Exercise

- **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 a lethal missile, and all of its missiles were sold to it by Colonel West, who is American. Prove that West is a criminal.**

$American(x) \wedge Weapon(y) \wedge Nation(z) \wedge Hostile(z) \wedge Sells(x, z, y) \Rightarrow Criminal(x)$	1
$Owns(Nono, M1)$	2
$Missile(M1)$	3
$Owns(Nono, x) \wedge Missile(x) \Rightarrow Sells(West, Nono, x)$	4
$Missile(x) \Rightarrow Weapon(x)$	5
$Enemy(x, America) \Rightarrow Hostile(x)$	6
$American(West)$	7
$Nation(Nono)$	8
$Enemy(Nono, America)$	9
$Nation(America)$	10

Exercise

- From (3) and (5) using Modus Ponens:
  - *Weapon(M)* (11)
- From (9) and (6) using Modus Ponens:
  - *Hostile(Nono)* (12)
- From (2), (3), and (4) using Modus Ponens:
  - *Sells(West,Nono,Ml)* (13)
- From (7), (11), (8), (12), (13) and (1), using Modus Ponens:
  - *Criminal(West)*

Elaboration of a Reasoning Program: Backward Chaining

Backward chaining is designed to find all answers to a question posed to the KB.

Process:

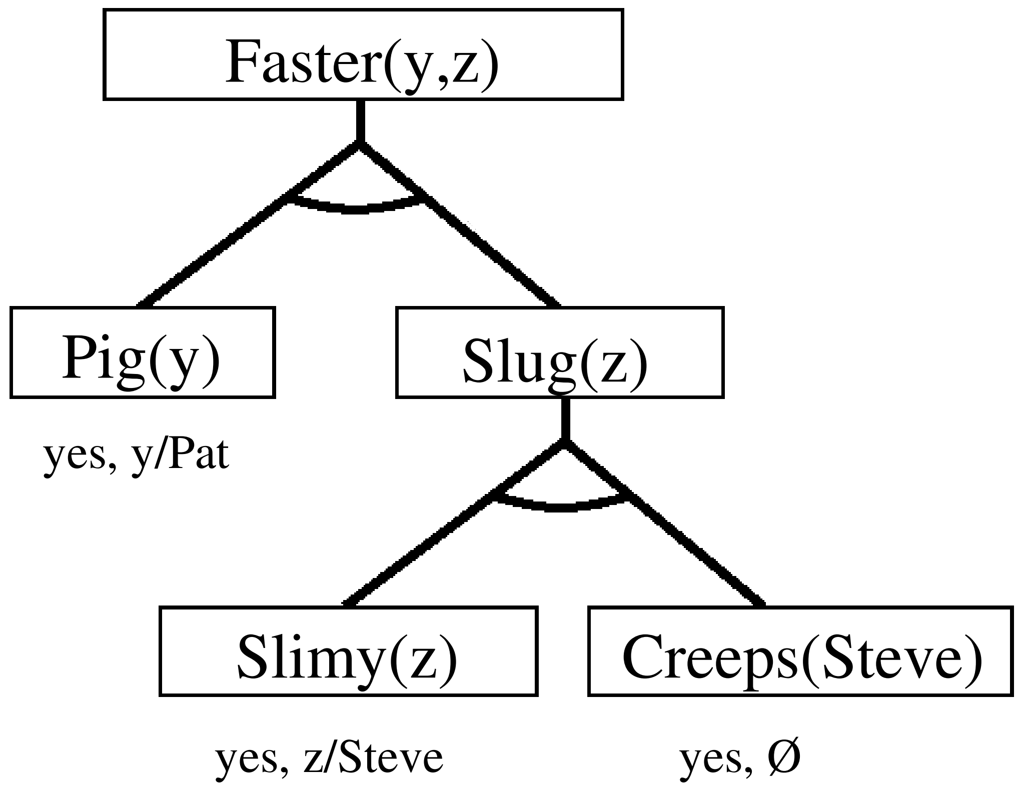
1. Checks to see if answers can be provided directly from sentences in the KB.
2. Finds all implications whose conclusion unifies with the query, and tries to establish the premises of those implications, also by backward chaining.
3. If the premise is a conjunction, then the algorithm processes the conjunction conjunct by conjunct, building up unification for the whole premise as it goes.

