Lecture 8: Knowledge 2

Previously...

```
**function** HYBRID-WUMPUS-AGENT($percept$) **returns** an $action$
  **inputs**: $percept$, a list, [$stench$,$breeze$,$glitter$,$bump$,$scream$]
  **persistent**: $KB$, a knowledge base, initially the atemporal “wumpus physics”
                  $t$, a counter, initially 0, indicating time
                  $plan$, an action sequence, initially empty

  TELL($KB$, MAKE-PERCEPT-SENTENCE($percept$, $t$))
  TELL the $KB$ the temporal “physics” sentences for time $t$
  $safe \leftarrow \{[x, y] : \text{ASK}(KB, OK^t_{x,y}) = true\}$
  **if** ASK($KB$, $Glitter^t$) $= true$ **then**
    $plan \leftarrow [Grab]$ + PLAN-ROUTE($current$, {[1,1]}, $safe$) + [$Climb$]
  **if** $plan$ is empty **then**
    $unvisited \leftarrow \{[x, y] : \text{ASK}(KB, L^{t'}_{x,y}) = false$ for all $t' \leq t\}$
    $plan \leftarrow$ PLAN-ROUTE($current$, $unvisited \cap safe$, $safe$)
  **if** $plan$ is empty and ASK($KB$, $HaveArrow^t$) $= true$ **then**
    $possible\_wumpus \leftarrow \{[x, y] : \text{ASK}(KB, \neg W_{x,y}) = false\}$
    $plan \leftarrow$ PLAN-SHOT($current$, $possible\_wumpus$, $safe$)
  **if** $plan$ is empty **then**   // no choice but to take a risk
    $not\_unsafe \leftarrow \{[x, y] : \text{ASK}(KB, \neg OK^t_{x,y}) = false\}$
    $plan \leftarrow$ PLAN-ROUTE($current$, $unvisited \cap not\_unsafe$, $safe$)
  **if** $plan$ is empty **then**
    $plan \leftarrow$ PLAN-ROUTE($current$, {[1, 1]}, $safe$) + [$Climb$]
  $action \leftarrow$ POP($plan$)
  TELL($KB$, MAKE-ACTION-SENTENCE($action$, $t$))
  $t \leftarrow t + 1$
  **return** $action$
```

```
**function** PLAN-ROUTE($current$,$goals$,$allowed$) **returns** an action sequence
  **inputs**: $current$, the agent’s current position
              $goals$, a set of squares; try to plan a route to one of them
              $allowed$, a set of squares that can form part of the route

  $problem \leftarrow$ ROUTE-PROBLEM($current$, $goals$,$allowed$)
  **return** A*-GRAPH-SEARCH($problem$)
```

Pros and cons of propositional logic

- 😊 Propositional logic is **declarative**: pieces of syntax correspond to facts
- 😊 Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
- 😊 Propositional logic is **compositional**: meaning of $B_{1,1} \land P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- 😊 Meaning in propositional logic is **context-independent** (unlike natural language, where meaning depends on context)
- ☹ Propositional logic has very limited expressive power (unlike natural language)
  E.g., cannot say "pits cause breezes in adjacent squares"
  except by writing one sentence for each square

First-order logic

Whereas propositional logic assumes world contains **facts**,
first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
- Relations: red, round, bogus, prime, multistoried . . .,
  brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
- Functions: father of, best friend, third inning of, one more than, end of . . .

Logics in general

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

Syntax of FOL: Basic elements

Constants	$KingJohn,\ 2,\ UCB, \ldots$
Predicates	$Brother,\ >, \ldots$
Functions	$Sqrt,\ LeftLegOf, \ldots$
Variables	$x,\ y,\ a,\ b, \ldots$
Connectives	$\wedge\ \ \vee\ \ \neg\ \ \Rightarrow\ \ \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall\ \exists$

Atomic sentences

Atomic sentence $=$ $predicate(term_1, \ldots, term_n)$
or $term_1 = term_2$

Term $=$ $function(term_1, \ldots, term_n)$
or $constant$ or $variable$

E.g., $Brother(KingJohn, RichardTheLionheart)$
$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \;\Rightarrow\; S_2, \quad S_1 \;\Leftrightarrow\; S_2$$

E.g. $Sibling(KingJohn, Richard) \;\Rightarrow\; Sibling(Richard, KingJohn)$

$>(1,2) \lor \leq(1,2)$

$>(1,2) \land \neg{>}(1,2)$

Truth in first-order logic

Sentences are true with respect to a model and an interpretation

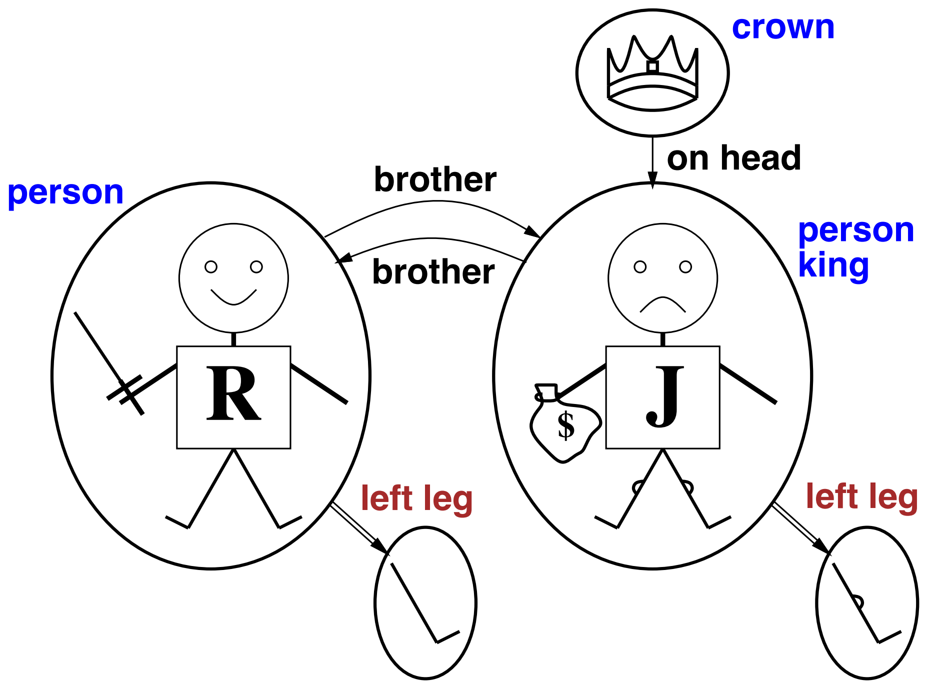
Model contains ≥ 1 objects (domain elements) and relations among them

Interpretation specifies referents for

- constant symbols $\rightarrow$ objects
- predicate symbols $\rightarrow$ relations
- function symbols $\rightarrow$ functional relations

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true
iff the objects referred to by $term_1, \ldots, term_n$
are in the relation referred to by $predicate$

Models for FOL: Example



Consider the interpretation in which
$Richard$ → Richard the Lionheart
$John$ → the evil King John
$Brother$ → the brotherhood relation

Under this interpretation, $Brother(Richard, John)$ is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Models for FOL: Lots!

Entailment in propositional logic can be computed by enumerating models

We **can** enumerate the FOL models for a given KB vocabulary:

```
For each number of domain elements $n$ from $1$ to $\infty$
    For each $k$-ary predicate $P_k$ in the vocabulary
        For each possible $k$-ary relation on $n$ objects
            For each constant symbol $C$ in the vocabulary
                For each choice of referent for $C$ from $n$ objects . . .
```

Computing entailment by enumerating FOL models is not easy!

Universal quantification

$\forall \langle variables \rangle \ \ \langle sentence \rangle$

Everyone at Berkeley is smart:
$\forall x \ \ At(x, Berkeley) \ \Rightarrow \ Smart(x)$

$\forall x \ \ P$ is true in a model m iff P is true with x being **each** possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of P

$$
\begin{aligned}
& (At(KingJohn, Berkeley) \ \Rightarrow \ Smart(KingJohn)) \\
\land \ & (At(Richard, Berkeley) \ \Rightarrow \ Smart(Richard)) \\
\land \ & (At(Berkeley, Berkeley) \ \Rightarrow \ Smart(Berkeley)) \\
\land \ & \ldots
\end{aligned}
$$

A common mistake to avoid

Typically, $\Rightarrow$ is the main connective with $\forall$

Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall\, x \;\; At(x, Berkeley) \wedge Smart(x)$$

means "Everyone is at Berkeley and everyone is smart"

Existential quantification

$\exists \langle variables \rangle \; \langle sentence \rangle$

Someone at Stanford is smart:
$\exists x \;\; At(x, Stanford) \land Smart(x)$

$\exists x \;\; P$ is true in a model m iff P is true with x being **some** possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of P

$$\begin{aligned}
& (At(KingJohn, Stanford) \land Smart(KingJohn)) \\
\lor\; & (At(Richard, Stanford) \land Smart(Richard)) \\
\lor\; & (At(Stanford, Stanford) \land Smart(Stanford)) \\
\lor\; & \ldots
\end{aligned}$$

Another common mistake to avoid

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists\, x \;\; At(x, Stanford) \;\Rightarrow\; Smart(x)$$

is true if there is anyone who is not at Stanford!

Properties of quantifiers

$\forall x\ \ \forall y$ is the same as $\forall y\ \ \forall x$ (why??)

$\exists x\ \ \exists y$ is the same as $\exists y\ \ \exists x$ (why??)

$\exists x\ \ \forall y$ is **not** the same as $\forall y\ \ \exists x$

$\exists x\ \ \forall y\ \ Loves(x, y)$
"There is a person who loves everyone in the world"

$\forall y\ \ \exists x\ \ Loves(x, y)$
"Everyone in the world is loved by at least one person"

Quantifier duality: each can be expressed using the other

$\forall x\ \ Likes(x, IceCream) \qquad \neg\exists x\ \ \neg Likes(x, IceCream)$

$\exists x\ \ Likes(x, Broccoli) \qquad \neg\forall x\ \ \neg Likes(x, Broccoli)$

Fun with sentences

Brothers are siblings

$\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$.

"Sibling" is symmetric

$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x)$.

One's mother is one's female parent

$\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \wedge Parent(x, y))$.

A first cousin is a child of a parent's sibling

$\forall x, y \;\; FirstCousin(x, y) \;\Leftrightarrow\; \exists\, p, ps \;\; Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$

Equality

$term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

E.g., $1 = 2$ and $\forall\, x \;\; \times(Sqrt(x), Sqrt(x)) = x$ are satisfiable
$2 = 2$ is valid

E.g., definition of (full) $Sibling$ in terms of $Parent$:

$$\forall\, x, y \;\; Sibling(x, y) \;\Leftrightarrow\; [\neg(x = y) \land \exists\, m, f \;\; \neg(m = f) \land Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)]$$

Interacting with FOL KBs

Suppose a wumpus-world agent is using an FOL KB
and perceives a smell and a breeze (but no glitter) at $t = 5$:

$Tell(KB, Percept([Smell, Breeze, None], 5))$
$Ask(KB, \exists a \;\; Action(a, 5))$

I.e., does KB entail any particular actions at $t = 5$?

Answer: $Yes, \;\; \{a/Shoot\}$ $\quad\leftarrow$ substitution (binding list)

Given a sentence S and a substitution σ,
$S\sigma$ denotes the result of plugging σ into S; e.g.,
$S = Smarter(x, y)$
$\sigma = \{x/Hillary, y/Bill\}$
$S\sigma = Smarter(Hillary, Bill)$

$Ask(KB, S)$ returns some/all σ such that $KB \models S\sigma$

Knowledge base for the wumpus world

"Perception"

$\forall\, b, g, t \;\; Percept([Smell, b, g], t) \;\Rightarrow\; Smelt(t)$

$\forall\, s, b, t \;\; Percept([s, b, Glitter], t) \;\Rightarrow\; AtGold(t)$

Reflex: $\forall\, t \;\; AtGold(t) \;\Rightarrow\; Action(Grab, t)$

Reflex with internal state: do we have the gold already?

$\forall\, t \;\; AtGold(t) \wedge \neg Holding(Gold, t) \;\Rightarrow\; Action(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

Deducing hidden properties

Properties of locations:
$\forall x,t \;\; At(Agent,x,t) \land Smelt(t) \;\Rightarrow\; Smelly(x)$
$\forall x,t \;\; At(Agent,x,t) \land Breeze(t) \;\Rightarrow\; Breezy(x)$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect
$$\forall y \;\; Breezy(y) \;\Rightarrow\; \exists x \;\; Pit(x) \land Adjacent(x,y)$$

Causal rule—infer effect from cause
$$\forall x,y \;\; Pit(x) \land Adjacent(x,y) \;\Rightarrow\; Breezy(y)$$

Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

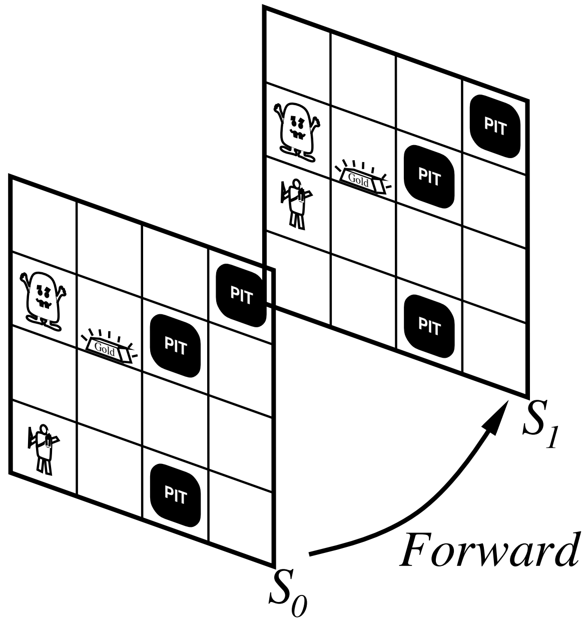
Definition for the $Breezy$ predicate:
$$\forall y \;\; Breezy(y) \;\Leftrightarrow\; [\exists x \;\; Pit(x) \land Adjacent(x,y)]$$

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



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, . . .

Describing actions II

Successor-state axioms solve the representational frame problem

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

$$\begin{array}{lcl} \text{P true afterwards} & \Leftrightarrow & [\text{an action made P true} \\ & \vee & \text{P true already and no action made P false}] \end{array}$$

For holding the gold:

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

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

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

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

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

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

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

Existential instantiation (EI)

For any sentence α, 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

Instantiation

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

Reduction to propositional inference

Suppose the KB contains just the following:

$$\begin{aligned}
&\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x) \\
&King(John) \\
&Greedy(John) \\
&Brother(Richard, John)
\end{aligned}$$

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

$$\begin{aligned}
&King(John) \land Greedy(John) \;\Rightarrow\; Evil(John) \\
&King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard) \\
&King(John) \\
&Greedy(John) \\
&Brother(Richard, John)
\end{aligned}$$

The new KB is propositionalized: proposition symbols are

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

Reduction to propositional inference

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 α 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 α is entailed, loops if α is not entailed

Theorem: Turing (1936), Church (1936), entailment in FOL is semidecidable

Problems with propositionalization

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

$$\begin{aligned}&\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)\\ &King(John)\\ &\forall y \;\; Greedy(y)\\ &Brother(Richard, John)\end{aligned}$$

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 θ 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	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	$fail$

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

Generalized Modus Ponens (GMP)

（前件推理）

$$\frac{p_1{}',\;\; p_2{}',\;\; \ldots,\;\; p_n{}',\;\;\; (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$ where $p_i{}'\theta = p_i\theta$ for all i

$p_1{}'$ is $King(John)$	p_1 is $King(x)$
$p_2{}'$ is $Greedy(y)$	p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$	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

Soundness of GMP

Need to show that

$$p_1', \ \ldots, \ p_n', \ \ (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$$

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. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta)$
2. $p_1', \ \ldots, \ p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

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

... 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 ...

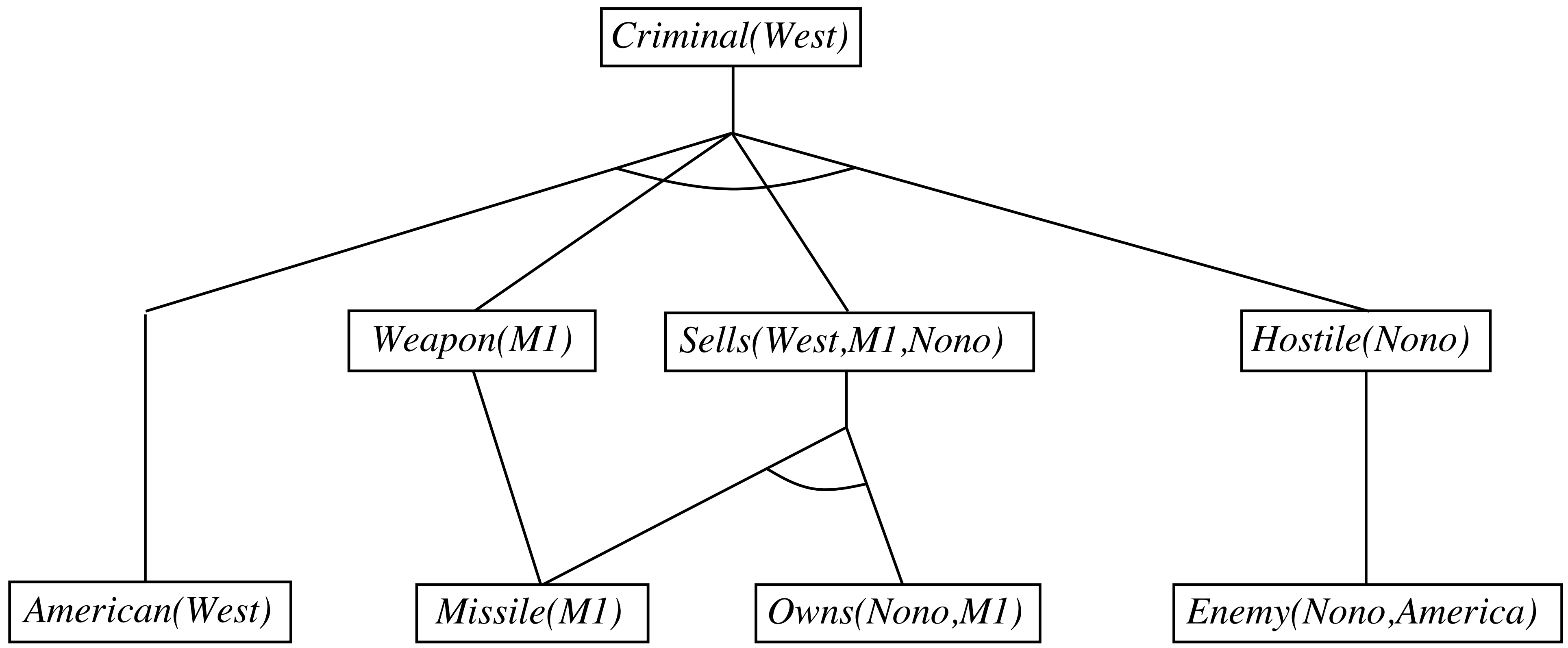
$Enemy(Nono, America)$

Forward chaining algorithm

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

    repeat until $new$ is empty
        $new \leftarrow \{\,\}$
        for each sentence $r$ in $KB$ do
            $(p_1 \land \ldots \land p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART($r$)
            for each $\theta$ such that $(p_1 \land \ldots \land p_n)\theta = (p'_1 \land \ldots \land p'_n)\theta$
                        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



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 α is not entailed

This is unavoidable: entailment with definite clauses is semidecidable

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

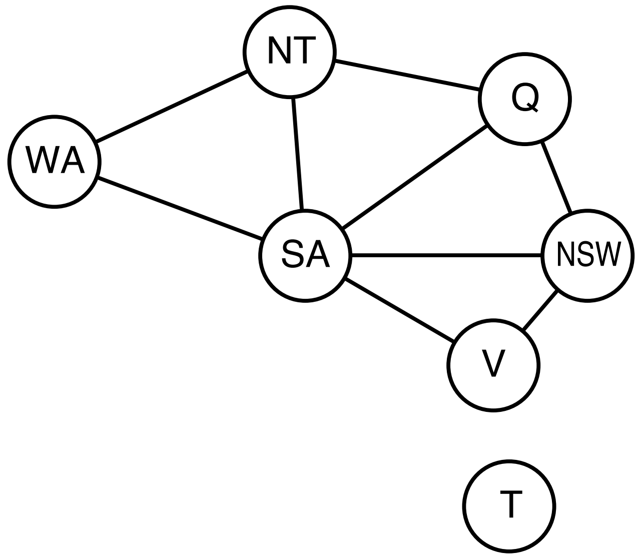
Matching itself can be expensive

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



$$\begin{aligned} &\mathit{Diff}(wa,nt) \land \mathit{Diff}(wa,sa) \land \\ &\quad \mathit{Diff}(nt,q) \mathit{Diff}(nt,sa) \land \\ &\quad \mathit{Diff}(q,nsw) \land \mathit{Diff}(q,sa) \land \\ &\quad \mathit{Diff}(nsw,v) \land \mathit{Diff}(nsw,sa) \land \\ &\quad \mathit{Diff}(v,sa) \;\Rightarrow\; \mathit{Colorable}() \end{aligned}$$

$$\mathit{Diff}(Red,Blue) \quad \mathit{Diff}(Red,Green)$$
$$\mathit{Diff}(Green,Red) \quad \mathit{Diff}(Green,Blue)$$
$$\mathit{Diff}(Blue,Red) \quad \mathit{Diff}(Blue,Green)$$

$\mathit{Colorable}()$ is inferred iff the CSP has a solution

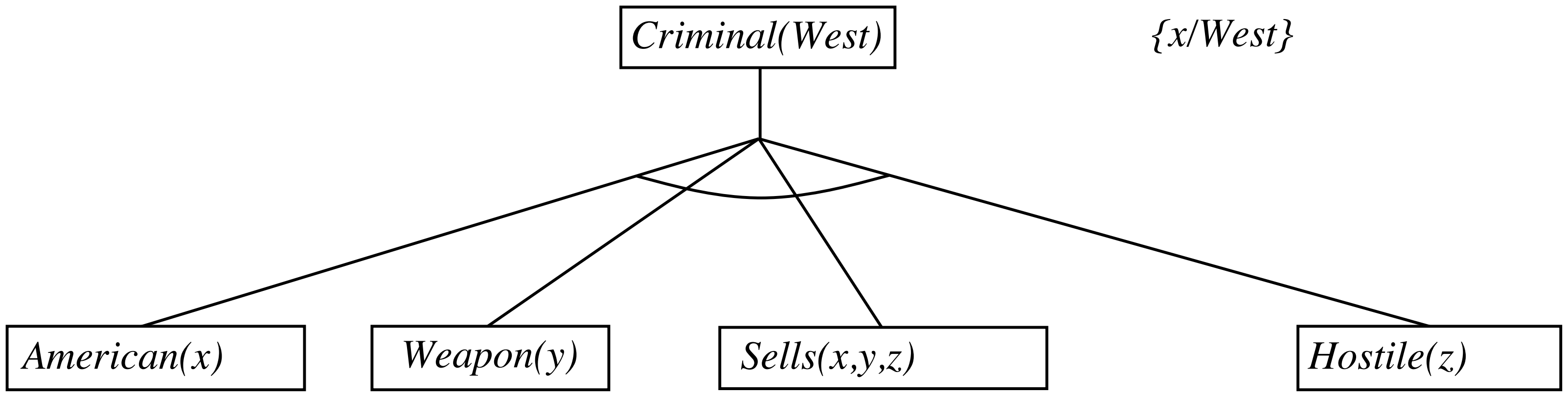
CSPs include 3SAT as a special case, hence matching is NP-hard

Backward chaining algorithm

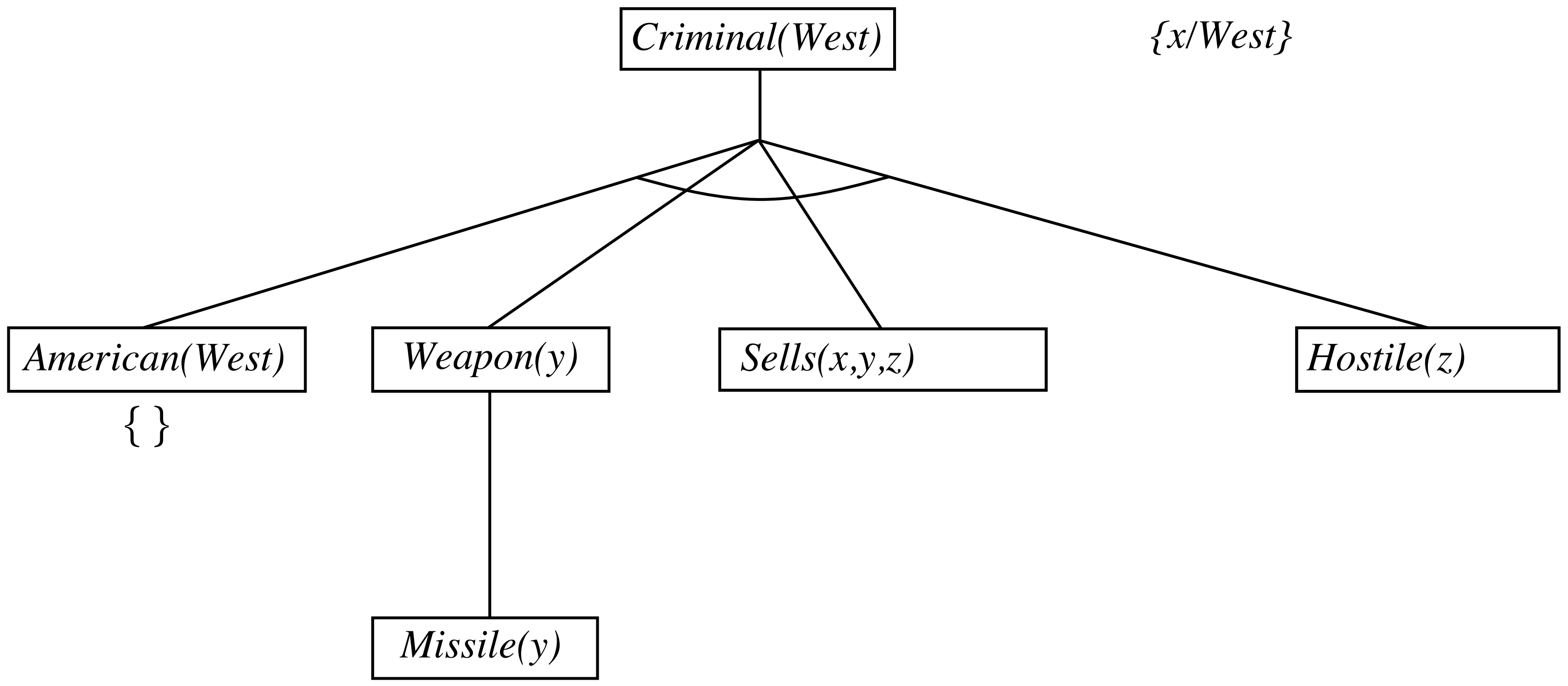
```
function FOL-BC-Ask($KB$, $goals$, $\theta$) returns a set of substitutions
    inputs: $KB$, a knowledge base
            $goals$, a list of conjuncts forming a query ($\theta$ already applied)
            $\theta$, the current substitution, initially the empty substitution $\{\,\}$
    local variables: $answers$, a set of substitutions, initially empty

    if $goals$ is empty then return $\{\theta\}$
    $q' \leftarrow$ Subst($\theta$, First($goals$))
    for each sentence $r$ in $KB$
            where Standardize-Apart($r$) $= (\, p_1 \,\wedge\, \ldots \,\wedge\, p_n \,\Rightarrow\, q)$
            and $\theta' \leftarrow$ Unify($q$, $q'$) succeeds
        $new\_goals \leftarrow [\, p_1, \ldots, p_n |$Rest($goals$)$]$
        $answers \leftarrow$ FOL-BC-Ask($KB$, $new\_goals$, Compose($\theta'$, $\theta$)) $\cup$ $answers$
    return $answers$
```

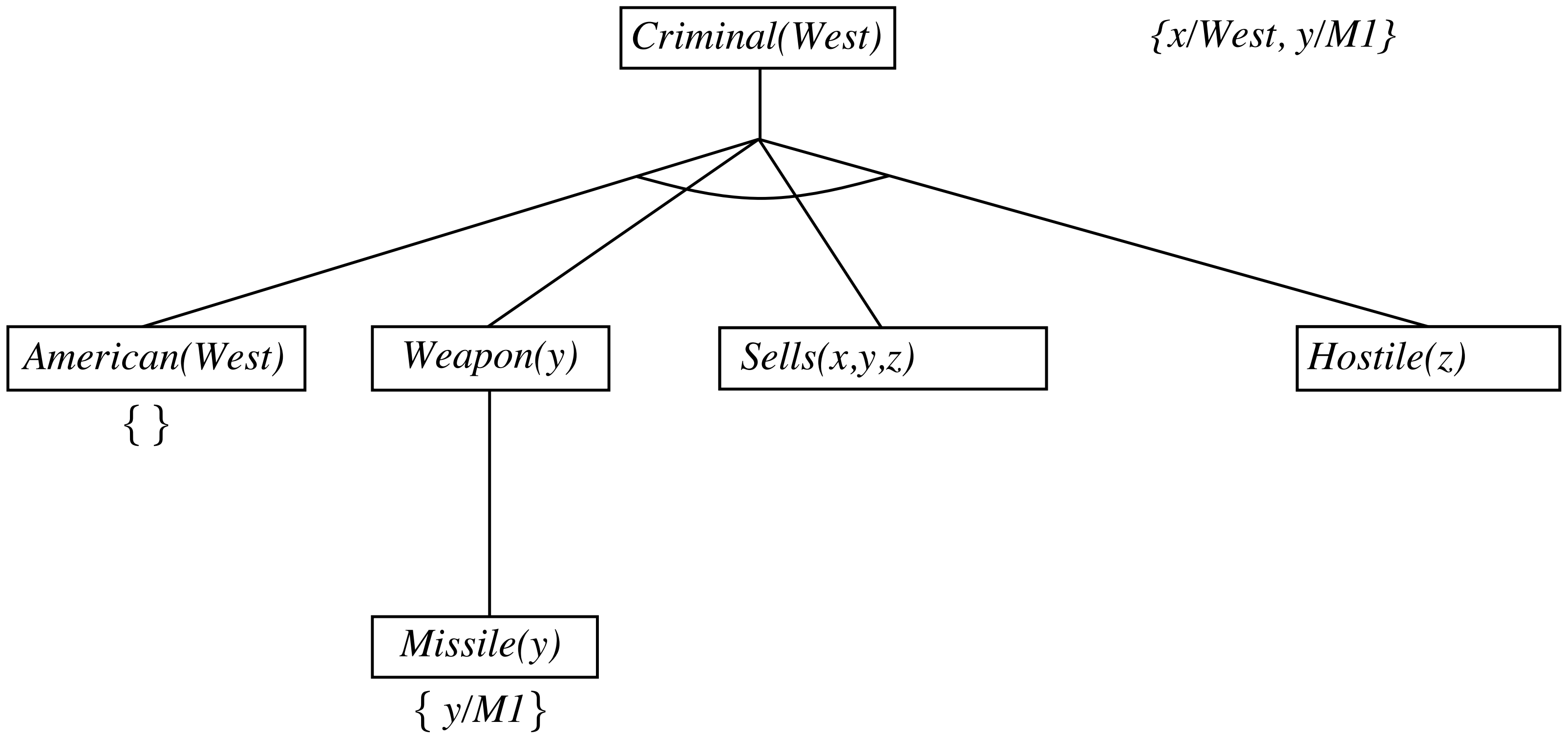
Backward chaining example



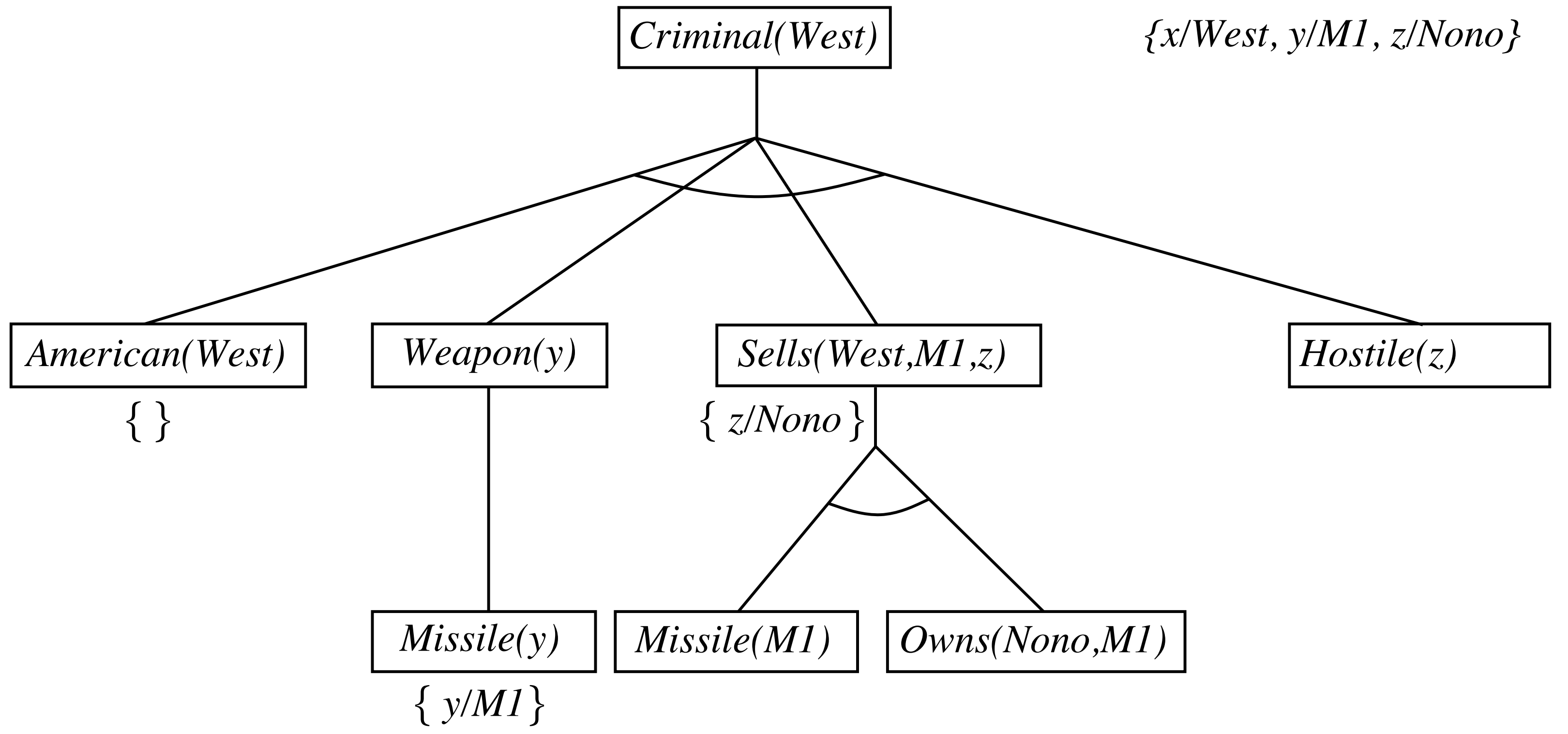
Backward chaining example



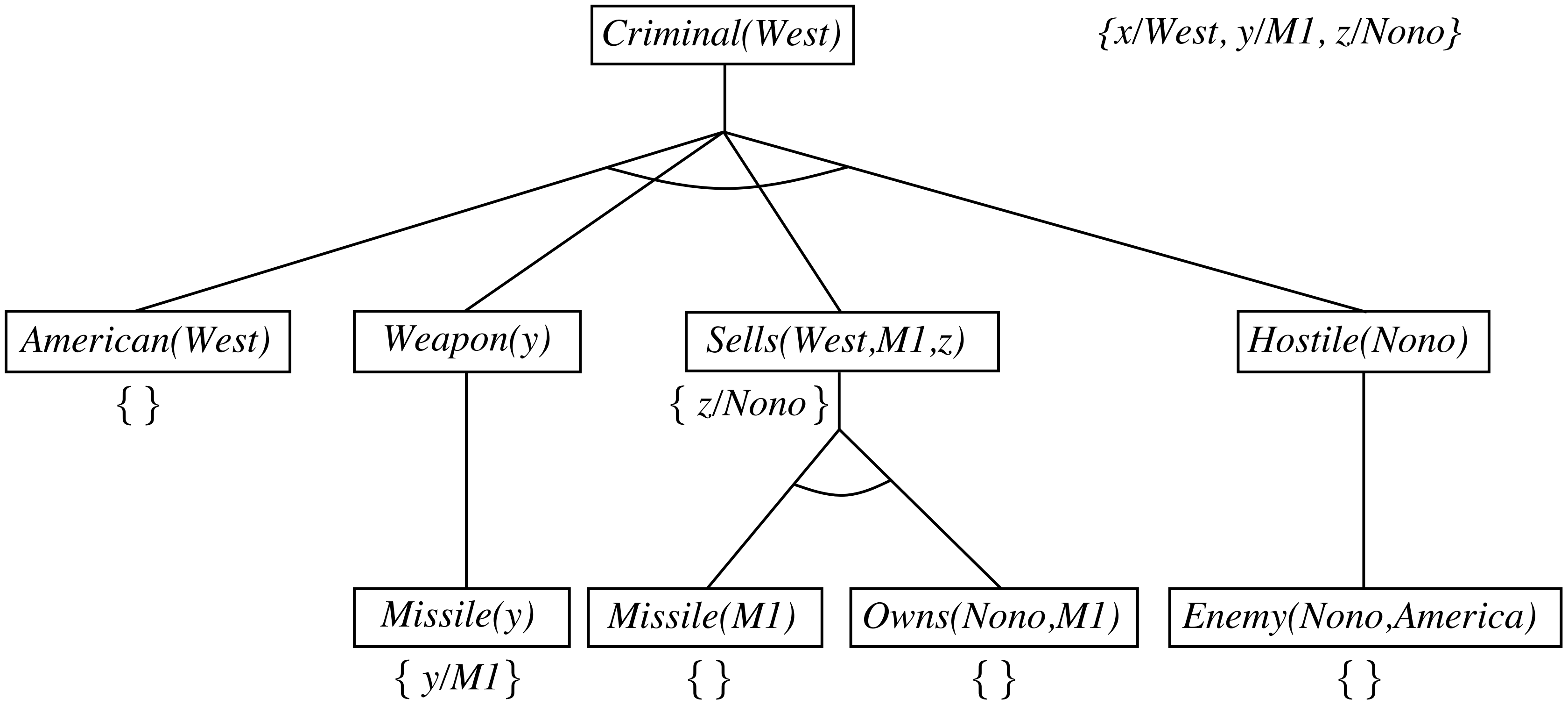
Backward chaining example



Backward chaining example



Backward chaining example



Properties of backward chaining

Depth-first recursive proof search: space is linear in size of proof

Incomplete due to infinite loops

$\Rightarrow$ fix by checking current goal against every goal on stack

Inefficient due to repeated subgoals (both success and failure)

$\Rightarrow$ fix using caching of previous results (extra space!)

