Deliberative Agents
Knowledge Representation: Logical

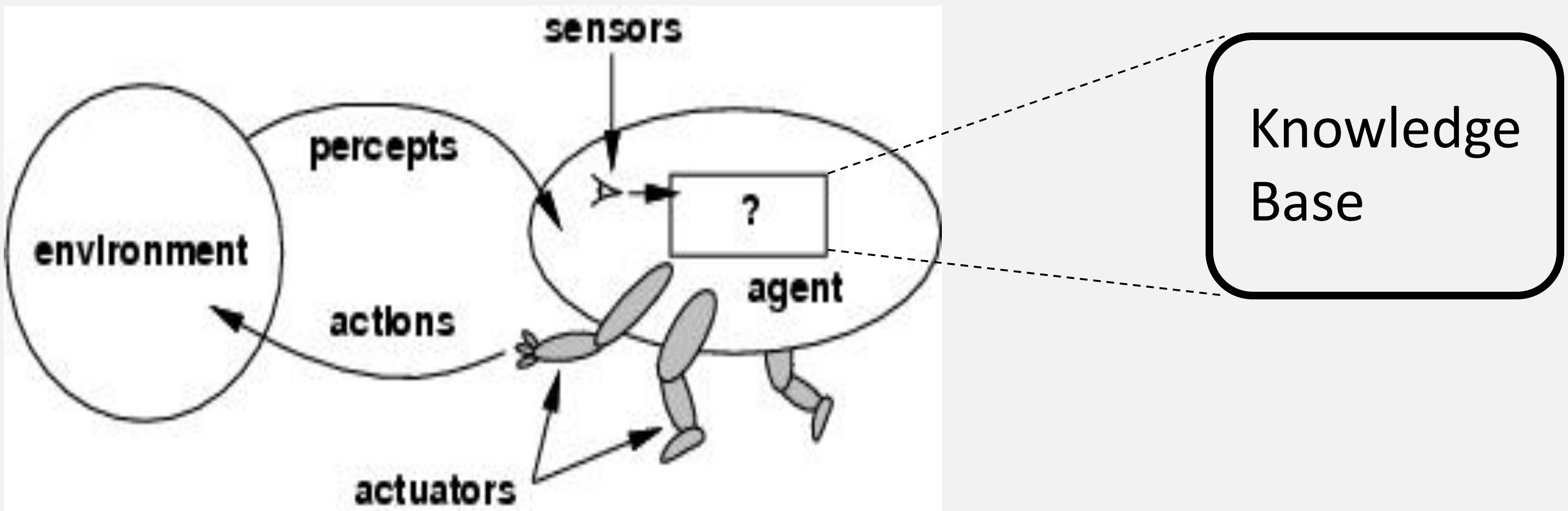
Vasant Honavar
Artificial Intelligence Research Laboratory
Informatics Graduate Program
Computer Science and Engineering Graduate Program
Bioinformatics and Genomics Graduate Program
Neuroscience Graduate Program

Center for Big Data Analytics and Discovery Informatics
Huck Institutes of the Life Sciences
Institute for Cyberscience
Clinical and Translational Sciences Institute
Northeast Big Data Hub
Pennsylvania State University

vhonavar@ist.psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

Deliberative Agents

- Intelligent behavior requires knowledge about the world



Deliberative Agents

- Intelligent behavior requires knowledge about the world
- Procedural, e.g., functions
  - Using knowledge = executing the procedure
- Declarative, e.g., facts
  - Using knowledge = performing inference

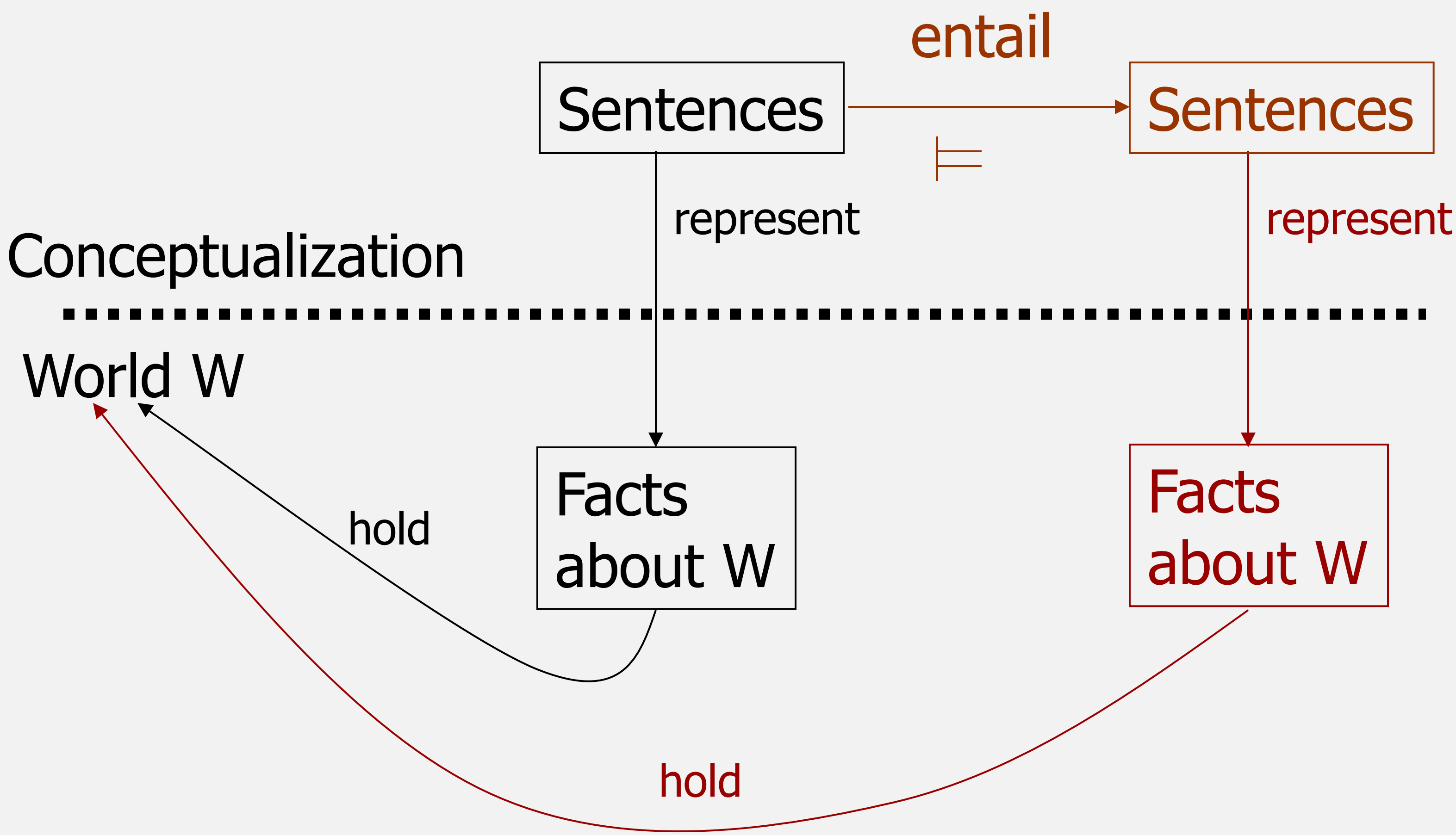
- Deliberative agents
  - Can represent and reason with knowledge
  - Exhibit logical rationality

Knowledge representation (KR) is a surrogate

A declarative knowledge representation

- Encodes facts that are true in the world into sentences
- Reasoning is performed by manipulating sentences according to sound rules of inference
- The results of inference are sentences that correspond to facts that are true in the world
- The correspondence between facts that hold in the world and sentences that describe the world ensures semantic grounding of the representation
- Allows agents to substitute thinking for acting in the world
  - Known facts: The coffee is hot; coffee is a liquid; a hot liquid will burn your tongue;
  - Inferred fact: Coffee will burn your tongue

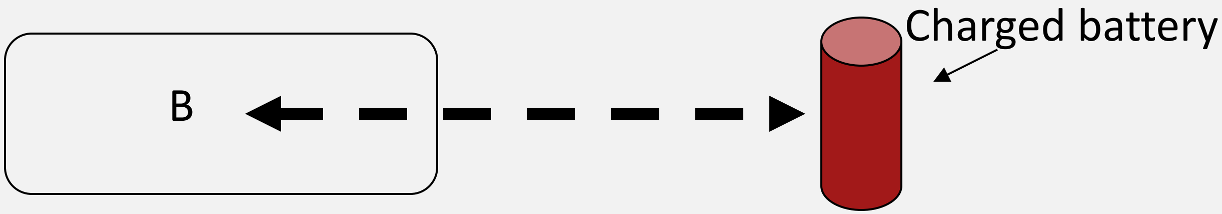
The nature of representation



Propositional Logic - Semantics

A proposition (sentence)

- does not have intrinsic meaning
- gets its meaning from correspondence with statements about the world (interpretation)



- Has a denotation in a given interpretation
  - e.g., proposition B denotes the fact that battery is charged
- Is True or False in a chosen interpretation
- Logical connectives have no intrinsic meaning – need interpretation

KR is a set of ontological commitments

- What does an agent care about?
  - ✓Entities
    - coffee, liquid, tongue
  - ✓Properties
    - being hot, being able to burn
  - ✓Relationships
    - Coffee is a liquid
- KR involves abstraction, simplification
  - A representation is
    - a (logical) model of the world
    - like a cartoon
  - All models are wrong, but some are useful

KR involves a set of epistemological commitments

- What can we know?
  - Propositional logic
    - Is a proposition *true* or *false*?
  - Probability theory
    - What is the *probability* that a given proposition true?
  - Decision theory
    - Which choice among a set of candidate choices is the most *rational*?
  - Logic of Knowledge
    - What does John *know* that Jim does not?

KR is a theory of intelligent reasoning

- How can knowledge be encoded ?
  - Syntax
- What does the encoded knowledge mean?
  - Semantics (entailment)
  - Inferences that are sanctioned by the semantics
- What can we infer from what we know?
  - Inferences that can be performed (by rules of inference)
  - Soundness, completeness, efficiency
- How can we manage inference?
  - What should we infer from among the things we can infer?

KR formalisms

KR formalisms provide provision for describing

- Individuals
- Sets of individuals (classes)
- Properties of individuals
- Properties of classes
- Relationships between individuals
- Relationships between classes
- Actions and their effects
- Locations and events in space and time
- Uncertainty
- Knowledge
- Beliefs
- Preferences
- …..

KR formalisms

- Logical
  - e.g., Propositional Logic, First order logic, Description logic
- Probabilistic
  - e.g., Bayesian networks
- Grammatical
  - e.g., Context free grammars
- Structured
  - e.g., frames – as in object-oriented programming
- Decision theoretic
- …

KR is a medium for efficient computation

- Reasoning = computation
- Anticipated by Leibnitz, Hilbert
  - Can all truths be reduced to calculation?
  - Is there an effective procedure for determining whether or not a conclusion is a logical consequence of a set of facts?
- KR involves tradeoffs between
  - Expressivity and tractability (decidability, efficiency) tradeoff
    - The more you can say (using a representational formalism), the less you can effectively do (within the formalism)
  - General purpose reasoning versus special-purpose, domain-specific inference
  - Declarative versus procedural knowledge

KR is a medium of expression and communication

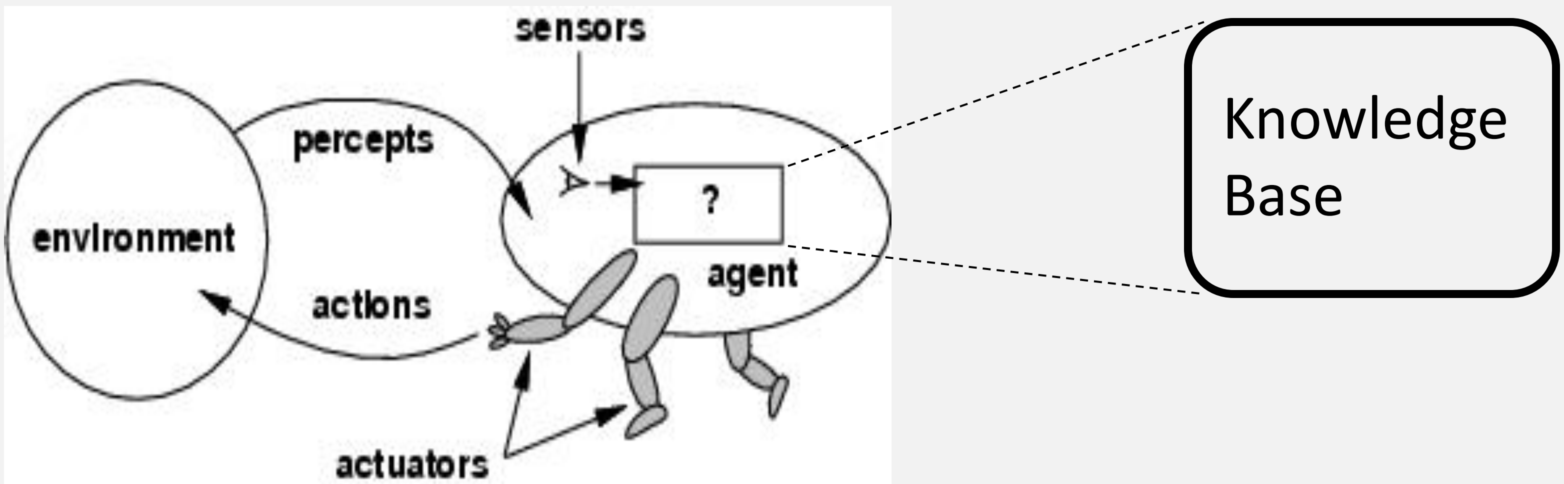
- If we assume shared
  - Ontological and epistemological commitments
  - KR formalism (syntax, semantics, reasoning)
- Then KR is a medium for
  - Expression
    - How general is it?
    - How precise is it?
    - Is it expressive enough?
  - Communication
    - Can we talk or think in the language?
    - What can we communicate the things we want to communicate?
    - What things are difficult to communicate?

Logical Agents I – Propositional Logic

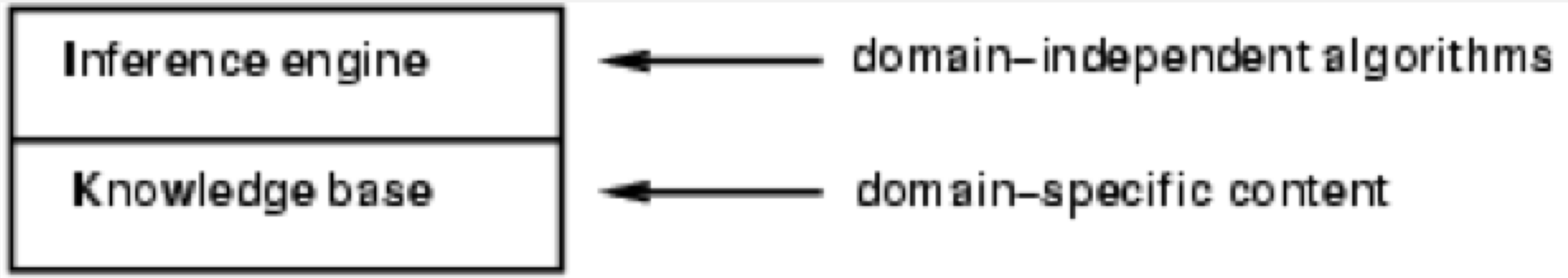
- Knowledge-based agents
- Logic in general - models and entailment
- Propositional (Boolean) logic
  - Syntax
  - Semantics
  - Equivalence, Validity, Satisfiability, Decidability
- Inference rules
  - Soundness, completeness
  - Resolution
- Inference procedures for automated theorem proving
  - Soundness, completeness, efficiency

Knowledge-Based Agents

- Knowledge-based agents represent and use knowledge about the world



Knowledge Base



- Knowledge base = set of sentences about the world in a declarative formal language
- Basic operations on the knowledge base
  - Tell
  - Ask
- Additional operations
  - Retract (in non monotonic logic)

A simple knowledge-based agent

Knowledge based agents

- Are not arbitrary programs
- Are amenable to knowledge level description
  - You can specify an agent's capability in terms of what it knows

Logic as a Knowledge Representation Formalism

Logic is a declarative language to:

- Assert sentences representing facts that hold in a world *W* (these sentences are given the value true)
- Deduce the true/false values to sentences representing other aspects of *W*

Examples of Logics

- Propositional logic A ▯ B →C
- First-order predicate logic ∀x ∃y Mother(y, x)
- Logic of Knowledge **K** (John, Father(Zeus, Cronus))
- Logic of Belief **B**(John, Father(Zeus, Cronus))

Knowledge versus belief:

- You can *believe* things that are false
- You cannot *know* things that are false

Components of Logical KR

- A logical formalism
  - Syntax for well-formed-formulae (wff)
  - Vocabulary of logical symbols (and, or, not, implies etc.)
  - Interpretation semantics for logical symbols
    - e.g. A or B holds in the world whenever A holds in the world or B holds in the world
- An ontology
  - Vocabulary of non logical symbols
    - objects, properties (e.g., A above), etc.
    - definitions of non-primitive symbols (e.g., iff)
    - axioms restricting the interpretation of primitive symbols (more on this later)
- Proof theory – sound rules of inference

Propositional Logic

- Atomic sentences - propositions
  - Propositions can be true or false
- Statements can be combined using *logical connectives*. Examples:
  - C = "It' s cold outside"
    - C is a proposition
  - O = "It's October"
    - O is a proposition
  - If O then C
    - if it's October then it' s cold outside

Propositional Logic: Syntax, logical symbols

The proposition symbols P_1, P_2 etc are sentences

If S is a sentence, $\neg S$ is a sentence (negation)

If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)

If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)

If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)

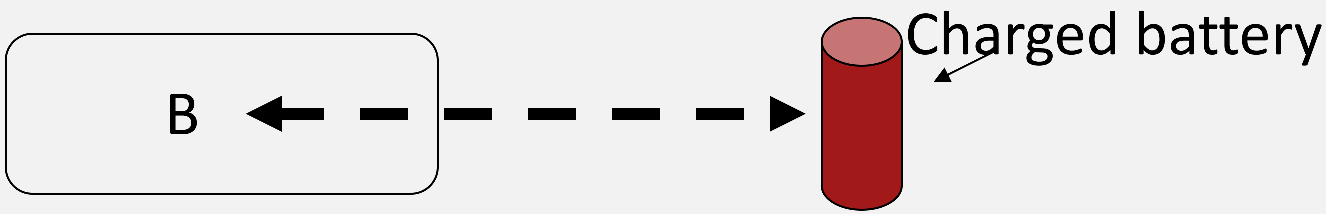
If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

We use extra-linguistic symbols (e.g., parentheses) to group sentences to avoid ambiguity

Propositional Logic - Semantics

A Proposition

- does not have intrinsic meaning
- gets its meaning from correspondence with statements about the world (interpretation)



- Has a denotation in a given interpretation
  - e.g., proposition B denotes the fact that *battery is charged*
- Is *True* or *False* in a chosen interpretation
- Connectives do not have intrinsic meaning – need interpretation

Propositional Logic – Model Theoretic Semantics

- Consider a logic with only two propositions:
  - *Rich*, *Poor*
  - denoting *Tom is rich* and *Tom is poor* respectively
- *A model M* is a subset of the set *A* of atomic sentences in the language
- By a model *M* we mean the state of affairs in which
  - every atomic sentence that is in *M* is *true* and
  - every atomic sentence that is not in *M* is *false*

Example

$$A = \{Rich, Poor\}$$

$$M_0 = \{\ \};\ M_1 = \{Rich\};\ M_2 = \{Poor\};\ \ M_3 = \{Rich, Poor\}$$

Propositional Logic: Semantics of Logical Symbols

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Note that $(P \Rightarrow Q) \Leftrightarrow (\neg P \vee Q)$

Model Theoretic Semantics

$$A = \{Rich, Poor\}$$

$$M_0 = \{\ \};\ M_1 = \{Rich\};\ M_2 = \{Poor\};\ \ M_3 = \{Rich, Poor\}$$

$Rich$ is $True$ in M_1, M_3

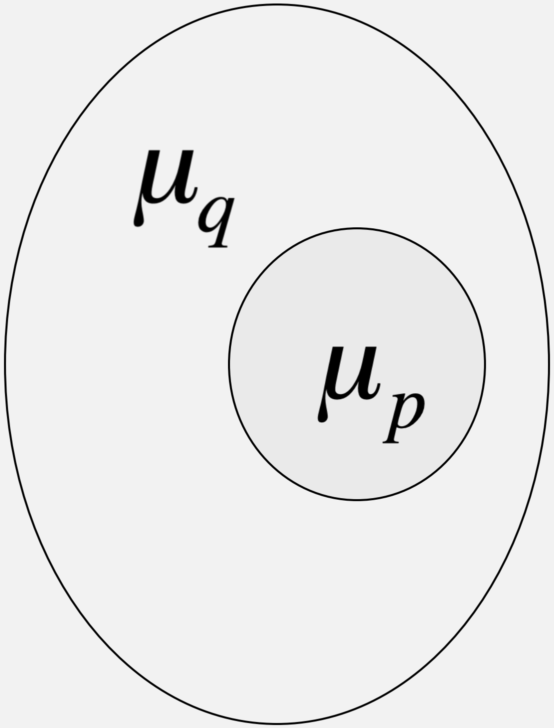
$Rich \vee Poor$ is $True$ in M_1, M_2, M_3

$Rich \wedge Poor$ is $True$ in M_3 Hmm !!

$Rich \Rightarrow \neg Poor$ is $True$ in M_0, M_1, M_2

Proof Theory: Entailment

- We say that *p* entails *q* (written as $p \models q$) if *q* holds in every model in which *p* holds



$\mu_q =$ set of models in which q holds

$\mu_p =$ set of models in which p holds

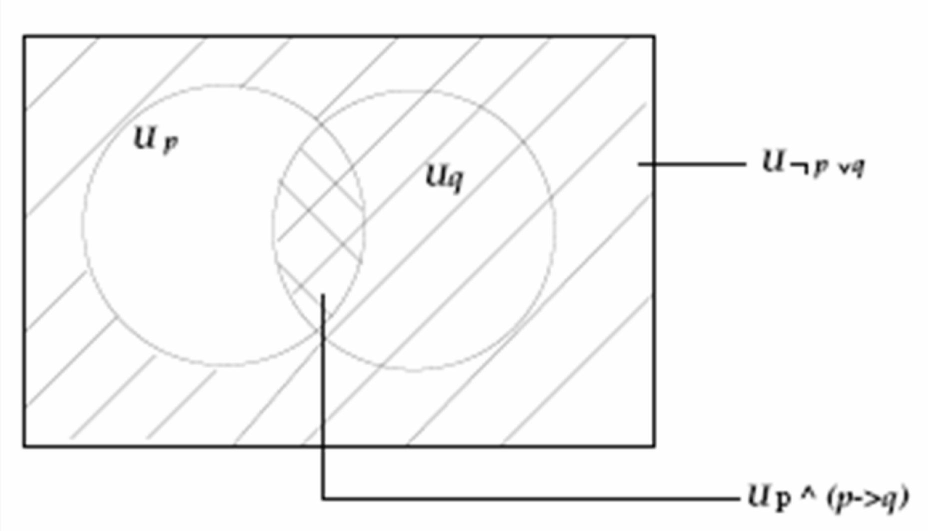
$p \models q$ if it is the case that $\mu_p \subseteq \mu_q$

Entailment – Example

$$p \wedge (p \Rightarrow q) \models q$$

Proof

$$(p \in M) \wedge ((p \Rightarrow q) \in M)$$
$$(p \in M) \wedge ((\neg p \vee q) \in M)$$
$$((p \wedge \neg p) \vee (p \wedge q)) \in M$$
$$p \wedge q \in M$$
$$\therefore \mu_{p \wedge (p \Rightarrow q)} = \mu_{p \wedge q} \subseteq \mu_q$$



Validity and satisfiability

- A sentence is valid if it is true in *all* models,
  - e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- A sentence is satisfiable if it is true in *some* model
  - e.g., $A \vee B$, C
- 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
  - Useful for proof by contradiction

Logical Rationality

- An propositional logic based agent *A* with a knowledge base KB_A is justified in inferring *q* if it is the case that

$$KB_A \models q$$

- How can the agent A decide whether in fact $KB_A \models q$?
  - Model checking
    - Enumerate all the models in which KB_A holds
    - Enumerate all the models in which *q* holds
    - Check whether $KB_A \subseteq \mu_q$
  - Inference algorithm based on inference rules

Searching for proofs: inference

- An inference rule $\{\xi, \psi\} \models \varphi$ consists of
  - 2 sentence patterns ξ and ψ called the premises and
  - one sentence pattern φ called the conclusion
- If ξ and ψ match two sentences of KB then
  - the corresponding φ can be inferred according to the rule
- Given a set of inference rules I and a knowledge base *KB*
  - inference is the process of successively applying inference rules from I to *KB*
  - each rule application adds its conclusion to *KB*

Inference rules

- *Modus ponens*

$$\frac{p \Rightarrow q \quad p}{q}$$

Modus ponens derives only inferences sanctioned by entailment

Modus ponens is sound

- *Loony tunes*

$$\frac{friday}{q}$$

Loony tunes can derive inferences that are not sanctioned by entailment

Loony tunes is not sound

Example: Inference using Modus Ponens

$$\{ p \Rightarrow q \, , p \} \vdash \quad q$$

$$\{\xi, \psi\} \vdash \varphi$$

KB:

Battery-OK ∧ *Bulbs-OK* ⇒ *Headlights-Work*

Battery-OK ∧ *Starter-OK* ∧ ¬*Empty-Gas-Tank* ⇒ *Engine-Starts*

Engine-Starts ∧ ¬*Flat-Tire* ∧ *Headlights-Work* ⇒ *Car-OK*

Battery-OK, Bulbs-OK, Starter-OK , ¬*Empty-Gas-Tank,* ¬*Flat-Tire*

Ask

Car-OK?

Soundness and Completeness of an inference rule $\vdash$

- We write $p \vdash q$ to denote that that p can be inferred from q using the inference rule $\vdash$

An inference rule $\vdash$ is said to be

- Sound if whenever $p \vdash q$, it is also the case that $p \models q$
- Complete if whenever $p \models q$, it is also the case that $p \vdash q$

Soundness and Completeness of an inference rule ⊢

- We can show that *modus ponens* is sound, but *not* complete unless the KB is *Horn* i.e., the KB can be written as a collection of sentences of the form

$$a_1 \wedge a_2 \wedge \ldots \wedge a_n \Rightarrow b$$

where each a_i and b are atomic sentences

Unsound inference rules are not necessarily useless!

Abduction (Charles Peirce) is not sound, but useful in diagnostic reasoning or hypothesis generation

$$\frac{\begin{array}{l} p \Rightarrow q \\ q \end{array}}{p}$$

$$\frac{\begin{array}{l} BlockedArtery \Rightarrow HeartAttack \\ HeartAttack \end{array}}{BlockedArtery}$$

Logical equivalence

- Two sentences are logically equivalent iff they are true in same set of models or $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$.

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Sound Inference rules in PL

- Modus Ponens $$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$
- And-elimination: from a conjunction any conjunct can be inferred: $$\frac{\alpha \wedge \beta}{\alpha}$$
- All logical equivalences can be used as inference rules $$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$$
- An inference procedure involves repeated application of applicable inference rules.
- An inference procedure is sound if it uses only sound inference rules.

Constructing proofs

- Finding proofs can be cast as a search problem
- Search can be
  - forward (forward chaining) to derive *goal* from *KB*
  - or backward (backward chaining) from the *goal*
- Searching for proofs
  - Is not more efficient than enumerating models in the worst case
  - Can be more efficient in practice with the use of suitable heuristics
- Propositional logic is monotonic
  - the set of entailed sentences can only increase as inferred facts are added to KB

$$\text{for any sentences } \alpha \text{ and } \beta :$$

$$\textit{if } KB \models \alpha \textit{ then } KB \wedge \beta \models \alpha$$

Soundness and Completeness

- An inference algorithm starts with the KB and applies applicable inference rules until the desired conclusion is reached
- An inference algorithm is sound if it uses a sound inference rule
- An inference algorithm is complete if
  - It uses a complete inference rule and
  - a complete search procedure

Soundness of Modus Ponens for Propositional Logic

- Modus Ponens is sound for Propositional Logic
- Modus Ponens is complete for Horn KB:
  - Whenever $KB \models q$, repeated application of Modus ponens is guaranteed to infer q if the KB consists of only Horn Clauses, i.e., only sentences of the form: $p_1 \wedge p_2 \wedge ..p_n \Rightarrow q$
  - Complexity of inference is polynomial in the size of the Horn KB
- Modus Ponens is not complete for arbitrary Propositional KB

Completeness of Modus Ponens for Propositional Logic

- Modus Ponens is not complete for Propositional Logic
- Suppose that all classes at some university meet either Mon/Wed/Fri or Tue/Thu. The AI course meets at 2:30 PM in the afternoon, and Jane has volleyball practice Thursdays and Fridays at that time.
- Can Jane take AI?

$$MonWedFriAI230pm \vee TueThuAI230pm$$
$$TueThuAI230pm \wedge JaneBusyThu230pm \rightarrow JaneConflictAI$$
$$MonWedFriAI230pm \wedge JaneBusyFri230pm \rightarrow JaneConflictAI$$
$$JaneBusyThu230pm$$
$$JaneBusyFri230pm$$

Completeness of Modus Ponens for Propositional Logic

- Modus Ponens is not complete for Propositional Logic
- Can Jane take AI?

$$MonWedFriAI230pm \vee TueThuAI230pm$$
$$TueThuAI230pm \wedge JaneBusyThu230pm \rightarrow JaneConflictAI$$
$$MonWedFriAI230pm \wedge JaneBusyFri230pm \rightarrow JaneConflictAI$$
$$JaneBusyThu230pm$$
$$JaneBusyFri230pm$$

- Of course not!
- Try proving this using Modus Ponens
- You can't!
- Why?

Completeness of Modus Ponens for Propositional Logic

$$1.\ MonWedFriAI230pm \vee TueThuAI230pm$$
$$2.\ TueThuAI230pm \wedge JaneBusyThu230pm \rightarrow JaneConflictAI$$
$$3.\ MonWedFriAI230pm \wedge JaneBusyFri230pm \rightarrow JaneConflictAI$$
$$4.\ JaneBusyThu230pm$$
$$5.\ JaneBusyFri230pm$$

- We can use Modus Ponens to establish

$$2\&4\text{:}\ TueThuAI230pm \rightarrow JaneConflictAI$$
$$3\&4\text{:}\ MonWedFriAI230pm \rightarrow JaneConflictAI$$

But Modus Ponens can't take us further to conclude $JaneConflictAI$!

- Modus Ponens is not complete for Propositional Logic (except in the restricted case when the KB is Horn)
- However, we can generalized Modus Ponens to obtain an inference rule for Propositional Logic

Proof

- The proof of a sentence α from a set of sentences *KB* is the derivation of α obtained through a series of applications of sound inference rules to *KB*

Inference with Horn KB

- Each Horn clause has only one positive, i.e., un-negated, literal
- Inference can be done with forward or backward chaining
- Entailment decidable in time linear in the size of propositional KB
- Prolog

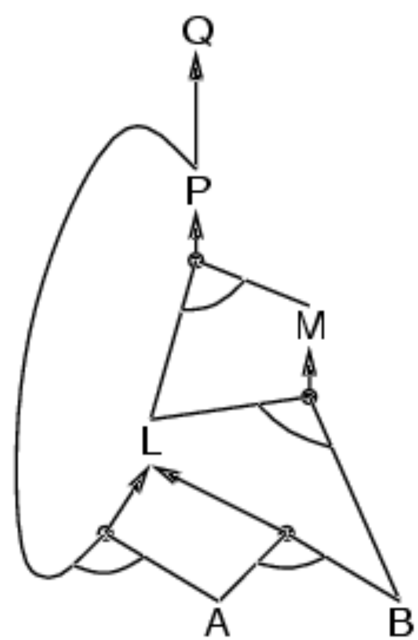
Forward chaining

- Idea: Apply any rule whose premises are satisfied in the *KB*,
  - add its conclusion to the *KB*, until query is proved.
  - Example: Query: Q

KB

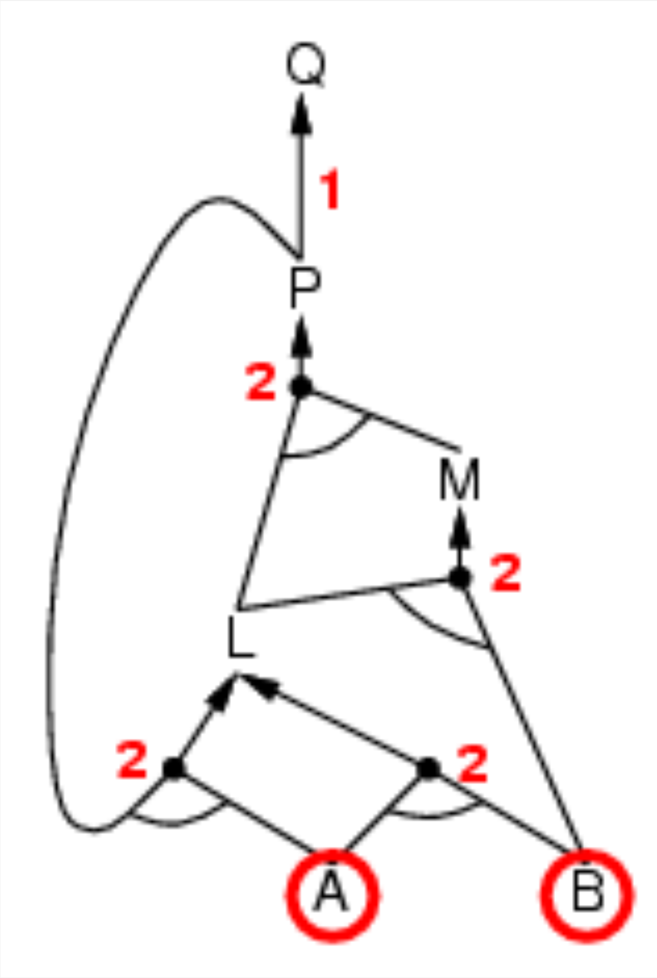
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Forward chaining proof of Q