Backward Chaining Example

1. $Pig(y) \wedge Slug(z) \;\Rightarrow\; Faster(y, z)$
2. $Slimy(z) \wedge Creeps(z) \;\Rightarrow\; Slug(z)$
3. $Pig(Pat)$ 4. $Slimy(Steve)$ 5. $Creeps(Steve)$

Notes:

1. Tree is read depth-first, left to right
2. Once one branch of a conjunction succeeds, its substitution is applied to subsequent branches



Completeness in FOL

Example

$$
\begin{aligned}
PhD(x) &\Rightarrow HighlyQualified(x)\\
\neg PhD(x) &\Rightarrow EarlyEarnings(x)\\
HighlyQualified(x) &\Rightarrow Rich(x)\\
EarlyEarnings(x) &\Rightarrow Rich(x)
\end{aligned}
$$

Should be able to infer Rich(Me) but backward chaining will not do it

Does a complete algorithm exist?

Validity and Satisfiability

A sentence is satisfiable if it is true in some model
e.g., $A \vee B$

A sentence is unsatisfiable if it is true in no models
e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:
$KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable
(Proof by contradiction)

Resolution Inference Rule

Resolution is a <u>refutation</u> procedure:
to prove $KB \models \alpha$, show that $KB \land \neg\alpha$ is unsatisfiable

Resolution uses KB, $\neg\alpha$ in CNF (conjunction of clauses)

$$\frac{\begin{array}{c} p_1 \lor \ldots\ p_j\ \ldots \lor p_m, \\ q_1 \lor \ldots\ q_k\ \ldots \lor q_n \end{array}}{(p_1 \lor \ldots\ p_{j-1} \lor p_{j+1}\ \ldots p_m \lor q_1 \ldots\ q_{k-1} \lor q_{k+1}\ \ldots \lor q_n)\sigma}$$

where $p_j\sigma = \neg q_k\sigma$

For example,

$$\frac{\begin{array}{l} \neg Rich(x) \lor Unhappy(x) \\ Rich(Me) \end{array}}{Unhappy(Me)}$$

with $\sigma = \{x/Me\}$

Conjunctive Normal Form

Literal = (possibly negated) atomic sentence, e.g., $\neg Rich(Me)$

Clause = disjunction of literals, e.g., $\neg Rich(Me) \vee Unhappy(Me)$

The KB is a conjunction of clauses

Any FOL KB can be converted to CNF as follows:

1. Replace $P \Rightarrow Q$ by $\neg P \vee Q$
2. Move $\neg$ inwards, e.g., $\neg \forall x\, P$ becomes $\exists x\, \neg P$
3. Standardize variables apart, e.g., $\forall x\, P \vee \exists x\, Q$ becomes $\forall x\, P \vee \exists y\, Q$
4. Move quantifiers left in order, e.g., $\forall x\, P \vee \exists x\, Q$ becomes $\forall x \exists y\, P \vee Q$
5. Eliminate $\exists$ by Skolemization (next slide)
6. Drop universal quantifiers
7. Distribute $\wedge$ over $\vee$, e.g., $(P \wedge Q) \vee R$ becomes $(R \vee Q) \wedge (P \vee R)$

Skolemization

$\exists x\, Rich(x)$ becomes $Rich(G1)$ where $G1$ is a new "Skolem constant"

More tricky when $\exists$ is inside $\forall$

E.g., "Everyone has a heart"

$\forall x \;\; Person(x) \;\Rightarrow\; \exists y \;\; Heart(y) \wedge Has(x, y)$

Incorrect:

$\forall x \;\; Person(x) \;\Rightarrow\; Heart(H1) \wedge Has(x, H1)$

Correct:

$\forall x \;\; Person(x) \;\Rightarrow\; Heart(H(x)) \wedge Has(x, H(x))$

where H is a new symbol ("Skolem function")