Widely used (without improvements!) for logic programming

Logic programming

Sound bite: computation as inference on logical KBs

	Logic programming	Ordinary programming
1.	Identify problem	Identify problem
2.	Assemble information	Assemble information
3.	Tea break	Figure out solution
4.	Encode information in KB	Program solution
5.	Encode problem instance as facts	Encode problem instance as data
6.	Ask queries	Apply program to data
7.	Find false facts	Debug procedural errors

Should be easier to debug $Capital(NewYork, US)$ than $x := x + 2$!

Prolog systems

Basis: backward chaining with Horn clauses + bells & whistles
Widely used in Europe, Japan (basis of 5th Generation project)
Compilation techniques $\Rightarrow$ approaching a billion LIPS

Program = set of clauses = `head :- literal`$_1$`, ... literal`$_n$`.`

```
criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).
```

Efficient unification by open coding
Efficient retrieval of matching clauses by direct linking
Depth-first, left-to-right backward chaining
Built-in predicates for arithmetic etc., e.g., `X is Y*Z+3`
Closed-world assumption ("negation as failure")
  e.g., given `alive(X) :- not dead(X).`
  `alive(joe)` succeeds if `dead(joe)` fails

Prolog examples

Depth-first search from a start state `X`:

```
dfs(X) :- goal(X).
dfs(X) :- successor(X,S),dfs(S).
```

No need to loop over `S`: `successor` succeeds for each

Appending two lists to produce a third:

```
append([],Y,Y).
append([X|L],Y,[X|Z]) :- append(L,Y,Z).

query:   append(A,B,[1,2]) ?
answers: A=[]    B=[1,2]
         A=[1]   B=[2]
         A=[1,2] B=[]
```

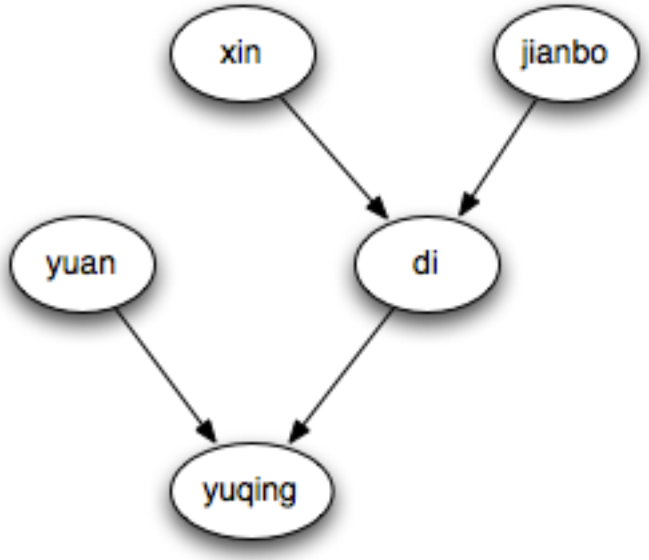
Prolog example

Let's try

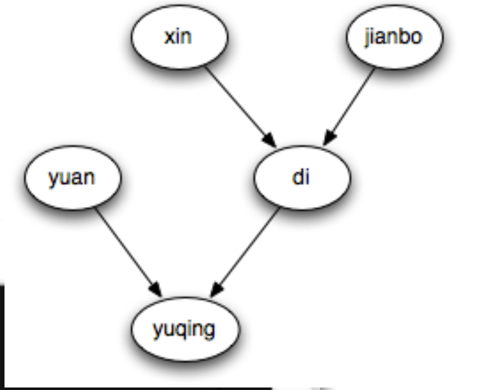
member(1,[1,2,3,4,5])

query: grandfather(X,yuqing)?

```
male(di).
male(jianbo).
female(xin).
female(yuan).
female(yuqing).
father(jianbo,di).
father(di,yuqing).
mother(xin,di).
mother(yuan,yuqing).
grandfather(X,Y):-father(X,Z),father(Z,Y).
grandmother(X,Y):-mother(X,Z),father(Z,Y).
daughter(X,Y):-father(X,Y),female(Y).
```



Prolog example



```
eyounxRMBP15:AI17 yuy$ 
```

Resolution: brief summary

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x) \\ Rich(Ken)\end{array}}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

Apply resolution steps to $CNF(KB \wedge \neg\alpha)$; complete for FOL

Conversion to CNF

Everyone who loves all animals is loved by someone:

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

1. Eliminate biconditionals and implications

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

2. Move $\neg$ inwards: $\neg\forall x, p \ \equiv \exists x\ \neg p$, $\neg\exists x, p \ \equiv \forall x\ \neg p$:

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

Conversion to CNF

3. Standardize variables: each quantifier should use a different one

$$\forall\, x \;\; [\exists\, y \;\; Animal(y) \land \neg Loves(x, y)] \lor [\exists\, z \;\; Loves(z, x)]$$

4. Skolemize: a more general form of existential instantiation.
   Each existential variable is replaced by a Skolem function
   of the enclosing universally quantified variables:

$$\forall\, x \;\; [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

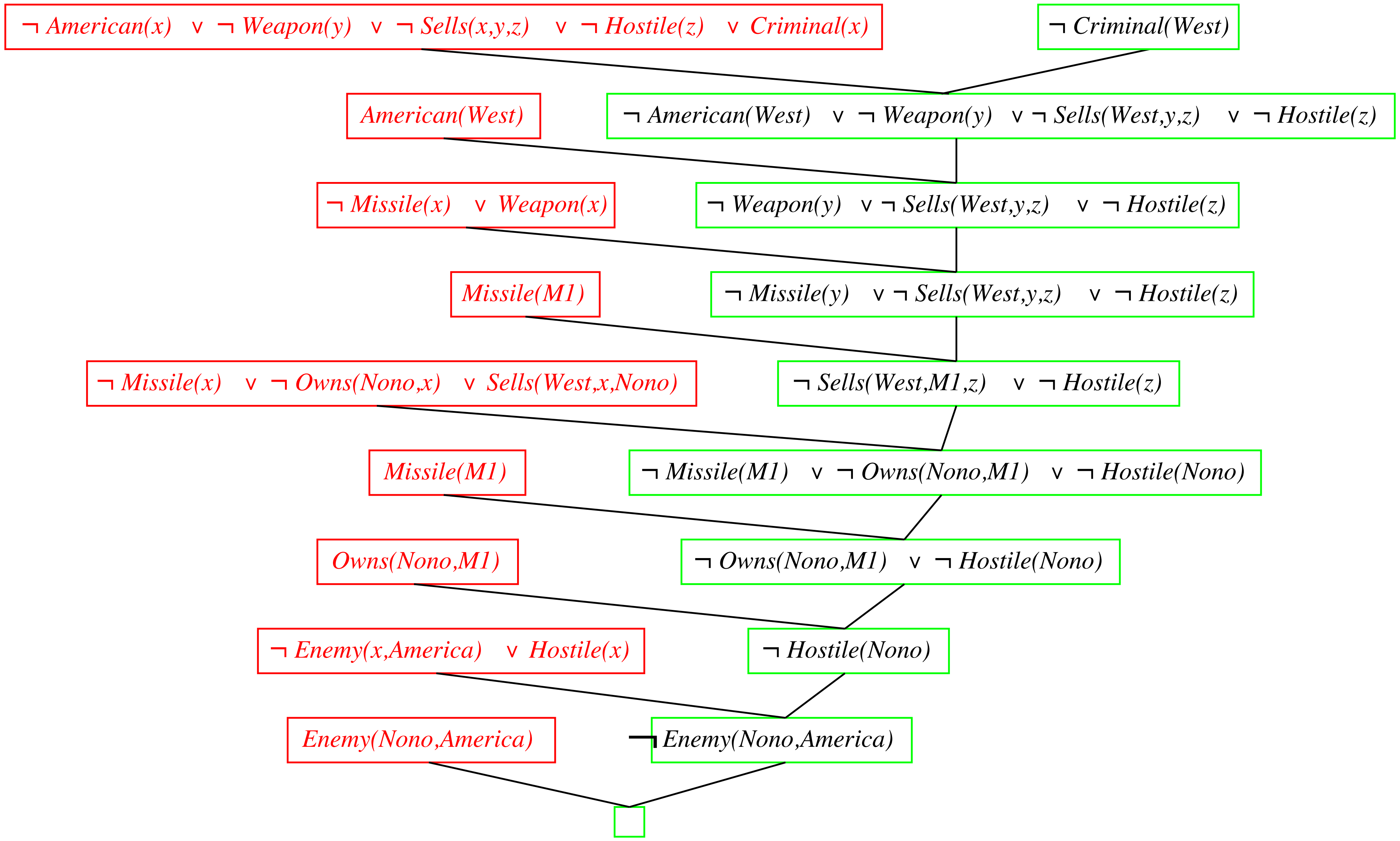
5. Drop universal quantifiers:

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

6. Distribute $\land$ over $\lor$:

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

Resolution proof: definite clauses


¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)
¬ Sells(West,M1,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
Owns(Nono,M1)
¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
¬ Enemy(x,America) ∨ Hostile(x)
¬ Hostile(Nono)
Enemy(Nono,America)
¬ Enemy(Nono,America)

Previously...

Propositional Logic

- PL-Forward chaining
- PL-Backward chaining
- PL-Resolution

First Order Logic (FOL)

- Instantiation
- FOL-Forward chaining
- FOL-Backward chaining
- FOL-Resolution

SAT problems

Propositional logic, CNF

literals: $x_1, x_2, \ldots, x_n$

clauses: $(x_1 \vee x_2 \vee x_5)$ $(\neg x_2 \vee x_3 \vee \neg x_7)$ ...

problem: find an assignment to literals so that the conjunction of the clauses is true, or prove unsatisfiable

$$(x_1 \vee x_2 \vee x_5) \wedge (\neg x_2 \vee x_3 \vee \neg x_7) \wedge \ldots$$

2SAT: every clause has at most 2 literals

P-solvable

3SAT: every clause has at most 3 literals

NP-hard

SAT solvers

SAT problems have many important applications

many SAT solvers are ready for use

- DPLL
- WalkSAT

DPLL

Davis–Putnam–Logemann–Loveland algorithm

```
function DPLL-SATISFIABLE?($s$) returns true or false
   inputs: $s$, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of $s$
   symbols ← a list of the proposition symbols in $s$
   return DPLL(clauses, symbols, { })
--------------------------------------------------------------------------------
function DPLL(clauses, symbols, model) returns true or false

   if every clause in clauses is true in model then return true
   if some clause in clauses is false in model then return false
   $P$, value ← FIND-PURE-SYMBOL(symbols, clauses, model)
   if $P$ is non-null then return DPLL(clauses, symbols – $P$, model $\cup$ {$P$=value})
   $P$, value ← FIND-UNIT-CLAUSE(clauses, model)
   if $P$ is non-null then return DPLL(clauses, symbols – $P$, model $\cup$ {$P$=value})
   $P$ ← FIRST(symbols); rest ← REST(symbols)
   return DPLL(clauses, rest, model $\cup$ {$P$=true}) or
             DPLL(clauses, rest, model $\cup$ {$P$=false}))
```

a deep-first search with heuristics

DPLL heuristics

Pure symbol heuristic: A **pure symbol** is a symbol that always appears with the same "sign" in all clauses.

$$(A \vee \neg B) \wedge (\neg B \vee \neg C) \wedge (C \vee A)$$

A and B is pure, but not C

Unit clause heuristic: A **unit clause** is a clause with just one literal.

$$(A \vee \neg B) \text{ with } A = \text{true}$$

is a unit clause

Other tricks

Component analysis : find disjoint subsets

Variable and value ordering : assign most frequent variable at first

Intelligent backtracking : remember conflicts

Random restart

Clever indexing

WalkSAT

a local search hill-climbing or others.

```
function WALKSAT($clauses, p, max\_flips$) returns a satisfying model or $failure$
   inputs: $clauses$, a set of clauses in propositional logic
           $p$, the probability of choosing to do a "random walk" move, typically around 0.5
           $max\_flips$, number of flips allowed before giving up

   $model \leftarrow$ a random assignment of $true/false$ to the symbols in $clauses$
   for $i = 1$ to $max\_flips$ do
      if $model$ satisfies $clauses$ then return $model$
      $clause \leftarrow$ a randomly selected clause from $clauses$ that is false in $model$
      with probability $p$ flip the value in $model$ of a randomly selected symbol from $clause$
      else flip whichever symbol in $clause$ maximizes the number of satisfied clauses
   return $failure$
```

failure ≠ unsatisfiable

The landscape of random SAT problems

Not all SAT instances are hard
under-constraint: a few clauses => easy to enumerate
over-constraint: too many clauses => unsatisfiable

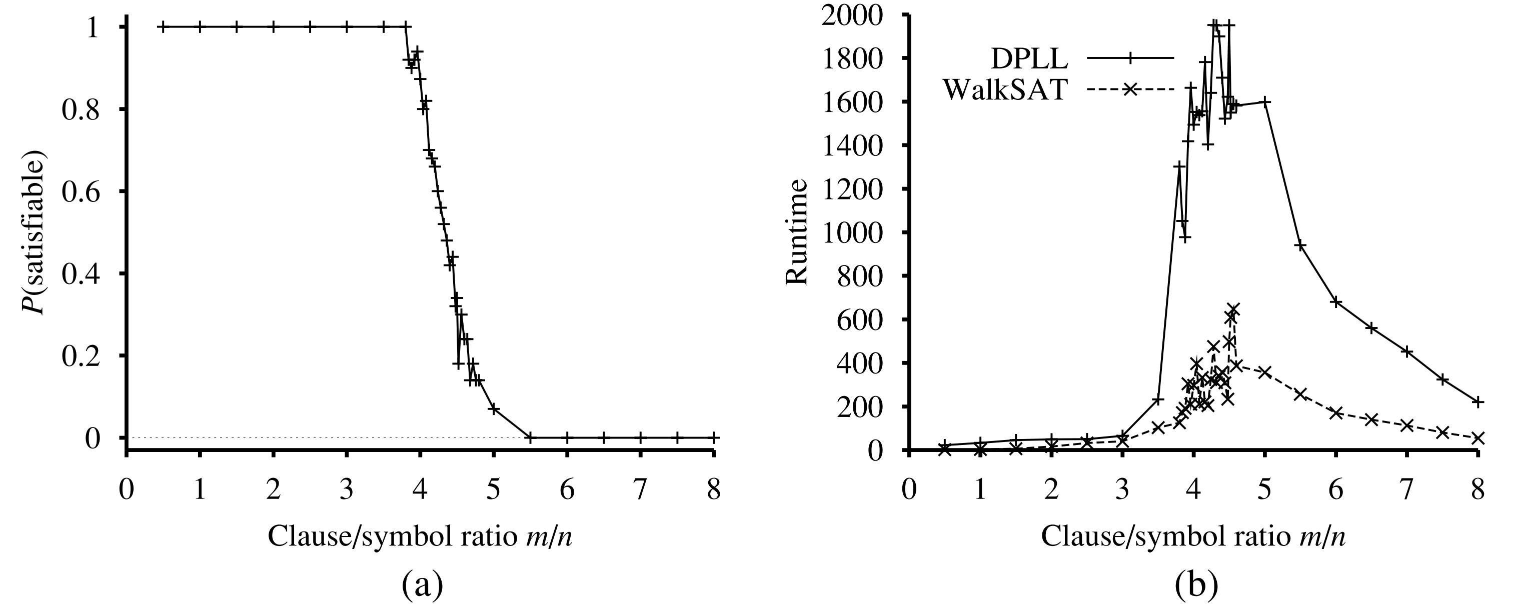


Figure 7.19 (a) Graph showing the probability that a random 3-CNF sentence with $n = 50$ symbols is satisfiable, as a function of the clause/symbol ratio m/n. (b) Graph of the median run time (measured in number of recursive calls to DPLL, a good proxy) on random 3-CNF sentences. The most difficult problems have a clause/symbol ratio of about 4.3.

Planning

Language

There are many languages description the world

Planning Domain Definition Language

1.2, 2.1, 2.2, 3.0, 3.1

state s

Action(s)

Result(s,a)

$$\begin{aligned}&Action(Fly(p, from, to),\\ &\quad \text{PRECOND:} At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)\\ &\quad \text{EFFECT:} \neg At(p, from) \land At(p, to))\end{aligned}$$

$$\begin{aligned}&Action(Fly(P_1, SFO, JFK),\\ &\quad \text{PRECOND:} At(P_1, SFO) \land Plane(P_1) \land Airport(SFO) \land Airport(JFK)\\ &\quad \text{EFFECT:} \neg At(P_1, SFO) \land At(P_1, JFK))\end{aligned}$$

Precondition

action a is **applicable** in state s if the preconditions are satisfied by s

$$(a \in \textsc{Actions}(s)) \;\Leftrightarrow\; s \models \textsc{Precond}(a)$$

$$\begin{aligned}&\forall\, p, \mathit{from}, \mathit{to}\;\; (\mathit{Fly}(p, \mathit{from}, \mathit{to}) \in \textsc{Actions}(s)) \;\Leftrightarrow\\ &\quad s \models (\mathit{At}(p, \mathit{from}) \land \mathit{Plane}(p) \land \mathit{Airport}(\mathit{from}) \land \mathit{Airport}(\mathit{to}))\end{aligned}$$

Result

removing the fluents that appear as negative literals in the action's effects (what we call the **delete list** or DEL(a)), and adding the fluents that are positive literals in the action's effects (what we call the **add list** or ADD(a))

$$\textsc{Result}(s, a) = (s - \textsc{Del}(a)) \cup \textsc{Add}(a) \; .$$

$$\begin{array}{l} Action(Fly(P_1, SFO, JFK), \\ \quad \textsc{Precond:} At(P_1, SFO) \wedge Plane(P_1) \wedge Airport(SFO) \wedge Airport(JFK) \\ \quad \textsc{Effect:} \neg At(P_1, SFO) \wedge At(P_1, JFK)) \end{array}$$

Example

$Init(On(A, Table) \wedge On(B, Table) \wedge On(C, A)$
$\quad \wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C))$
$Goal(On(A, B) \wedge On(B, C))$
$Action(Move(b, x, y),$
$\quad$ PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge Block(y) \wedge$
$\qquad (b \neq x) \wedge (b \neq y) \wedge (x \neq y),$
$\quad$ EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$
$Action(MoveToTable(b, x),$
$\quad$ PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x),$
$\quad$ EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$

Figure 10.3 A planning problem in the blocks world: building a three-block tower. One solution is the sequence $[MoveToTable(C, A), Move(B, Table, C), Move(A, Table, B)]$.

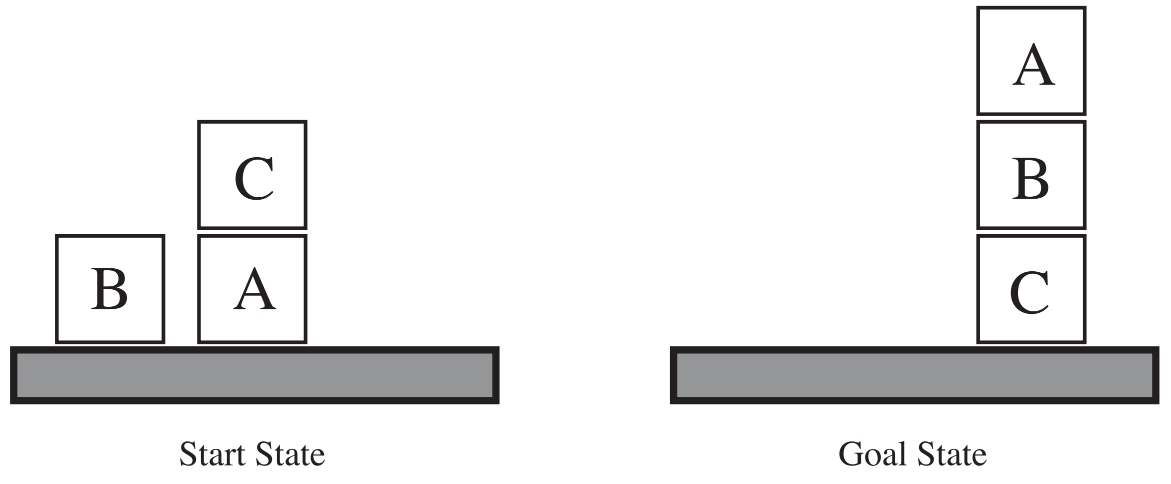
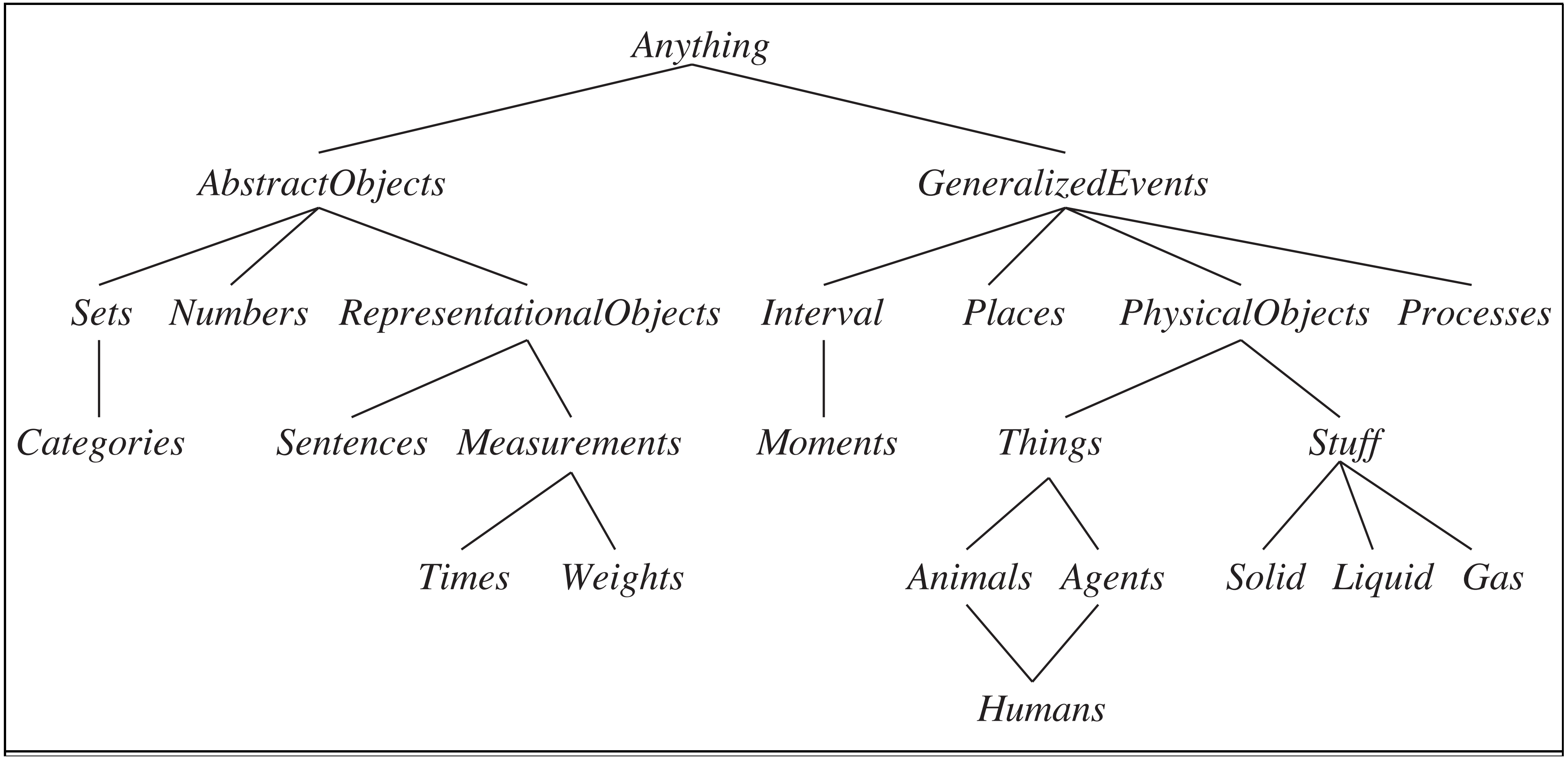


Figure 10.4 Diagram of the blocks-world problem in Figure 10.3.

Ontology and Semantic Web

Up ontology


Anything
AbstractObjects
GeneralizedEvents
Sets
Numbers
RepresentationalObjects
Interval
Places
PhysicalObjects
Processes
Categories
Sentences
Measurements
Moments
Things
Stuff
Times
Weights
Animals
Agents
Solid
Liquid
Gas
Humans

Domain ontology

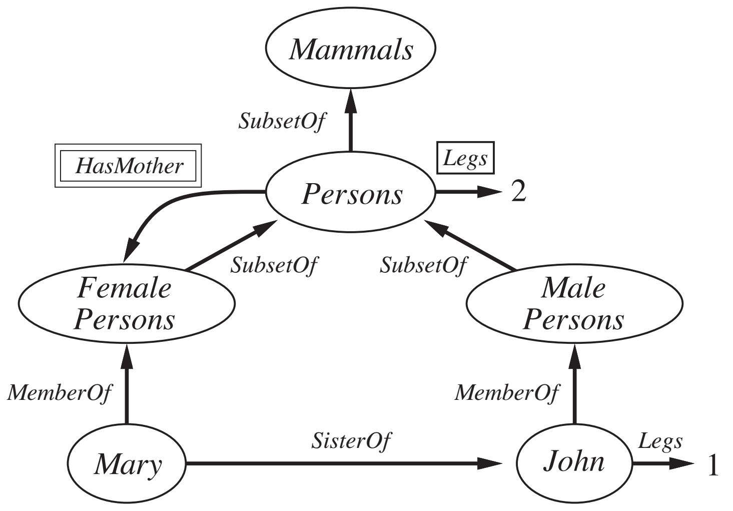


Figure 12.5 A semantic network with four objects (John, Mary, 1, and 2) and four categories. Relations are denoted by labeled links.

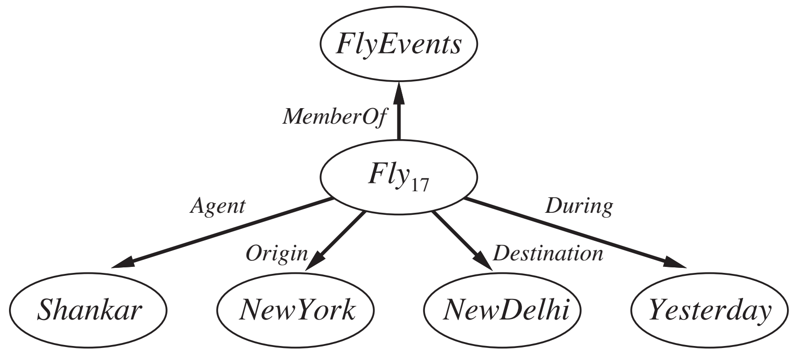
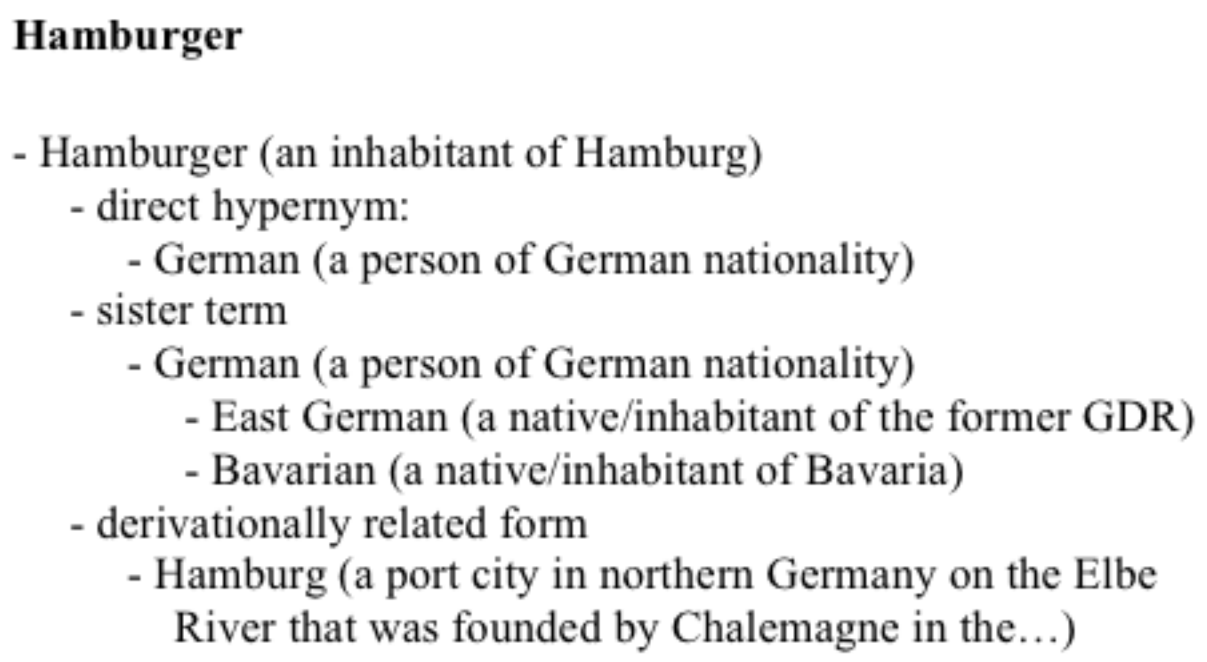


Figure 12.6 A fragment of a semantic network showing the representation of the logical assertion $Fly(Shankar, NewYork, NewDelhi, Yesterday)$.

Example: Wordnet



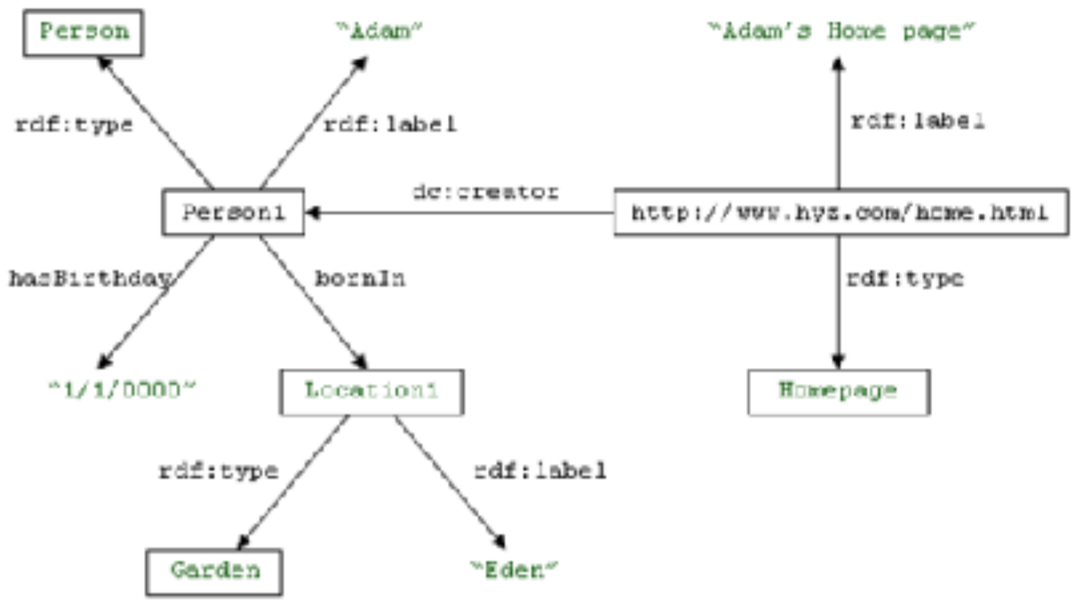
Hamburger

- Hamburger (an inhabitant of Hamburg)
    - direct hypernym:
        - German (a person of German nationality)
    - sister term
        - German (a person of German nationality)
            - East German (a native/inhabitant of the former GDR)
            - Bavarian (a native/inhabitant of Bavaria)
    - derivationally related form
        - Hamburg (a port city in northern Germany on the Elbe River that was founded by Chalemagne in the…)

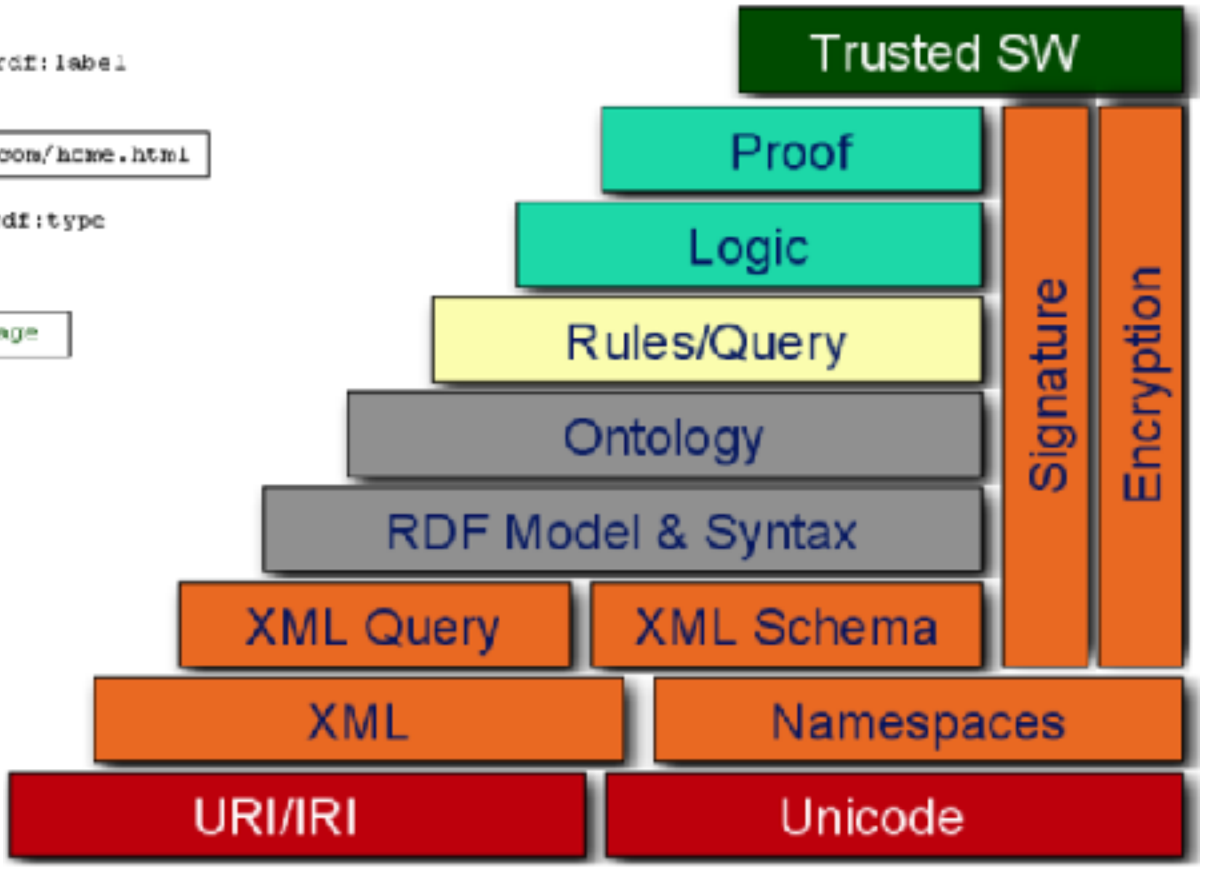
[from wikipedia]

Semantic web

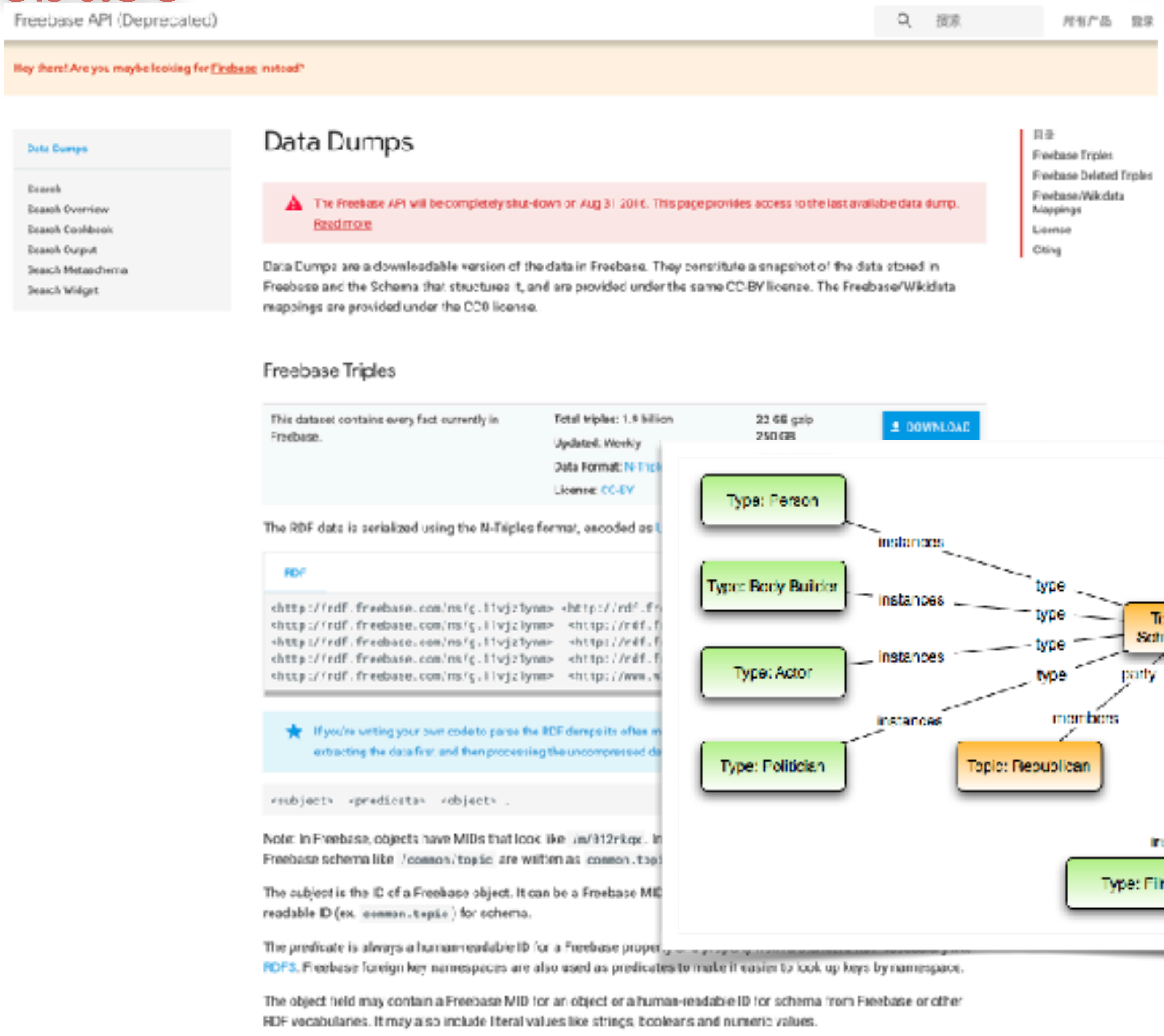
- handling complex and heterogeneous information resources
- retrieving documents based on a set of relationships that are external to these documents
- providing multiple search options for richer investigation
- targeting and sifting results more efficiently
- using authoritative information resources more effectively as guides to searching



An RDF graph



Freebase



Freebase API (Deprecated)

Hey there! Are you maybe looking for Firebase instead?

Data Dumps

Search
Search Overview
Search Cookbook
Search Output
Search Metaschema
Search Widget

Data Dumps

The Freebase API will be completely shut-down on Aug 31 2016. This page provides access to the last available data dump. Read more

Data Dumps are a downloadable version of the data in Freebase. They constitute a snapshot of the data stored in Freebase and the Schema that structures it, and are provided under the same CC-BY license. The Freebase/Wikidata mappings are provided under the CC0 license.

目录
Freebase Triples
Freebase Deleted Triples
Freebase/Wikidata Mappings
License
Citing

Freebase Triples

This dataset contains every fact currently in Freebase.

Total triples: 1.9 billion
Updated: Weekly
Data Format: N-Tripl
License: CC-BY

22 GB gzip
250 GB

DOWNLOAD

The RDF data is serialized using the N-Triples format, encoded as U

RDF

```
<http://rdf.freebase.com/ns/g.11vjz1ynm> <http://rdf.f
<http://rdf.freebase.com/ns/g.11vjz1ynm>  <http://rdf.f
<http://rdf.freebase.com/ns/g.11vjz1ynm>  <http://rdf.f
<http://rdf.freebase.com/ns/g.11vjz1ynm>  <http://rdf.f
<http://rdf.freebase.com/ns/g.11vjz1ynm>  <http://www.w
```

If you're writing your own code to parse the RDF dumps its often m
extracting the data first and then processing the uncompressed da

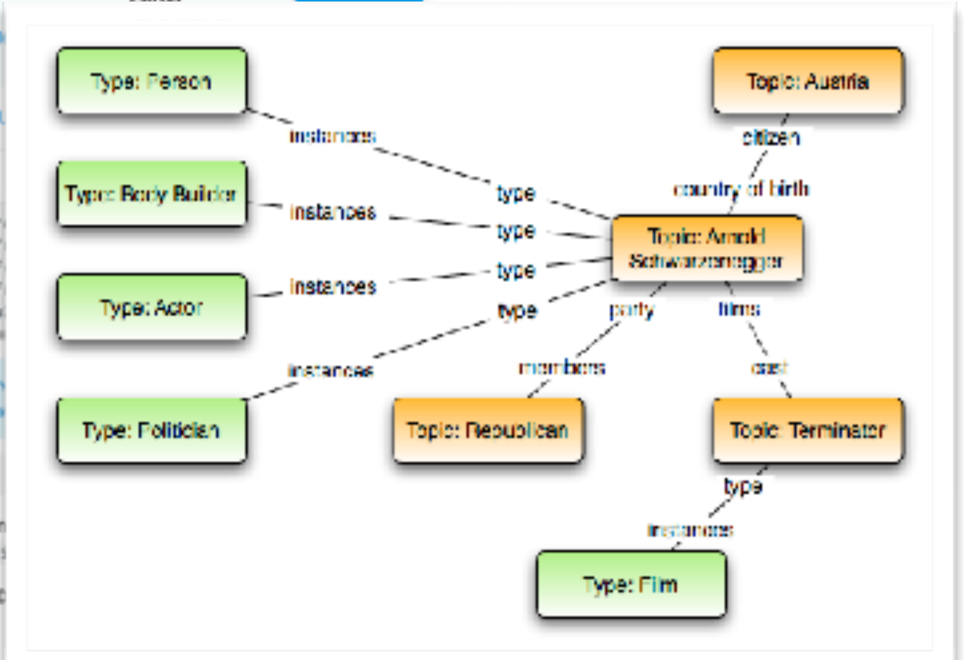
```
<subject>  <predicate>  <object> .
```

Note: In Freebase, objects have MIDs that look like /m/012rkqx. In
Freebase schema like /common/topic are written as common.top

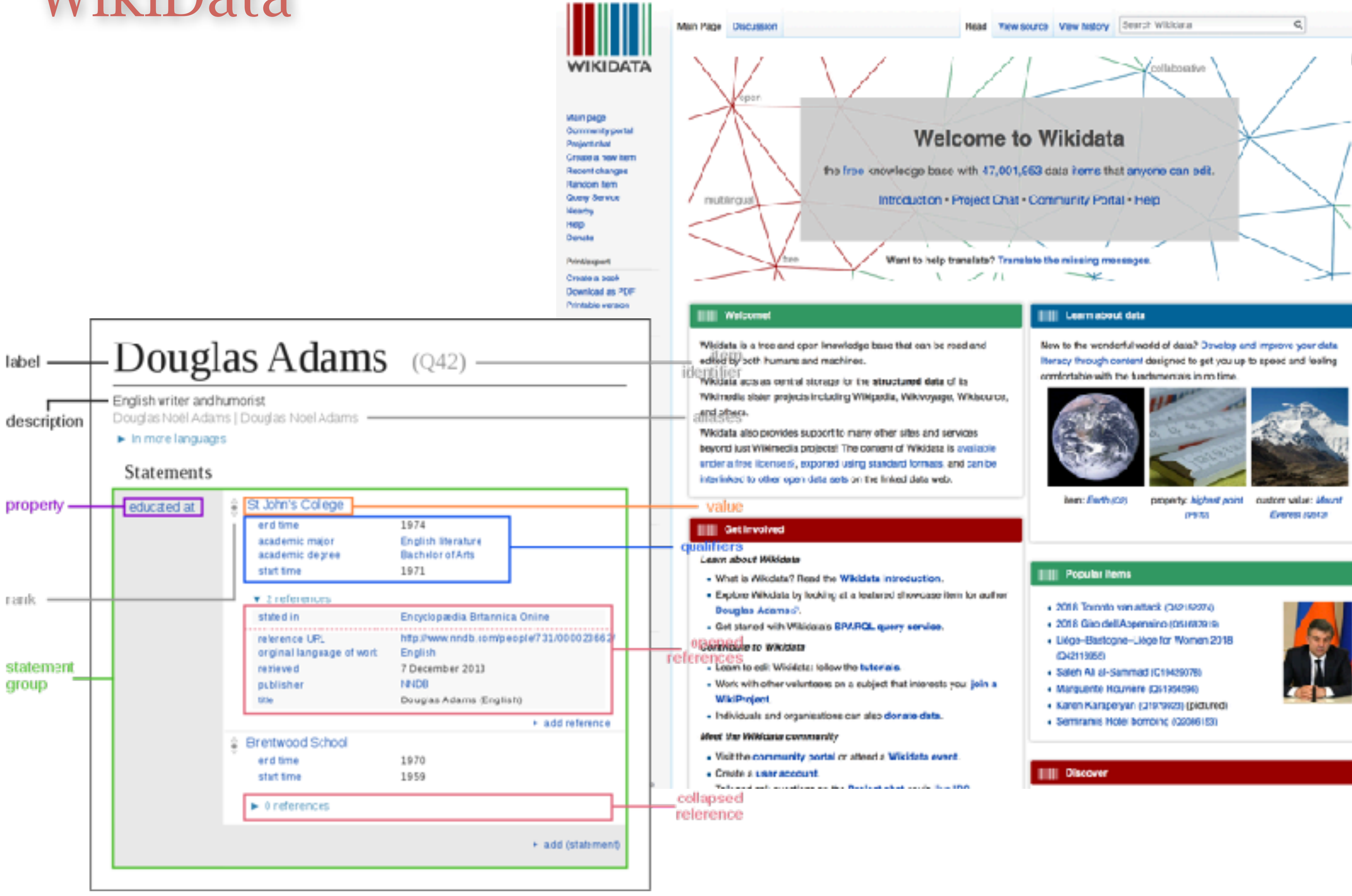
The subject is the ID of a Freebase object. It can be a Freebase MID
readable ID (ex. common.topic) for schema.

The predicate is always a human-readable ID for a Freebase proper
RDFS. Freebase foreign key namespaces are also used as predicates to make it easier to look up keys by namespace.

The object field may contain a Freebase MID for an object or a human-readable ID for schema from Freebase or other RDF vocabularies. It may also include literal values like strings, booleans and numeric values.



WikiData



WIKIDATA

Main page
Community portal
Project chat
Create a new item
Recent changes
Random item
Query Service
Nearby
Help
Donate

Print/export
Create a book
Download as PDF
Printable version

Main Page | Discussion | Read | View source | View history | Search Wikidata

Welcome to Wikidata

the free knowledge base with 47,001,663 data items that anyone can edit.

Introduction • Project Chat • Community Portal • Help

Want to help translate? Translate the missing messages.

Welcome!

Wikidata is a free and open knowledge base that can be read and edited by both humans and machines.

Wikidata acts as central storage for the structured data of its Wikimedia sister projects including Wikipedia, Wikivoyage, Wikisource, and others.

Wikidata also provides support to many other sites and services beyond just Wikimedia projects! The content of Wikidata is available under a free license, exported using standard formats, and can be interlinked to other open data sets on the linked data web.

Learn about data

New to the wonderful world of data? Develop and improve your data literacy through content designed to get you up to speed and feeling comfortable with the fundamentals in no time.

Item: Earth (Q2) | property: highest point (P610) | custom value: Mount Everest (Q513)

Get involved

Learn about Wikidata

- What is Wikidata? Read the Wikidata introduction.
- Explore Wikidata by looking at a featured showcase item for author Douglas Adams.
- Get started with Wikidata's SPARQL query service.

Contribute to Wikidata

- Learn to edit Wikidata: follow the tutorials.
- Work with other volunteers on a subject that interests you: join a WikiProject.
- Individuals and organizations can also donate data.

Meet the Wikidata community

- Visit the community portal or attend a Wikidata event.
- Create a user account.

Popular items

- 2018 Toronto van attack (Q52162274)
- 2018 Giro dell'Appennino (Q51682918)
- Liège–Bastogne–Liège for Women 2018 (Q42113056)
- Saleh Ali al-Sammad (Q19429078)
- Marguerite Houmere (Q51354596)
- Karen Karapetyan (Q19799823) (pictured)
- Semiramis Hotel bombing (Q30886153)

Discover

Douglas Adams (Q42)

English writer and humorist

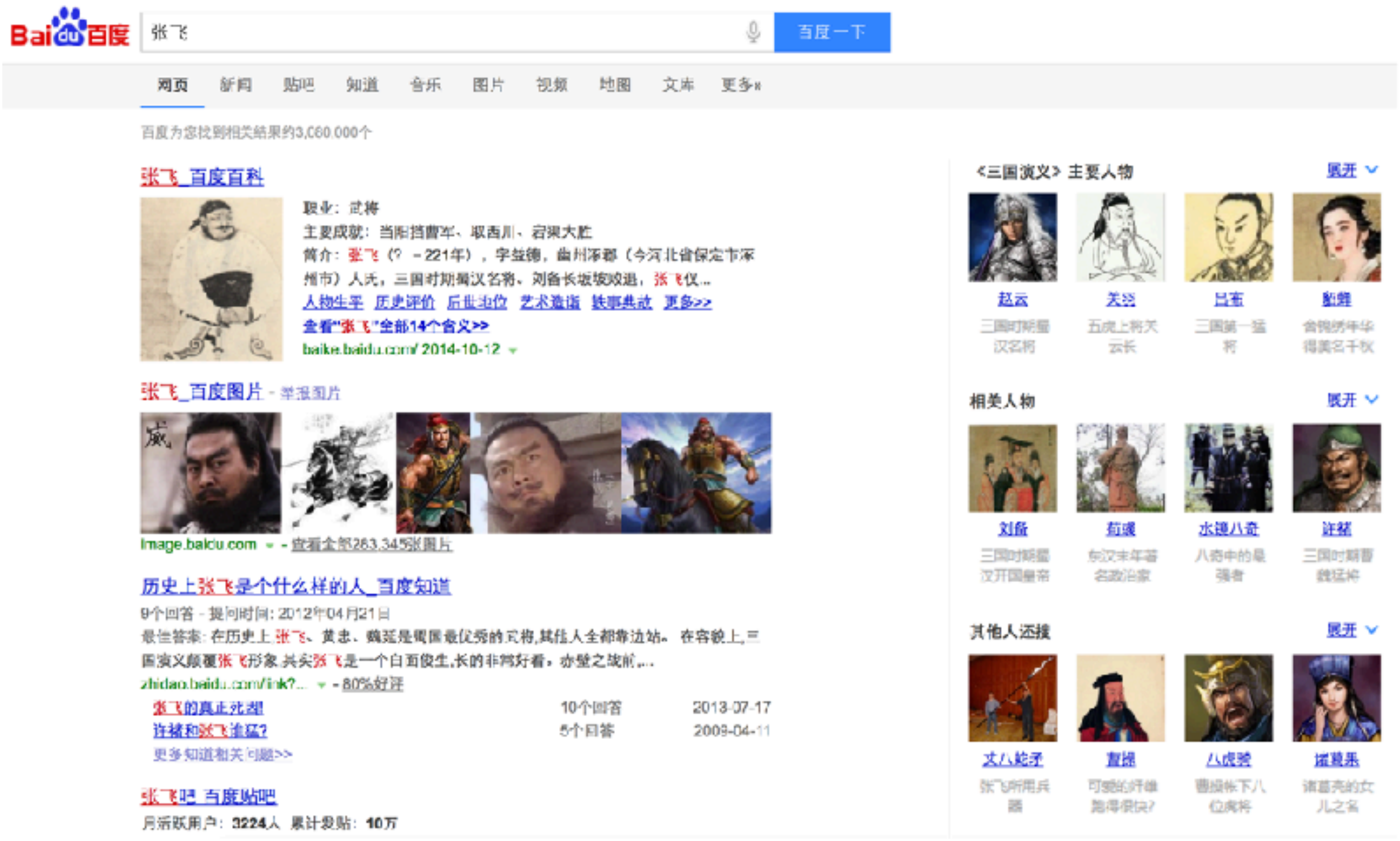
Douglas Noël Adams | Douglas Noel Adams

▸ In more languages

Statements

educated at	St John's College	
	end time	1974
	academic major	English literature
	academic degree	Bachelor of Arts
	start time	1971
	▾ 2 references	
	stated in	Encyclopædia Britannica Online
	reference URL	http://www.nndb.com/people/731/000023662/
	original language of work	English
	retrieved	7 December 2013
	publisher	NNDB
	title	Douglas Adams (English)
	▸ add reference	
	Brentwood School	
	end time	1970
	start time	1959
	▸ 0 references	
	▸ add (statement)	

Example application



百度为您找到相关结果约3,060,000个

张飞_百度百科

职业：武将
主要成就：当阳挡曹军、取西川、宕渠大胜
简介：张飞（? －221年），字益德，幽州涿郡（今河北省保定市涿州市）人氏，三国时期蜀汉名将。刘备长坂坡败退，张飞仅...
人物生平 历史评价 后世地位 艺术造诣 轶事典故 更多>>
查看"张飞"全部14个含义>>
baike.baidu.com/ 2014-10-12

张飞_百度图片 - 举报图片

image.baidu.com - 查看全部263,345张图片

历史上张飞是个什么样的人_百度知道

9个回答 - 提问时间: 2012年04月21日
最佳答案: 在历史上,张飞、黄忠、魏延是蜀国最优秀的武将,其他人全都靠边站。 在客観上,三国演义颠覆张飞形象,其实张飞是一个白面俊生,长的非常好看，赤壁之战前,...
zhidao.baidu.com/link?... - 80%好评

张飞的真正死因	10个回答	2013-07-17
许褚和张飞谁猛?	5个回答	2009-04-11

更多知道相关问题>>

张飞吧_百度贴吧

月活跃用户: 3224人 累计发贴: 10万

《三国演义》主要人物 展开

赵云	关羽	吕布	貂蝉
三国时期蜀汉名将	五虎上将关云长	三国第一猛将	含锦绣年华得美名千秋

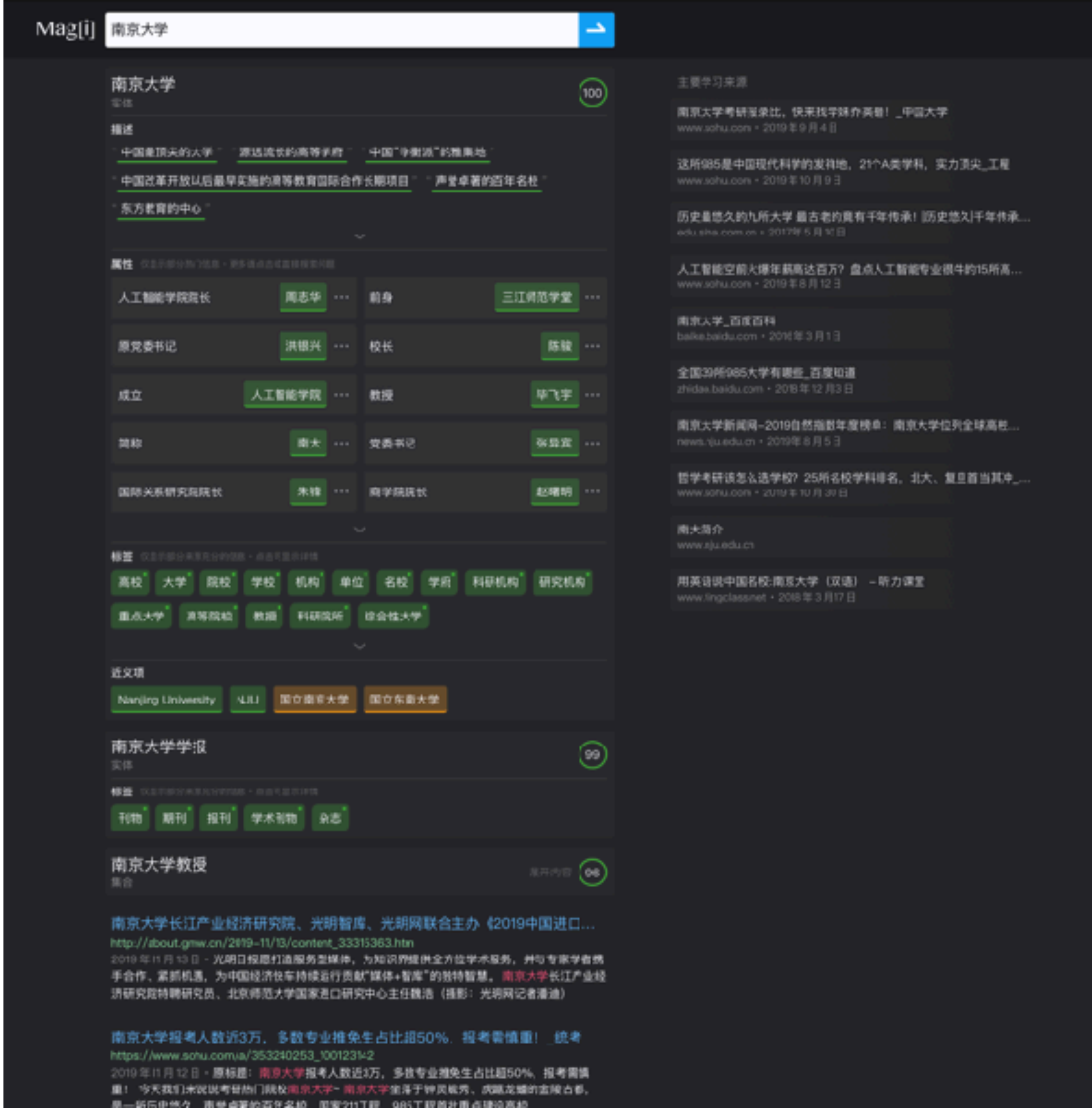
相关人物 展开

刘备	荀彧	水镜八奇	许褚
三国时期蜀汉开国皇帝	东汉末年著名政治家	八奇中的最强者	三国时期曹魏猛将

其他人还搜 展开

丈八蛇矛	曹操	八虎骑	诸葛果
张飞所用兵器	可爱的奸雄跑得很快?	曹操帐下八位虎将	诸葛亮的女儿之名

magi.com



Mag[i] 南京大学

南京大学
实体 100

描述
"中国最顶尖的大学" "源远流长的高等学府" "中国"学衡派"的聚集地"
"中国改革开放以后最早实施的高等教育国际合作长期项目" "声誉卓著的百年名校"
"东方教育的中心"

属性 [illegible]

人工智能学院院长 周志华	前身 三江师范学堂
原党委书记 洪银兴	校长 陈骏
成立 人工智能学院	教授 毕飞宇
简称 南大	党委书记 张异宾
国际关系研究院院长 朱锋	商学院院长 赵曙明

标签 [illegible]
高校 大学 院校 学校 机构 单位 名校 学府 科研机构 研究机构
重点大学 高等院校 教育 科研院所 综合性大学

近义项
Nanjing University NJU 国立南京大学 国立东南大学

南京大学学报
实体 99

标签 [illegible]
刊物 期刊 报刊 学术刊物 杂志

南京大学教授
集合 展开内容 96

南京大学长江产业经济研究院、光明智库、光明网联合主办《2019中国进口...
http://about.gmw.cn/2019-11/13/content_33315363.htm
2019年11月13日 - 光明日报思打造服务型媒体，为知识界提供全方位学术服务，并与专家学者携手合作，紧抓机遇，为中国经济快车持续运行贡献"媒体+智库"的独特智慧。南京大学长江产业经济研究院特聘研究员、北京师范大学国家进口研究中心主任魏浩（摄影：光明网记者潘迪）

南京大学报考人数近3万，多数专业推免生占比超50%，报考需慎重！_统考
https://www.sohu.com/a/353240253_100123142
2019年11月12日 - 原标题：南京大学报考人数近3万，多数专业推免生占比超50%，报考需慎重！ 今天我们来说说考研热门院校南京大学~ 南京大学坐落于钟灵毓秀、虎踞龙蟠的金陵古都，是一所历史悠久、声誉卓著的百年名校，国家211工程、985工程首批重点建设高校

主要学习来源

南京大学考研报录比，快来找学姐介绍吧！_中国大学
www.sohu.com · 2019年9月4日

这所985是中国现代科学的发祥地，21个A类学科，实力顶尖_工程
www.sohu.com · 2019年10月9日

历史最悠久的九所大学 最古老的拥有千年传承！|历史悠久|千年传承...
edu.sina.com.cn · 2017年5月10日

人工智能空前火爆年薪高达百万？盘点人工智能专业很牛的15所高...
www.sohu.com · 2019年8月12日

南京大学_百度百科
baike.baidu.com · 2016年3月1日

全国39所985大学有哪些_百度知道
zhidao.baidu.com · 2018年12月3日

南京大学新闻网-2019自然指数年度榜单：南京大学位列全球高校...
news.nju.edu.cn · 2019年6月5日

哲学考研该怎么选学校？25所名校学科排名，北大、复旦首当其冲_...
www.sohu.com · 2019年10月30日

南大简介
www.nju.edu.cn

用英语说中国名校:南京大学（双语） - 听力课堂
www.tingclass.net · 2018年3月17日