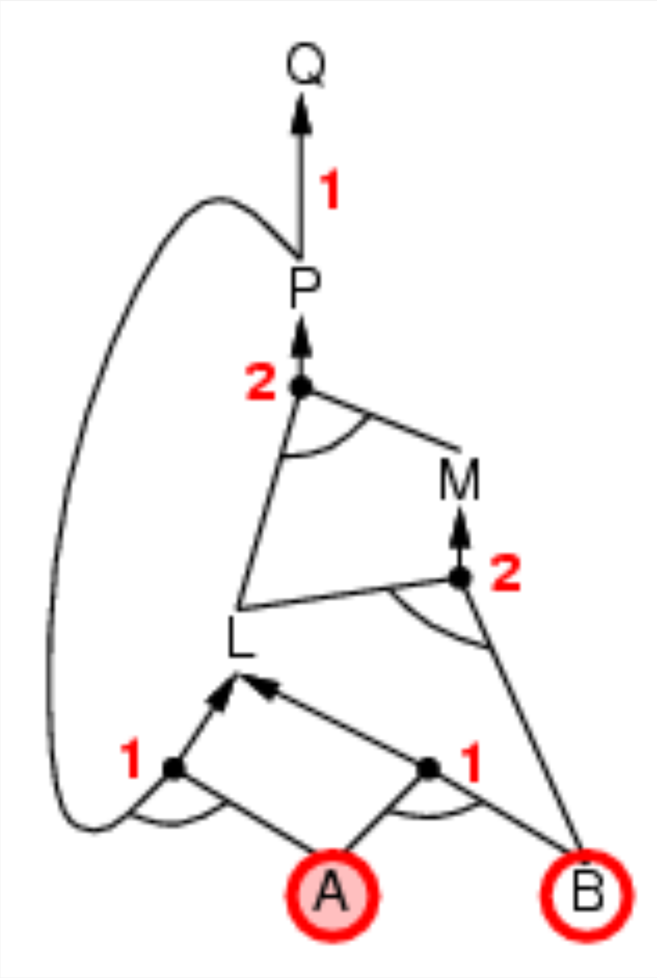


- Forward chaining is sound and complete for Horn KB

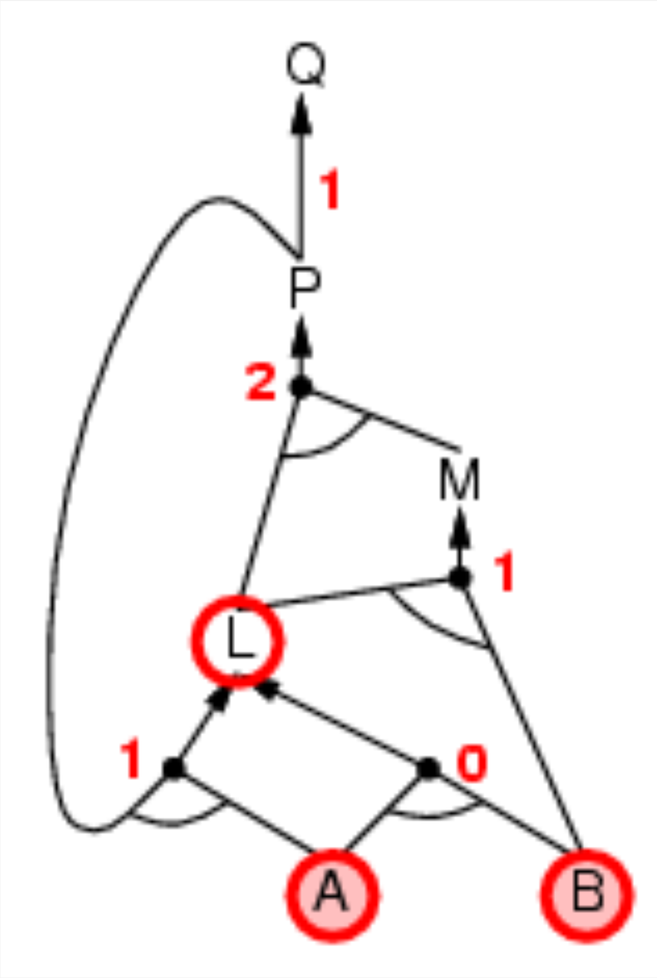
Forward chaining example



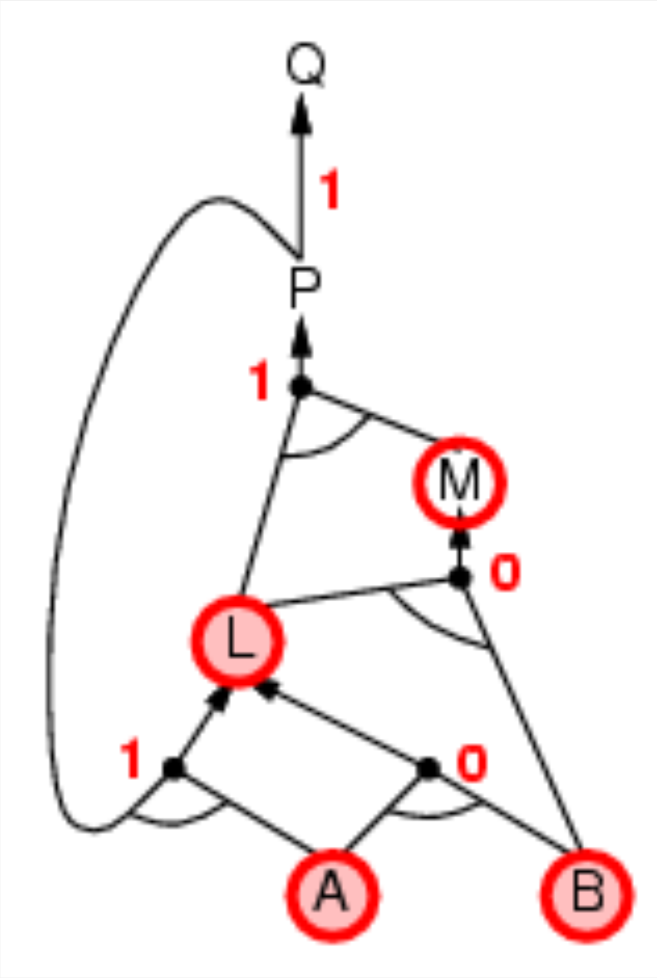
Forward chaining example



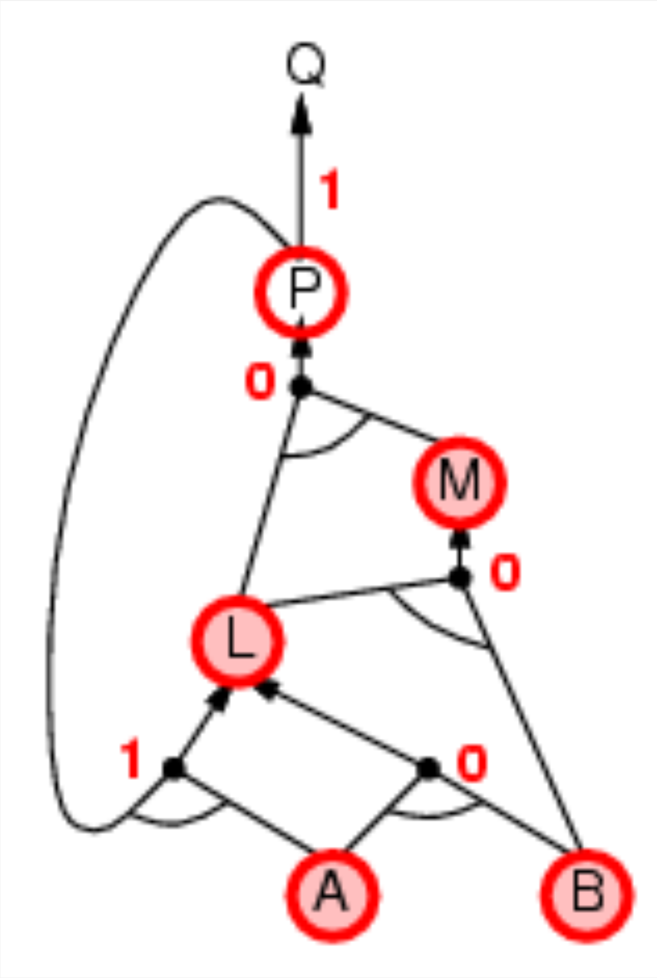
Forward chaining example



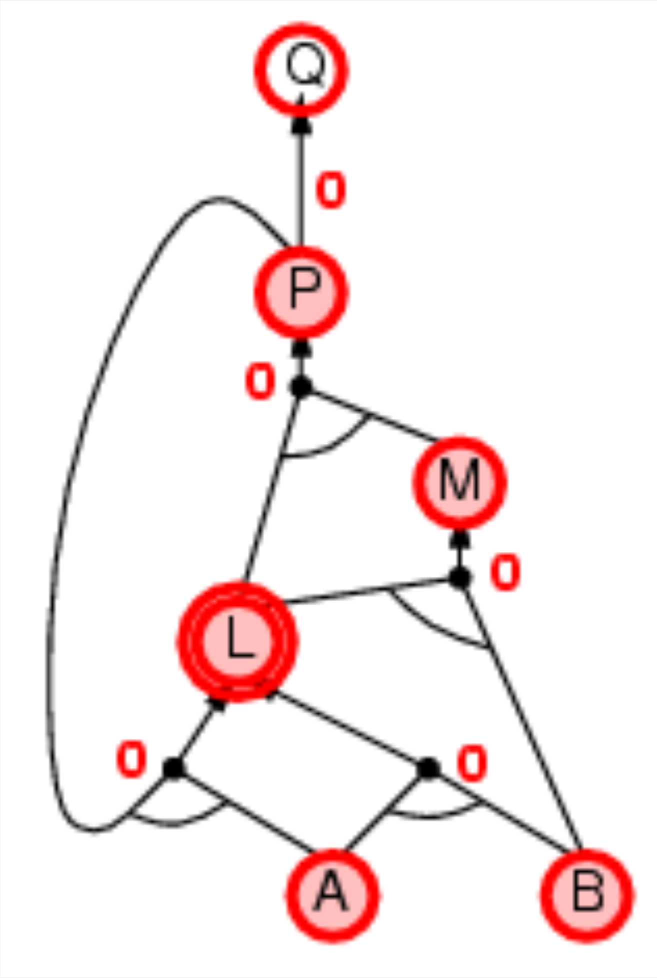
Forward chaining example



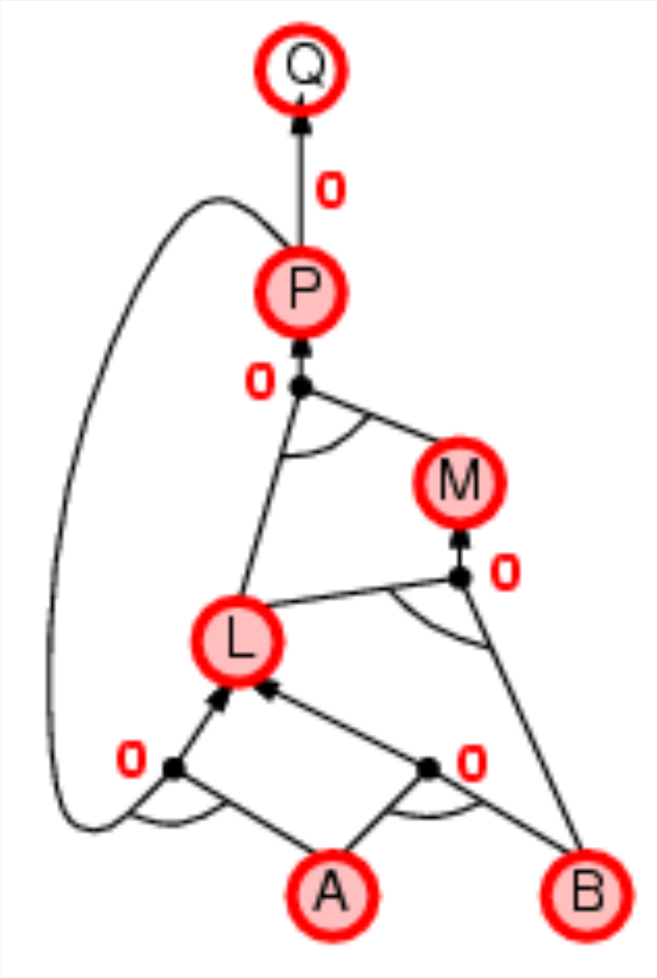
Forward chaining example



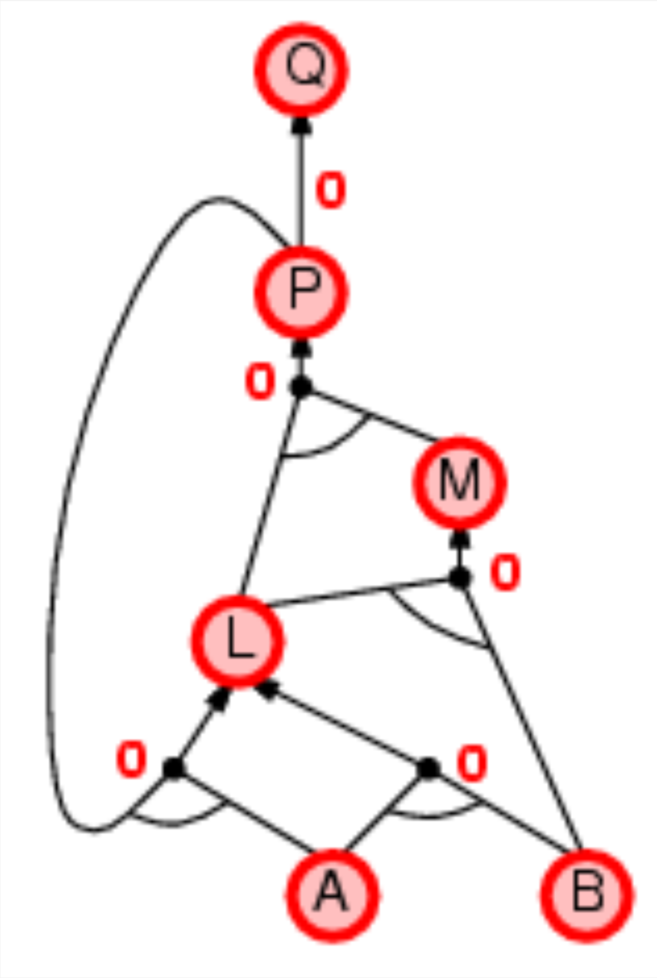
Forward chaining example



Forward chaining example



Forward chaining example



Soundness and Completeness of Forward Chaining

- An inference algorithm starts with the KB and applies applicable inference rules until the desired conclusion is reached
- An inference algorithm is sound if it uses a sound inference rule
- An inference algorithm is complete if
  - It uses a complete inference rule and
  - a complete search procedure
- Forward chaining using Modus Ponens is sound and complete for Horn knowledge bases (i.e., knowledge bases that contain only Horn clauses)

Backward chaining

Idea: work backwards from the query *q*:

to prove *q* by BC,

check if *q* is known already, or

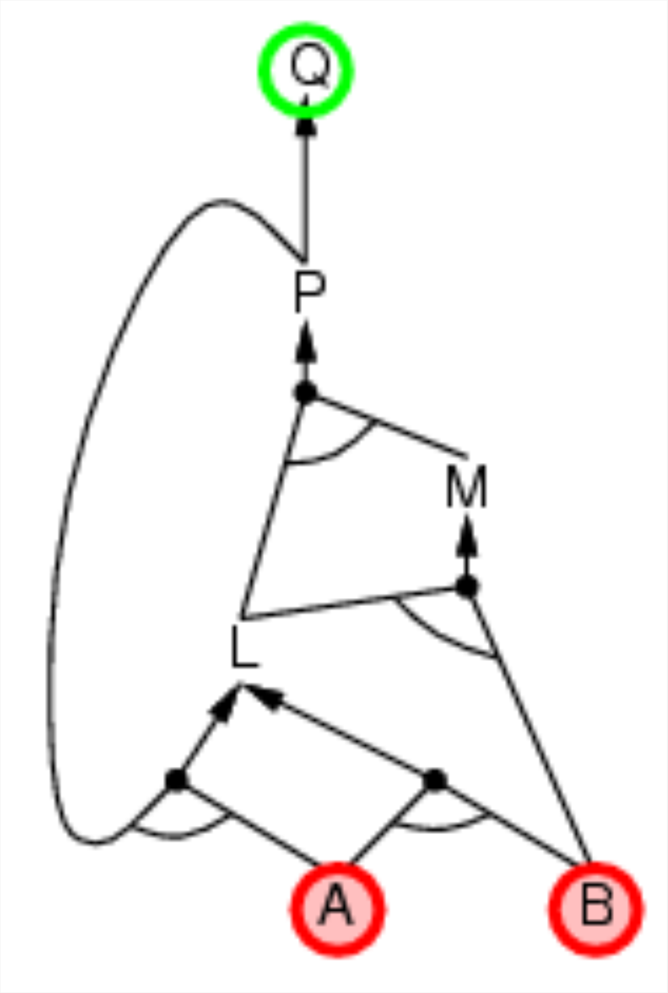
prove by BC all premises of some rule concluding *q*

Avoid loops: check if new subgoal is already on the goal stack

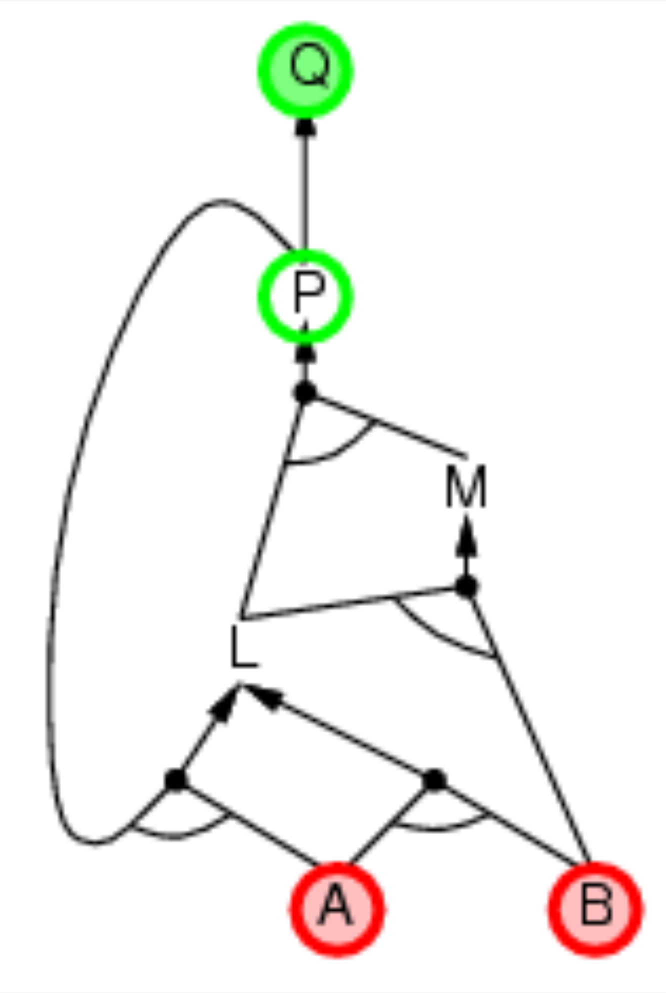
Avoid repeated work: check if new subgoal

- has already been proved true, or
- has already failed

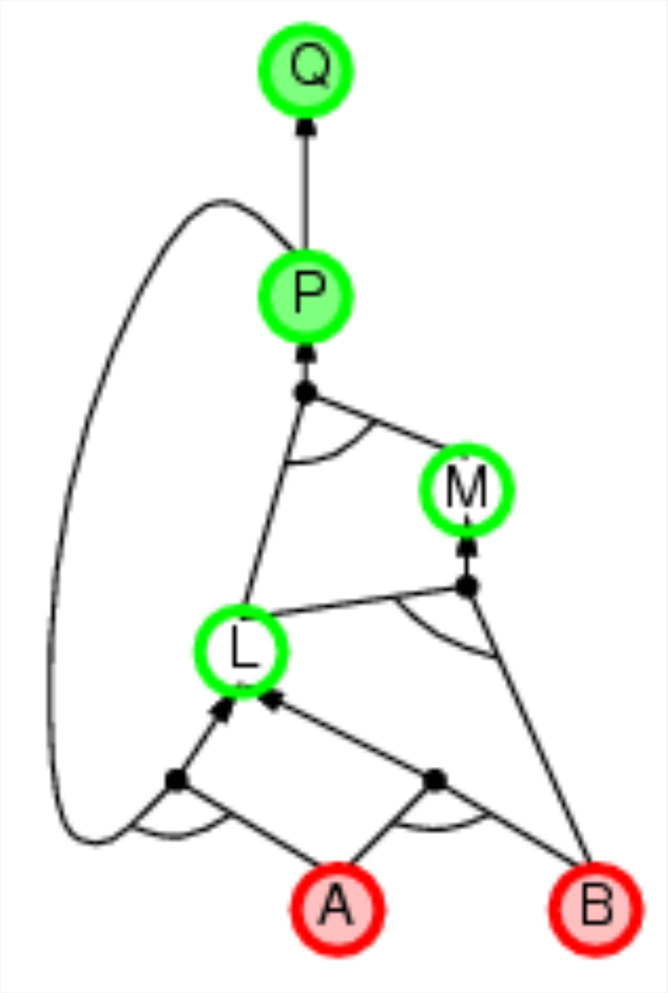
Backward chaining example



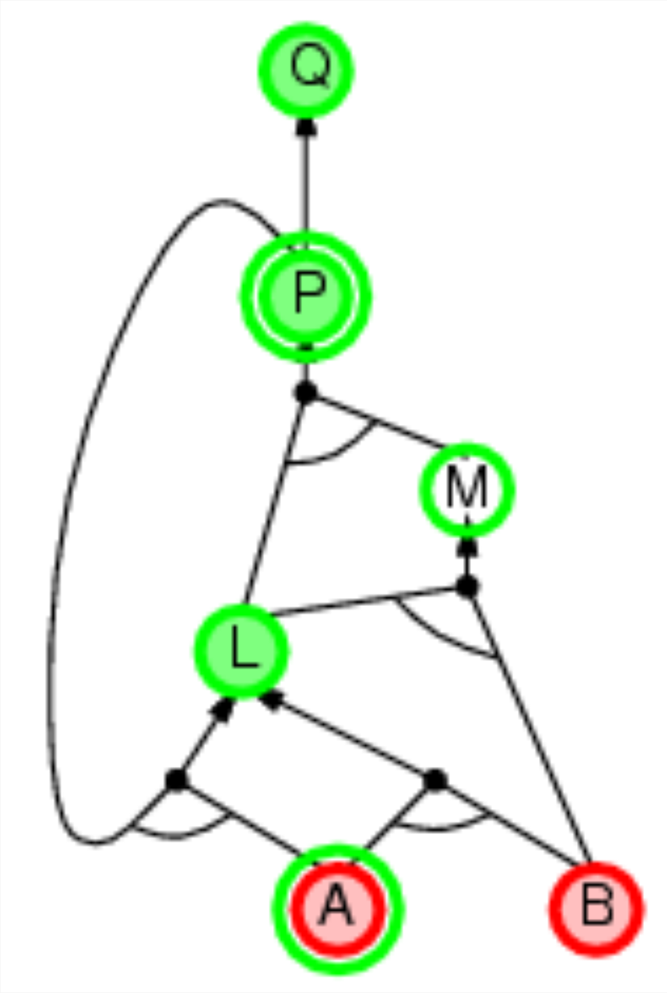
Backward chaining example



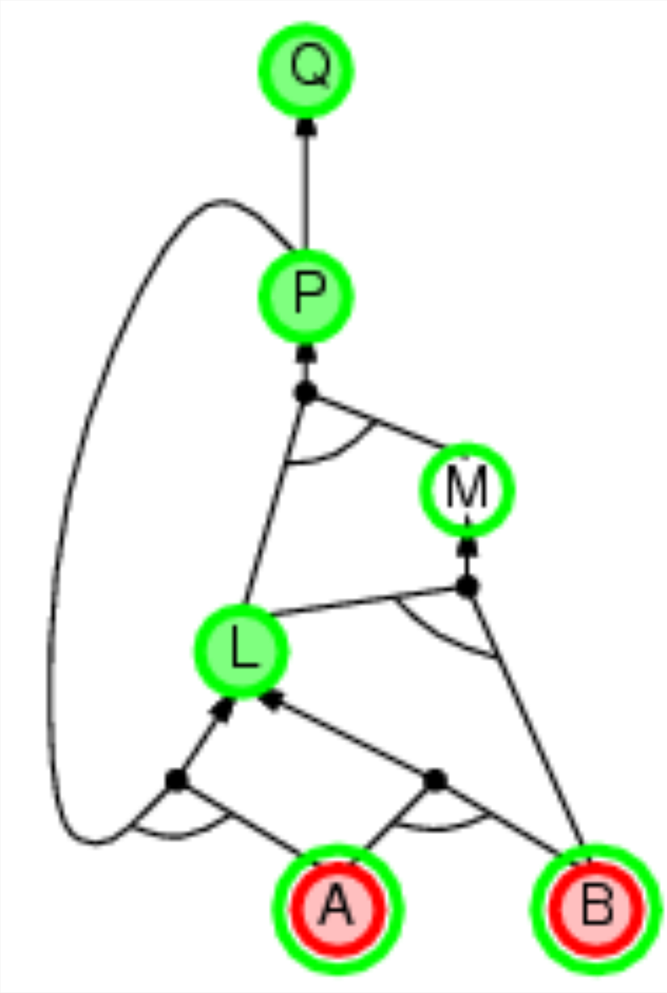
Backward chaining example



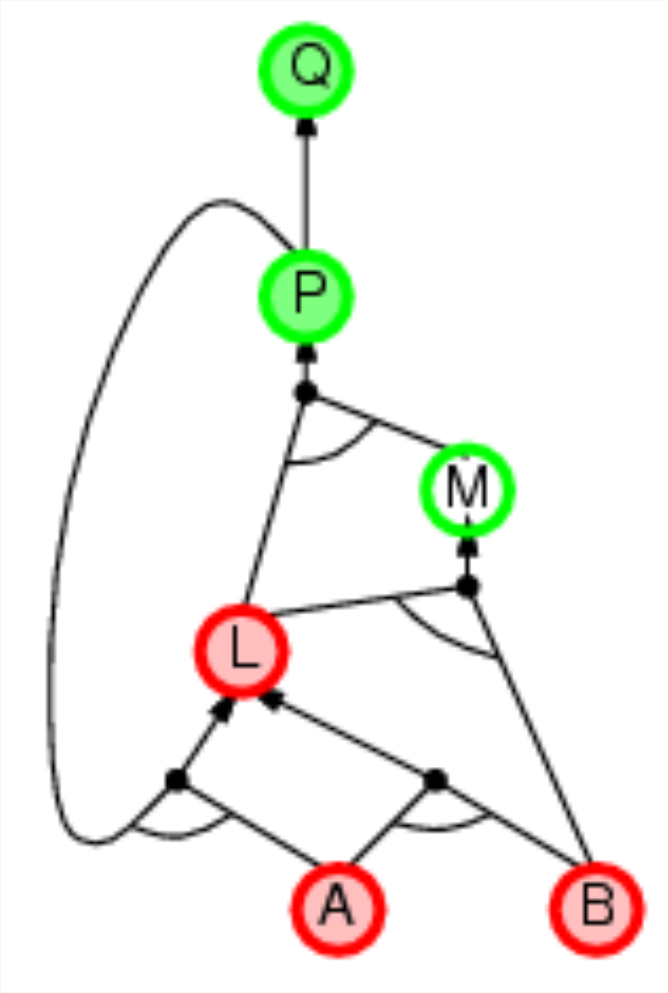
Backward chaining example



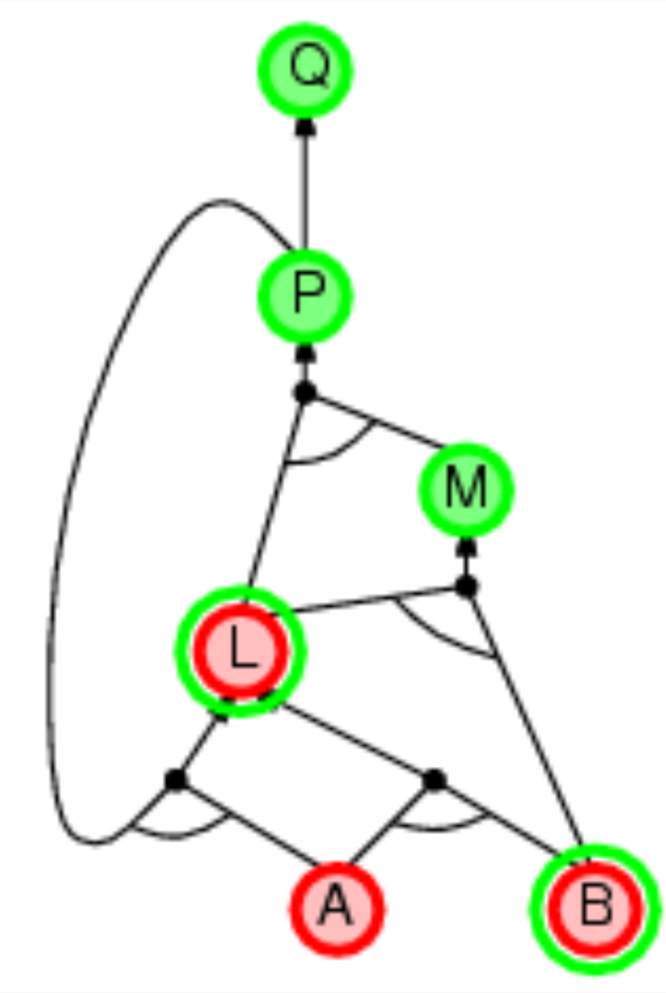
Backward chaining example



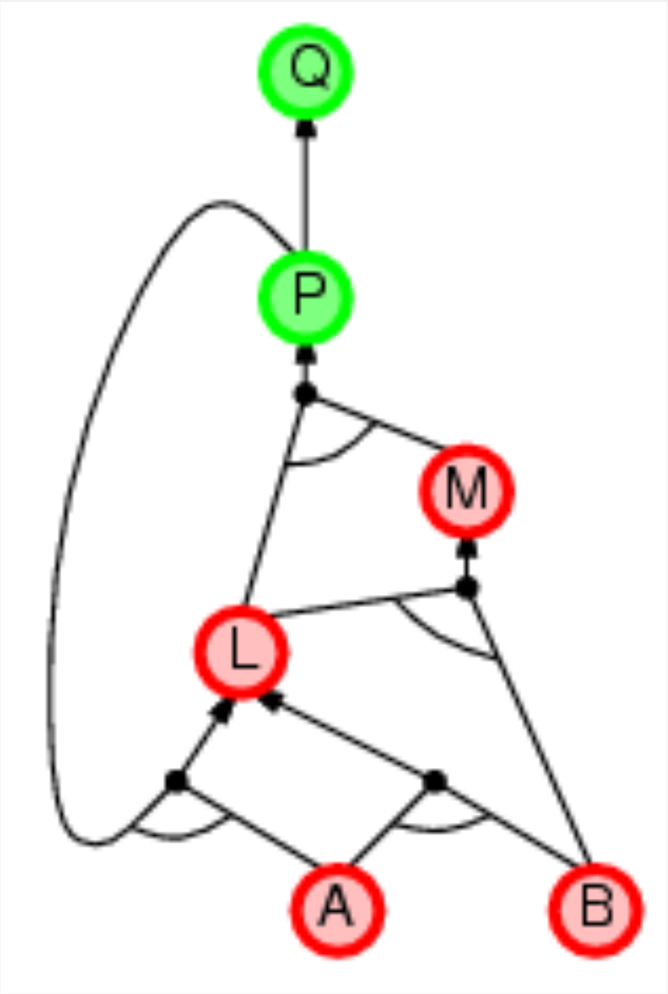
Backward chaining example



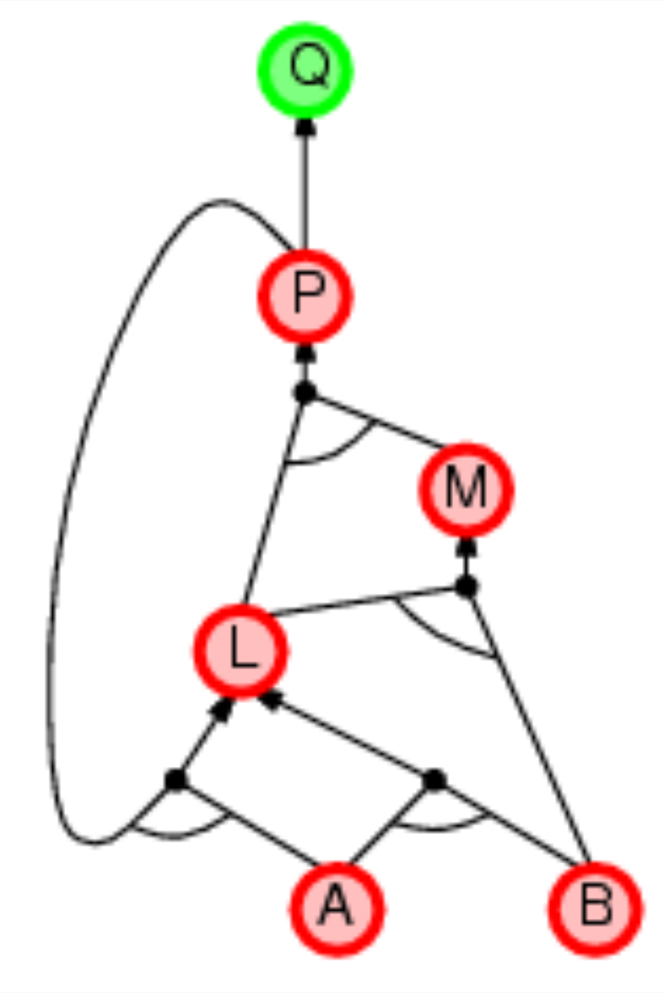
Backward chaining example



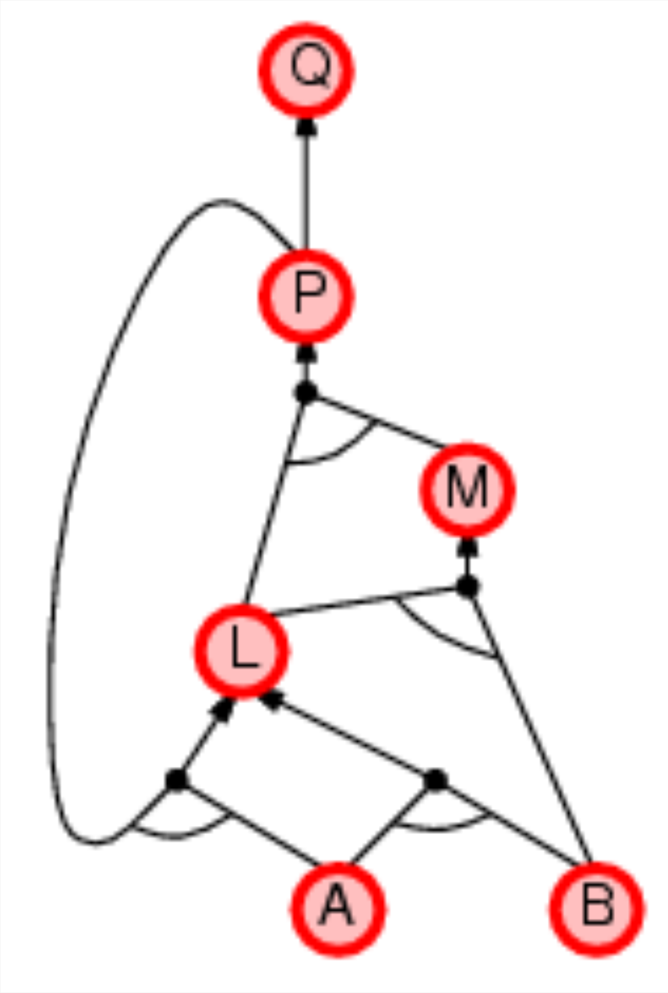
Backward chaining example



Backward chaining example



Backward chaining example



Soundness and Completeness of Forward Chaining

- An inference algorithm starts with the KB and applies applicable inference rules until the desired conclusion is reached
- An inference algorithm is sound if it uses a sound inference rule
- An inference algorithm is complete if
  - It uses a complete inference rule and
  - a complete search procedure
- Backward chaining using Modus Ponens is sound and complete for Horn knowledge bases (i.e., knowledge bases that contain only Horn clauses)

Forward vs. backward chaining

- FC is data-driven, automatic, unconscious processing,
  - Akin to day dreaming…
- May do lots of work that is irrelevant to the goal
- BC is goal-driven, appropriate for problem-solving
  - e.g., Where are my keys? How do I get into a PhD program?
- The run time of FC is linear in the size of the KB.
- The run time of BC can be, in practice, much less than linear in size of *KB*

Towards a sound and complete inference rule for propositional KB

We know that *Modus ponens* is sound $\{p \Rightarrow q, p\}\{q\}$

We observe that p does not have to be an atomic sentence

$$\frac{\begin{array}{c} a_1 \wedge a_2 \wedge a_3 \cdots a_{i-1} \wedge a_i \wedge a_{i+1} \cdots a_n \Rightarrow b \\ T \Rightarrow a_i \end{array}}{a_1 \wedge a_2 \wedge a_3 \cdots a_{i-1} \wedge a_{i+1} \cdots a_n \Rightarrow b}$$

q may be contingent on other facts

$$\frac{\begin{array}{c} a_1 \wedge a_2 \wedge a_3 \cdots a_{i-1} \wedge a_i \wedge a_{i+1} \cdots a_n \Rightarrow b \\ d_1 \wedge d_2 \cdots d_m \Rightarrow c \end{array}}{a_1 \wedge a_2 \wedge a_3 \cdots a_{i-1} \wedge a_{i+1} \cdots a_n \wedge d_1 \wedge d_2 \cdots d_m \Rightarrow b}$$

Assume $a_i = c$

Resolution principle

b, c do not have to be an atomic sentences

$$a_1 \wedge a_2 \wedge \cdots a_{i-1} \wedge a_i \wedge a_{i+1} \wedge \cdots a_n \Rightarrow b_1 \vee b_2 \vee \cdots \vee b_k$$
$$d_1 \wedge d_2 \cdots d_m \Rightarrow c \text{ (assume } a_i = c)$$

$$(a_1 \wedge a_2 \cdots a_{i-1} \wedge a_{i+1} \cdots \wedge a_n) \wedge (d_1 \wedge d_2 \cdots \wedge d_m) \Rightarrow b_1 \vee b_2 \cdots \vee b_k$$

As before, this rule can be shown to be sound.

$$a_1 \wedge a_2 \wedge \cdots a_{i-1} \wedge a_i \wedge a_{i+1} \wedge \cdots a_n \Rightarrow b_1 \vee b_2 \vee \cdots \vee b_k$$
$$d_1 \wedge d_2 \cdots d_m \Rightarrow c_1 \vee c_2 \vee \cdots c_{j-1} \vee c_j \vee c_{j+1} \vee \cdots c_l$$

$$(a_1 \wedge a_2 \cdots a_{i-1} \wedge a_{i+1} \cdots \wedge a_n) \wedge (d_1 \wedge \cdots \wedge d_m) \Rightarrow$$
$$(b_1 \vee b_2 \cdots \vee b_k) \vee (c_1 \vee c_2 \vee \cdots c_{j-1} \vee c_{j+1} \vee \cdots c_l) \text{ (assume } c_j = a_i)$$

Resolution is sound and complete for propositional *KB*

Soundness and completeness of resolution

- Resolution is sound and complete for propositional *KB*
- *Formal Proof Omitted*

Applying resolution

- Transform *KB* into an *equivalent* Conjunctive normal form (CNF)
  - Each sentence in *KB* is a disjunction of literals or their negations using known logical equivalences
  - *KB* is a conjunction of disjunctions
- Any propositional KB can be converted into CNF

Transformation to Clause Form (CNF)

Example:

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

1. Eliminate $\Rightarrow$

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

2. Reduce scope of $\neg$ using De Morgan's laws

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

3. Distribute $\vee$ over $\wedge$

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

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

KB in the form of a set of clauses or conjunction of disjunctions (CNF):

$\{\neg A \vee C , \neg A \vee D , B \vee C , B \vee D\}$

Proof

- The proof of a sentence α from a set of sentences *KB* is the derivation of α obtained through a series of applications of sound inference rules to *KB*
- $KB \models \alpha$ if and only if $\{KB, \neg\alpha\}$ is unsatisfiable

  (contradiction, $T \rightarrow F$, $\blacksquare$, empty sentence)
- Proving α from *KB* is equivalent to deriving a contradiction from *KB* augmented with the negation of α

Resolution by Refutation Algorithm

Add negation of goal to KB, derive empty clause (contradiction)

RESOLUTION-REFUTATION (KB,α)

clauses $\leftarrow$ set of clauses obtained from KB and $\neg\alpha$

new $\leftarrow$ {}

Repeat:

For each C, C' in clauses do
res $\leftarrow$ RESOLVE(C,C')
If res contains the empty clause then return yes
new $\leftarrow$ new U res

If new $\subseteq$ clauses then return no

clauses $\leftarrow$ clauses U new

Example: Applying Resolution

- Suppose that all classes at some university meet either Mon/Wed/Fri or Tue/Thu. The AI course meets at 2:30 PM in the afternoon, and Jane has volleyball practice Thursdays and Fridays at that time.
- Does Jane have a conflict with AI? Assume not.

$$1.\ MonWedFriAI230pm \vee TueThuAI230pm$$
$$2.\ TueThuAI230pm \wedge JaneBusyThu230pm \rightarrow JaneConflictAI$$
$$3.\ MonWedFriAI230pm \wedge JaneBusyFri230pm \rightarrow JaneConflictAI$$
$$4.\ JaneBusyThu230pm$$
$$5.\ JaneBusyFri230pm$$
$$6.\ \neg JaneConflictAI$$

Example: Applying Resolution

$$1.\ MonWedFriAI230pm \vee TueThuAI230pm$$

$$2.\ \neg TueThuAI230pm \vee \neg JaneBusyThu230pm \vee JaneConflictAI$$

$$3.\ \neg MonWedFriAI230pm \vee \neg JaneBusyFri230pm \vee JaneConflictAI$$

$$4.\ JaneBusyThu230pm$$

$$5.\ JaneBusyFri230pm$$

$$6.\ \neg JaneConflictAI$$

$Proof$ ——————————————————————

$$2,4.\ \neg TueThuAI230pm \vee JaneConflictAI$$

$$3,5.\ \neg MonWedFriAI230pm \vee JaneConflictAI$$

$$1,(2,4).\ MonWedFriAI230pm \vee JaneConflictAI$$

$$(3,5),(1,(2,4).\ JaneConflictAI \vee JaneConflictAI$$

$$6,((3,5),(1,(2,4)).\ JaneConflictAI$$

$$6,(6,((3,5),\left(1,(2,4)\right)).\ \blacksquare$$

Exercise: Prove *Car-OK* using resolution by refutation

KB:

$Battery\text{-}OK \wedge Bulbs\text{-}OK \Rightarrow Headlights\text{-}Work$

$Battery\text{-}OK \wedge Starter\text{-}OK \wedge \neg Empty\text{-}Gas\text{-}Tank \Rightarrow Engine\text{-}Starts$

$Engine\text{-}Starts \wedge \neg Flat\text{-}Tire \wedge Headlights\text{-}Work \Rightarrow Car\text{-}OK$

$Battery\text{-}OK \wedge Bulbs\text{-}OK$

$Starter\text{-}OK \wedge \neg Empty\text{-}Gas\text{-}Tank \wedge \neg Flat\text{-}Tire$

Ask

Car-OK?

Inference by model checking

Example KB

KB:

$A \lor B$

$\neg C \lor A$

S:

$A \land C$

A	B	C	KB	S
F	F	F	F	F
F	F	T	F	F
F	T	F	T	F
F	T	T	F	F
T	F	F	T	F
T	F	T	T	T
T	T	F	T	F
T	T	T	T	T

KB $\not\models$ S because KB is true but *S* is false

Propositional model checking

- Start with a finite propositional KB
- To find proof of a theorem (query "$KB \models s$")
- Enumerate the set of models $\boldsymbol{\mu}_{KB}$ in which KB holds
- Enumerate the models $\boldsymbol{\mu}_s$ in which *s* holds
- Check if $\boldsymbol{\mu}_{KB} \subseteq \boldsymbol{\mu}_s$
- Because *KB* is finite, there are only a finite number of models to enumerate
- Model checking is sound and complete
- Worst case complexity 2^N where N is the number of propositions
- Model checking can be made more efficient in practice
    - Reduce theorem proving to satisfiability checking

For *n* symbols, worst case time complexity is $O(2^n)$, space complexity is $O(n)$.

- In practice, much more efficient inference possible

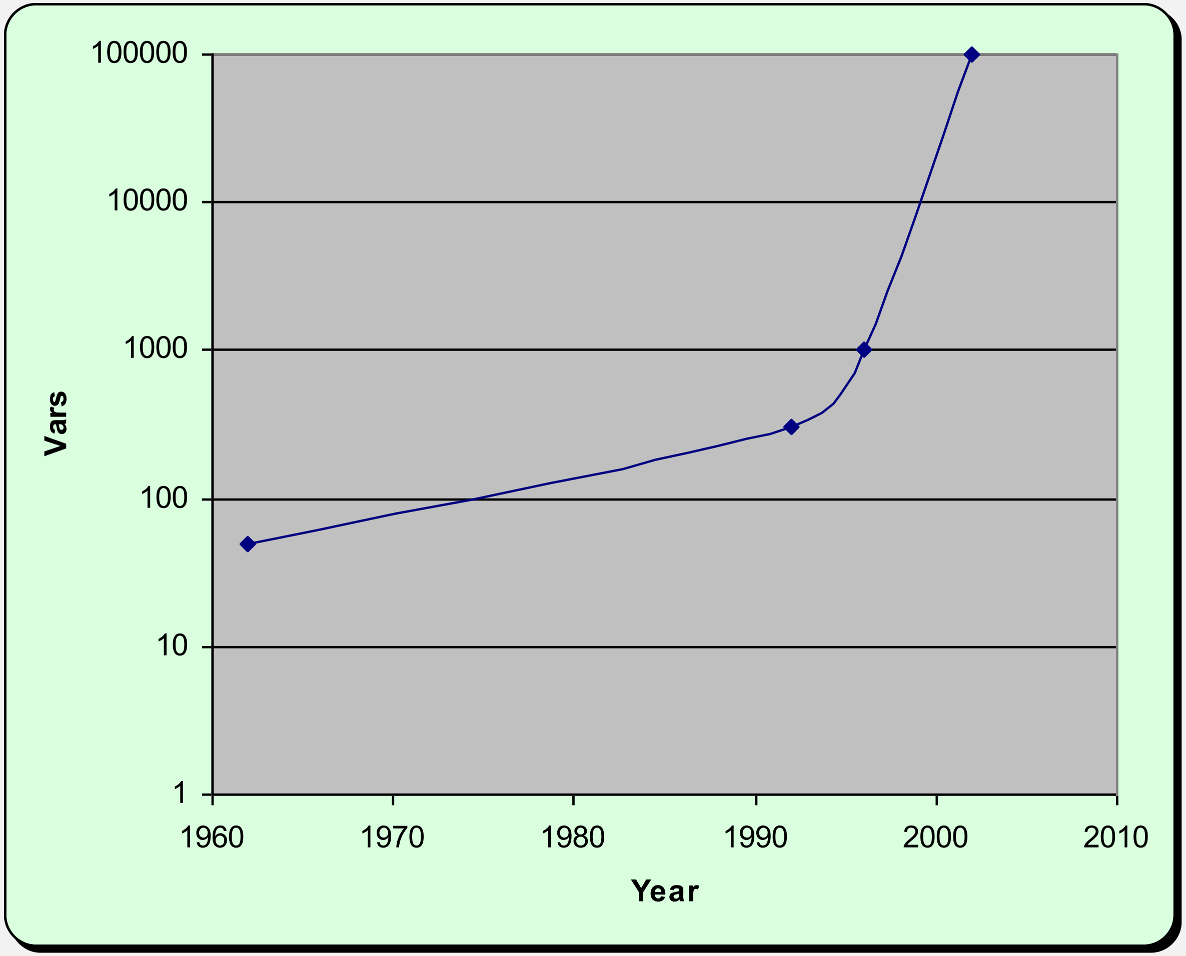
Propositional model checking

- Model checking can be made more efficient in practice
  - Reduce theorem proving to satisfiability checking
  - $KB \models s$ if and only if
  - $(KB \wedge \neg s)$ is unsatisfiable, or
  - $\neg(KB \wedge \neg s)$ is satisfiable
- There has been a great deal of progress in SAT solvers
  - Input: CNF a clause; Output: Yes/No
  - DPLL algorithm (Davis, Putnam, Logemann, Loveland) and its variants
    - Depth-first search through the space of possible models
    - Until an model in which the query holds is found or all models have been enumerated
  - Many optimizations possible

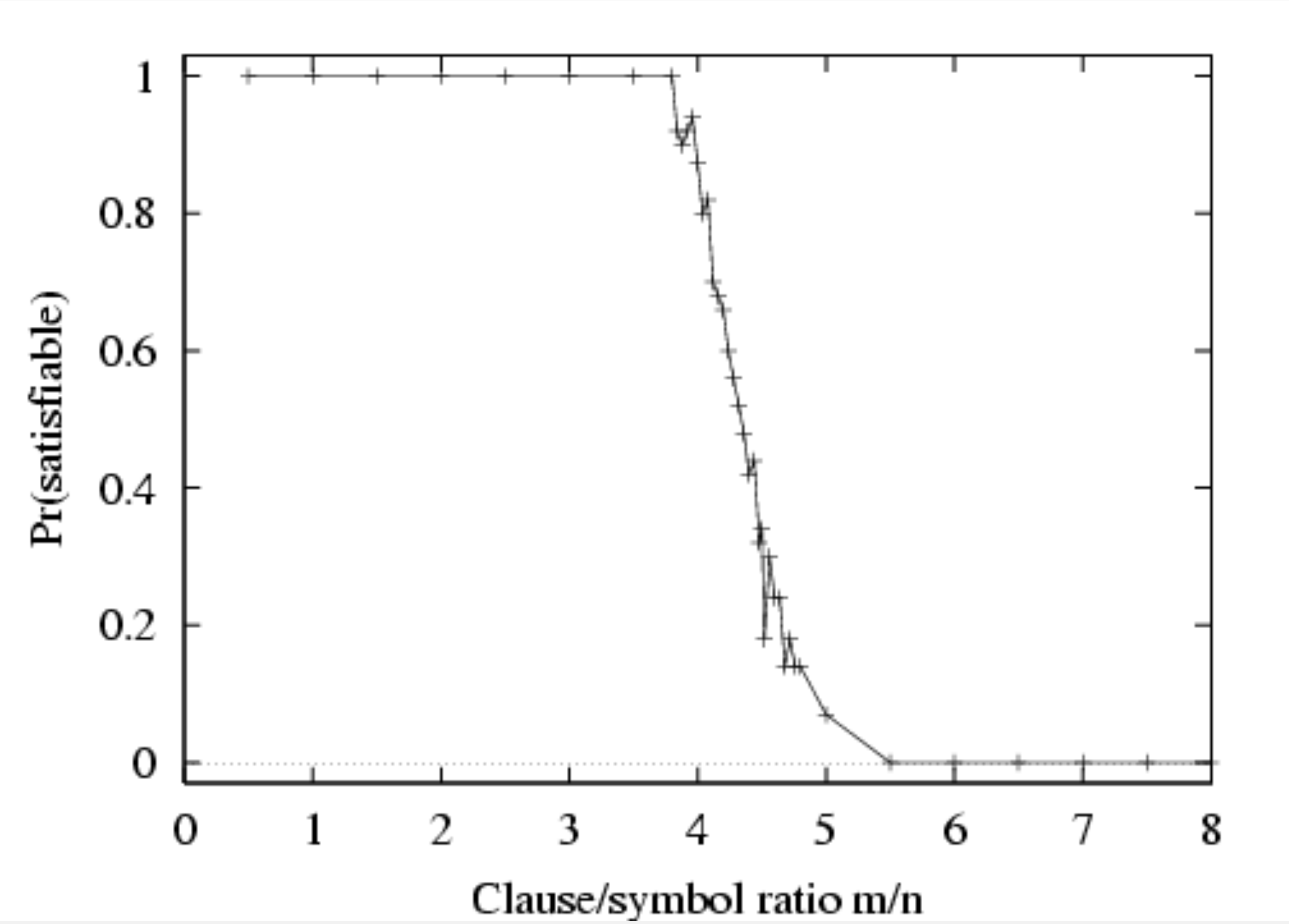
SAT as CSP

- SAT is about determining the satisfiability of a CNF formula
- SAT can be formulated as constraint satisfaction problem
  - Variables are propositions
  - Domain is T, F
  - Clauses are constraints

Progress in SAT solvers



Hard satisfiability problems



Modern SAT solvers

- SAT is NP-hard
- But most instances of SAT are not hard
- If *m* = number of clauses, *n* = number of propositions, hard SAT instances seem to cluster near *m/n* = 4.3 (critical point)
- Modern SAT solvers include many additional tricks
  - Conflict analysis
  - Constraint propagation
  - Advanced data structures
    - Binary decision diagrams (BDD)

Circuit-based implementation

- Intelligence without representation? (Brooks)
- Circuit-based agents
  - Reflex agents with internal state
  - Implemented using sequential circuits (logic gates plus registers)
  - Circuits evaluated in dataflow fashion
  - Inference linear in circuit size
  - Circuit size may be exponentially larger than the inference based agent's KB in some environments
- There are tradeoffs between inference-based and circuit-based agents
- Best of both worlds –
  - Inference based
  - Routinely used inferences compiled into circuits

First Order Predicate Logic

- Propositional logic
  - assumes the world can be represented using propositions
  - has limited expressive power
- First-order predicate logic (like natural language)
  - assumes the world contains
    - Objects:
      - people, flowers, houses, numbers, students,
    - Relations:
      - red, round, prime, brother of, bigger than, part of
    - Functions:
      - father of, best friend, plus, …
  - Allows one to talk about some or all of the objects

Ontological and Epistemological Commitments

	Ontological Commitments	Epistemological Commitments
Propositional Logic	Facts	True, False
First Order Predicate Logic	Facts, Objects, Relations, Functions	True, False
Probability Theory	Facts	Degree of belief $\in \{0,1\}$

Syntax of FOL: Basic elements

- Constants
  - Oksana, 2, Penn-State-University
- Predicates
  - Brother, Father, Teacher, Red
- Functions
  - Successor ( ), Plus
- Variables x, y, a, b,...
- Connectives $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- Equality =
- Quantifiers $\forall, \exists$

Atomic sentences

- Term
  - *function* (*term*$_1$,...,*term*$_n$),
    - e.g. house_of (*John*)
  - *constant*
    - e.g., *John*, 5
  - or *variable*

    e.g., x, y, z
- Predicates
  - E.g., *Brother(George, Jeb)*

Compound sentences

- Compound sentences are made from atomic sentences using connectives

$\neg S$,

$S_1 \wedge S_2$,

$S_1 \vee S_2$,

$S_1 \Rightarrow S_2$,

$S_1 \Leftrightarrow S_2$,

E.g. *Brother(George, Jeb)* $\Rightarrow$ *Sibling(George,Jeb)*

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains relations among objects
-
- Interpretation specifies referents for
  - constant symbols → objects
  - predicate symbols → relations
  - function symbols → functions

- An atomic sentence *predicate(term*$_1$*,...,term*$_n$*)* is true iff the objects referred to by *term*$_1$*,...,term*$_n$ are in the relation referred to by *predicate*

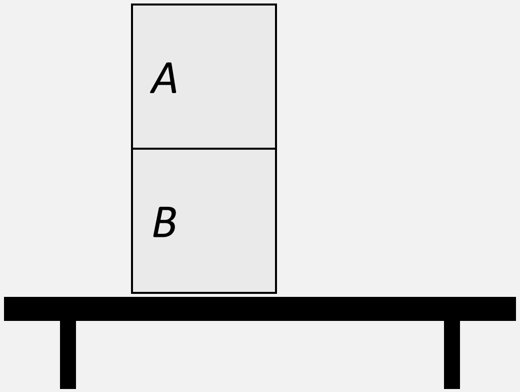
FOL Models - Example

- Object Constants: *A, B, Table*
- Relation Constant: *On*

Model

On (*A*, *B*)

On (*B*, *Table*)



Models for FOL

- In principle, we can enumerate the models for a given KB vocabulary
- Computing entailment by enumerating the models will not be easy !!

For each number of domain elements n from 1 to ∞
    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 ...

Quantifiers

Quantifiers

- Allow us to express properties of collections of objects instead of enumerating objects by name
  - Universal: “for all” $\forall$
  - Existential: “there exists” $\exists$

$$\forall x \; Human(x) \Rightarrow Mortal(x)$$

$$\forall z \; Petdog(z) \Rightarrow \exists y \; Human(y) \wedge Caresfor(y,z)$$

Universal quantification

- $\forall$<*variables*> <*sentence*>
- Everyone at PSU is smart: $$\forall x\ At(x, PSU) \Rightarrow Smart(x)$$
- $\forall$x *P*(*x*) is true in a model *m* iff *P* is true with *x* instantiated to each possible object in the world
- Roughly speaking, $\forall$x *P*(*x*) is equivalent to the conjunction of instantiations of *P*

$$\left(At(Matt, PSU) \Rightarrow Smart(Matt)\right) \wedge$$
$$\left(At(Oksana, PSU) \Rightarrow Smart(Oksana)\right) \wedge$$
$$\left(At(Fido, PSU) \Rightarrow Smart(Fido)\right) \wedge$$
.....

A common mistake to avoid

- A universally quantifier is also equivalent to a set of implications over all objects
- Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall x\ At(x, PSU) \wedge Smart(x)$$

Means

- "Everyone is at PSU and everyone is smart"

as opposed to

- "Everyone at PSU is smart"

Existential quantification

∃<*variables*> <*sentence*>

Someone at PSU is smart: $\exists x \; At(x, PSU) \wedge Smart(x)$

∃*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* (*x*)

$$\left(At(Matt, PSU) \wedge Smart(Matt)\right) \vee$$

$$\left(At(Oksana, PSU) \wedge Smart(Oksana)\right) \vee$$

$$\left(At(Fido, PSU) \wedge Smart(Fido)\right) \vee$$

.....

Another common mistake to avoid

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

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

- The above assertion is true even if there is someone that is smart who is not at PSU!
- What we wanted to assert was instead that there is someone at PSU who is smart!

Properties of quantifiers

$\forall x\ \forall y$ is the same as $\forall y\ \forall x$

$\exists x\ \exists y$ is the same as $\exists y\ \exists x$

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

$\exists x\ \forall y$ Loves(x,y)

- "There is someone who loves everyone"

$\forall y\ \exists x$ Loves(x,y)

- "Everyone is loved by someone"

Quantifier Duality

Duality: "Everyone dislikes Parsnips" $\equiv$
"there is no one who likes Parsnips"

$$\forall x \neg \texttt{Likes}(x, \texttt{Parsnips}) \quad \equiv \quad \neg \exists x \texttt{Likes}(x, \texttt{Parsnips})$$

De Morgan Rules:

$$\begin{array}{llll}
\forall x\, \neg P & \equiv \neg \exists x\, P & \neg P \wedge \neg Q & \equiv \neg (P \vee Q) \\
\neg \forall x\, P & \equiv \exists x\, \neg P & \neg (P \wedge Q) & \equiv \neg P \vee \neg Q \\
\forall x\, P & \equiv \neg \exists x\, \neg P & P \wedge Q & \equiv \neg (\neg P \vee \neg Q) \\
\neg \forall x\, \neg P & \equiv \exists x\, P & \neg (\neg P \wedge \neg Q) & \equiv P \vee Q
\end{array}$$

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., definition of *Sibling* in terms of *Parent*:

$$\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \wedge \exists m,f\ \neg (m = f) \wedge \mathrm{Parent}(m,x) \wedge \mathrm{Parent}(f,x) \wedge \mathrm{Parent}(m,y) \wedge \mathrm{Parent}(f,y)]$$

Interacting with FOL KBs

- Given a sentence S and a substitution α,
  - $S\alpha$ denotes the result of plugging α into S; e.g.,

    S = Smarter(x,y)

    α = {x/Hillary,y/Bill}

    $S\alpha$ = Smarter(Hillary,Bill)

- `Ask`(KB,S) returns some/all α such that KB |= Sα.

Using FOL

A KB about kinship

- Brothers are siblings

  $\forall$x,y *Brother(x,y)* $\rightarrow$ *Sibling(x,y)*

- One's mother is one's female parent

  $\forall$m,c *Mother(c)* = m $\Leftrightarrow$ *(Female(m)* $\wedge$ *Parent(m,c))*

- "Sibling" is symmetric

  $\forall$x,y *Sibling(x,y)* $\Leftrightarrow$ *Sibling(y,x)*

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

  $\forall$x,y *FirstCousin(x,y)* $\Leftrightarrow$ $\exists$*u,v Parent(u,x)* $\wedge$ *Sibling(v,u)* $\wedge$ *Parent(v,y)*

FOL Examples

- Predicates:

$$Purple(x),\ Mushroom(x),\ Poisonous(x),\ Equal(x,y)$$

- All purple mushrooms are poisonous

$$\forall x\ Purple(x) \wedge Mushroom(x) \Rightarrow Poisonous(x)$$

- Some purple mushrooms are poisonous

$$\exists x\ Purple(x) \wedge Mushroom(x) \wedge Poisonous(x)$$

- No purple mushrooms are poisonous

$$\forall x\ Purple(x) \wedge Mushroom(x) \Rightarrow \neg Poisonous(x)$$

- There is exactly one mushroom

$$\exists x\ Mushroom(x) \wedge \forall y\ Mushroom(y) \Rightarrow Equal(x,y)$$

Knowledge engineering in FOL

- Identify the task (the purpose for which a KB will be used)
- Assemble the relevant knowledge (knowledge acquisition)
- Decide on a vocabulary of predicates, functions, and constants
  - Translate domain-level knowledge into logic-level names
- Encode general knowledge about the domain
  - define axioms
- Encode a description of the specific problem instance
- Pose queries to the inference procedure and get answers
- Debug the knowledge base

Summary

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

- Increased expressive power

Inference in FOL

- Adapt techniques from propositional logic
- Adapt techniques developed for propositional inference
  - How to eliminate universal quantifiers?
    - Instantiate variables
  - How to convert existential quantifiers?
    - Skolemization

Universal instantiation (UI)

Example

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$:

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$

$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

- Every instantiation of a universally quantified sentence is entailed by it
- $\forall v\ \alpha$, entails instantiations obtained by substituting v with ground terms:
- Subst({v/g}, α) denotes instantiation of α by substituting variable *v* with term *g*

  (Subst (x|y) = substitution of y by x)

Existential instantiation (EI)

- E.g., $\exists x\ House(x) \wedge \text{Ownedby}(x,John)$
- There exists a *house owned by John*
- Let us name the house whose existence is asserted by the above, *John-Villa*
- Now, *John-Villa* is a *house*, and it is *owned by John*

$$House(John\text{-}Villa) \wedge Ownedby(John\text{-}Villa,John)$$

John-Villa, a unique name that refers to the house obtained by eliminating the existential quantifier above is called a Skolem constant

Skolemization Examples

Eg: "Everyone has a heart."

$$\boxed{\forall X\, \mathbf{person}(X) \Rightarrow \exists Y\, \mathbf{heart}(Y) \wedge \mathbf{has}(X, Y)}$$

Incorrect: $\forall X\, \mathtt{Person}(X) \Rightarrow \mathtt{heart}(\mathtt{H}_1) \wedge \mathtt{has}(\mathtt{x}, \mathtt{H}_1)$

?everyone has the SAME heart $\mathtt{H}_1$?

Correct: $\boxed{\forall X \mathtt{person}(X) \Rightarrow \mathtt{heart}(\mathtt{h}(X)) \wedge \mathtt{has}(X, \mathtt{h}(X))}$

where $\mathtt{h}$ is a new symbol ("Skolem function")

- Skolem function arguments:
  all enclosing universally quantified variables

Skolemization

- **Skolemizing** procedure (to remove existentials)

For each existential X, let $Y_1, \ldots, Y_m$ be the universally quantified variables that are quantified to the LEFT of X's "$\exists X$". Generate *new* function symbol, g_X, of m variables. Replace each X with $g_X(Y_1, \ldots, Y_m)$. (Write $g_X()$ as g_X.)

$$\forall Y \exists X \ \phi(X) \wedge \rho(Y) \quad \mapsto \quad \forall Y \ \phi(\boxed{g_X(Y)}) \wedge \rho(Y)$$

$$\exists X \forall Y \ \phi(X) \wedge \rho(Y) \quad \mapsto \quad \forall Y \ \phi(\boxed{g_X}) \wedge \rho(Y)$$

Skolemization Theorem

If $T_1 = \left\{ \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \ldots \\ \exists X \, \forall Y \;\; \varphi(X, Y) \\ \ldots \end{array} \right\}$ is consistent

then $s(T_1) = \left\{ \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \ldots \\ \forall Y \;\; \varphi(c_1, Y) \\ \ldots \end{array} \right\}$ is consistent.

. . . if $s(T)$ is inconsistent, then T is inconsistent . . .

Universal versus Existential Instantiation

- Universal Instantiation
    - can be applied many times to *add* new sentences;
    - the new KB is logically equivalent to the old
- Existential Instantiation (Skolemization)
    - can be applied once to eliminate each existential quantifier;
    - the resulting existential quantifier free KB is not equivalent to the old
    - The new KB is satisfiable if the old KB was satisfiable

Reduction to propositional inference

- Suppose the KB contains just the following:

$$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$
$$King(\mathrm{John})$$
$$Greedy(\mathrm{John})$$
$$Brother(\mathrm{Richard},\mathrm{John})$$

- After universal instantiation we get a variable-free, quantifier-free KB – a propositionalized KB

$$King(\mathrm{John}) \wedge Greedy(\mathrm{John}) \Rightarrow Evil(\mathrm{John})$$
$$King(\mathrm{Richard}) \wedge Greedy(\mathrm{Richard}) \Rightarrow Evil(\mathrm{Richard})$$
$$King(\mathrm{John})$$
$$Greedy(\mathrm{John})$$
$$Brother(\mathrm{Richard},\mathrm{John})$$

Reduction of FOL inference to PL inference

- *CLAIM*: A (ground) instantiation of a sentence is entailed by a new KB iff entailed by the original KB.
- *CLAIM*: Every FOL KB can be propositionalized so as to preserve entailment
- *IDEA*: propositionalize KB and query, apply resolution, return result

- *PROBLEM*: when function symbols are present, it is possible to generate infinitely many ground terms:

  e.g., *Father*(*Father*(*Father*(*John*)))

Reduction of FOL inference to PL inference

- THEOREM: Herbrand (1930).

  If a sentence α is entailed by a FOL KB, it is entailed by a finite subset of the propositionalized KB

- *IDEA*: For n = 0 to ∞ do
  - create a propositional KB by instantiating with depth-n terms
  - see if α is entailed by this KB

Reduction of FOL inference to PL inference

- THEOREM: Turing (1936), Church (1936) Entailment for FOL is semi decidable
  - algorithms exist that affirm every sentence that is entailed by the KB
    - Prove a theorem that in fact follows from the axioms
  - No algorithm exists that also says no to sentence that is not entailed by the KB
    - Algorithm may not terminate

Problems with propositionalization

Given:

$$\forall x\; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$
$$King(\text{John})$$
$$\forall y\; Greedy(y)$$
$$Brother(\text{Richard},\text{John})$$

- 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!
  - Can we avoid unnecessary instantiation of unneeded facts?

Lifting and Unification

- Instead of translating the knowledge base to PL, we can redefine the inference rules into FOL.
  - *Lifting:* only make those substitutions that are needed to allow particular inferences to proceed
  - *Unification:* identify the relevant substitutions

Unification

- We can get the inference immediately if we can find a substitution α such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

Substituting x by John and y by John works

α = {x/John,y/John}

Unification

- To unify *Knows(John,x)* and *Knows(y,z)*,

  α = {y/John, x/z } or α = {y/John, x/John, z/John}

- The first unifier is more general than the second.

- There is a single most general unifier (MGU) that is unique up to renaming of variables.

  MGU = { y/John, x/z }

Unification Examples

$$p = P(x, f(y), B)$$
$$q = P(z, f(w), B)$$

$$\alpha = \{z|x, w|y\}$$

$$p = P(x, f(y), B)$$
$$q = Q(z, f(w), B)$$

p and *q* not unifiable (why?)

$$p = P(x, B)$$
$$q = P(f(x), B)$$

p and *q* not unifiable (why?)

$$p = P(y, B)$$
$$q = P(f(x), B)$$

$$\alpha = \{y|f(x)\}$$

Unification examples

$$p = P(g(x), B)$$
$$q = P(f(x), B)$$

p and *q* are not unifiable (Why?)

$$p = P(x, A)$$
$$q = P(y, B)$$

p and *q* are not unifiable (Why?)

$$p = P(x, y, z, f(w))$$
$$q = P(A, y, z, f(u))$$

$$\alpha = \{x|A, w|u\}$$

The unification algorithm

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
    inputs: x, a variable, constant, list, or compound
            y, a variable, constant, list, or compound
            θ, the substitution built up so far

    if θ = failure then return failure
    else if x = y then return θ
    else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
    else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
    else if COMPOUND?(x) and COMPOUND?(y) then
        return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
    else if LIST?(x) and LIST?(y) then
        return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
    else return failure
```

The unification algorithm

function UNIFY-VAR(var, x, θ) **returns** a substitution
  inputs: var, a variable
    x, any expression
    θ, the substitution built up so far

  if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ

Applying Substitution

- Given $\begin{cases} t & \texttt{-- a term} \\ \sigma & \texttt{-- a substitution} \end{cases}$

  "$t\sigma$" is the term resulting from applying substitution σ to term t.

- Small Examples:

  $$\texttt{X\{X/a\} = a}$$

  $$\texttt{f(X)\{X/a\} = f(a)}$$

Examples

- Example: Using $t = \texttt{f(a, h(Y,b), X)}$

$$\texttt{f(a, h(Y,b), X)}\{\texttt{X/b}\} = \texttt{f(a, h(Y,b), b)}$$

$$\texttt{f(a, h(Y,b), X)}\{\texttt{X/b Y/f(Z)}\} = \texttt{f(a, h(f(Z),b), b)}$$

$$\texttt{f(a, h(Y,b), X)}\{\texttt{X/Z Y/f(Z,a)}\} = \texttt{f(a, h(f(Z,a),b), Z)}$$

$$\texttt{f(a, h(Y,b), X)}\{\texttt{W/Z}\} = \texttt{f(a, h(Y,b), X)}$$

- σ need not include all variables in t;
  σ can include variables not in t.

Most General Unifier

- σ is a *mgu* for t_1 and t_2 iff
  - σ unifies t_1 and t_2, and
  - $\forall\ \mu$: unifier of t_1 and t_2,
    $\exists$ subsitution, θ, s.t. $\sigma \circ \theta = \mu$.
    (Ie, for all terms t, $t\mu = (t\sigma)\theta$.)

MGU example

- Example: $\sigma = \{X/Y\}$ is mgu for f(X) and f(Y).
  Consider unifier $\mu = \{X/a \quad Y/a\}$.
  Use substitution $\theta = \{Y/a\}$:

$$\begin{aligned} f(X)\mu &= f(X)\{X/a\ Y/a\} \\ &= f(a) \end{aligned}$$

$$\begin{aligned} f(X)[\sigma \circ \theta] &= (f(X)\sigma) \quad \theta \\ &= (f(X)\{X/Y\})\theta \\ &= f(Y)\{Y/a\} \\ &= f(a) \end{aligned}$$

Similarly, $f(Y)\mu = f(a) = f(Y)[\sigma \circ \theta]$

(μ is NOT a mgu, as $\neg\exists\theta'$ s.t. $\mu \circ \theta' = \sigma$!)

Notes on MGU

- If two terms are unifiable, then there exists a MGU
- There can be more than one MGU, but they differ only in variable names
- Not every unifier is a MGU
- A MGU uses constants only as necessary

FOL Modus Ponens Example

$$\frac{\begin{array}{l} \textit{All Men are Mortal} \\ \textit{Socrates is a Man} \end{array}}{\textit{Socrates is mortal}}$$

$$\frac{\begin{array}{l} \forall x\ Man(x) \Rightarrow Mortal(x) \\ Man(Socrates) \end{array}}{Mortal(Socrates)}$$

$$MGU = \{Socrates \mid x\}$$

Generalized Modus Ponens (GMP)

$$\frac{p_1 \wedge p_2 \wedge \ldots p_n \Rightarrow q \qquad p_1' \wedge p_2' \wedge \ldots p_n'}{q\theta}$$

∀x King(x) ∧ Greedy(x) ⇒ Evil(x)
King(John) ∧Greedy(John)

$$\text{where } (p_1 \wedge p_2 \wedge \ldots p_n)\theta = p_1' \wedge p_2' \wedge \ldots p_n'$$

θ is {x/John,y/John}

q is *Evil*(*x*)

qθ 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 $p_1', \ldots, p_n', (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \mid= q\theta$

provided that $p_i'\theta = p_i\theta$ for all i

- *LEMMA*: For any sentence p, we have $p \mid= p\theta$ by UI

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

Generalized resolution principle

$$\frac{\begin{array}{c} p_1 \vee p_2 \vee \ldots p_n \\ p_1^{'} \vee p_2^{'} \vee p_m^{'} \end{array}}{\left(p_1 \vee ..p_{i-1} \vee p_{i+1} .. \vee p_n \vee p_1^{'} .. \vee p_{j-1}^{'} \vee p_{j+1}^{'} .. \vee p_m^{'}\right)\theta}$$

$$\text{where } \left(p_i\right)\theta = \neg p_j^{'}$$

Resolution Rule in FOL

- Example:
  - father(John, Kim),
  - $\forall x \, \forall y \, \neg father(x,y) \lor parent(x, y)$
  - parent(John, Kim)?
- Resolution with propositional logic:
  - Find complementary literals
- Resolution with FOL
  - Create complementary literals with substitution

Conversion to CNF

0: $\forall x\ [(\forall y\ P(x, y)) \Rightarrow \neg\forall y\ Q(x, y) \Rightarrow R(x, y)]$

1: **Eliminate implication, iff, . . .**
$\forall x\ [\neg(\forall y\ P(x, y)) \vee [\neg\forall y\ [\neg Q(x, y) \vee R(x, y)]]]$

2: **Move $\neg$ inwards**
$\forall x\ [(\exists y\ \neg P(x, y)) \vee [\exists y\ Q(x, y) \wedge \neg R(x, y)]]$

3: **Standarize variables**
$\forall x\ [(\exists y\ \neg P(x, y)) \vee [\exists z\ Q(x, z) \wedge \neg R(x, z)]]$

4: **Move quantifiers left**
$\forall x\ \exists y\ \exists z\ [\neg P(x, y)) \vee [Q(x, z) \wedge \neg R(x, z)]]$

Conversion to CNF

5: Skolemize (remove existentials); Drop $\forall$s

$$\neg P(\mathtt{x},\ \mathtt{F}_1(\mathtt{x})) \vee [\, Q(\mathtt{x},\ \mathtt{F}_2(\mathtt{x})) \wedge \neg R(\mathtt{x},\ \mathtt{F}_2(\mathtt{x}))\,]$$

6: Distribute $\wedge$ over $\vee$

$$[\neg P(\mathtt{x},\ \mathtt{F}_1(\mathtt{x})) \vee Q(\mathtt{x},\ \mathtt{F}_2(\mathtt{x}))]$$
$$\wedge [\neg P(\mathtt{x},\ \mathtt{F}_1(\mathtt{x})) \vee \neg R(\mathtt{x},\ \mathtt{F}_2(\mathtt{x}))]$$

7 : Change to SET notation

$$\left\{ \begin{array}{c} \neg P(\mathtt{x},\ \mathtt{F}_1(\mathtt{x}))\ \vee Q(\mathtt{x},\ \mathtt{F}_2(\mathtt{x})), \\ \neg P(\mathtt{x},\ \mathtt{F}_1(\mathtt{x}))\ \vee\ \neg R(\mathtt{x},\ \mathtt{F}_2(\mathtt{x}))\ \} \end{array} \right\}$$

8 : Make variables unique

$$\left\{ \begin{array}{c} \neg P(\mathtt{x}_1,\ \mathtt{F}_1(\mathtt{x}_1))\ \vee Q(\mathtt{x}_1,\ \mathtt{F}_2(\mathtt{x}_1)), \\ \neg P(\mathtt{x}_2,\ \mathtt{F}_1(\mathtt{x}_2))\ \vee\ \neg R(\mathtt{x}_2,\ \mathtt{F}_2(\mathtt{x}_2))\ \} \end{array} \right\}$$

Example

- Jack owns a dog
- Every dog owner is an animal lover
- No animal lover kills an animal
- Either Jack or Curiosity killed the cat (named Tuna)
- Did Curiosity kill the cat?

Example

- The law says that it is a crime for an American to sell weapons to hostile nations
- Missiles are weapons
- The country Nono, an enemy of America, has some missiles
- All of Nono's missiles were sold to it by Colonel West
- Colonel West is an American

- Prove that Col. West is a criminal

Example knowledge base

… 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

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

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 …

$American(West)$

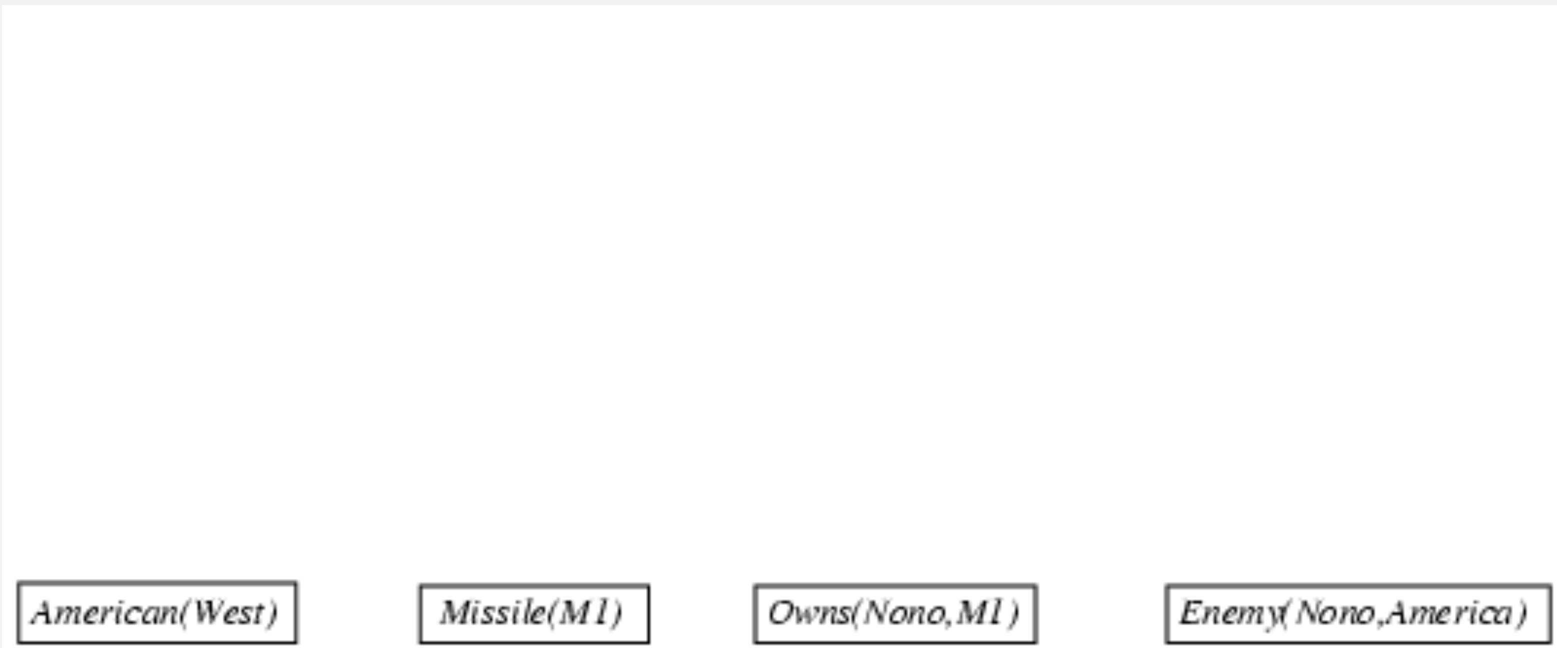
The country Nono, an enemy of America …

$Enemy(Nono,America)$

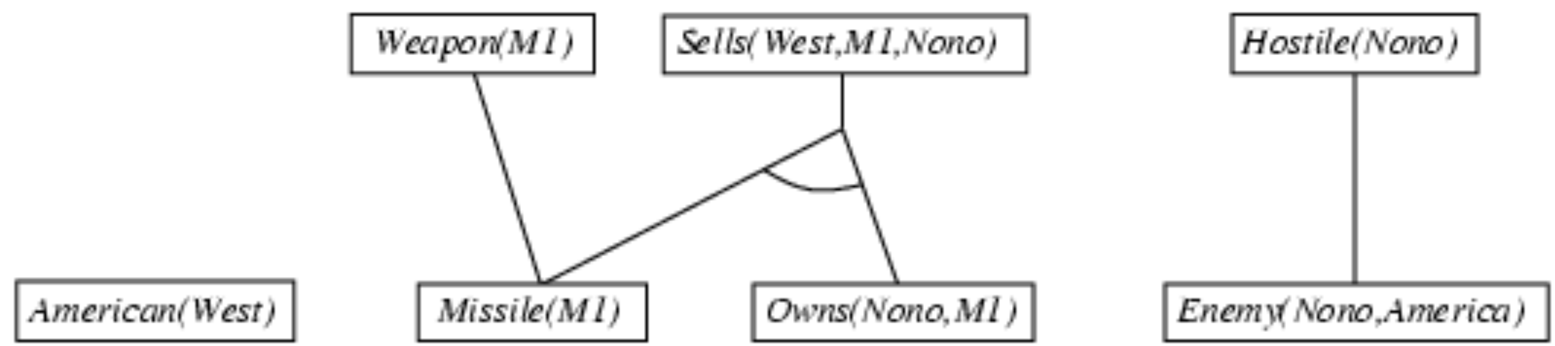
Inference

- Forward chaining – typically used in deductive databases e.g., Datalog (no functions, horn clauses)
- Backward chaining –typically used in logic programming

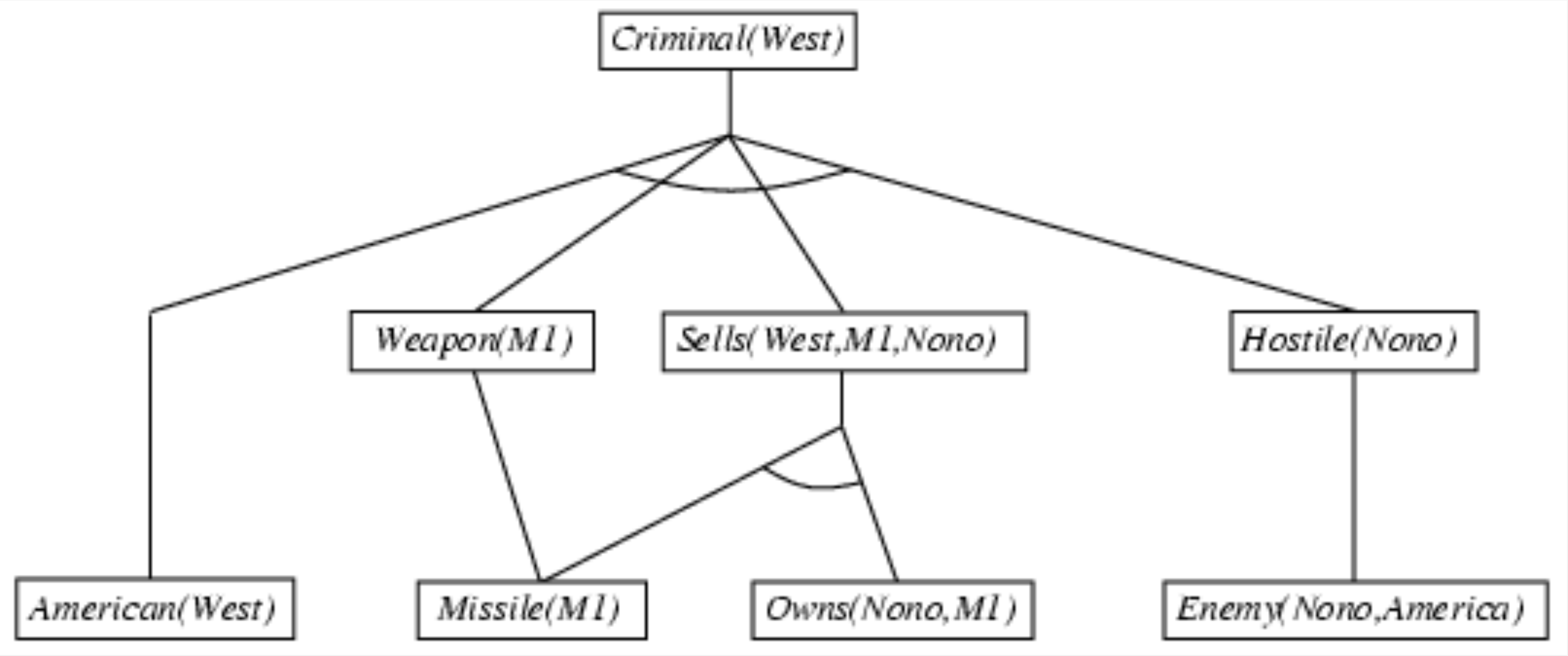
Forward chaining example



Forward chaining example



Forward chaining example



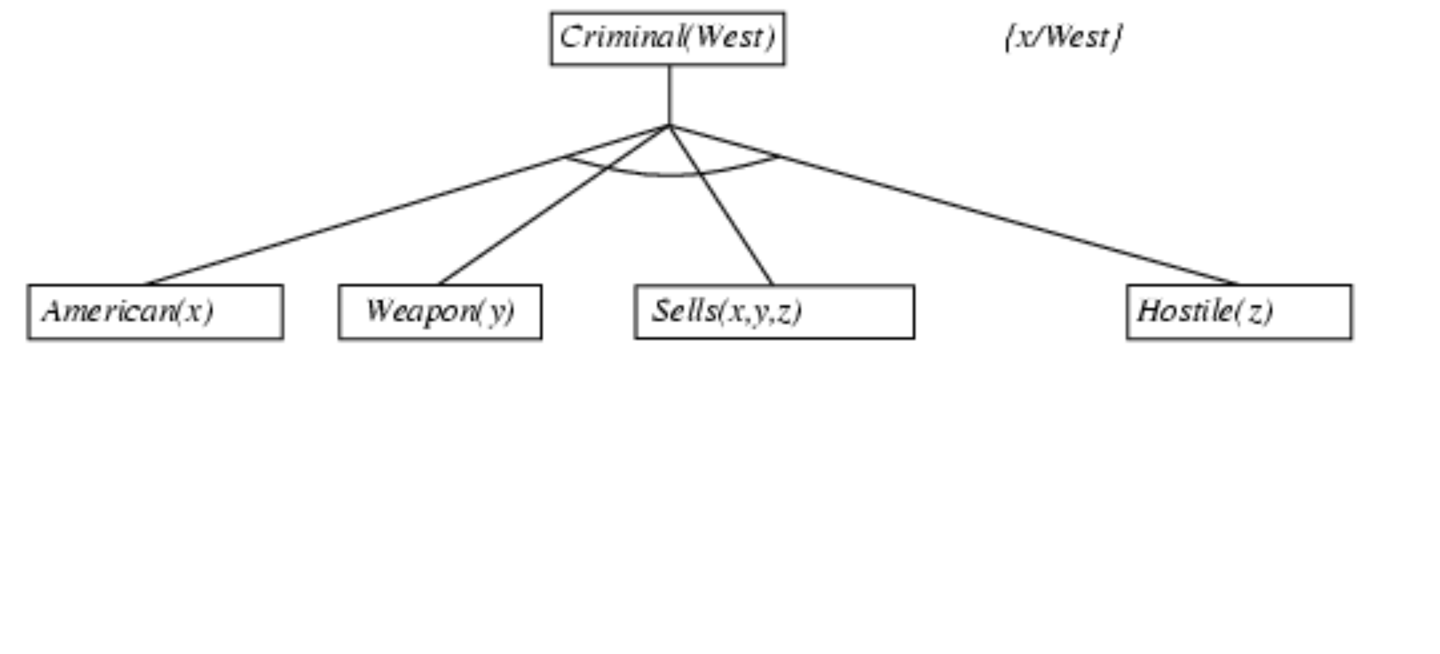
Properties of forward chaining

- Sound and complete for first-order definite clauses
- *Datalog* = first-order definite clauses + *no functions* (e.g. crime KB)
    - FC terminates for Datalog in finite number of iterations
- May not terminate in general DF clauses with functions if α is not entailed
    - This is unavoidable: entailment with definite clauses is semidecidable

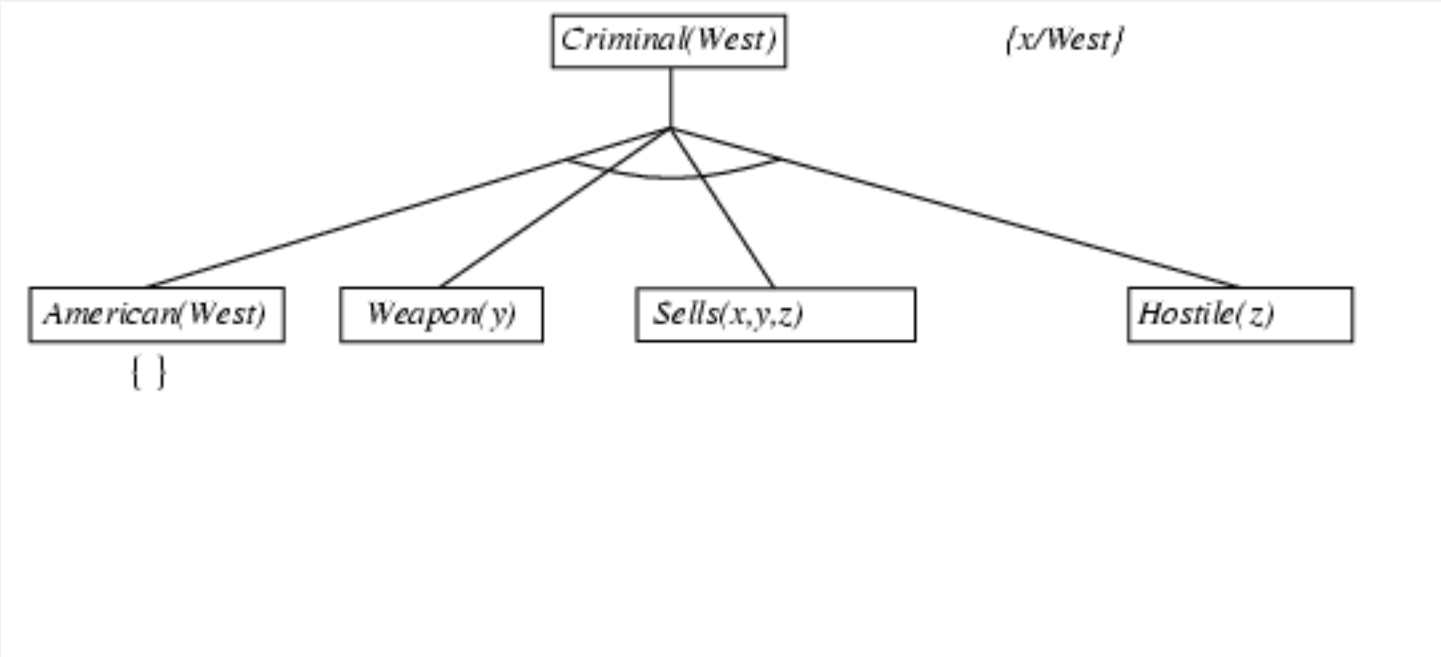
Backward chaining example

$\boxed{Criminal(West)}$

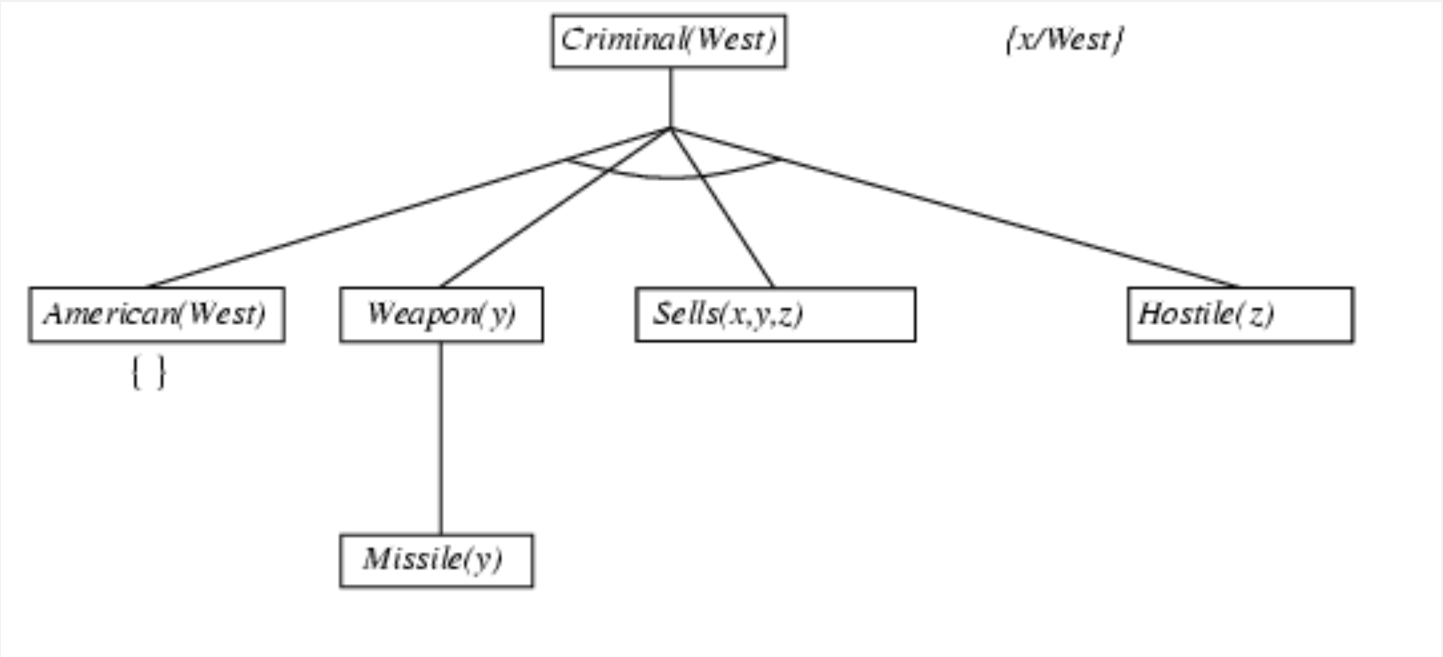
Backward chaining example



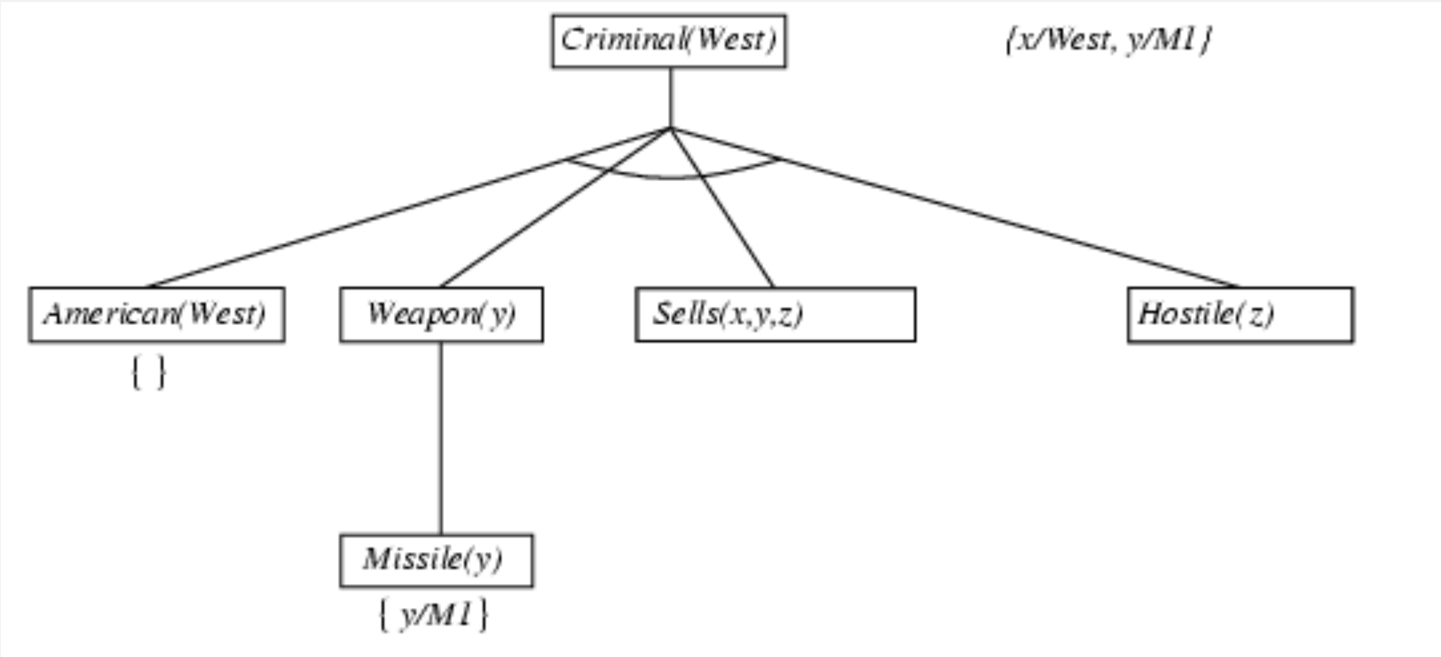
Backward chaining example



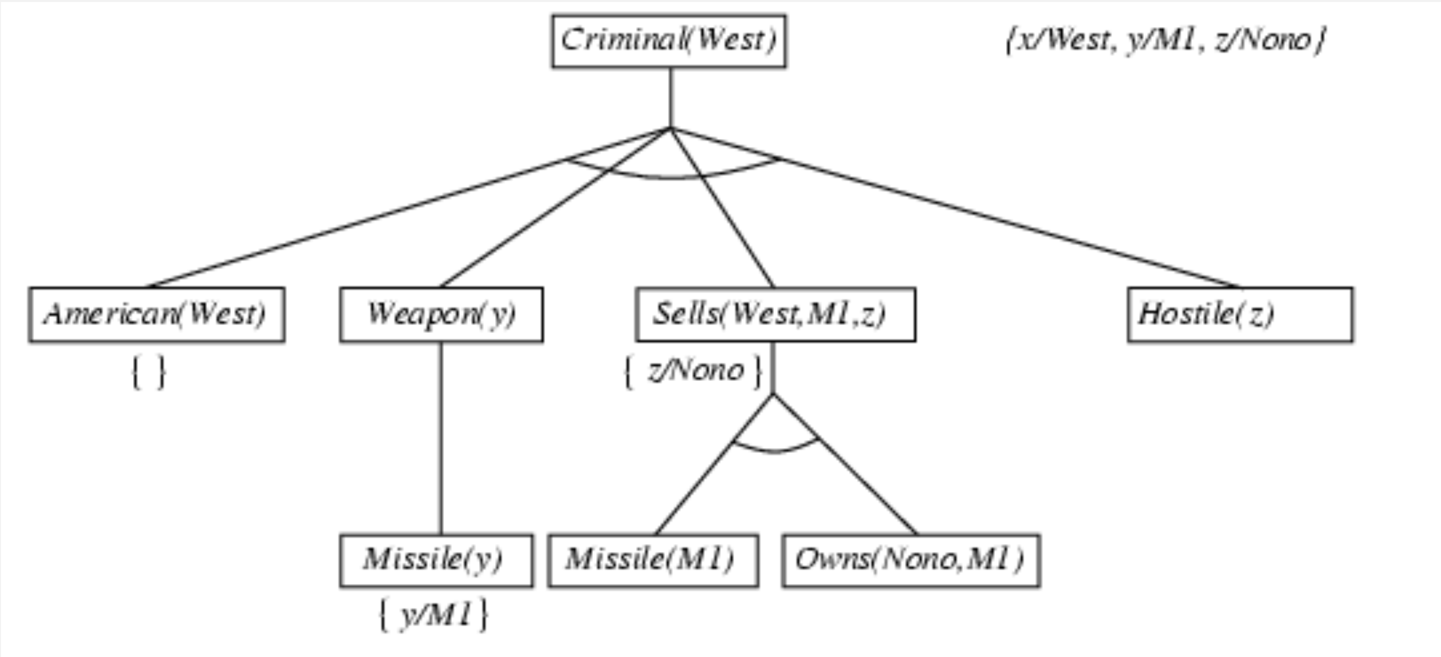
Backward chaining example



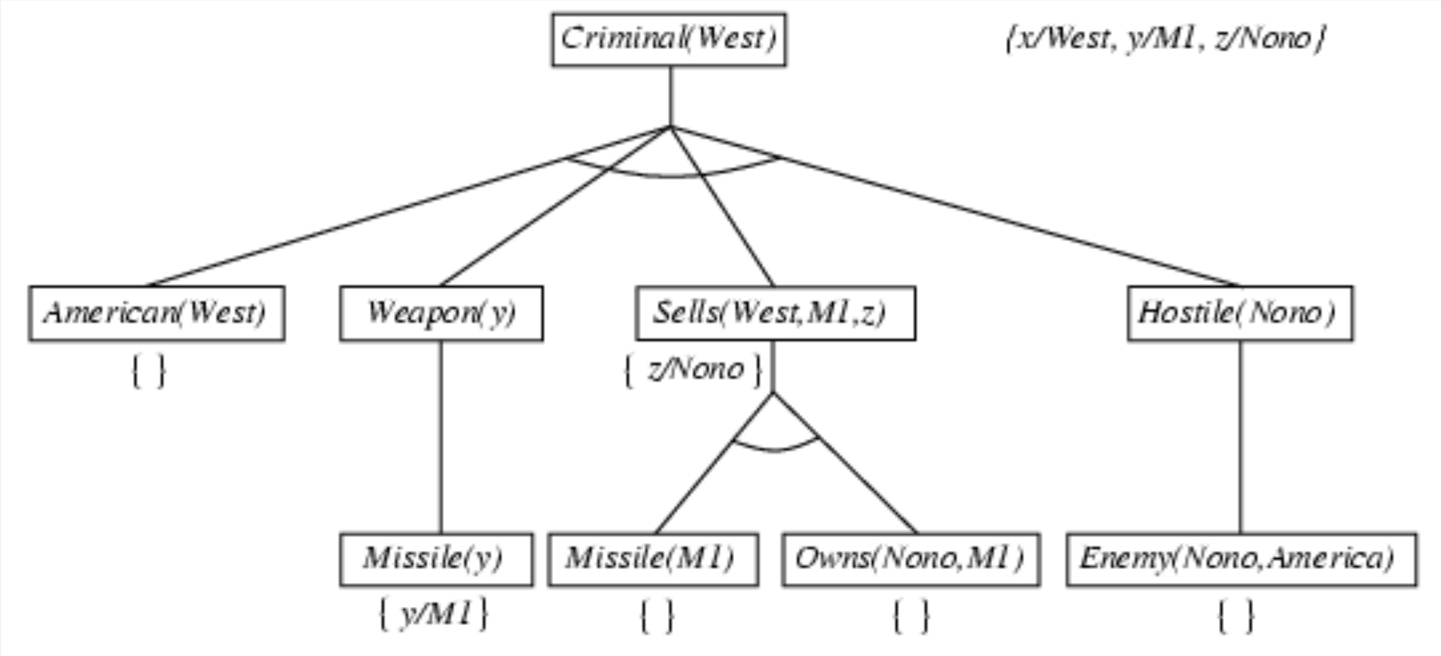
Backward chaining example



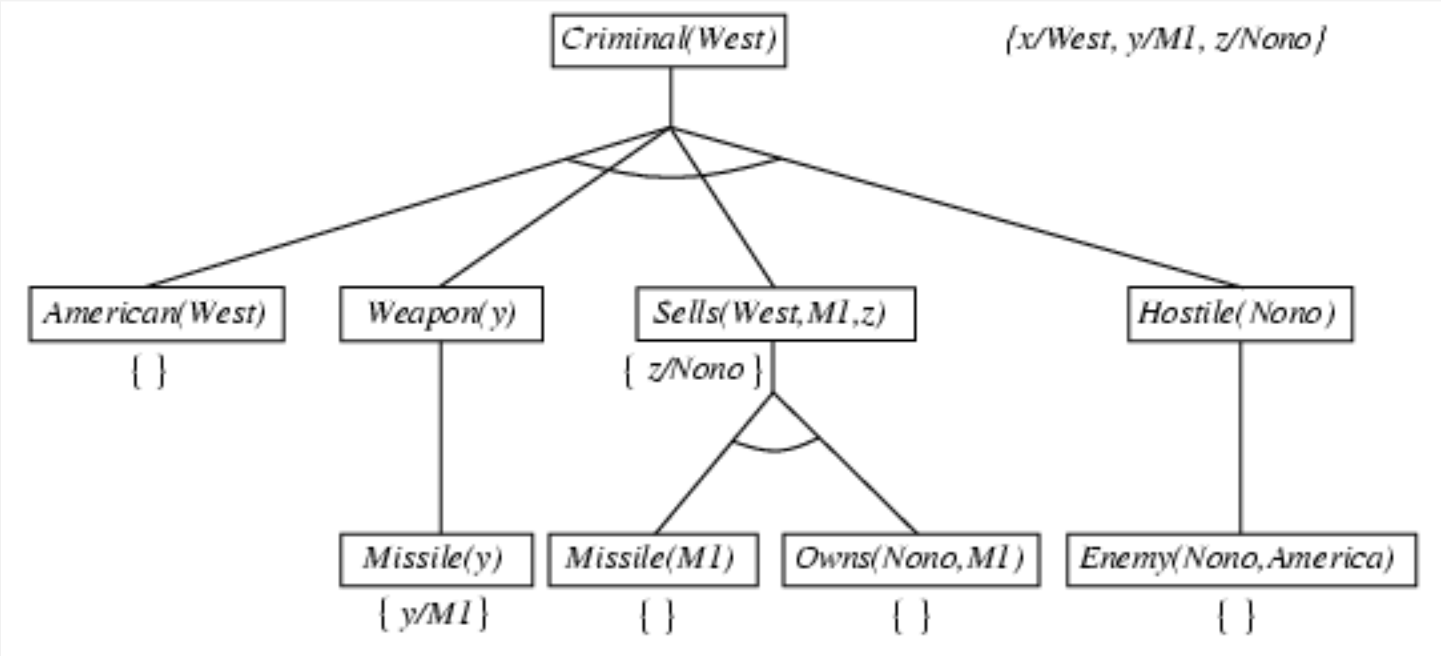
Backward chaining example



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
  - fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  - fix using caching of previous results (extra space!!)
- Widely used for logic programming

Logic programming

- Logic programming
  - Identify problem
  - Assemble information
  - Encode info in KB
  - Encode problem instances as facts
  - Ask queries
  - Find false facts.

- Procedural programming
  - Identify problem
  - Assemble information
  - Figure out solution
  - Program solution
  - Encode problem instance as data
  - Apply program to data
  - Debug procedural errors

Theorem Proving in Predicate Logic Resolution by Refutation

If $KB \models \sigma$
then $\exists$ resolution proof of $\{\}$
from $KB \cup \{\neg\sigma\}$

- Add ¬σ to KB
- Convert KB to CNF
- Apply Resolution Procedure
  - Derive {}: σ is proved
  - Dead end:
    σ is not a consequence of KB

Properties of Resolution

Resolution by refutation is

- Sound
- Refutation Complete
  - If KB $|= \alpha$, refutation will prove it
  - Otherwise, in the general setting (infinite number of models) refutation procedure may not terminate
- Complexity
  - Exponential in the size of KB for Propositional Logic (worst case)

Theorem Proving in FOL

If a course is interesting, some students are happy.
if a course has a final, no student is happy.
Prove: If a course has a final, then it is not interesting.

Putting this in FOPL we get:

1. $\forall c\, Interesting(c) \Rightarrow \exists s\, [Student(s,c) \wedge Happy(s)]$
2. $\forall s\, \forall_c\, [Final(c) \wedge Student(s,c) \Rightarrow \neg Happy(s)]$

Theorem to prove : $\forall c\, Final(c) \Rightarrow \neg Interesting(c)$

Negation of theorem :

3. $\neg[\forall c\, Final(c) \Rightarrow \neg Interesting(c)]$

Theorem Proving in FOL

By inspection we can translate the above into clause normal form :

a. $\neg Interesting(c) \vee Student(skf(c), c)$

b. $\neg Interesting(x) \vee happy(skf(x))$

c. $\neg Final(z) \vee \neg Student(s, z) \vee \neg Happy(s)$

d. $Final(sk\phi)$

e. $Interesting(sk\phi)$

Proof that if a course has a final, it is not interesting

a. $\neg Interesting(c) \vee Student(skf(c), c)$
e. $Interesting(sk\phi)$ $\quad \sigma = \{sk\phi|c\}$

---

f. $[Student(skf(sk\phi), sk\phi)]$
c. $\neg Final(z) \vee \neg Student(s, z) \vee \neg Happy(s)$ $\quad \sigma = \{skf(sk\phi)|s, sk\phi|z\}$

---

g. $[\neg Final(sk\phi) \vee \neg Happy(skf(sk\phi))]$
d. $Final(sk\phi)$ $\quad \sigma = \{\}$

---

h. $\neg Happy(skf(sk\phi))$
b. $\neg Interesting(x) \vee Happy(skf(x))$ $\quad \sigma = \{sk\phi|x\}$

---

i. $\neg Interesting(sk\phi)$
e. $Interesting(sk\phi)$

[Null clause]

Resolution Proof: Example

$$\text{Axioms}: \{I(A),\ D(A),\ \neg R(x) \vee L(x),\ \neg D(y) \vee \neg L(y)\}$$

$$\text{Negated Theorem}: \{\neg I(z) \vee R(z)\}$$

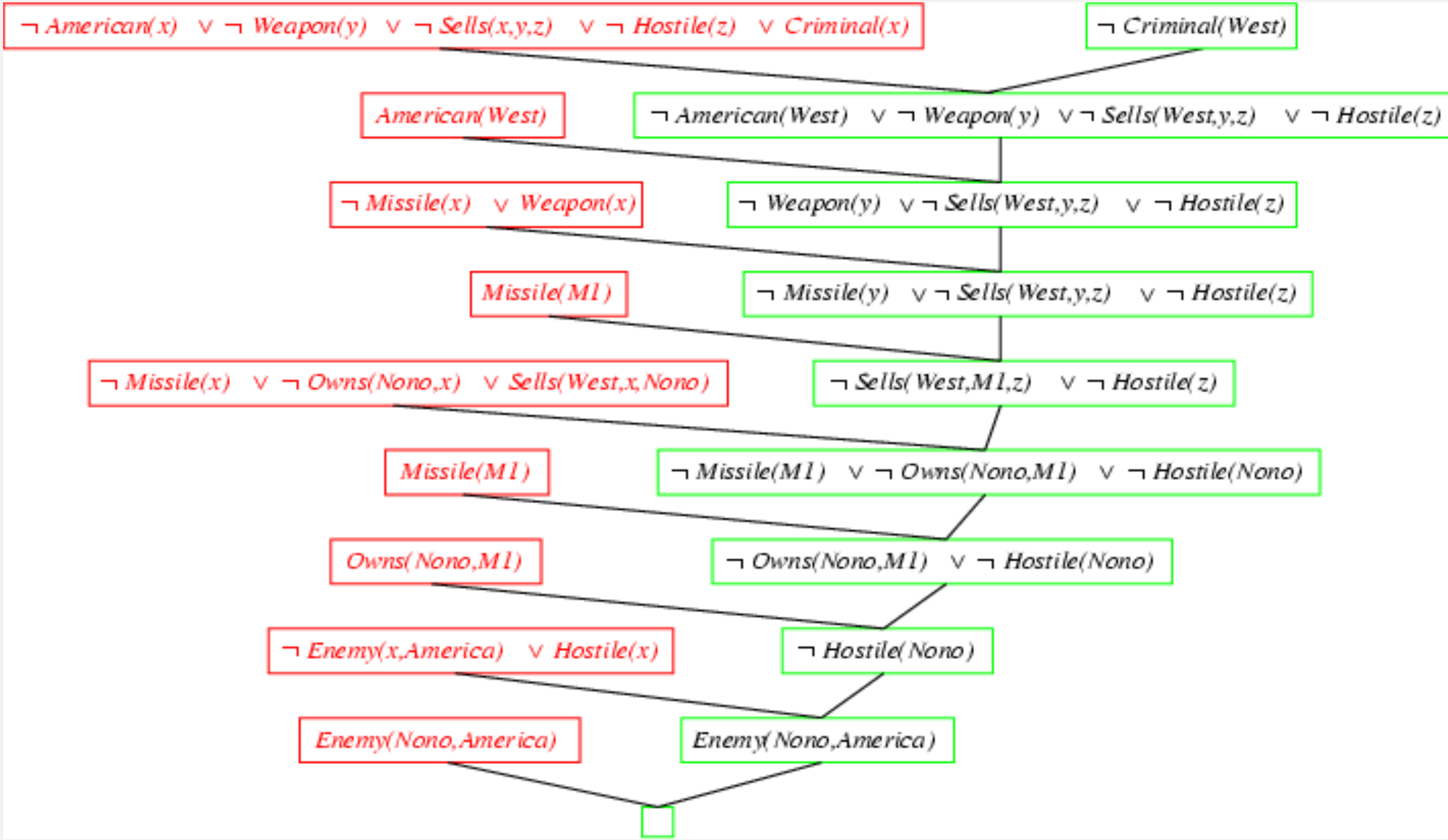
$$\neg I(z) \vee R(z),\quad I(A) \ \longrightarrow\ R(A)$$

$$R(A),\quad \neg R(x) \vee L(x) \ \longrightarrow\ L(A)$$

$$L(A),\quad \neg D(y) \vee \neg L(y) \ \longrightarrow\ \neg D(A)$$

$$\neg D(A),\quad D(A) \ \longrightarrow\ \blacksquare$$

Resolution Proof



Search Control in Theorem Proving

- Unit preference strategy

$$P(x)$$
$$\neg P(y) \vee R(y) \vee Q(y)$$
$$P(z) \vee \neg S(z)$$

- Which pair of clauses to choose?

$$P(x)$$
$$\neg P(y) \vee R(y) \vee Q(y)$$

- Why?

Search Control in Theorem Proving

- Set of support (SOS)
    - All clauses in negated theorem belong to SOS
    - Any clause derived from resolving a member of SOS with another clause belongs to SOS
- Set of support strategy
    - Each resolution step must choose a member of SOS as one of the two clauses to be resolved
- Theorem: SOS is refutation complete for FOL. That is, if there is a proof for a theorem, it can be found using SOS strategy.

Resolution Proof: Example

$$\text{Axioms}: \{I(A),\ D(A),\ \neg R(x) \vee L(x), \neg D(y) \vee \neg L(y)\}$$

$$\text{Negated Theorem}: \ \{\neg I(z) \vee R(z)\}$$

$$\begin{tikzcd}
\neg I(z) \vee R(z) \arrow[dr] & & I(A) \arrow[dl] & & & \\
& R(A) \arrow[dr] & & \neg R(x) \vee L(x) \arrow[dl] & & \\
& & L(A) \arrow[dr] & & \neg D(y) \vee \neg L(y) \arrow[dl] & \\
& & & \neg D(A) \arrow[dr] & & D(A) \arrow[dl] \\
& & & & \blacksquare &
\end{tikzcd}$$

Search Control for Theorem Proving

- Eliminate
  - clauses containing pure literals (literals whose complements do not appear in any other clause in the KB)
  - tautologies e.g., $R(x) \vee \neg R(x)$
  - any clause that is subsumed by another clause

A clause ϕ subsumes a clause ψ iff

$\exists$ a substitution σ such that $\phi\sigma \subseteq \psi$

$P(x)$ subsumes $P(x) \vee R(y)$

$P(x) \vee Q(y)$ subsumes $P(f(A)) \vee R(z) \vee Q(B)$

Elimination of subsumed clauses

- Theorem: Unsatisfiability of a set *S* of clauses is unaffected by elimination of clauses in *S* that are subsumed by other clauses in *S*
- Proof: WLOG consider propositional KB

$$\text{Let } S = \{c_1 \ldots c_n, c, c'\} \models q$$
$$S' = \{c_1 \ldots c_n, c,\} = S - \{c'\}$$
$$S'' = \{c_1 \ldots c_n\} = S - \{c, c'\} = S' - \{c'\}$$

$$\text{Let } c = P;\ c' = P \vee Q; \text{So } c \text{ subsumes } c'$$
$$M_S = M_{S''} \cap M_{c \wedge c'} = M_{S''} \cap M_c \cap M_{c'}$$
$$= M_{S''} \cap M_c = M_{S'}$$

Green's Trick for Answer Extraction

- We are often interested in instantiation that makes a theorem true (e.g., queries in deductive databases)

KB:

$$\forall x \ At(Bumstead, x) \Rightarrow At(Daisy, x)$$

$$At(Bumstead, Couch)$$

Query:

$$\exists x \ At(Daisy, x)$$

Substitute in $At(Daisy, x)$

the *same* subsitutions used to prove the query

to answer the question *Where is Daisy?*

Green’s Trick for Answer Extraction

$\neg At(Daisy, z)$ $\quad$ $\neg At(Bumstead, x) \vee At(Daisy, x)$ $\quad\quad$ $At(Daisy, z)$

$z \mid x$

$\neg At(Bumstead, z)$ $\quad$ $At(Bumstead, Couch)$

$Couch \mid z$

$\blacksquare$ $\quad\quad$ $At(Daisy, Couch)$

Knowledge Representation
Semantic Web and Description Logics

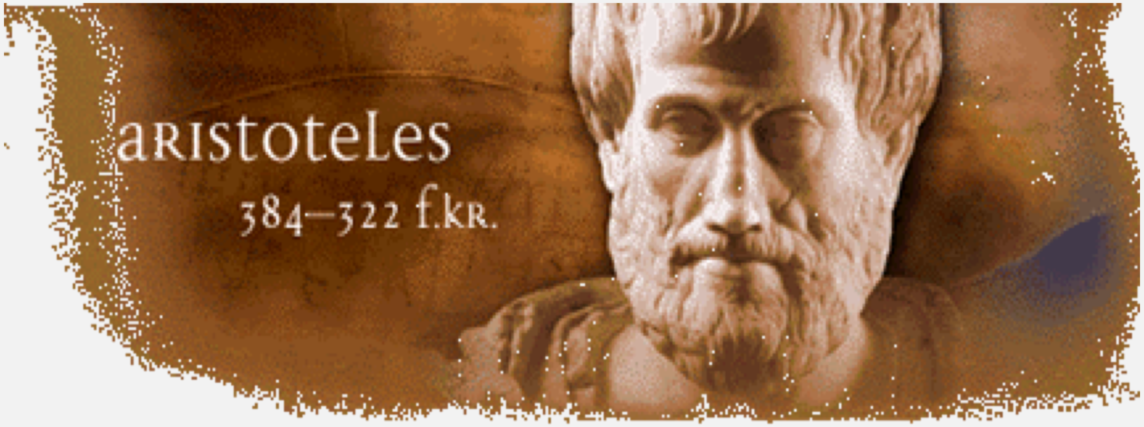
Vasant Honavar
Artificial Intelligence Research Laboratory
Informatics Graduate Program
Computer Science and Engineering Graduate Program
Bioinformatics and Genomics Graduate Program
Neuroscience Graduate Program

Center for Big Data Analytics and Discovery Informatics
Huck Institutes of the Life Sciences
Institute for Cyberscience
Clinical and Translational Sciences Institute
Northeast Big Data Hub
Pennsylvania State University

vhonavar@ist.psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

Ontology…

- Philosophical origins date back to Aristotle
  - "The metaphysical study of the nature of being and existence"



Ontology…

- In Artificial Intelligence,
  - "an ontology is a formal, explicit specification of a shared conceptualization "
- Conceptualization
  - What does our world consist of?
  - Entities, Properties, Relationships
- Formal
  - Machine interpretable
  - Syntax, semantics, proof theory
- Shared
  - Within a domain of inquiry (e.g., physics), a community (e.g., fans of pop music) etc.

What does our world consist of?

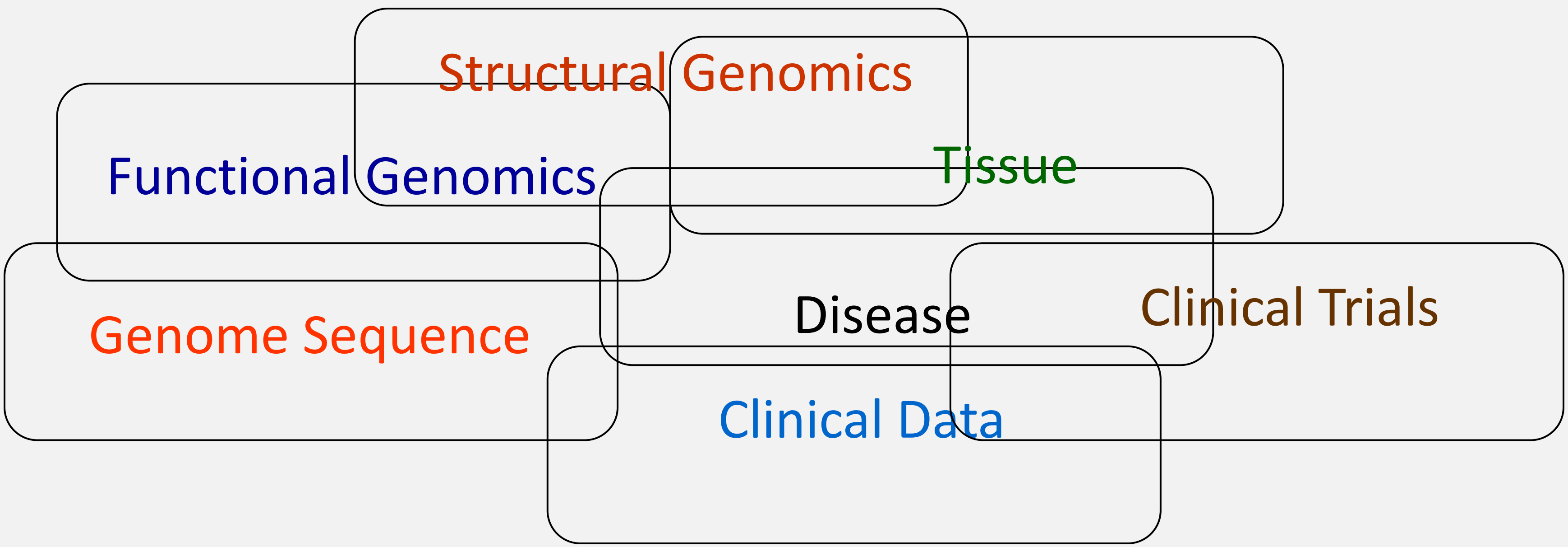
- Objects or instances or individuals
  - Correspond to constants in FOL
- Classes or concepts usually organized in taxonomies
  - Sets of objects sharing certain characteristics
  - Equivalent to unary predicates in FOL
  - e.g. a university ontology, might include the concepts like student and professor
- Relations, roles between concepts (often limited to binary)
  - Binary relations define sets of pairs (tuples) of objects
  - Binary relations are equivalent to binary predicates in FOL

What does our world consist of?

- Functions:
  - Can be modeled by relations in which the *n* th element of the relation is unique given the *n*-1 preceding elements
  - Price-of-a-used-car function can calculate the price of a used car given the car model, and mileage
- Axioms
  - Sentences that are always true in our world
  - Definitions that restrict the use of concepts and relationships
  - e.g., every Data Sciences major should have a 3.0 or better GPA in DS courses

Kinds of ontologies

- General ontologies
  - vocabulary of things, events, time, space, units, etc.
- Domain ontologies
  - Ontology reusable within a domain
  - e.g., gene ontology, disease ontology, e-commerce ontology, weather ontology

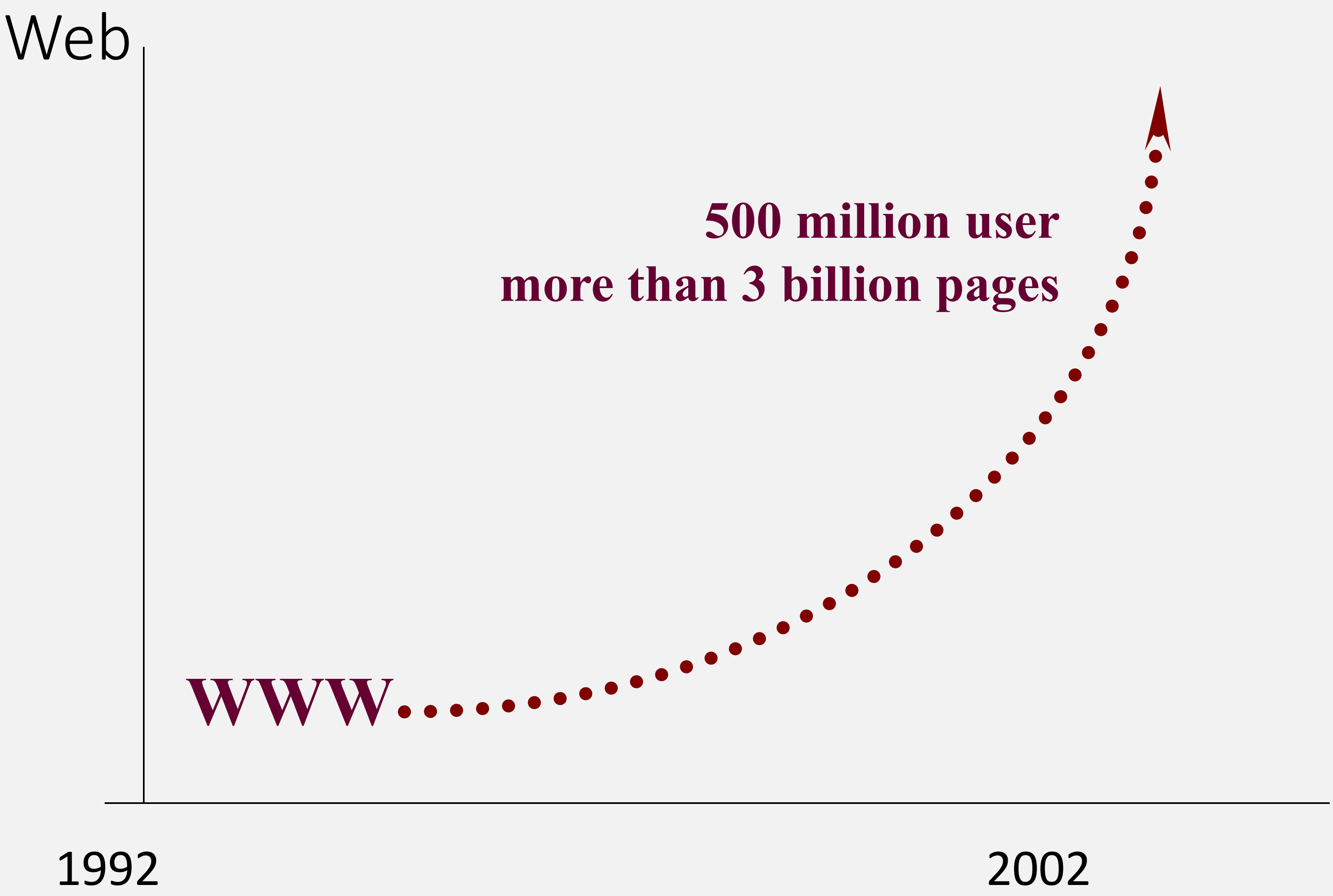


Domain ontologies

- Gene ontology
- Disease ontology
- Weather ontology
- Pizza ontology
- Scheduling ontology
- Clinical procedures ontology

Ontology Applications

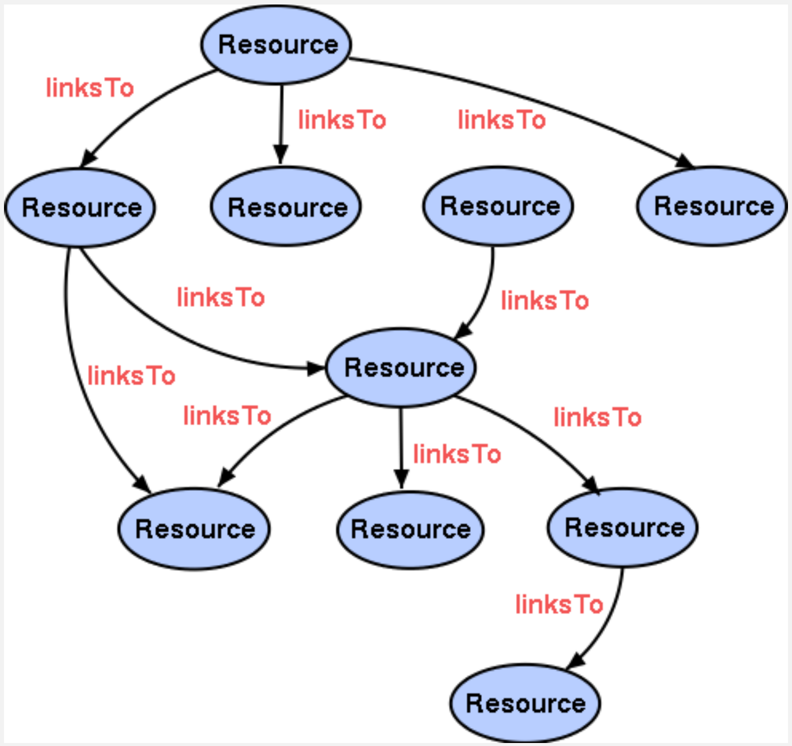
- Semantic search
  - search for news on high tech stocks should return news on Intel, Yahoo, etc.
- e-commerce
- Querying multiple data sources
- Enterprise Application Integration
- e-science
- Semantic web
- …


Web
500 million user
more than 3 billion pages
WWW
1992
2002

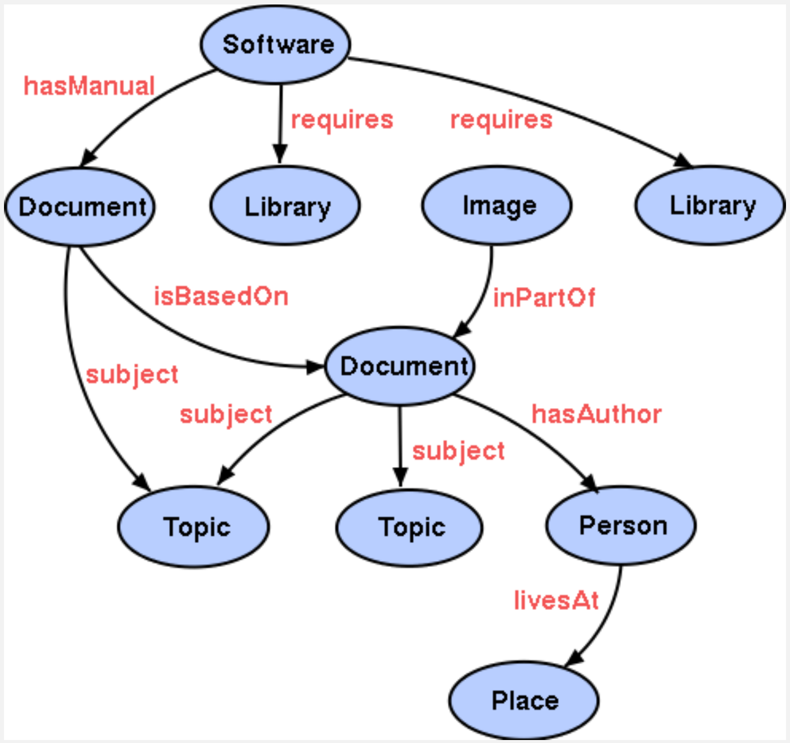
The Web

- The World Wide Web has been a success beyond anyone had dreamed of at the time of its invention in terms of
  - of the amount of information available
  - the number of users
- This success is based on
  - its simplicity
    - simple protocol HTTP
    - simple markup language HTML
  - Network effect
- But.. HTML is primarily for formatting information for presentation to human readers
- The web is impoverished in terms of semantics

The web of hyperlinked documents is not enough!



- Homogeneous resources
- Semantically empty
- Needs human interpretation



- ☐ Identified resources
- ☐ Meaningful links
- ☐ Machine processable

Semantic technologies and semantic web

- Semantic Web Vision
  - machine-interpretable data and knowledge grounded in domain specific, task specific, or even user-specific semantics
  - specialized reasoning services
  - services of querying, integrating, and analyzing, and acting on information
- The semantic Web needs ontologies for
  - formal and consensual specifications of conceptualizations…
  - providing a shared and common understanding of a domain
  - Communicating data and knowledge between humans and computers

Semantic Web Technology

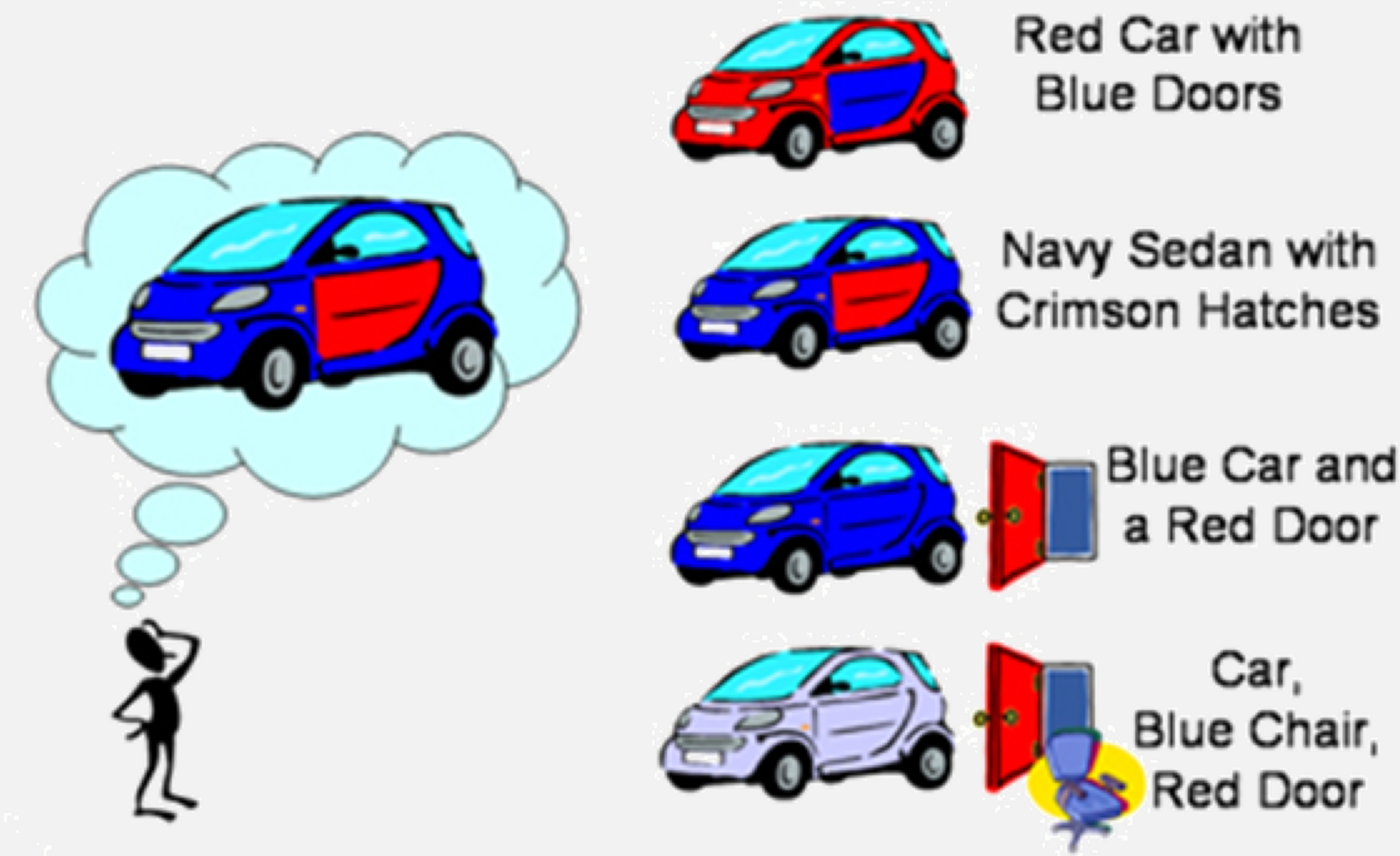
- Ontologies
  - establish a formal semantics for data and knowledge making it possible for computers to process information
  - enable communication of machine interpretable content between humans, between machines, and between humans and machines

Semantic search

- Semantic search
- Complex queries involving background knowledge
  - Find information about “animals that use sonar but are not either bats, dolphins or whales”
- Integrating information from multiple sources
  - Book me a holiday next weekend somewhere warm, not too far away, and where they speak French or English

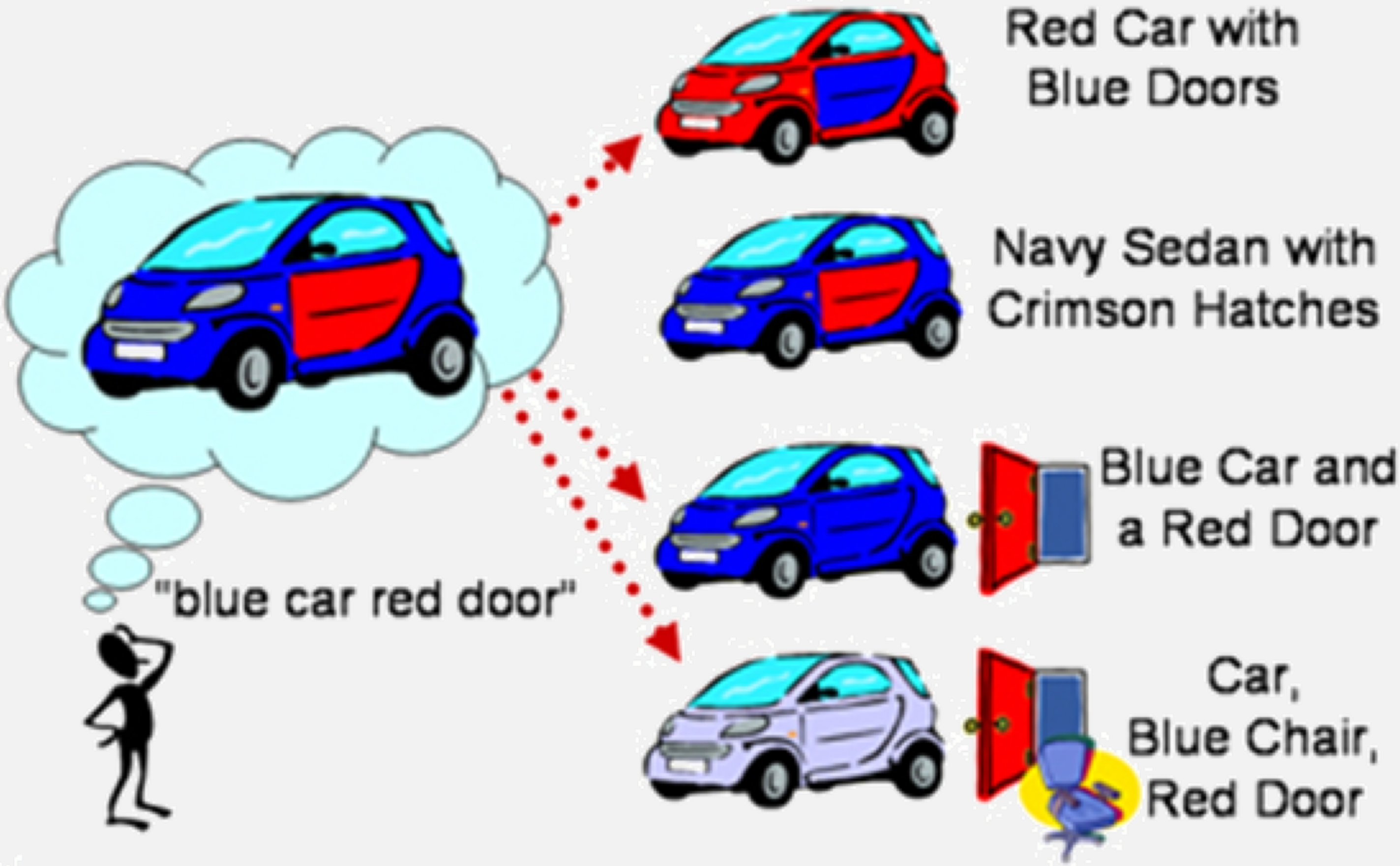
Hopeless for machines and tedious for people

Looking for a "Blue Car with Red Doors"

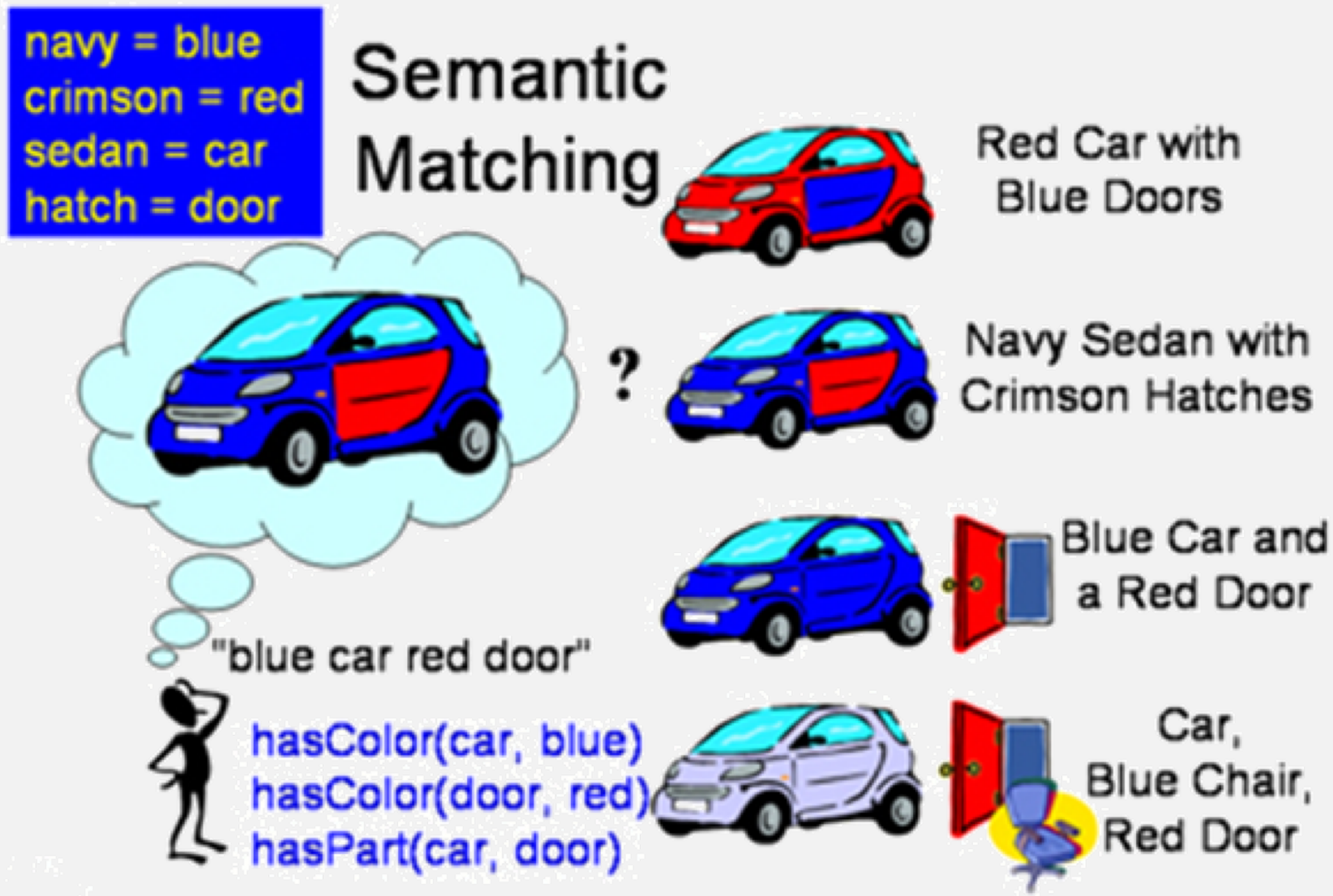


Steven Kraines, Science Integration Project – Human, The University of Tokyo

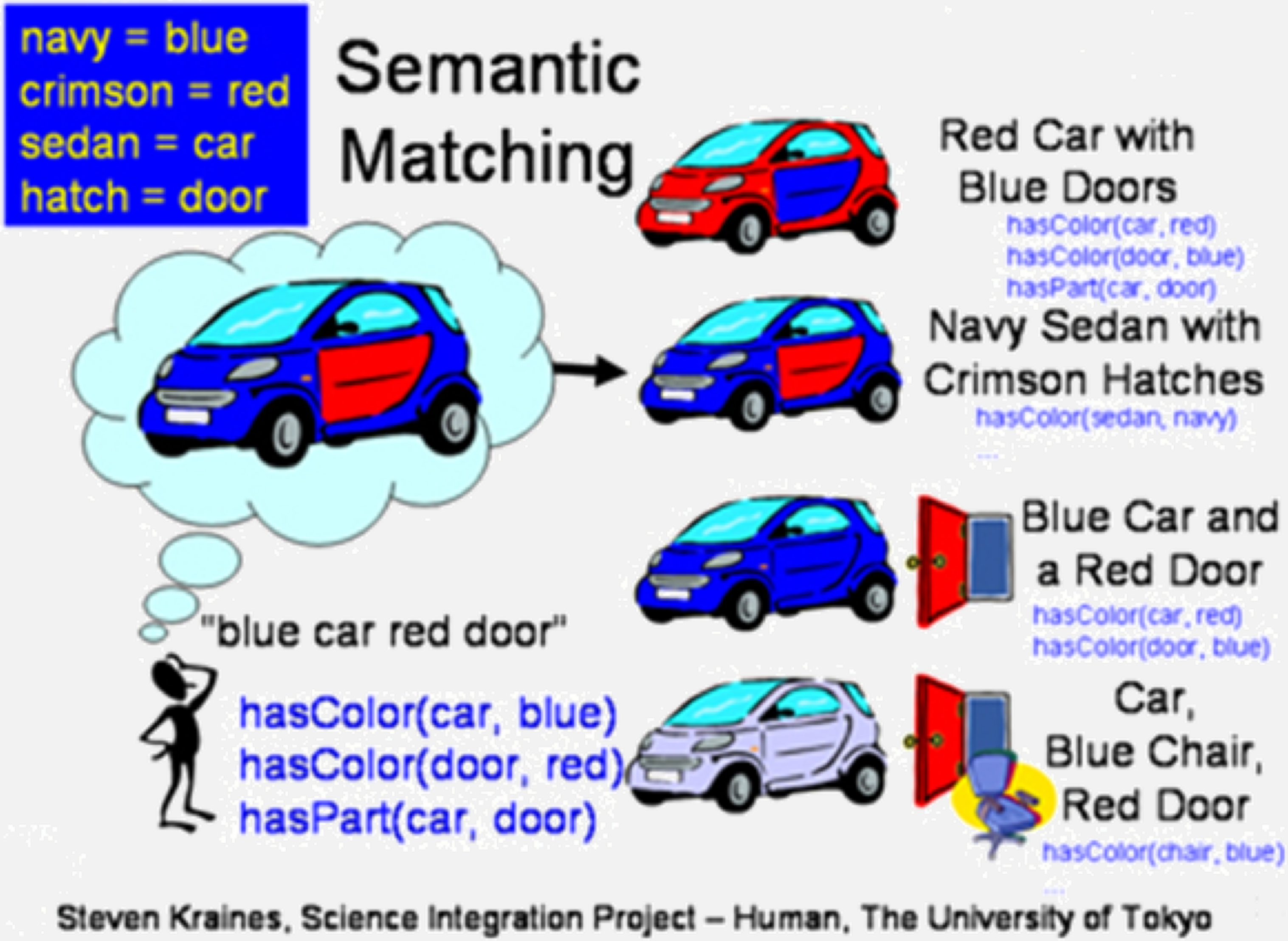
Simple word-matching



Steven Kraines, Science Integration Project – Human, The University of Tokyo



Steven Kraines, Science Integration Project – Human, The University of Tokyo


navy = blue
crimson = red
sedan = car
hatch = door
Semantic Matching
Red Car with Blue Doors
hasColor(car, red)
hasColor(door, blue)
hasPart(car, door)
Navy Sedan with Crimson Hatches
hasColor(sedan, navy)
Blue Car and a Red Door
hasColor(car, red)
hasColor(door, blue)
Car, Blue Chair, Red Door
hasColor(chair, blue)
"blue car red door"
hasColor(car, blue)
hasColor(door, red)
hasPart(car, door)
Steven Kraines, Science Integration Project – Human, The University of Tokyo

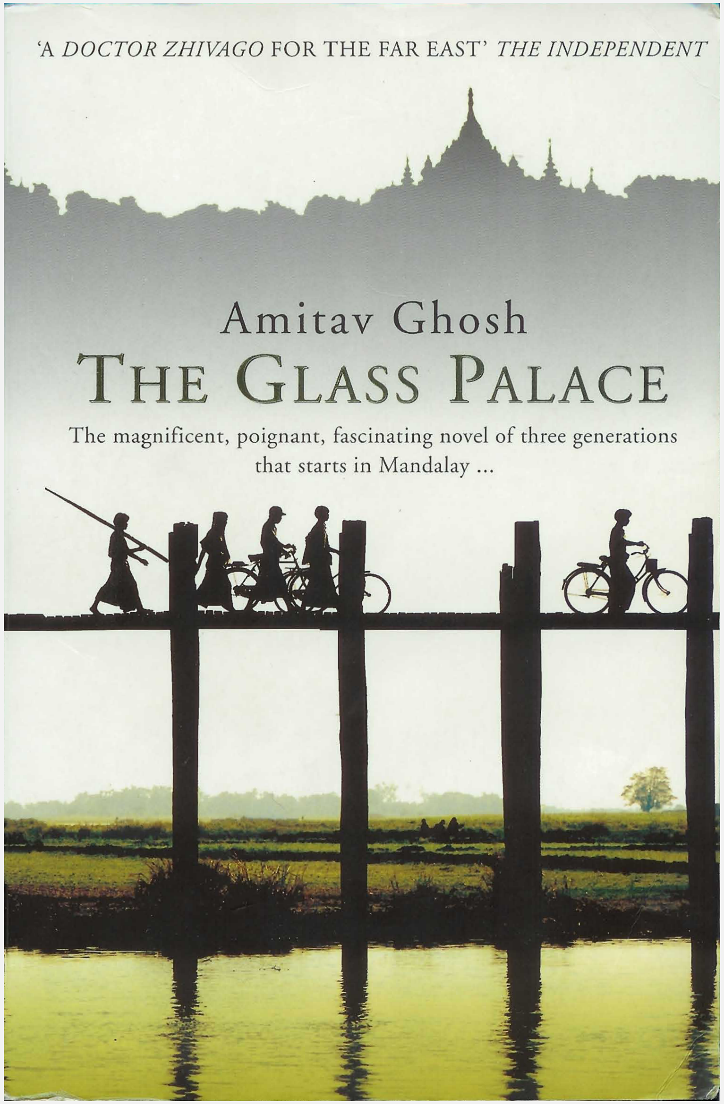
Semantic Web

- Expose data on the web in an interoperable form (RDF)
- Expose knowledge on the web with interoperable semantics (ontologies: RDF Schema, OWL)
- Apply (lightweight) inference for
  - Searching for information
  - Answering complex queries
  - Integrating multiple sources of knowledge and data
  - Composing composite services from component services
  - Learning predictive models from disparate data sources
  - Unexpected reuse

Data integration

- Represent data in RDF stores
- Combine data from RDF stores
- Query the integrated RDF data

A book in English



[Ivan Herman, W3C]

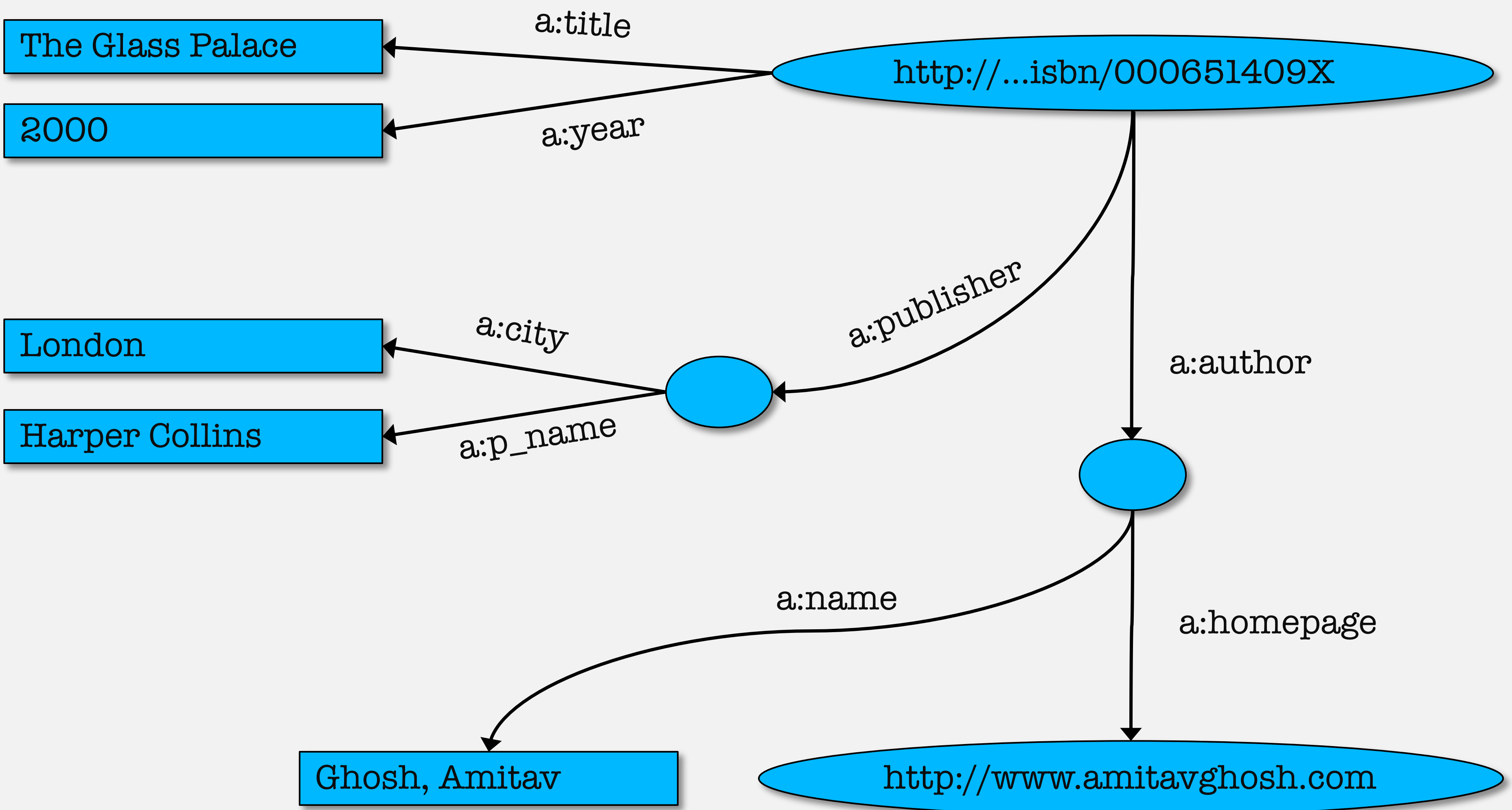
A simplified bookstore data (dataset "A")

ID	Author	Title	Publisher	Year
ISBN 0-00-6511409-X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

ID	Publisher's name	City
id_qpr	Harper Collins	London

1st: export data as a set of relations



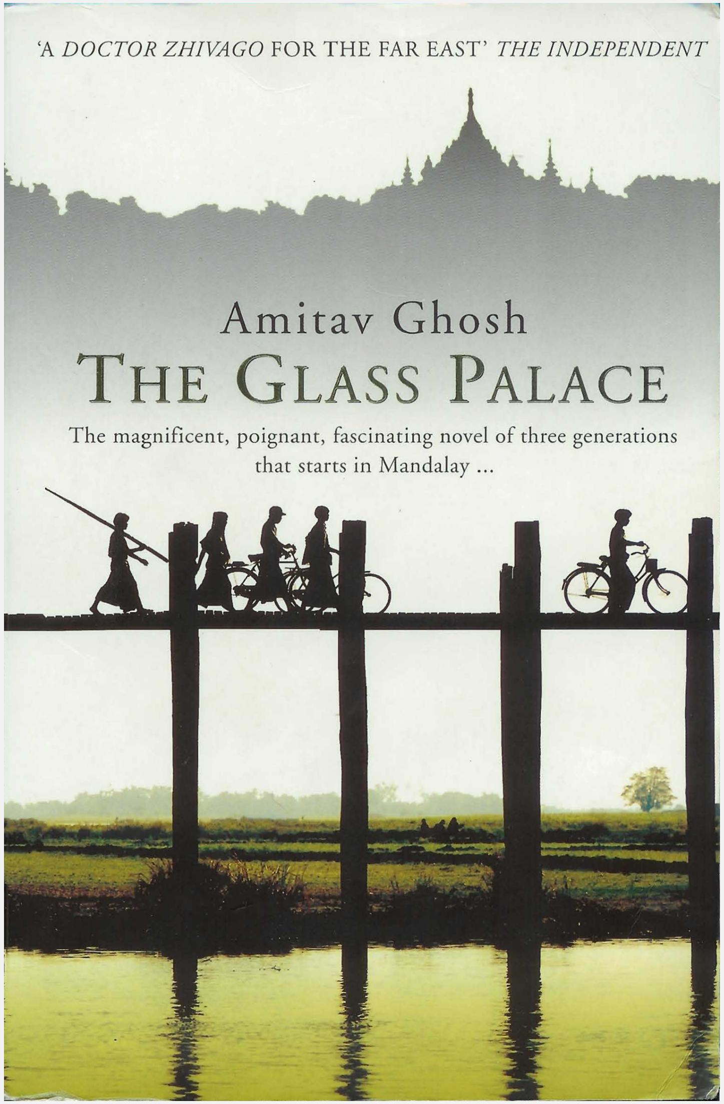
Some notes on the exporting the data

- Relations form a graph
  - the nodes refer to the “real” data or contain some literal
  - how the graph is represented in the machine is immaterial for now
- Data export does not necessarily mean physical conversion of the data
  - Relations can be generated on-the-fly at query time
    - via SQL “bridges”
    - scraping HTML pages
    - extracting data from Excel sheets
    - Extracting data from text

Data integration

- Represent data in RDF stores
- Combine data from RDF stores
- Query the integrated RDF data

A book in English



[Ivan Herman, W3C]

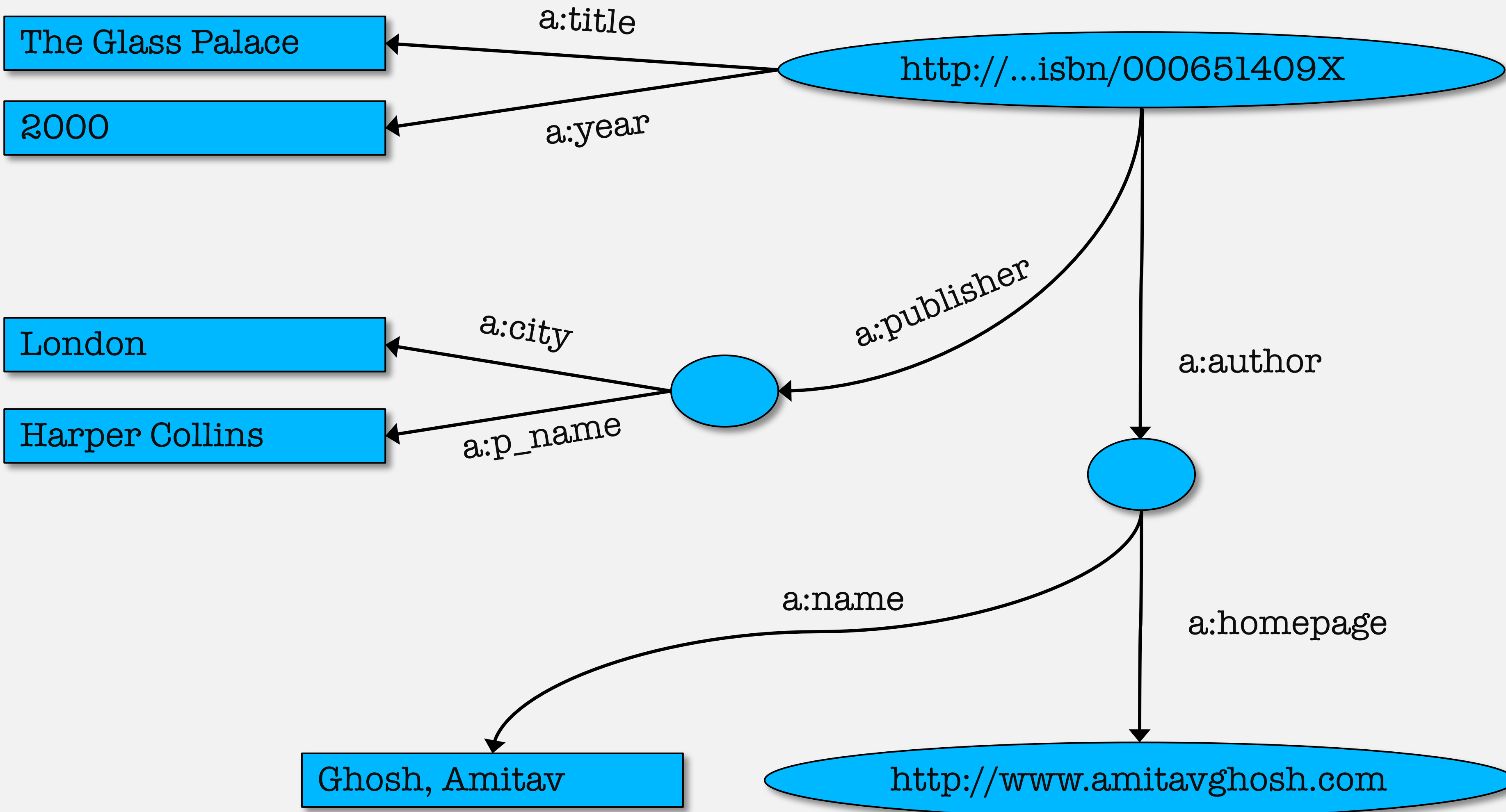
A simplified bookstore data (dataset "A")

ID	Author	Title	Publisher	Year
ISBN 0-00-6511409-X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

ID	Publisher's name	City
id_qpr	Harper Collins	London

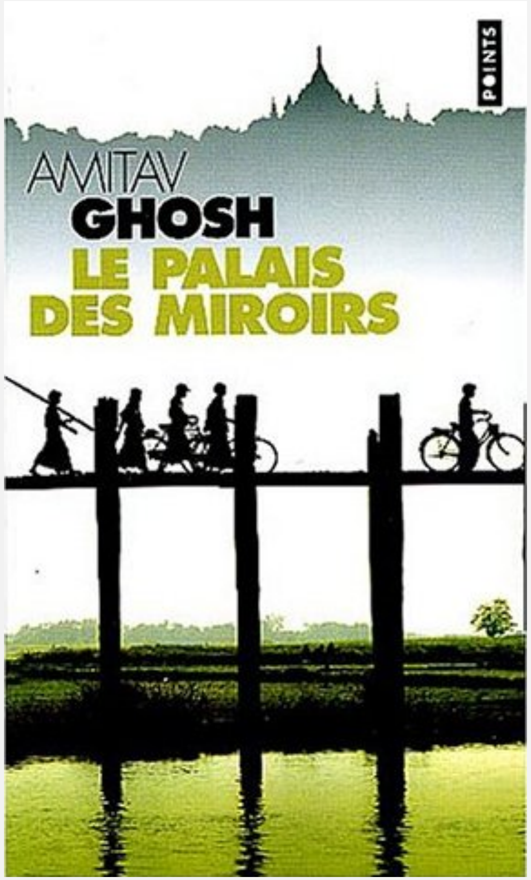
1st: export data as a set of relations



Some notes on the exporting the data

- Relations form a graph
  - the nodes refer to the "real" data or contain some literal
  - how the graph is represented in the machine is immaterial for now
- Data export does not necessarily mean physical conversion of the data
  - Relations can be generated on-the-fly at query time
    - via SQL "bridges"
    - scraping HTML pages
    - extracting data from Excel sheets
    - Extracting data from text
- One can export part of the data

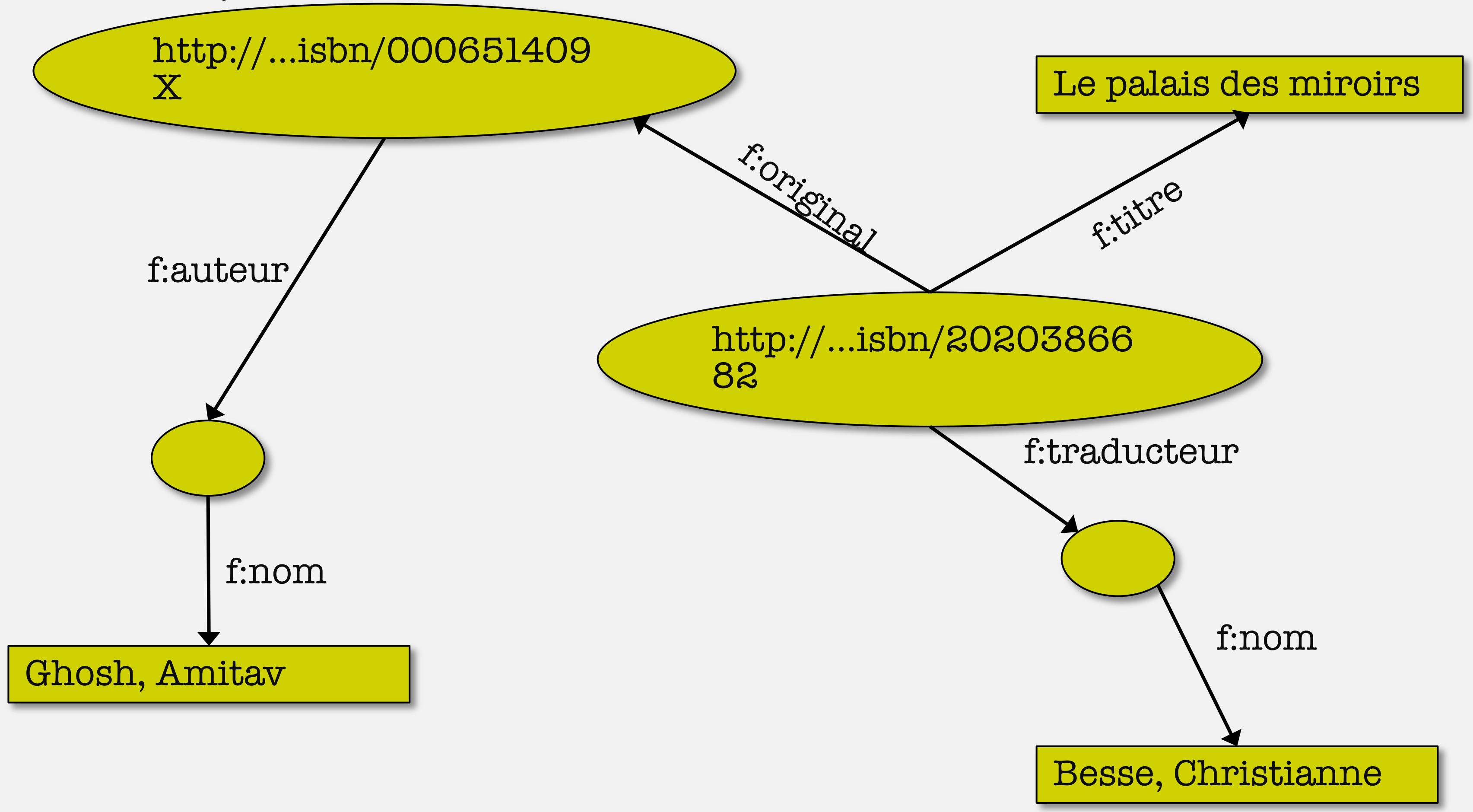
The same book in French…



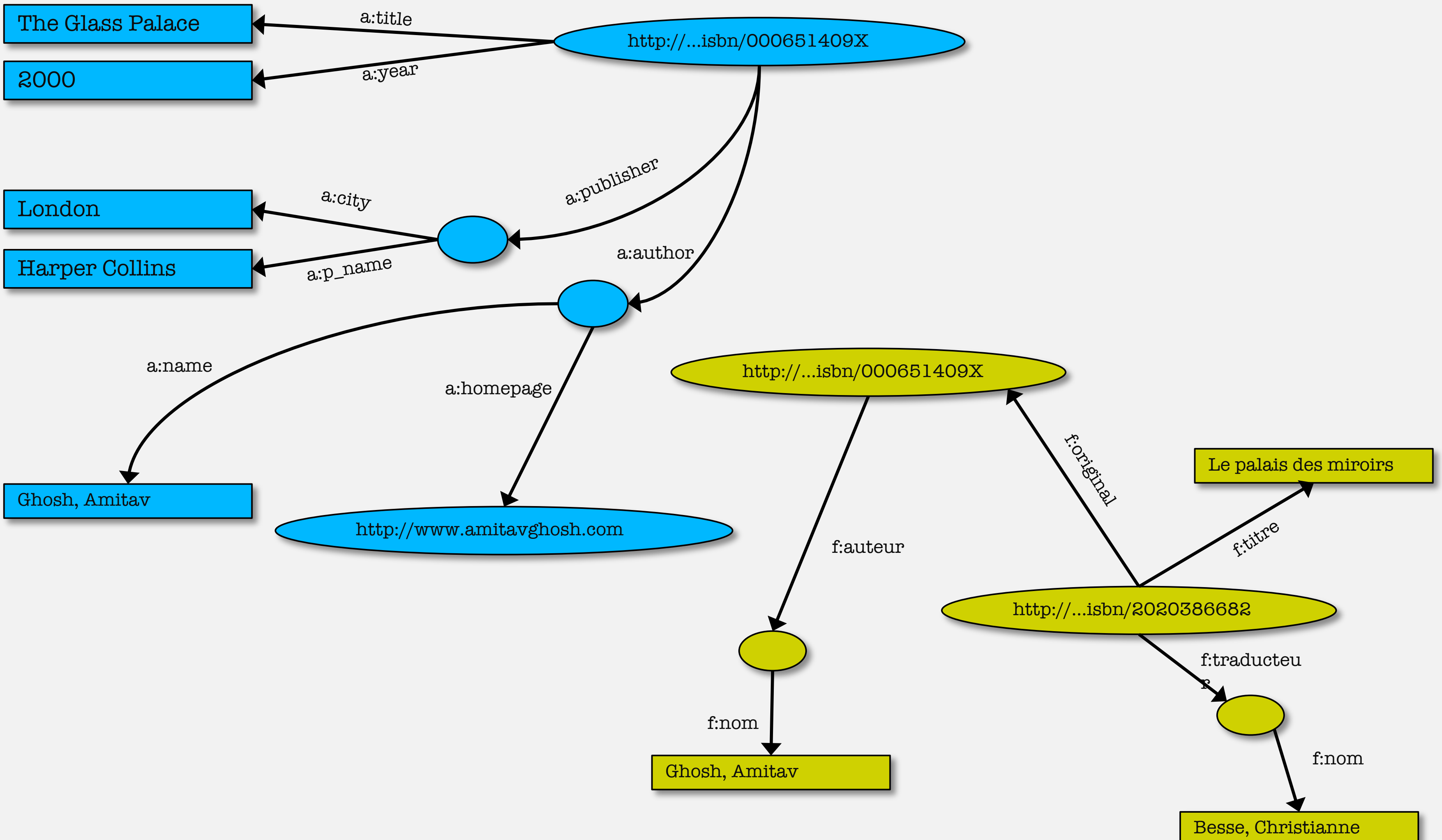
A second bookstore data (dataset "F")

	A	B	C	D
1	**ID**	**Titre**	**Traducteur**	**Original**
2	ISBN 2020286682	Le Palais des Miroirs	$A12$	ISBN 0-00-6511409-X
3				
4				
5				
6	**ID**	**Auteur**		
7	ISBN 0-00-6511409-X	$A11$		
8				
9				
10	**Nom**			
11	Ghosh, Amitav			
12	Besse, Christianne			

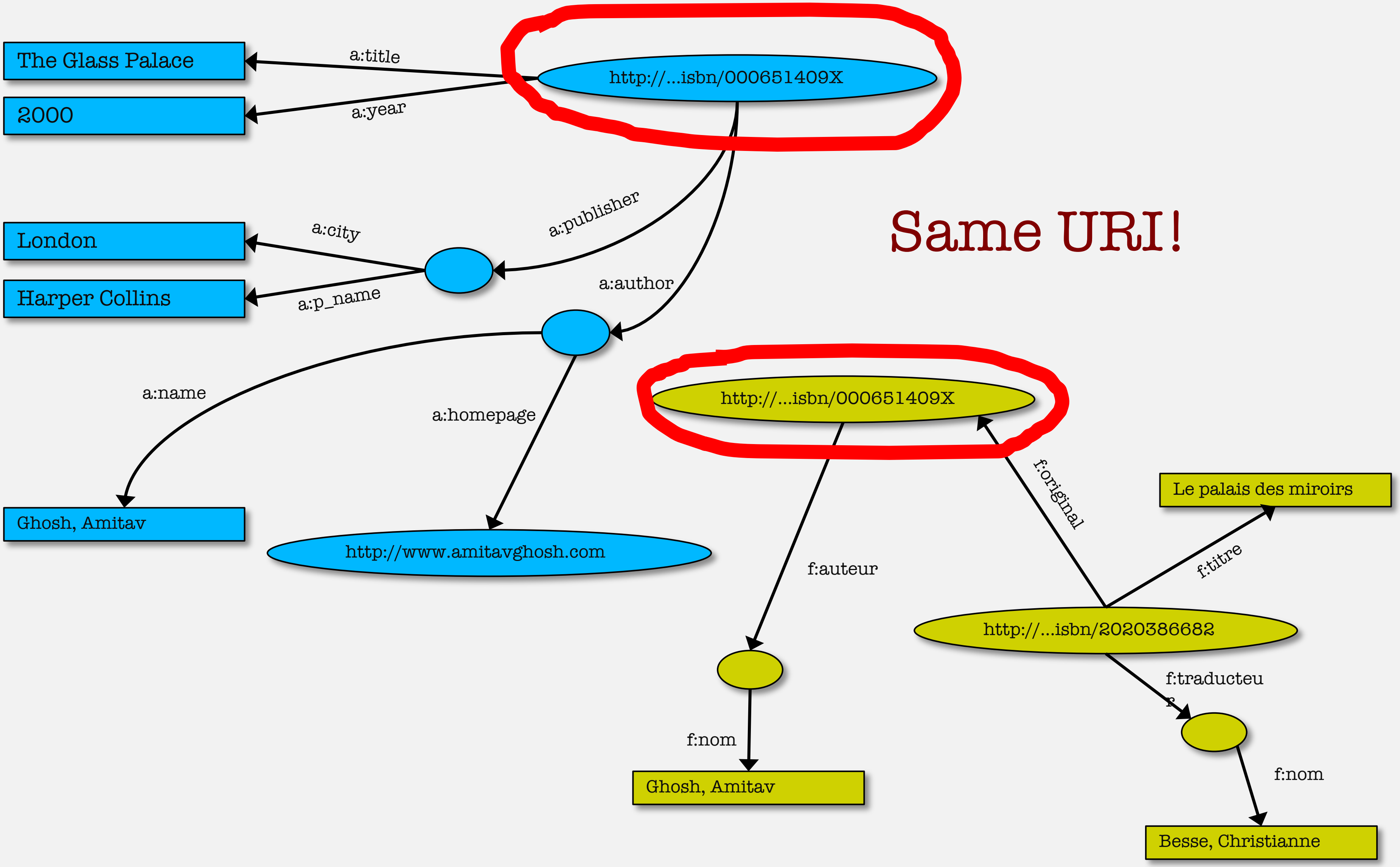
2nd: export the second set of data



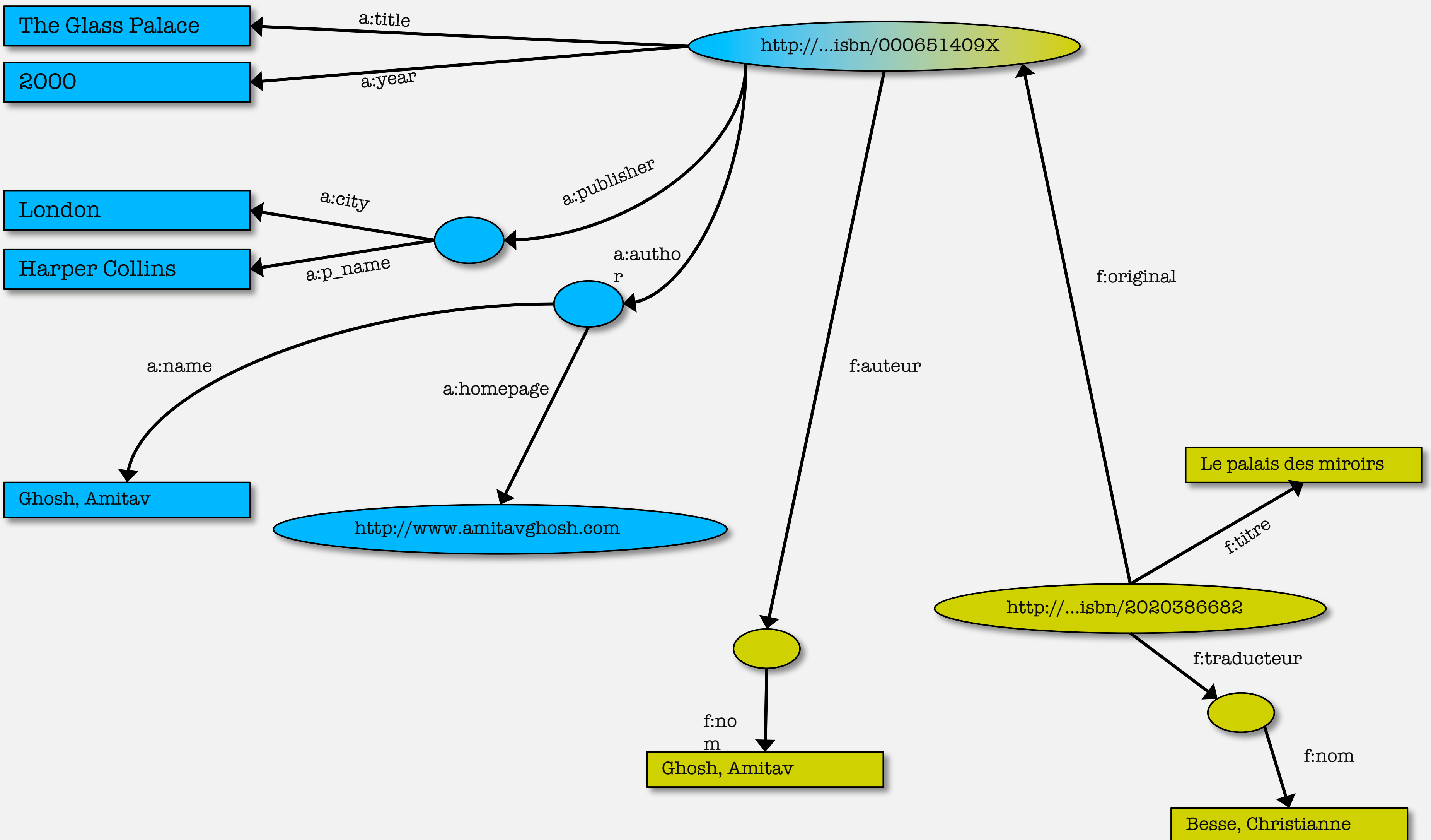
3rd: merge your data



3rd: merge your data (cont)

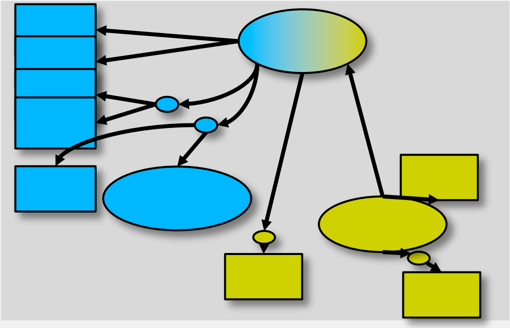


3rd: merge your data



Start making queries…

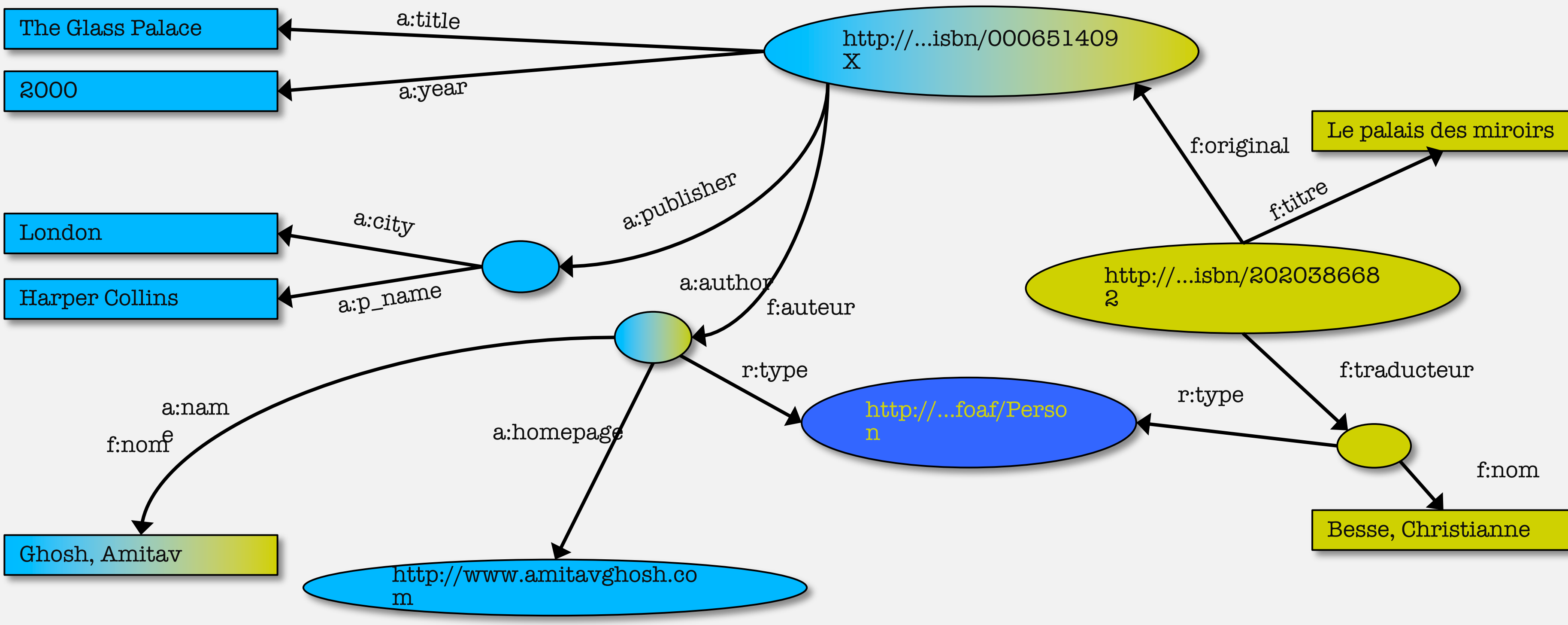
- User of data “F” can now ask queries like:
    - “give me the title of the original”
        - well, … « donnes-moi le titre de l’original »
- This information is not in the dataset “F”…
- …but can be retrieved by merging with dataset “A”!

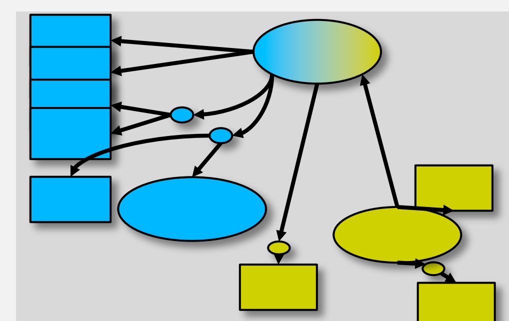


There is more we can do!

- We “think” that a:author and f:auteur should be the same
- But an automatic merge does not know that!
- Let us add some extra information to the merged data:
  - a:author same as f:auteur
  - both identify a “Person”
  - a term that a community may have already defined:
    - a “Person” is uniquely identified by his/her name and, say, homepage
    - it can be used as a “category” for certain type of resources

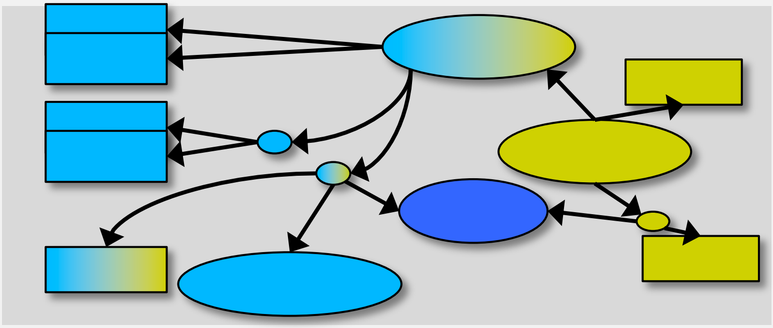
Querying revisited: use the extra knowledge





Start making richer queries!

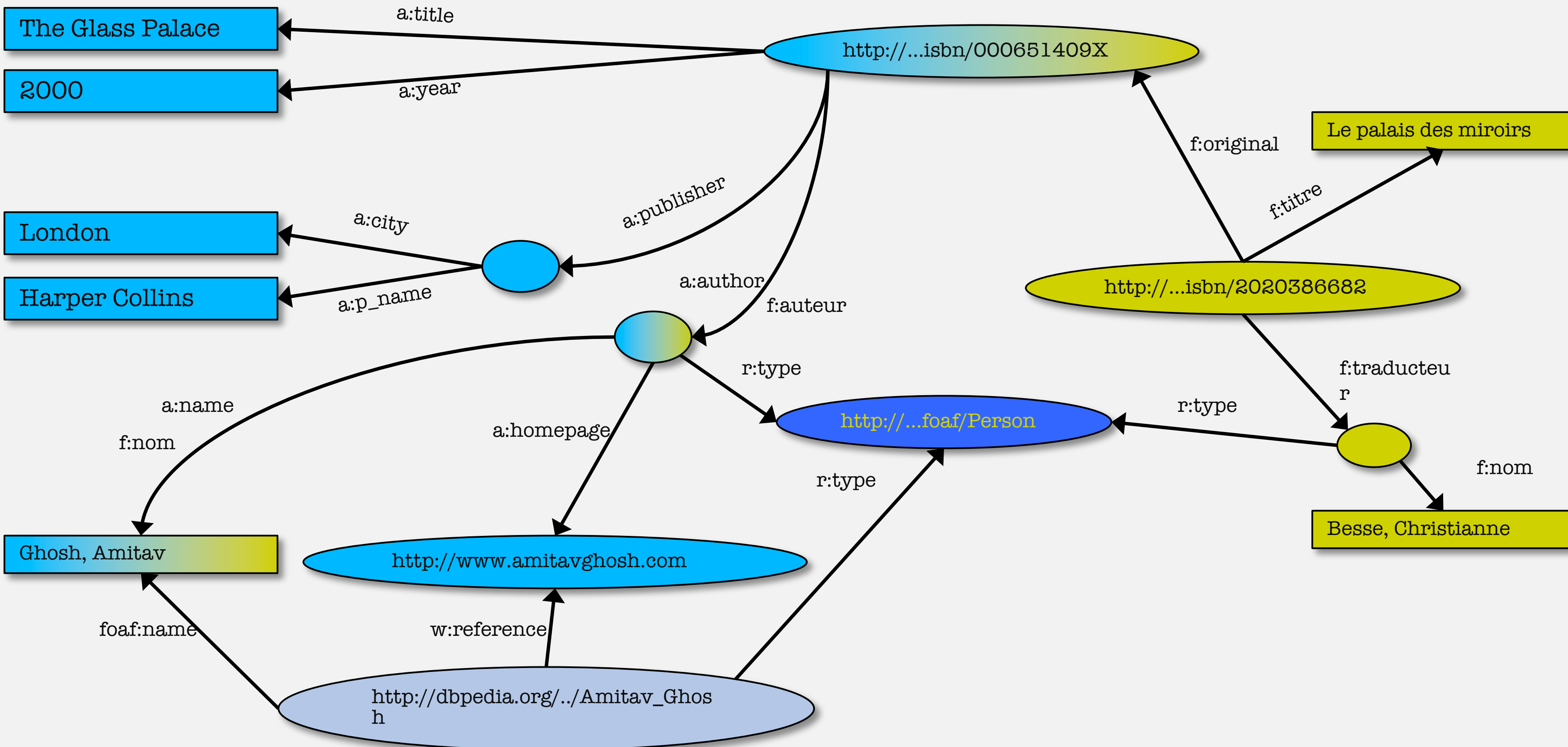
- User of dataset “F” can now query:
  - “donnes-moi la page d’accueil de l’auteur de l’original”
    - well… “give me the home page of the original’s ‘auteur’”
- The information is not in datasets “F” or “A” but was made available by:
  - merging dataset “A” and dataset “F”
  - adding three simple extra statements as an extra “glue”



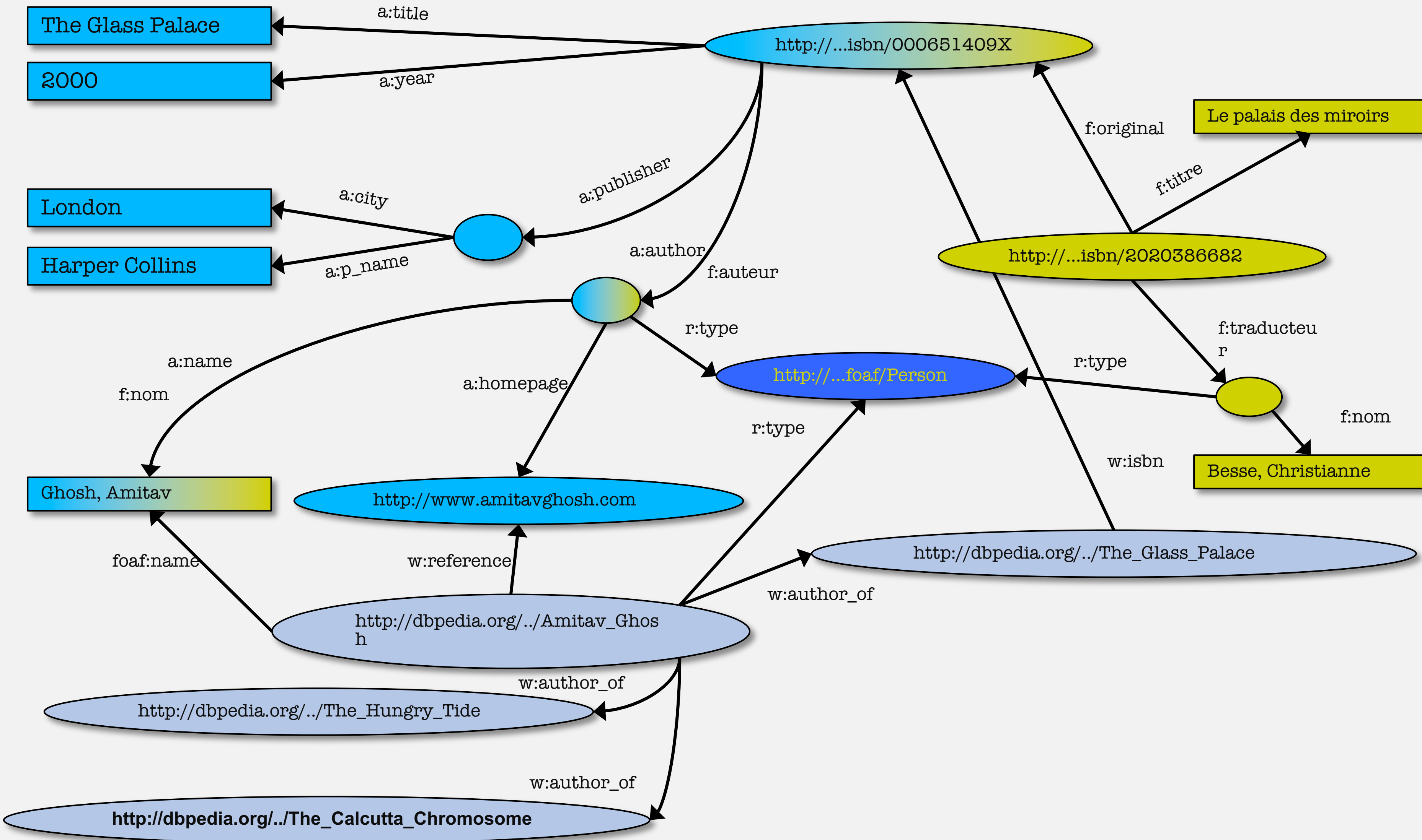
Combine with different datasets

- Using, e.g., the “Person”, the dataset can be combined with other sources
- For example, data in Wikipedia can be extracted using dedicated tools
  - e.g., the “dbpedia” project can extract the “infobox” information from Wikipedia already…

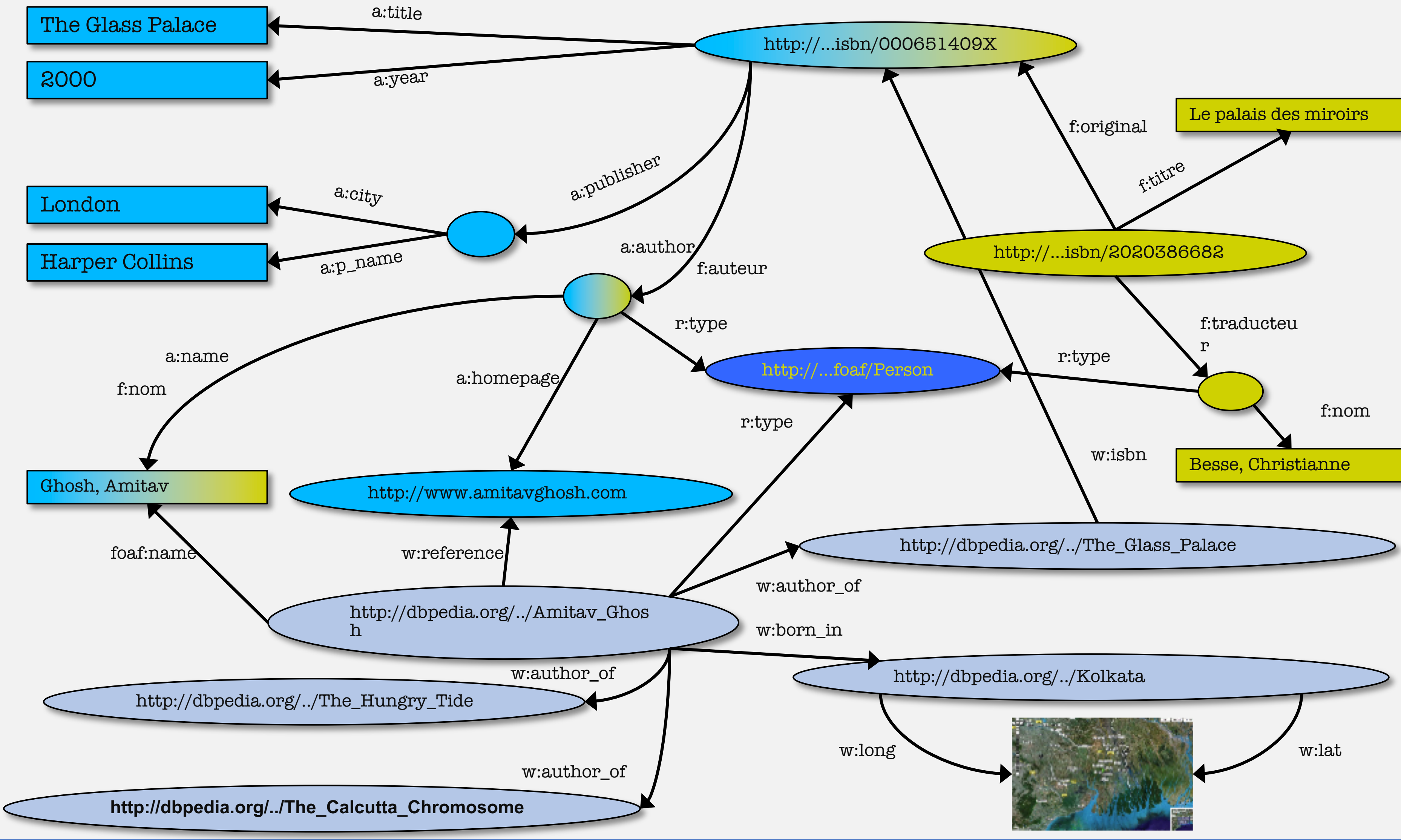
Merge with Wikipedia data



Merge with Wikipedia data



Merge with Wikipedia data



We can do more

- We could add extra knowledge to the merged datasets
  - e.g., a full classification of various types of library data
  - geographical information
  - etc.
- This is where ontologies, extra rules, etc. come in
  - Ontologies/rule sets can be relatively simple and small, or very complex, or anything in between…
- Even more powerful queries can be asked as a result

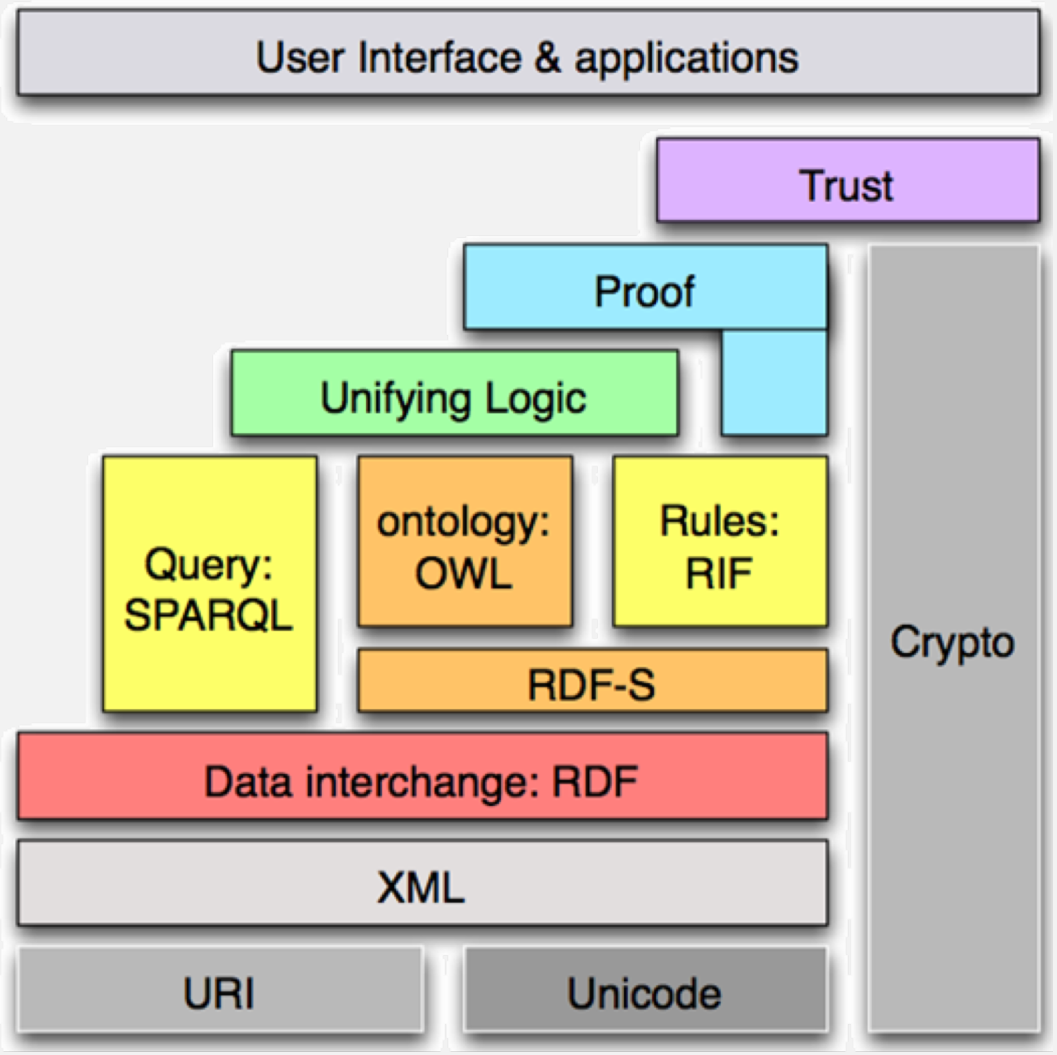
What did we do?

- Expose data on the web in an interoperable form (RDF)
- Expose knowledge on the web with interoperable semantics (ontologies: RDF Schema, OWL)
- Apply (lightweight) inference for
  - Answering complex queries
  - Integrating multiple sources of knowledge and data

Semantic Web

- Semantic web utilizes
  - Low expressivity logic (RDF) that allows for some inference
    - Property inheritance
    - Domain/range inference
  - Medium expressivity logic (OWL) that supports additional inference
    - In(equality)
    - Number restrictions
    - Data types …

Semantic Web Stack (W3C, 2006)

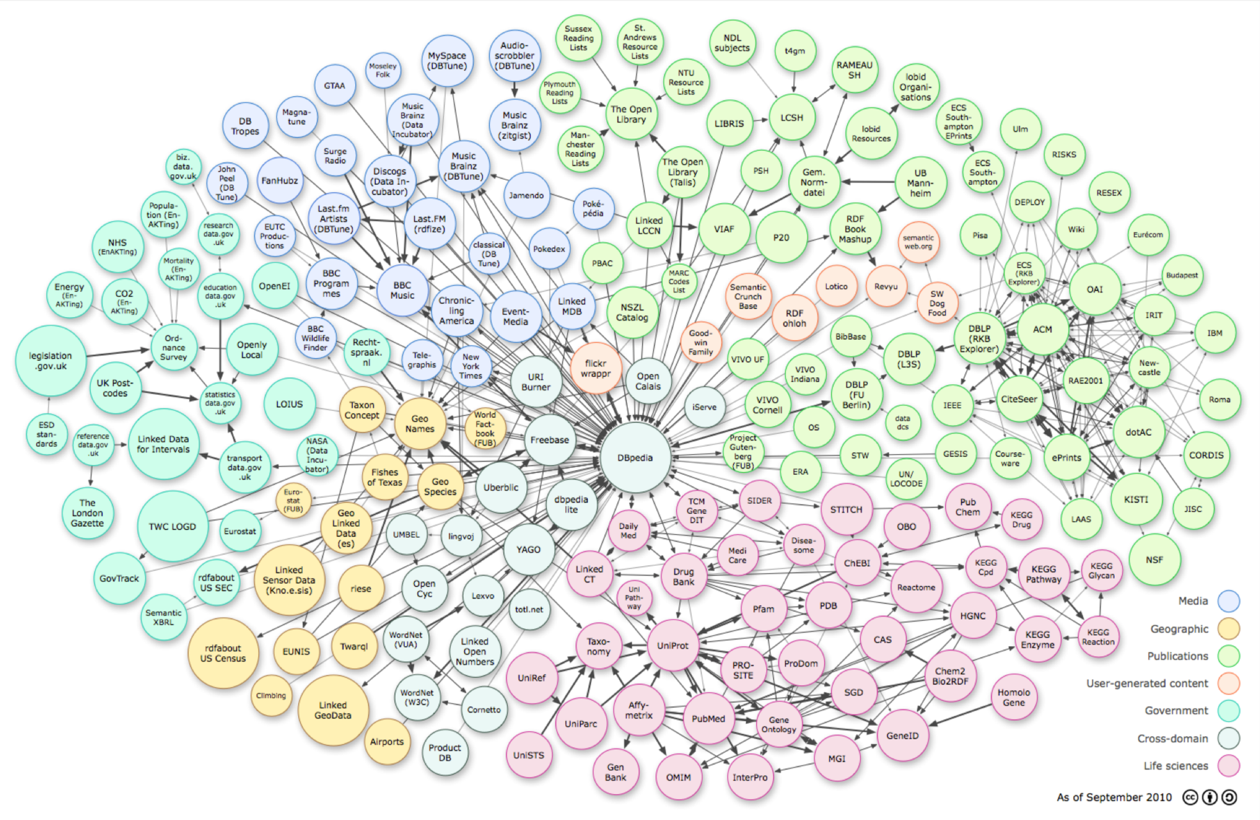


Alternative perspectives on the semantic web

- Semantic web
  - annotated web
  - web of data (RDF triple stores)
- Semantic web
  - In the large
  - In the small

Linked Open Data Initiative

- Goal: “expose” open datasets in RDF
- Set RDF links among the data items from different datasets
- Set up, if possible, query endpoints



Example data source: DBpedia

- DBpedia is a community effort to
  - extract structured ("infobox") information from Wikipedia
  - provide a query endpoint to the dataset
  - interlink the DBpedia dataset with other datasets on the Web

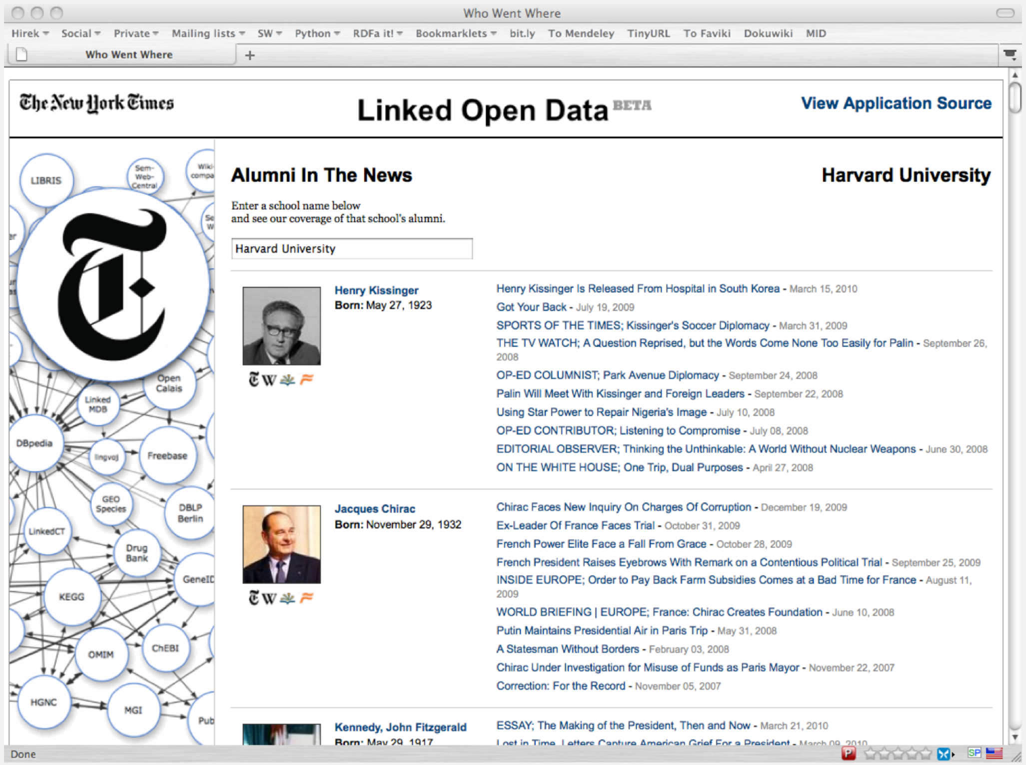
Freie Universität Berlin

UNIVERSITÄT LEIPZIG

OPENLINK SOFTWARE

Linked data enable new applications

- Build your own NYT linked data application



The NYT Linked open data is no longer available from http://data.nytimes.com

However, you can retrieve the data from the Wayback Machine of the Internet Archive.

Description Logic systems

- Brachman and Levesque [1984]:
  - "There is a tradeoff between the expressiveness of a representation language and the difficulty of reasoning over the representations built using that language".
  - The more expressive the language, the harder the reasoning!
- Schmidt-Schauss and Smolka [1991] specialized classical settings for deductive reasoning to Description Logics – tractable subsets of first-order logics
- Best of both worlds
  - Unary predicates for names of classes
  - Binary predicates for properties
  - Conclusions based on inheritance

What Are Description Logics?

- A family of logic based Knowledge Representation formalisms
  - Descendants of semantic networks and KL-ONE [Brachman and Schmolze, 1985]
  - Describe domain in terms of concepts (classes), roles (relationships) and individuals
- Distinguished by:
  - Formal semantics (typically model theoretic)
    - Decidable fragments of FOL
  - Provision of inference services
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimized)

Description Logics

- Description Logics are a subset of first-order logic:
  - Only unary predicates (called concepts) and binary predicates (called roles, properties).
- Knowledge bases are composed of:
  - T-box: defining the concepts and the roles
  - A-box: including ground facts about individuals
- Complex concepts are defined by concept descriptions:
  - The expressive power of the language is determined by the set of constructors in the grammar of concept descriptions
  - Complex roles can also be defined via constructors

Description Logics

- TBox: *terminology*
  - the vocabulary of an application domain:
    - Concepts: sets of individuals
    - Roles: binary relationships between individuals.
  - Examples:
    - Concepts: Person, Female, Mother
    - Role: hasChild
- ABox: *assertions*
  - about named individuals in terms of this vocabulary

DL knowledge base

- Atomic concepts (unary predicates)
- Atomic roles (binary predicates)
- Complex concepts built using constructors

An example Grammar for Concept Descriptions

C,*D* are complex concepts. *A* is a primitive concept.

$C, D \rightarrow$	$A \mid$	(base concept)
	$\top \mid$	(top, the set of all individuals)
	$C \sqcap D \mid$	(conjunction)
	$\neg A \mid$	(complement)
	$\forall R.C \mid$	(universal quantification)
	$(\geq n R) \mid (\leq n R)$	(number restrictions)

Many other constructors possible: union, existential quantification, equality on role paths,…

Example Terminology

a_1. Italian $\sqsubseteq$ Person

a_2. Comedy $\sqsubseteq$ Movie

a_3. Comedy $\sqsubseteq$ $\neg$Documentary

a_4. Movie $\sqsubseteq$ ($\leq$ 1 director)

a_5. AwardMovies := Movie $\sqcap$ ($\geq$ 1 award)

a_6. ItalianHits := AwardMovie $\sqcap$ ($\forall$director.Italian)

a_1: Italians are people

a_2: Comedies are movies

a_3: Comedies are disjoint from documentaries

a_4: Movies have at most one director

a_5: Award movies are those that have at least one award

a_6: Italian hits are award movies whose director is Italian

Abox: the Ground Facts

- A set of assertions of the form *C(a)*, or *R(a,b)*
  - *b* is called an *R*-successor of *a*.
- C and R can be concept descriptions

Comedy(LifeIsBeautiful)
director(LifeIsBeautiful, Benigni)
Italian(Benigni)
award(LifeIsBeautiful, Oscar1997)
ItalianHits(LaStrada)

Semantics of Description Logics

- Semantics are based on interpretations.
- Given a knowledge base Δ, the *models* of Δ are the interpretations that are consistent Δ's T-box and A-box.
- Any fact that is true in all models of Δ is said to be entailed by Δ.

A “Family” Knowledge Base

- Man is a Male Person
- A Woman is a Female Person
- A Man is not a Woman
- A Father is a Man who has a Child
- A Mother is a Woman who has a Child
- Both Father and Mother are Parents
- Grandmother is a Mother of a Parent
- A Mother Without Daughter is a Mother without female Children

DL for Family KB

1. Person
2. Female
3. $\mathsf{Woman} \equiv \mathsf{Person} \sqcap \mathsf{Female}$
4. $\mathsf{Man} \equiv \mathsf{Person} \sqcap \neg\mathsf{Woman}$
5. $\mathsf{Mother} \equiv \mathsf{Woman} \sqcap \exists hasChild.\mathsf{Person}$
6. $\mathsf{Father} \equiv \mathsf{Man} \sqcap \exists hasChild.\mathsf{Person}$
7. $\mathsf{Parent} \equiv (\mathsf{Father} \sqcup \mathsf{Mother})$
8. $\mathsf{Grandmother} \equiv \mathsf{Mother} \sqcap \exists hasChild.\mathsf{Parent}$

Value restrictions

- Individuals all of whose children are male

$$\forall hasChild.Male$$

- Individuals with a female child

$$\exists hasChild.Female$$

- Individuals with at most 3 children and 2 dogs

$$(\leq 3 hasChild) \sqcap (\leq 2 hasDog)$$

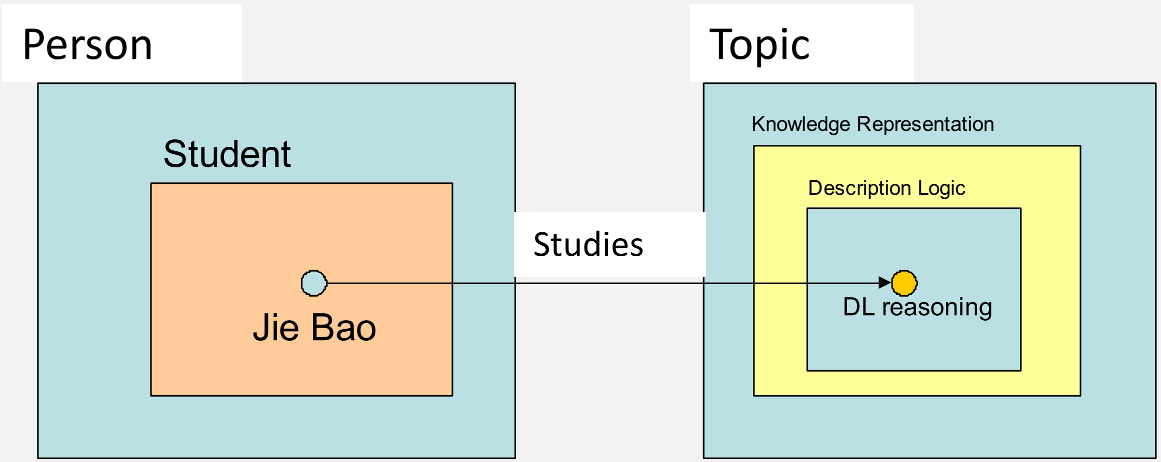
DL Semantics

- DL Ontology: is a set of terms and their relations
- Interpretation of a DL Ontology: A possible world ("model") that materializes the ontology

Ontology:

Student $\subseteq$ Person
Student $\subseteq$ $\exists$Studies.Topic
KR $\subseteq$ Topic
DL $\subseteq$ KR

Interpretation



DL Semantics

- DL semantics defined by interpretations: $I = (\Delta^I, .^I)$, where
  - Δ^I is the domain (a non-empty set)
  - $.^I$ is an interpretation function that maps:
    - Concept (class) name A -> subset A^I of Δ^I
    - Role (property) name R -> binary relation R^I over Δ^I
    - Individual name i -> i^I element of Δ^I
- Interpretation function $.^I$ tells us how to interpret atomic concepts, properties and individuals.
  - The semantics of concept forming operators is given by extending the interpretation function in an obvious way.

DL Semantics: example

- $I = (\Delta^I, \cdot^I)$
- Δ^I = {Jie_Bao, DL_Reasoning}
- $Person^I$=$Student^I$={Jie_Bao}
- $Topic^I$=KR^I=DL^I={DL_Reasoning}
- $Studies^I$={(Jie_Bao, DL_Reasoning)}

An interpretation that satisfies all of axioms in a DL ontology is also called a model of the ontology.

Extensions of Complex Expressions

The extensions of concept and role descriptions are given by the following equations. (#S denotes the cardinality of the set S).

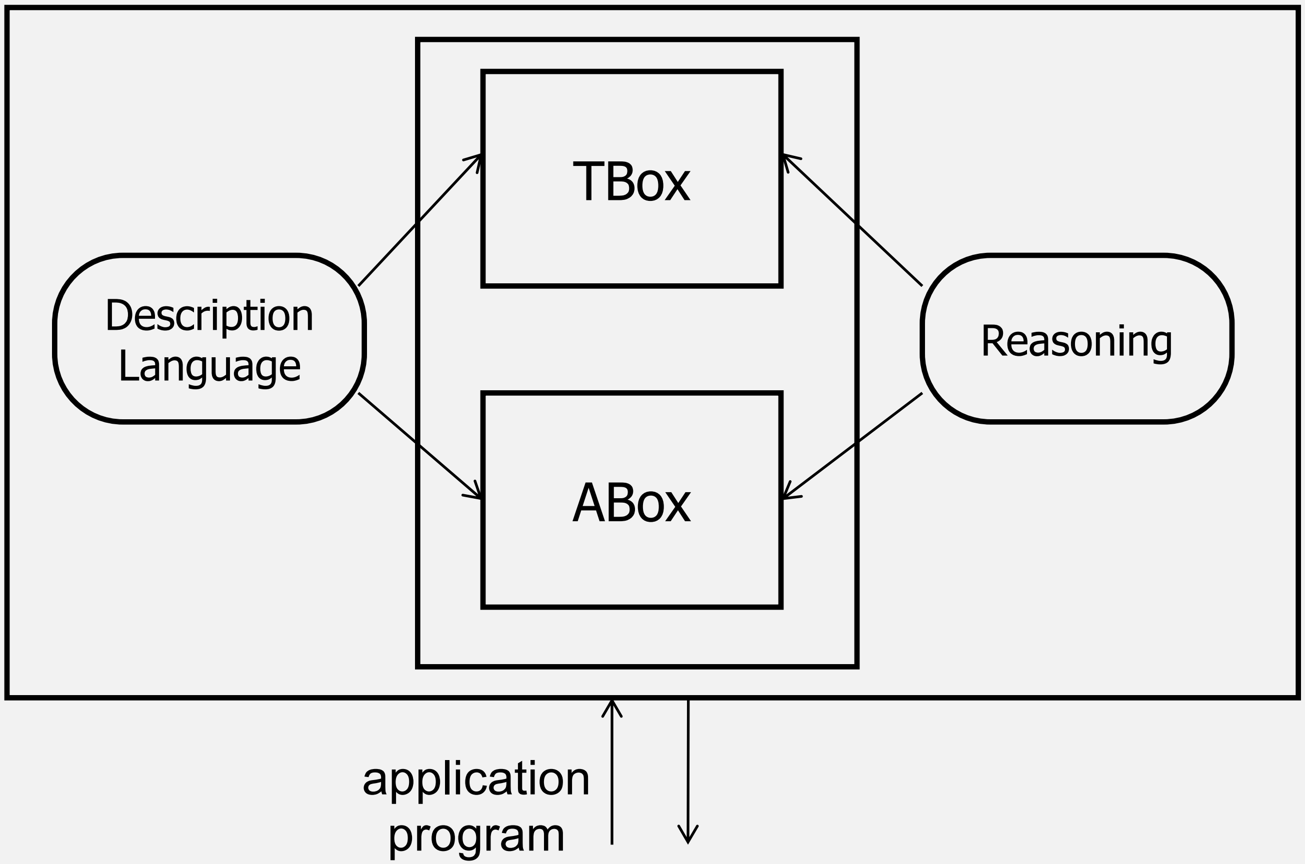
$$
\begin{aligned}
&\top^I = \mathcal{O}^I, \\
&(C \sqcap D)^I = C^I \cap D^I, \\
&(\neg A)^I = \mathcal{O}^I \setminus A^I, \\
&(\forall R.C)^I = \{d \in \mathcal{O}^I \mid \forall e : (d,e) \in R^I \rightarrow e \in C^I\} \\
&(\geq n\, R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \geq n\} \\
&(\leq n\, R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \leq n\}
\end{aligned}
$$

DL and First Order Logic

- Any DL concept C is a unary FOL predicate
- Any DL role is R is a binary FOL predicate
- $\forall R.C \equiv \forall y R(x,y) \Rightarrow C(y)$

  $\exists R.C \equiv \exists y R(x,y) \wedge C(y)$

Description Logic



Description logic

- DL vocabulary consists of **concepts**, that denote sets of individuals, and **roles**, that denote binary relationships between individuals
- DL allow complex descriptions of concepts and roles (Tbox can be used to assign names to them)
- Statements in the TBox and ABox can be identified with formulae in first-order logic

DL Reasoning Tasks

- Determining if a description is satisfiable – satisfiability
- Determining whether one description is more general than another – subsumption
- A-box reasoning
  - to find out if its set of assertions is consistent (has a model, and if individuals are instances of concept descriptions)

Example TBox

$$Woman \equiv Person \sqcap Female$$

$$Man \equiv Person \sqcap \neg Woman$$

$$Mother \equiv Woman \sqcap \exists hasChild.Person$$

$$Father \equiv Man \sqcap \exists hasChild.Person$$

$$Parent \equiv Father \sqcup Mother$$

$$Grandmother \equiv Mother \sqcap \exists hasChild.Parent$$

$$MotherWithManyChildren \equiv Mother \sqcap \geq 3\, hasChild$$

$$MotherWithoutDaughter \equiv Mother \sqcap \forall hasChild.\neg Woman$$

$$Wife \equiv Woman \sqcap \exists hasHusband.Man$$

ABox …

In ABox we describe a specific state of affairs of a given application domain

We introduce individuals, by giving them names, and we asserts properties of these individuals

ABox …

The semantics of ABox is "open-world semantics" – we cannot assume that the knowledge in the knowledge base is complete (contrast with the"closed-world" semantics of classical databases)

DL Syntax and Semantics

Introduction to DL: Syntax and Semantics of $\mathcal{ALC}$

Semantics given by means of an **interpretation** $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$:

Constructor	Syntax	Example	Semantics
atomic concept	A	Human	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
atomic role	R	likes	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
For C, D concepts and R a role name			
conjunction	$C \sqcap D$	Human $\sqcap$ Male	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	Nice $\sqcup$ Rich	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$\neg$ Meat	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
exists restrict.	$\exists R.C$	$\exists$has-child.Human	$\{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restrict.	$\forall R.C$	$\forall$has-child.Blond	$\{x \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$

Other DL Constructors

Introduction to DL: Other DL Constructors

Constructor	Syntax	Example	Semantics
number restriction	$(\geq n\ R)$	($\geq$ 7 has-child)	$\{x \mid \lvert\{y.\langle x, y\rangle \in R^{\mathcal{I}}\}\rvert \geq n$
	$(\leq n\ R)$	($\leq$ 1 has-mother)	$\{x \mid \lvert\{y.\langle x, y\rangle \in R^{\mathcal{I}}\}\rvert \leq n$
inverse role	R^{-}	has-child^{-}	$\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$
trans. role	R^{*}	has-child*	$(R^{\mathcal{I}})^{*}$
concrete domain	$u_1, \ldots, u_n.P$	h-father·age, age. $>$	$\{x \mid \langle u_1^{\mathcal{I}}, \ldots, u_n^{\mathcal{I}}\rangle \in P\}$
etc.			

Many different DLs/DL constructors have been investigated

DL and FOL

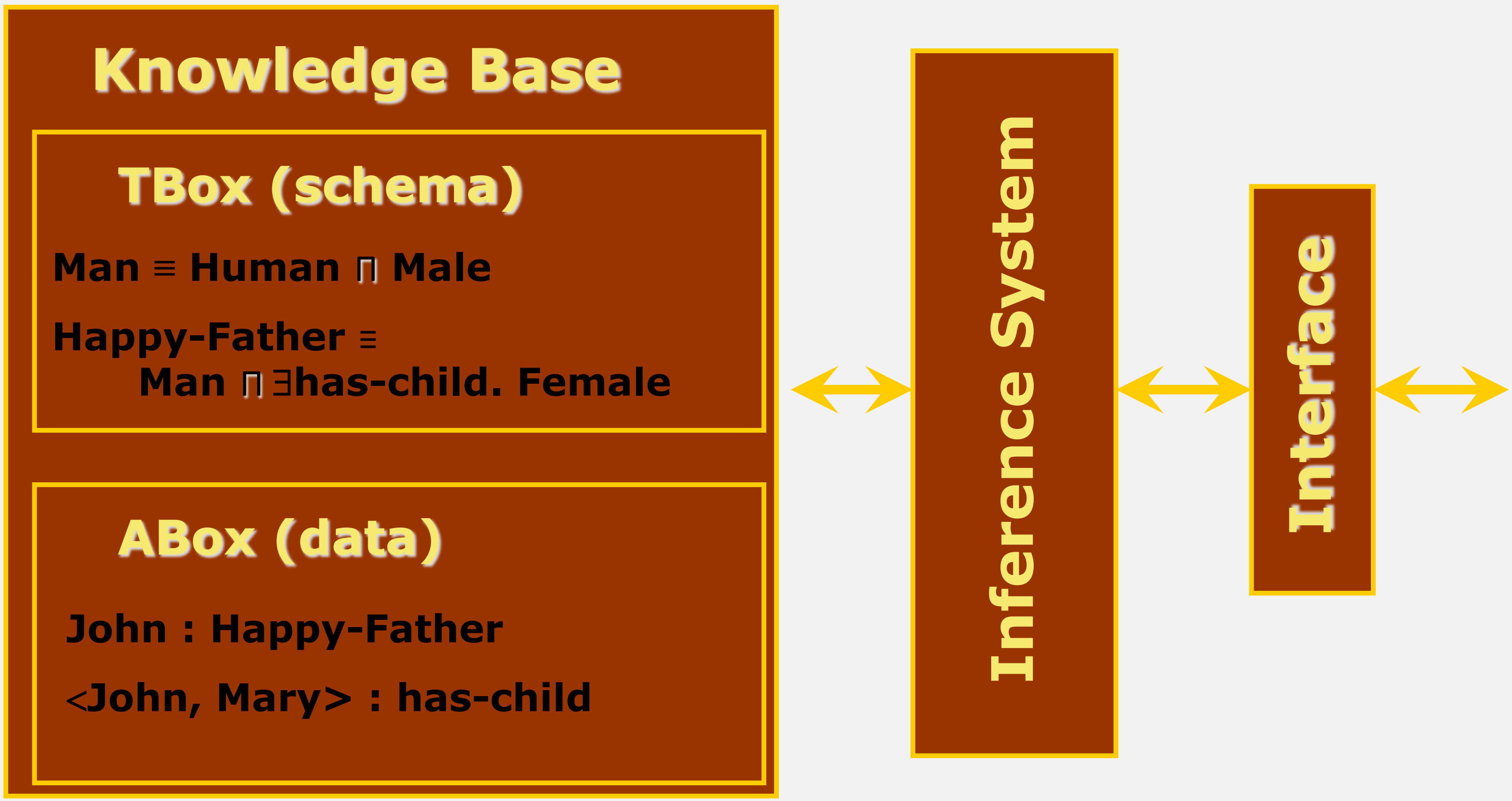
- Syntactic feature of DL: variable free notation.
- Most DLs are fragments of FOL, e.g. ACL.
- ACL expressions can be translated into FOL:
  - A unary predicate Φ_A is introduced for each concept C, and a binary relation ρ_R for each role R.
  - Translation ACL →FOL:

    artist ⊓ *(∃ CREATES. song)* →

    ∀x∃ *y: artist (x) ∧ (CREATES (x, y) ∧ song (y))*
- Why not use FOL?

  The expressive power is too high for having good computational properties and efficient inference procedures.

DL Systems Architecture



DL TBox

Introduction to DL: Knowledge Bases: TBoxes

For terminological knowledge: TBox contains

Concept definitions $A \doteq C$ (A a concept name, C a complex concept)

$\text{Father} \doteq \text{Man} \sqcap \exists\text{has-child.Human}$

$\text{Human} \doteq \text{Mammal} \sqcap \forall\text{has-child}^{-}\text{.Human}$

$\leadsto$ introduce macros/names for concepts, can be (a)cyclic

Axioms $C_1 \sqsubseteq C_2$ (C_i complex concepts)

$\exists\text{favourite.Brewery} \sqsubseteq \exists\text{drinks.Beer}$

$\leadsto$ restrict your models

An interpretation $\mathcal{I}$ satisfies

a concept definition $A \doteq C$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$

an axiom $C_1 \sqsubseteq C_2$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$

a TBox $\mathcal{T}$ iff $\mathcal{I}$ satisfies all definitions and axioms in $\mathcal{T}$

$\leadsto$ $\mathcal{I}$ is a model of $\mathcal{T}$

DL ABox

Introduction to DL: Knowledge Bases: ABoxes

For assertional knowledge: ABox contains

Concept assertions $a : C$ (a an individual name, C a complex concept)

$$\text{John} : \text{Man} \sqcap \forall \text{has-child}.(\text{Male} \sqcap \text{Happy})$$

Role assertions $\langle a_1, a_2 \rangle : R$ (a_i individual names, R a role)

$$\langle \text{John}, \text{Bill} \rangle : \text{has-child}$$

An interpretation $\mathcal{I}$ satisfies

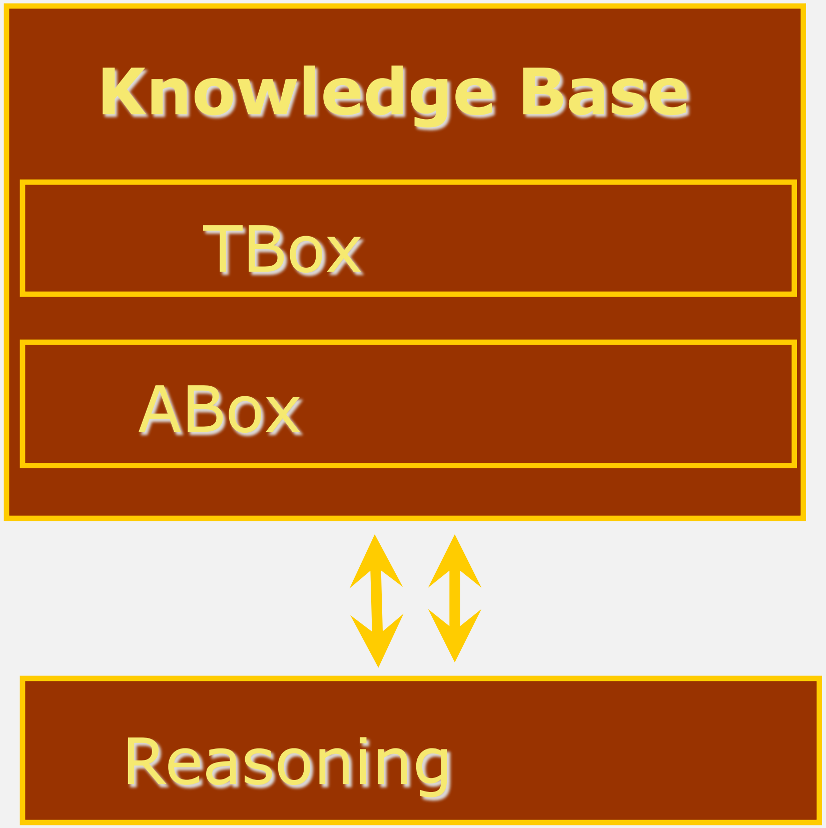
a concept assertion $a : C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$

a role assertion $\langle a_1, a_2 \rangle : R$ iff $\langle a_1^{\mathcal{I}}, a_2^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$

an ABox $\mathcal{A}$ iff $\mathcal{I}$ satisfies all assertions in $\mathcal{A}$

$\leadsto$ $\mathcal{I}$ is a model of $\mathcal{A}$

Knowledge to Reasoning



Reasoning about the knowledge

- Add new knowledge to the KB that follows logically.
- Ask KB if a statement is valid.

Reasoning / Inference

Basic Inference Problems, for TBox T:

- Consistency:

  "A concept C is consistent with respect to T, if there exists a model $\mathcal{I}$ of T with $C^{\mathcal{I}} \neq \emptyset$. [$\mathcal{I}$ is a model of C]".

  Inconsistent:

  $$\text{songwriter} \equiv \text{artist} \sqcap (\exists \text{CREATES. song})$$