Skolem function arguments: all enclosing universally quantified variables

Resolution Proof

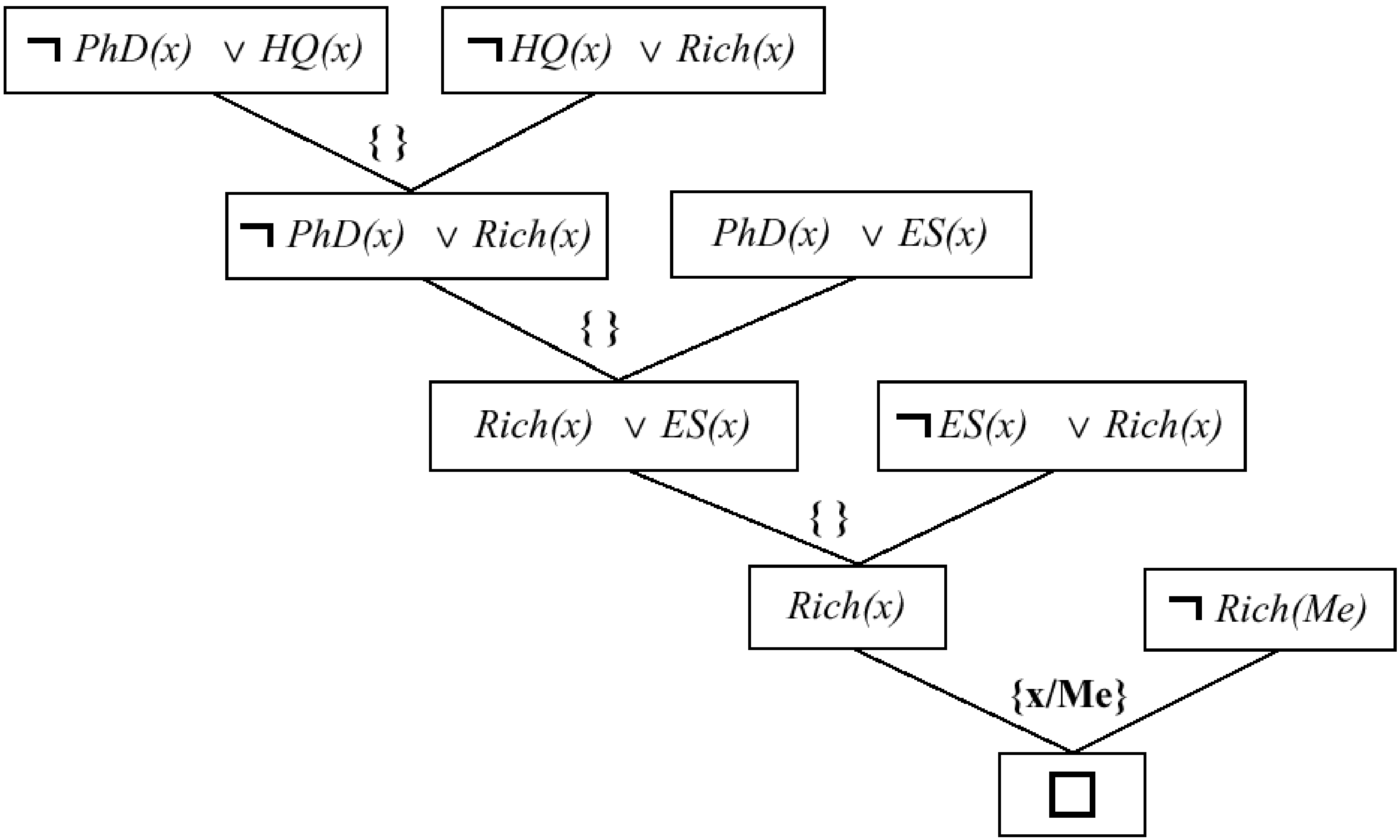
To prove α:

- negate it
- convert to CNF
- add to CNF KB
- infer contradiction

E.g., to prove $Rich(me)$, add $\neg Rich(me)$ to the CNF KB

$$\neg PhD(x) \vee HighlyQualified(x)$$
$$PhD(x) \vee EarlyEarnings(x)$$
$$\neg HighlyQualified(x) \vee Rich(x)$$
$$\neg EarlyEarnings(x) \vee Rich(x)$$

Resolution Proof



Exercise

Translate the following English sentences to First-Order Logic (FOL) using the following predicates: Cat(x), Dog(x), and IsFriendOf(x,y). Convert the knowledgebase to CNF and then use resolution to show that Sam is not a dog.

- Joe is a cat.
- No cat has a dog friend.
- Sam is Joe's friend.

Exercise

English sentences	CNF
Joe is a cat.	Cat(Joe)
No cat has a dog friend	~Cat(x) v ~Dog(y) v ~Friend(x,y)
Sam is Joe's friend	Friend (Joe, Sam)

Resolution works by adding the negated query to the knowledge base and resolving to false. Since we want to prove ~Dog(Sam), we add Dog(Sam) to the knowledge base and then: