CS5010 Artificial Intelligence Principles:
Lecture 9

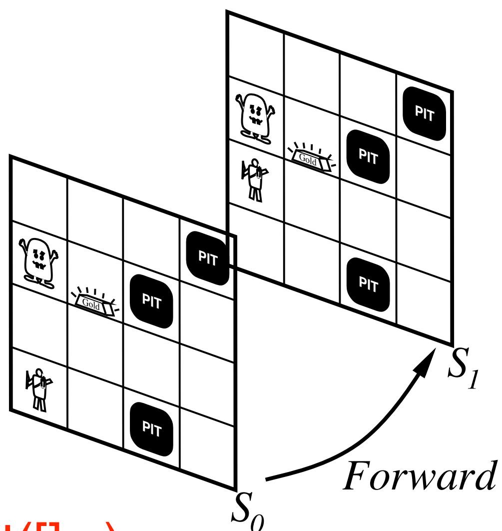
# Keeping track of change

Facts hold in situations, rather than eternally
E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

Situation calculus is one way to represent change in FOL:

Adds a situation argument to each non-eternal predicate
E.g., $Now$ in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function
$Result(a, s)$ is the situation that results from doing $a$ in $s$




forall s: Result([],s) = s
forall s,a: Result([a | seq], s) = Result(seq, Result(a,s))



So a situation is (the result of) a sequence of actions

# Describing actions I

"Effect" axiom—describe changes due to action
$\forall s \;\; AtGold(s) \;\Rightarrow\; Holding(Gold, Result(Grab, s))$

"Frame" axiom—describe **non-changes** due to action
$\forall s \;\; HaveArrow(s) \;\Rightarrow\; HaveArrow(Result(Grab, s))$

Frame problem: find an elegant way to handle non-change
(a) representation—avoid frame axioms
(b) inference—avoid repeated "copy-overs" to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .

Very easy to (a) miss an important frame axiom and/or (b) fail to modify them to remain consistent with current situation

# Describing actions II

Successor-state axioms solve the representational frame problem

Each axiom is "about" a **predicate** (not an action per se):

P true afterwards $\Leftrightarrow$ [an action made P true
$\lor$ P true already and no action made P false]

For holding the gold:

$$\forall\, a, s \;\; Holding(Gold, Result(a, s)) \;\Leftrightarrow$$
$$[(a = Grab \land AtGold(s))$$
$$\lor\, (Holding(Gold, s) \land a \neq Release)]$$

# Making plans

Initial condition in KB:

$At(Agent, [1,1], S_0)$
$At(Gold, [1,2], S_0)$

Query: $Ask(KB, \exists\, s \;\; Holding(Gold, s))$
i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$
i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at $S_0$ and that $S_0$ is the only situation described in the KB

# Making plans: A better way

Recast the question as:
How do I get to a situation in which I'm holding the gold from here?

Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$

$PlanResult(p, s)$ is the result of executing $p$ in $s$

Then the query $Ask(KB, \exists\, p \;\; Holding(Gold, PlanResult(p, S_0)))$
has the solution $\{p/[Forward, Grab]\}$

Definition of $PlanResult$ in terms of $Result$:

$$\forall\, s \;\; PlanResult([\,], s) = s$$
$$\forall\, a, p, s \;\; PlanResult([a|p], s) = PlanResult(p, Result(a, s))$$

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

Planning is NP-hard as a class of problems

Solutions can be possible, feasible, optimal, or best so far

Simple instances can have more possible solutions than the number of atoms in the universe, so search and optimisation are crucial

# Summary

First-order logic:
– objects and relations are semantic primitives
– syntax: constants, functions, predicates, equality, quantifiers

Increased expressive power: sufficient to define wumpus world

Situation calculus:
– conventions for describing actions and change in FOL
– can formulate planning as inference on a situation calculus KB

# Inference in first-order logic

## Chapter 9

# Outline

◇ Reducing first-order inference to propositional inference

◇ Unification

◇ Generalized Modus Ponens

◇ Forward and backward chaining

◇ Logic programming

◇ Resolution

# A brief history of reasoning

| | | |
|---|---|---|
| 450B.C. | Stoics | propositional logic, inference (maybe) |
| 322B.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 |

School of Mohism 479 - 22 BC
Inference
Basic principles, but dynamic rather than axiomatic

# Universal instantiation (UI)

Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall\, v \;\; \alpha}{\textsc{Subst}(\{v/g\}, \alpha)}$$

for any variable $v$ and ground term $g$

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

$$\begin{array}{l} King(John) \land Greedy(John) \;\Rightarrow\; Evil(John) \\ King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard) \\ King(Father(John)) \land Greedy(Father(John)) \;\Rightarrow\; Evil(Father(John)) \\ \quad\vdots \end{array}$$

# Existential instantiation (EI)

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

$$\frac{\exists\, v \;\; \alpha}{\textsc{Subst}(\{v/k\}, \alpha)}$$

E.g., $\exists\, x \;\; Crown(x) \land OnHead(x, John)$ yields

$$Crown(C_1) \land OnHead(C_1, John)$$

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

Another example: from $\exists\, x \;\; d(x^y)/dy = x^y$ we obtain

$$d(e^y)/dy = e^y$$

provided $e$ is a new constant symbol

Skolem constant is a specific value for something known to exist in general terms

Must be new to avoid confusion, but we have an endless supply of new symbols.
Named after Thoraf Skolem

# 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 iff the old KB was satisfiable

So a proof in the new one forms a proof in the old one

And failure to prove in the new one means failure to prove in the new one

# Reduction to propositional inference

Suppose the KB contains just the following:

$$\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

Instantiating the universal sentence in **all possible** ways, we have

$$King(John) \land Greedy(John) \;\Rightarrow\; Evil(John)$$
$$King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

The new KB is propositionalized: proposition symbols are

$King(John),\; Greedy(John),\; Evil(John), King(Richard)$ etc.

# Reduction contd.

A term with no variables is a ground term

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., $Father(Father(Father(John)))$

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

If something is true, we eventually prove it.
If it's false, we might prove it false, but might loop forever

# Problems with propositionalization

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

$$\forall\, x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$\forall\, y \;\; Greedy(y)$$
$$Brother(Richard, John)$$

it seems obvious that $Evil(John)$, but propositionalization produces lots of facts such as $Greedy(Richard)$ that are irrelevant

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

With function symbols, it gets nuch much worse!

# 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

$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | |
| $Knows(John, x)$ | $Knows(y, OJ)$ | |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | |
| $Knows(John, x)$ | $Knows(x, OJ)$ | |

# 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

$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | |
| $Knows(John, x)$ | $Knows(x, OJ)$ | |

# 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

$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | |
| $Knows(John, x)$ | $Knows(x, OJ)$ | |

# 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

$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, OJ)$ | |

Knows(x,OJ) means “Everyone knows OJ”

So we should be able to infer that John knows OJ, but there is a problem

# 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

$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

x can't refer to John and OJ at the same time

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, OJ)$ | $fail$ |

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

Now {x/OJ, z17/John} works

# Generalized Modus Ponens (GMP)

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

where $p_i{}'\theta = p_i\theta$ for all $i$

$p_1{}'$ is $King(John)$ $\qquad p_1$ is $King(x)$
$p_2{}'$ is $Greedy(y)$ $\qquad p_2$ is $Greedy(x)$
$\theta$ is $\{x/John, y/John\}$ $\quad q$ is $Evil(x)$
$q\theta$ is $Evil(John)$

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

This is MP for propositional logic in Horn form (see L07) but lifted to FOL.

The idea is to only make the substitutions needed to make our particular inferences to proceed

# 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

1. We can treat the whole premise as a definite clause, then move the theta inside the brackets

2. Everything in the premise is a definite clause, so we apply UI one at a time

3. All the implicit foralls have been removed, so just apply propositional MP

# Example knowledge base

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:

$$American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)$$

Nono . . . has some missiles

# Example knowledge base contd.

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

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono . . . has some missiles, i.e., $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ and $Missile(M_1)$

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

Here Mi is a Skolem constant use to eliminate the exists.

If there exists at least one x, then we can assume it is M1.

Recall that the KB is now changed, but proof attempts are not affected

# Example knowledge base contd.

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

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono . . . has some missiles, i.e., $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ and $Missile(M_1)$

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

$\forall\, x\ \ Missile(x) \wedge Owns(Nono, x)\ \Rightarrow\ Sells(West, x, Nono)$

Missiles are weapons:

# Example knowledge base contd.

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

$$American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)$$

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

$Owns(Nono, M_1)$ and $Missile(M_1)$

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

$$\forall\, x\ \ Missile(x) \land Owns(Nono,x)\ \Rightarrow\ Sells(West,x,Nono)$$

Missiles are weapons:

$$Missile(x) \Rightarrow Weapon(x)$$

An enemy of America counts as "hostile":

# Example knowledge base contd.

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

$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono ... has some missiles, i.e., $\exists\, x\ Owns(Nono,x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ and $Missile(M_1)$

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

$\forall\, x\ \ Missile(x) \wedge Owns(Nono,x)\ \Rightarrow\ Sells(West,x,Nono)$

Missiles are weapons:

$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":

$Enemy(x, America)\ \Rightarrow\ Hostile(x)$

West, who is American ...

$American(West)$

The country Nono, an enemy of America ...

$Enemy(Nono, America)$

Each clause in the KB is definite: either atomic, or a conjunction of positive literals implying a single positive literal.

Similar to Horn clauses for PL.

Note that not every KB can be converted into this form

# Forward chaining algorithm

**function** FOL-FC-ASK($KB$, $\alpha$) **returns** a substitution or *false*

**repeat until** *new* is empty
    *new* ← { }
    **for each** sentence $r$ **in** $KB$ **do**
        $(\, p_1 \wedge \ldots \wedge \; p_n \; \Rightarrow \;\; q) \leftarrow$ STANDARDIZE-APART($r$)
        **for each** $\theta$ such that $(p_1 \; \wedge \; \ldots \; \wedge \; p_n)\theta \; = \; (p'_1 \; \wedge \; \ldots \; \wedge \; p'_n)\theta$
                for some $p'_1, \ldots, p'_n$ in $KB$
            $q' \leftarrow$ SUBST($\theta$, $q$)
            **if** $q'$ is not a renaming of a sentence already in $KB$ or *new* **then do**
                add $q'$ to *new*
                $\phi \leftarrow$ UNIFY($q'$, $\alpha$)
                **if** $\phi$ is not *fail* **then return** $\phi$
    add *new* to $KB$
**return** *false*

# Forward chaining proof

These are the sentences from the KB that are not implications

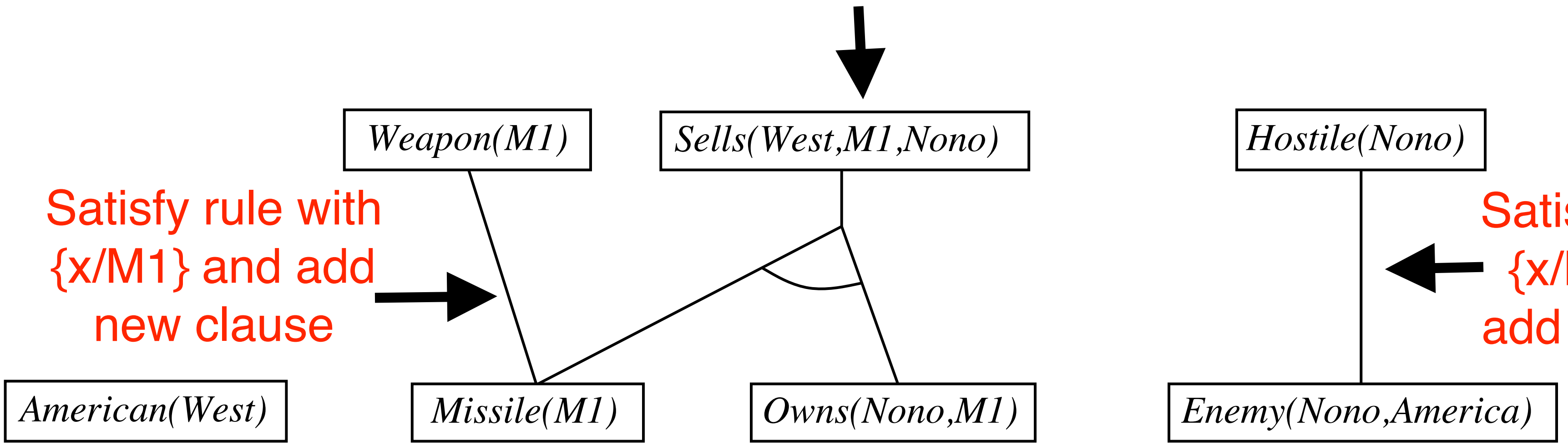
$\boxed{American(West)}$ $\boxed{Missile(M1)}$ $\boxed{Owns(Nono,M1)}$ $\boxed{Enemy(Nono,America)}$

# Forward chaining proof

Satisfy rule with {x/M1} and and add new clause

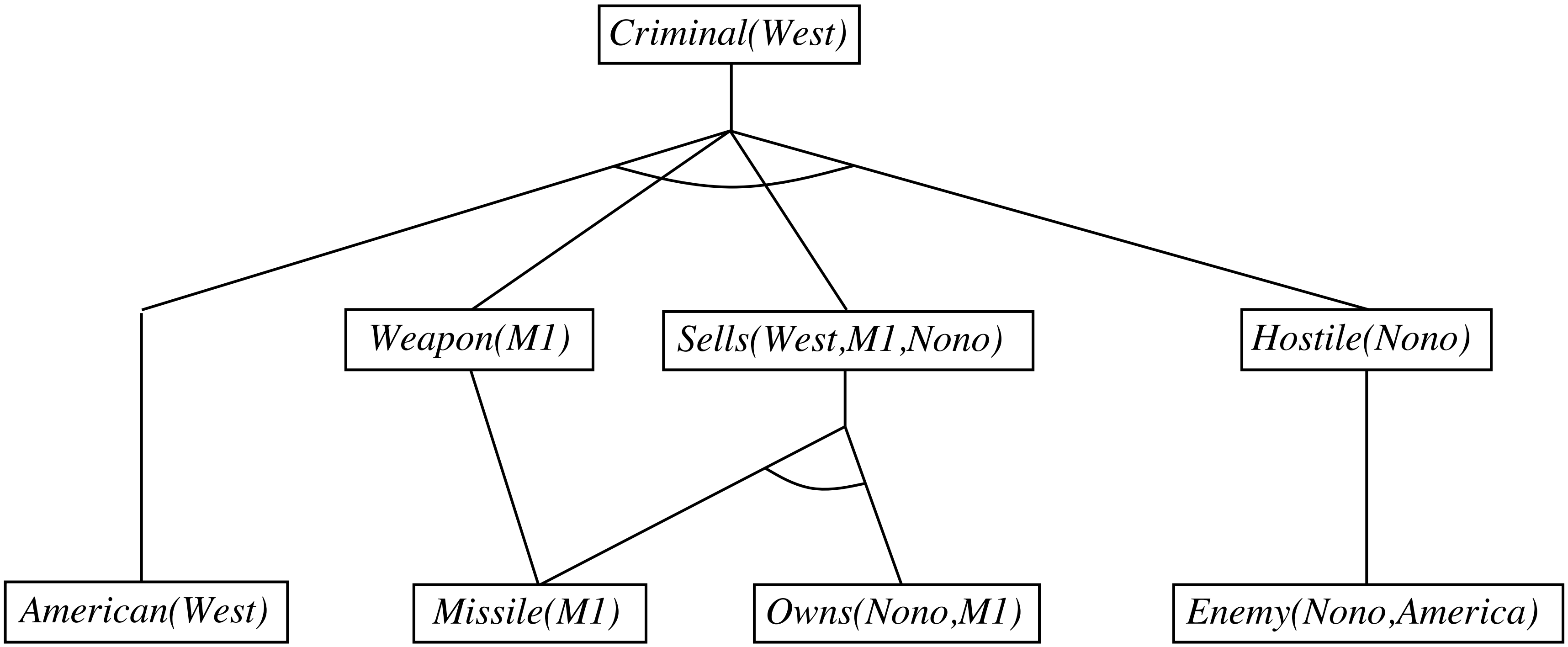
Satisfy rule with {x/M1} and add new clause

Satisfy rule with {x/Nono} and add new clause




Keep track of modified KB using AND-OR graph notation

# Forward chaining proof




On the 2nd pass, we can apply {x/West, y/M1, z/Nono} and add the goal clause

The tree is now a proof tree, showing that the goal is entailed by the KB once suitable substitutions are made

# Properties of forward chaining

Sound and complete for first-order definite clauses
(proof similar to propositional proof)

Datalog = first-order definite clauses + **no functions** (e.g., crime KB)
FC terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals

May not terminate in general if $\alpha$ is not entailed

This is unavoidable: entailment with definite clauses is semidecidable

This version of FC is inefficient:

All possible unifiers are searched for at each stage.
Each rule is rechecked at every iteration.
Facts are generated that are not important w.r.t. the goal

# Efficiency of forward chaining

Simple observation: no need to match a rule on iteration $k$
if a premise wasn't added on iteration $k-1$

$\Rightarrow$ match each rule whose premise contains a newly added literal

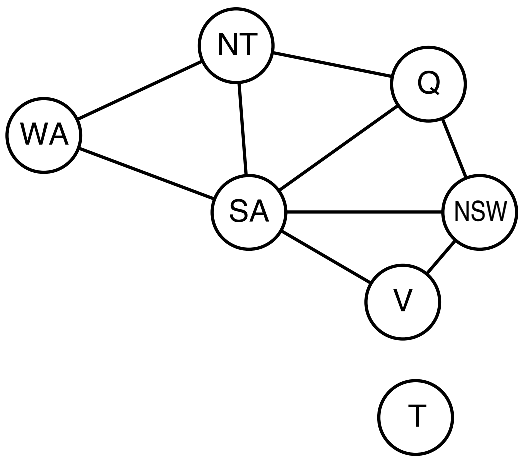
Matching itself can be expensive

Database indexing allows $O(1)$ retrieval of known facts
e.g., query $Missile(x)$ retrieves $Missile(M_1)$

Matching conjunctive premises against known facts is NP-hard

Forward chaining is widely used in deductive databases

# Hard matching example




$$Diff(wa, nt) \wedge Diff(wa, sa) \wedge$$
$$Diff(nt, q) Diff(nt, sa) \wedge$$
$$Diff(q, nsw) \wedge Diff(q, sa) \wedge$$
$$Diff(nsw, v) \wedge Diff(nsw, sa) \wedge$$
$$Diff(v, sa) \Rightarrow Colorable()$$

$Diff(Red, Blue)$ $Diff(Red, Green)$
$Diff(Green, Red)$ $Diff(Green, Blue)$
$Diff(Blue, Red)$ $Diff(Blue, Green)$

$Colorable()$ is inferred iff the CSP has a solution
CSPs include 3SAT as a special case, hence matching is NP-hard

3-colouring can be formulated as a Constraint Satisfaction Problem
3-colouring can also be formulated as FC matching with definite clauses

CSPs are as hard as 3-SAT, which are as hard as SAT
So FC matching is NP-hard