proof by Resolution–Refutation (INF)

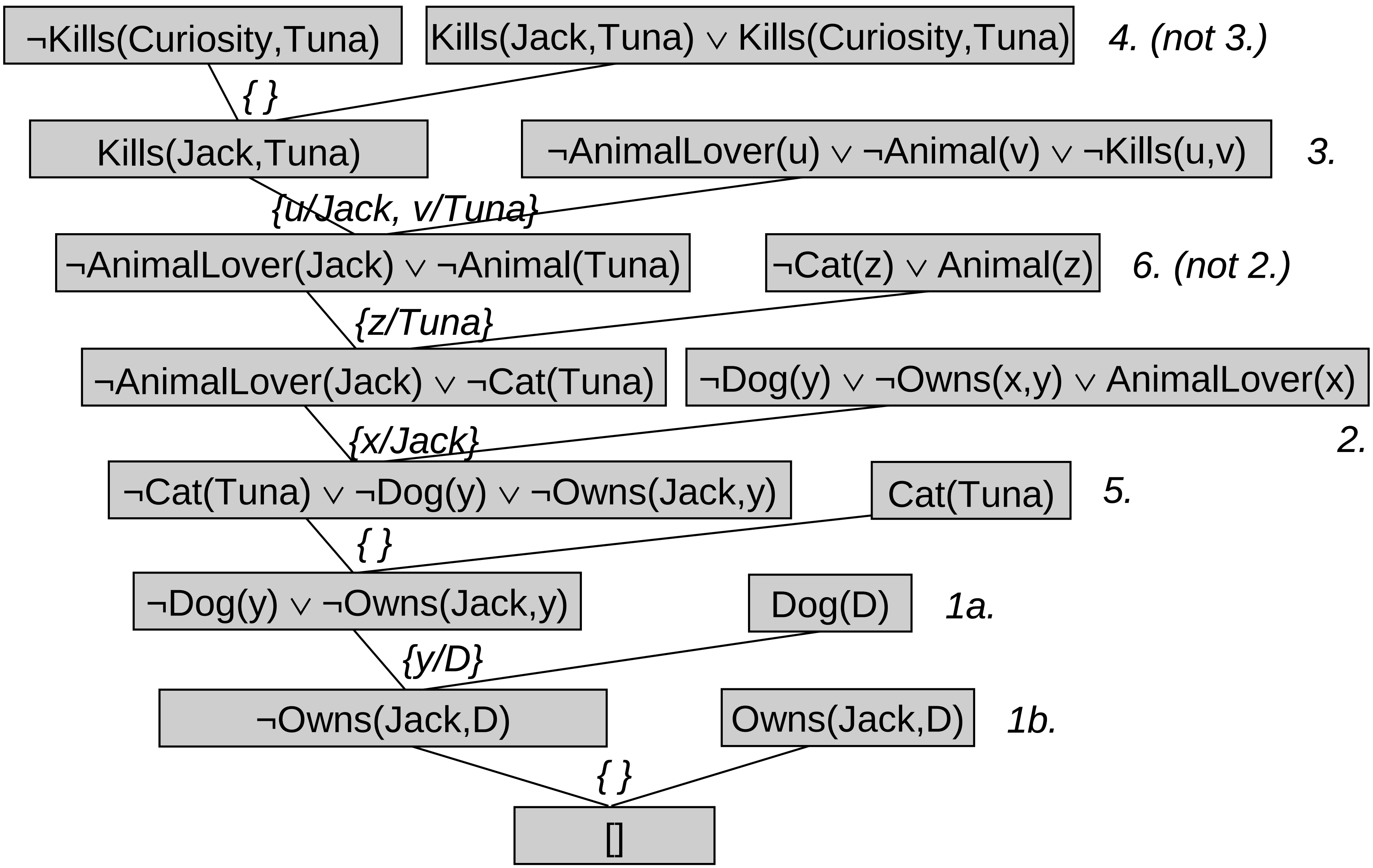


Resolution Strategies

– *Resolution is guaranteed to find a solution, but how?*

- **Unit Preference**
  – Favour short sentences (even <u>unit clauses</u>, if possible)
- **Set of Support**
  – Define a subset of the KB and use those sentences only
    - e.g. { negated query }, as in refutation
- **Input Resolution**
  – Combine KB sentences some inferred sentence
- **Subsumption**
  – Eliminate all sentences subsumed (more specific than) existing sentences in the KB, e.g. if P(x) then P(A) not needed

Proof Using IR and UP Strategies



Theorem Provers

- **Differences with LPL**
  - Use full first-order logic
  - Control kept distinct from knowledge
    - Order of writing does not matter, e.g. $A \Leftarrow B \wedge C$ or $A \Leftarrow C \wedge B$
- **Design of a theorem prover**
  - Design of a control strategy, e.g. OTTER
    - Using a set of support, unit preference, other strategies …
  - Extending Prolog , e.g. PTTP
    - Make search sound: use Occur-Check in unification
    - Make search complete: use IDS instead of depth-first search
    - Make inference complete: using linear input resolution
    - Use locking: store rules different ways
    - Allow negated literals

Using Theorem Provers

- **Practical uses**
  - Assistant, e.g. decision making
  - Proof-checker, supervised by a mathematician
  - Socratic reasoner
    - Provide partial answers, so that a series of “right” questions always leads to the solution
  - Verification of software and hardware
    - e.g. computer algorithms (RSA, Boyer-Moore, etc.), logic design
  - Synthesis of software and hardware
    - i.e. prove “there exists a program p satisfying the specification s”
    - e.g. hand-guided synthesis of new algorithms,
      deductive synthesis in circuit design, large-scale integration

Summary

- **Inference rules for first-order logic …**
  - Are simply extended from propositional logic.
  - Are complex to use, because of a huge branching factor.
- **Unification …**
  - Improves efficiency by identifying appropriate variable substitutions.
- **The Generalised Modus Ponens …**
  - Uses unification to provide a powerful inference rule.
  - Can be either data-driven, using forward-chaining, or goal-oriented, using backward-chaining.

    …

Summary

– Uses sentences in Horn form, i.e. $\alpha_1 \land \ldots \land \alpha_n \Rightarrow \beta$.

– Is not complete.

- **The Generalised Resolution …**

  – Provides a complete system for proof by refutation.

  – Requires sentences in either Conjunctive Normal Form or Implicative Normal Form, i.e. $\alpha_1 \land \ldots \land \alpha_m \Rightarrow \beta_1 \lor \ldots \lor \beta_n$ (which are equivalent).

  – Can use several strategies (heuristics) to improve efficiency and reduce the size of the search space.

References

– Boolos, G. S. and Jeffrey, R. C. (1989). *Computability and Logic*. Cambridge University Press, Cambridge, third edition.

– Genesereth, M. R. and Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, California.

– Hunter, G. (1971). *Metalogic: An Introduction to the Metatheory of Standard First-Order Logic*. University of California Press, Berkeley and Los Angeles.

– Siekmann, J. and Wrightson, G., editors (1983). *Automation of Reasoning*. Springer-Verlag, Berlin. Two volumes.

– Wos, L., Overbeek, R., Lusk, E., and Boyle, J. (1992). *Automated Reasoning: Introduction and Applications*. McGraw-Hill, New York, second edition.

end