  $$\text{song} \equiv \neg \forall \text{IS\_CREATED\_BY. songwriter}$$

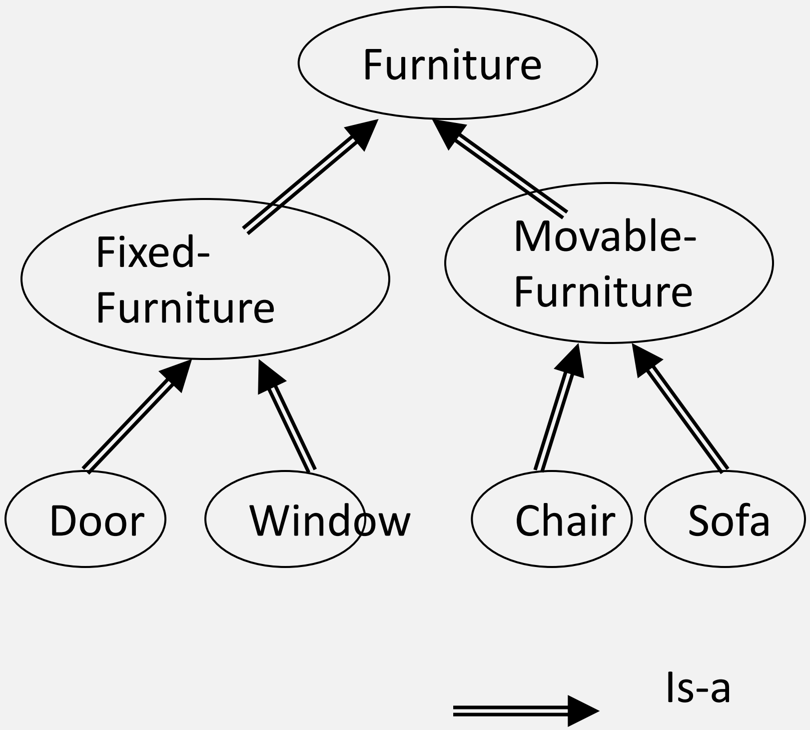
- Subsumption:

  "A concept C is subsumed by a concept D with respect to T if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of T".

  $$\text{male} \subseteq \text{person}$$

Classification

- Classification of concepts
  - allows to structure the terminology in the form of a subsumption hierarchy
- Classification of instances
  - determines whether an individual is an instance of a certain concept.



Reasoning Algorithms

- Structural subsumption algorithms
  - Check if a concept is subsumed by another
  - Complete for simple languages with limited expressive power
  - Used by KL-ONE, LOOM and other systems.
- Tableau–based algorithms
  - Efficient
  - Sound, complete and decidable
  - Used in tools such as RACER
- Resolution based algorithms
  - Transform DL into FOL and FOL reasoners

Tableau-based Algorithms

- Construct a *model* for the input concept description C_0.
- Model is represented by tree form.
- The formula has been translated into Negation Normal Form (NNM).
- To decide satisfiability of the concept C_0, start with the initial tree (root node).
- Repeatedly apply expansion rules until find contradiction or expansion completed.
- Satisfiable *iff* a complete and contradiction-free tree is found.

Reasoning

- Many inference tasks can be reduced to subsumption reasoning
  - C and D are equivalent $\Leftrightarrow C \sqsubseteq D$ and $D \sqsubseteq C$
  - C and D are disjoint $\Leftrightarrow C \sqcap D \sqsubseteq \bot$.
  - a is a member of C $\Leftrightarrow \{a\} \sqsubseteq C$
- Subsumption can be reduced to satisfiability

$$C \sqsubseteq D \Leftrightarrow C \sqcap \neg D \text{ is unsatisfiable}$$

Reasoning

Reasoning services like subsumption and consistency

- Speed-up the inference procedures for query.
- Help to infer implicitly represented knowledge from the explicitly contained knowledge of KB.

T-Box	A-Box
Female $\cup$ Male $\subseteq$ Human	Mary: Mother
Mother $\subseteq$ Female	John: Father
Father $\subseteq$ Male	Mary: $\exists$parent.Child
Child $\subseteq$ $\exists$has.Mother $\sqcap$ $\exists$has.Father	John: $\exists$parent.Child

- Able to deduce implicit knowledge, like Mary is a Human.

Reasoning: Decidability vs. Expressivity

- KR system should
  - answer queries in a reasonable time.
  - The reasoning procedures should terminate.
- Trade-off between the expressivity of DLs and the complexity of their reasoning.
  - Very expressive DLs can have inference problems of high complexity, they may even be undecidable.
  - Very Weak DLs my not be sufficiently expressive to represent the important concepts of an application.

Representative DLs

- ALC: the smallest DL that is propositionally closed
  - Constructors include Booleans (and, or, not),
  - Restrictions on role successors
- SHOIQ = OWL DL
  - S=ALCR$^+$: ALC with transitive role
  - H = role hierarchy
  - O = nominal .e.g WeekEnd = {Saturday, Sunday}
  - I = Inverse role
  - Q = qualified number restriction e.g. >=1 hasChild.Man
    - N = number restriction e.g. >=1 hasChild

DL Concept and Role Constructors

- Range of other constructors found in DLs, including:
  - Number restrictions (cardinality constraints) on roles, e.g., $\geq$3 hasChild, $\leq$1 hasMother
  - Qualified number restrictions, e.g., , $\leq$ 1hasChild.Female, , $\leq$ hasParent.Male
  - Nominals (singleton concepts), e.g., {Italy}
  - Concrete domains (datatypes),
  - Inverse roles, e.g., hasChild^{-} (hasParent)
  - Transitive roles, e.g., hasChild* (descendant)
  - Role composition, e.g., hasParent o hasBrother (uncle)

OWL as DL: Class Constructors

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	$\neg$Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} $\sqcup$ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	$\forall$hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	$\exists$hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	$\leqslant$1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	$\geqslant$2hasChild	$\exists^{\geqslant n} y.P(x,y)$

OWL as DL: Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

OWL DL Semantics

- Mapping OWL to equivalent DL (SHOIN($\mathbf{D}_n$)):
  - Facilitates provision of reasoning services (using DL systems)
  - Provides well defined semantics
- DL semantics defined by interpretations: $I = (\Delta^I, .^I)$, where
  - Δ^I is the domain (a non-empty set)
  - $\cdot^I$ is an interpretation function that maps:
    - Concept (class) name A is a subset A^I of Δ^I
    - Role (property) name R is a binary relation R^I over Δ^I
    - Individual name i is an element of Δ^I

Multiple Models -v- Single Model

- DL KB doesn't define a single model, it is a set of constraints that define a set of possible models
  - No constraints (empty KB) means any model is possible
  - More constraints means fewer models
  - Too many constraints may mean no possible model (inconsistent KB)
- In contrast, DBs make assumptions such that DB defines a single model
  - Unique name assumption
    - Different names always interpreted as different individuals
  - Closed world assumption
    - Domain consists only of individuals named in the DB
  - Minimal models
    - Extensions are as small as possible

Summary

- DL are logic based knowledge representation formalisms.
- DL systems provide efficient inference of consistency, subsumption, etc.
- DLs are effective in a range of applications.

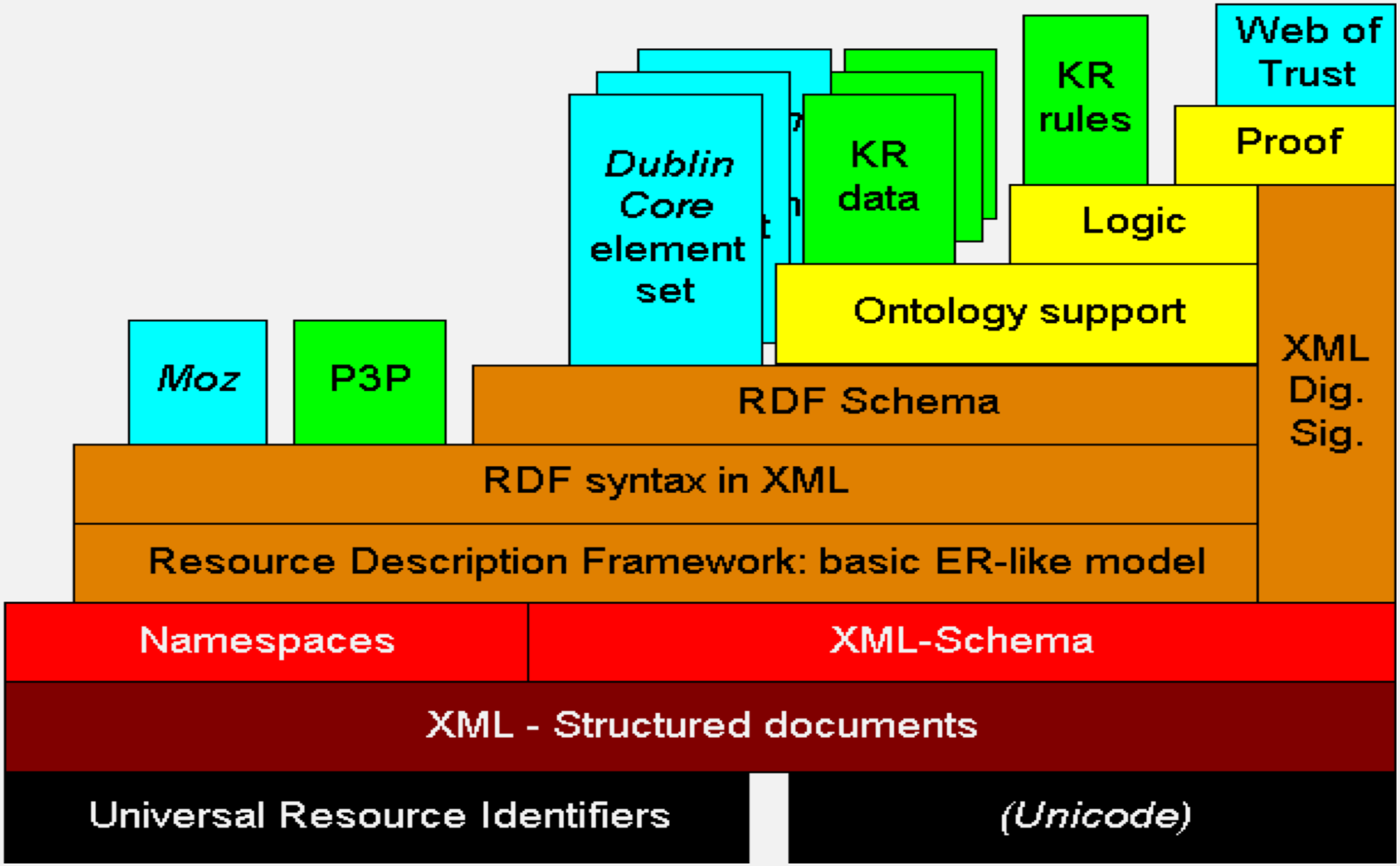
References

- F. Baader, W. Nutt. *Basic Description Logics*. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 47-100.
- Ian Horrocks and Ulrike Sattler. *Description Logics Tutorial*, ECAI-2002, Lyon, France, July 23rd, 2002.
- Ian Horrocks and Ulrike Sattler. *A tableaux decision procedure for SHOIQ*. In Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), 2005.
- I. Horrocks and U. Sattler. *A description logic with transitive and inverse roles and role hierarchies*. Journal of Logic and Computation, 9(3):385-410, 1999.

Semantic web

- URIs
  - You must name something before you can use it
- Triples:
  - subject, predicate, object as a basis for all communication
- RDF Model:
  - Nodes and arcs
- RDF serialization:
  - Graph, RDF/XML, N3
- Ontologies:
  - RDF Schema, OWL

Semantic Web - Language tower



Web “Schema” Languages

- Existing Web languages extended to facilitate content description
  - RDF $\rightarrow$ RDF Schema (RDFS)
- RDFS *is* recognisable as an ontology language
  - Classes and properties
  - Sub/super-classes (and properties)
  - Range and domain (of properties)

Semantic Web

- Expose data on the web in an interoperable form (RDF)
- Expose knowledge on the web with interoperable semantics (ontologies: RDF Schema, OWL)
- Apply (lightweight) inference for
  - Searching for information
  - Answering complex queries
  - Integrating multiple sources of knowledge and data
  - Composing composite services from component services
  - Learning predictive models from disparate data sources
  - Unexpected reuse

Semantic search

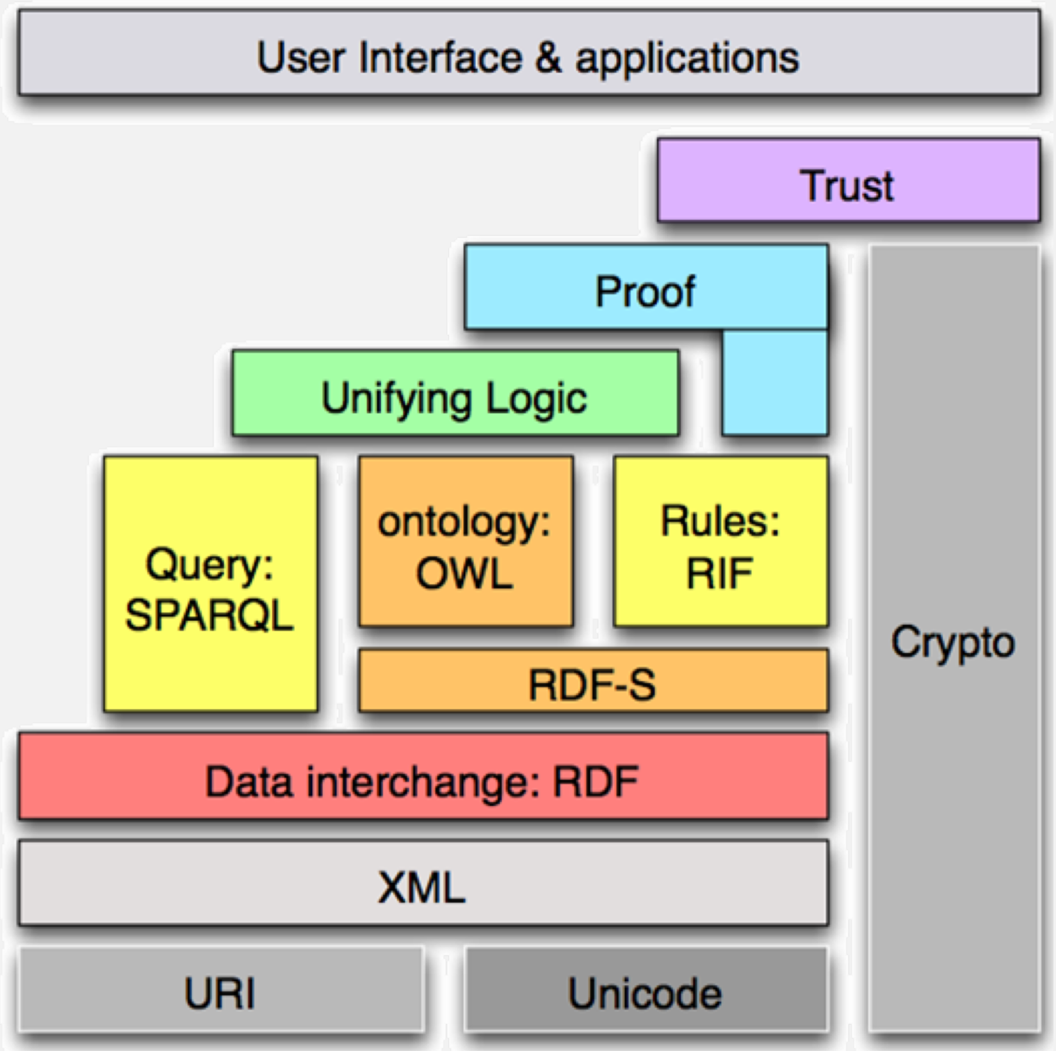
- Semantic search
- Complex queries involving background knowledge
  - Find information about "animals that use sonar but are not either bats, dolphins or whales"
- Integrating information from multiple sources
  - Book me a holiday next weekend somewhere warm, not too far away, and where they speak French or English

Hopeless for machines and tedious for people

Semantic Web

- Expose data on the web in an interoperable form (RDF)
- Expose knowledge on the web with interoperable semantics (ontologies: RDF Schema, OWL)
- Apply (lightweight) inference for
  - Searching for information
  - Answering complex queries
  - Integrating multiple sources of knowledge and data
  - Composing composite services from component services
  - Learning predictive models from disparate data sources
  - Unexpected reuse

Semantic Web Stack (updated, W3C, 2006)



We can do a lot already, but many problems remain

Practically useful yet computationally tractable approaches to

- Representing and using context
- Selective knowledge sharing across ontologies
- Privacy preserving query answering
- Incorporating uncertainty, imprecision
- Representing and reasoning about time and space
- Representing and reasoning about preferences
- Capturing, representing, and reasoning about provenance
- Learning from data and knowledge
- Learning semantics-preserving mappings across ontologies

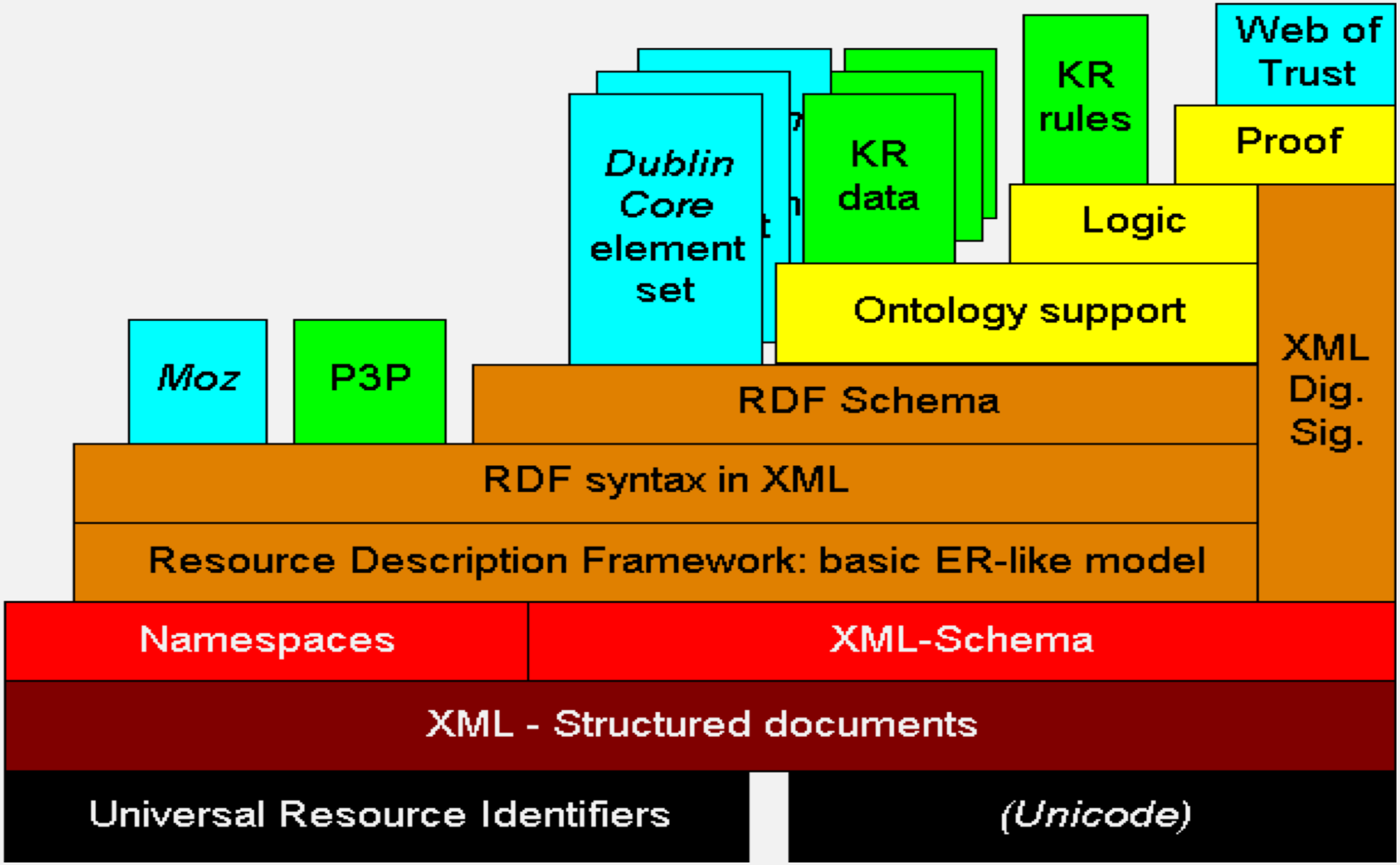
From RDF to OWL

- Two languages developed to satisfy above requirements
  - OIL: developed by group of (largely) European researchers (several from EU OntoKnowledge project)
  - DAML-ONT: developed by group of (largely) US researchers (in DARPA DAML programme)
- Efforts merged to produce DAML+OIL
  - Extends ("Description logic subset" of) RDF
- DAML+OIL submitted to W3C as basis for standardisation
  - WebOnt group developed OWL language based on DAML+OIL
  - OWL language now a W3C Recommendation (i.e., a standard like HTML and XML)

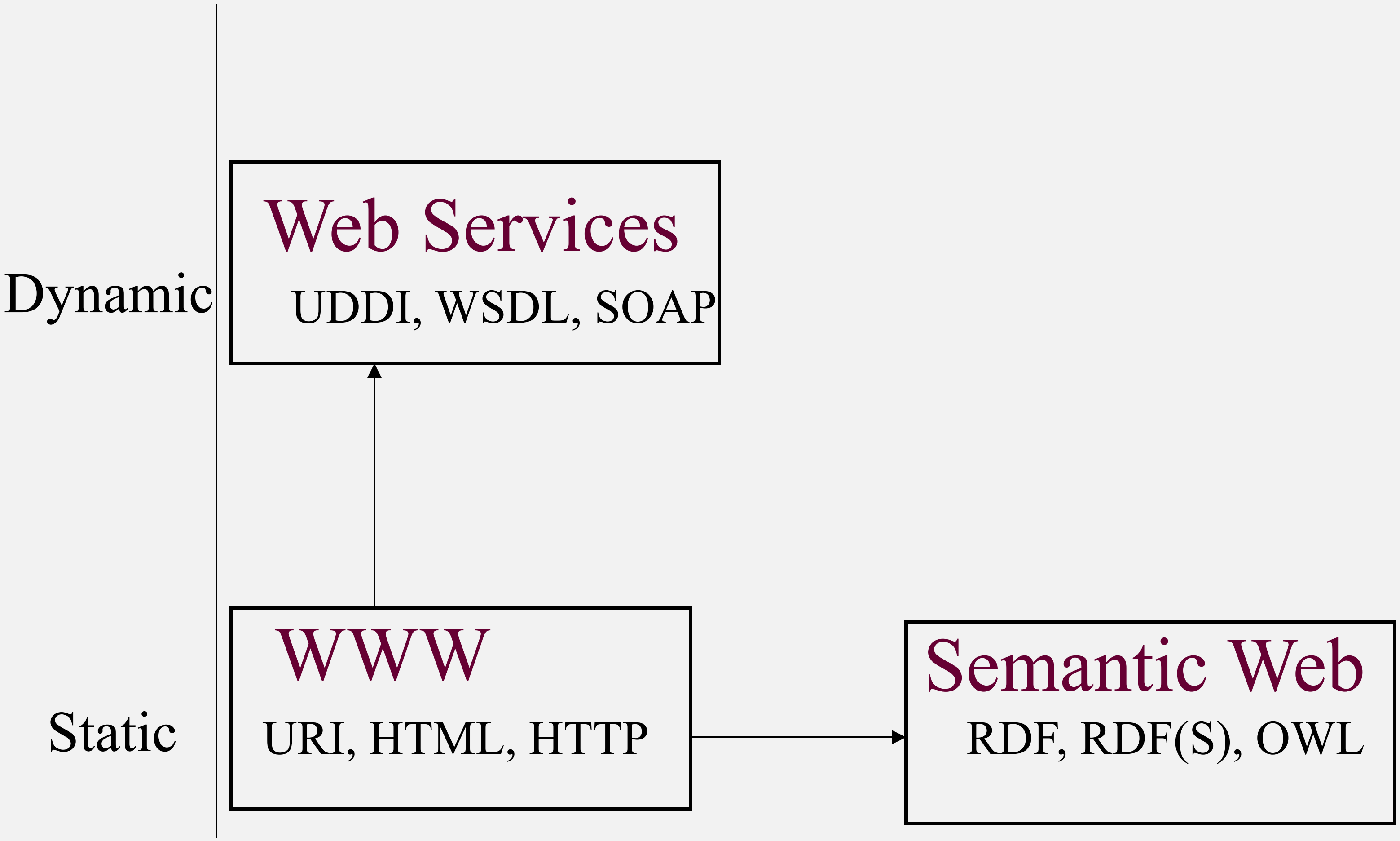
OWL Language

- Three species of OWL
  - OWL full is union of OWL syntax and RDF
  - OWL DL restricted to FOL fragment (¼ DAML+OIL)
  - OWL Lite is "easier to implement" subset of OWL DL
  - DL semantics officially definitive
- OWL DL based on Description Logic
- OWL DL Benefits from many years of DL research
  - Well defined semantics
  - Formal properties well understood (complexity, decidability)
  - Known reasoning algorithms
  - Implemented systems (highly optimised)

Semantic Web - Language tower



Web Services



Web Services

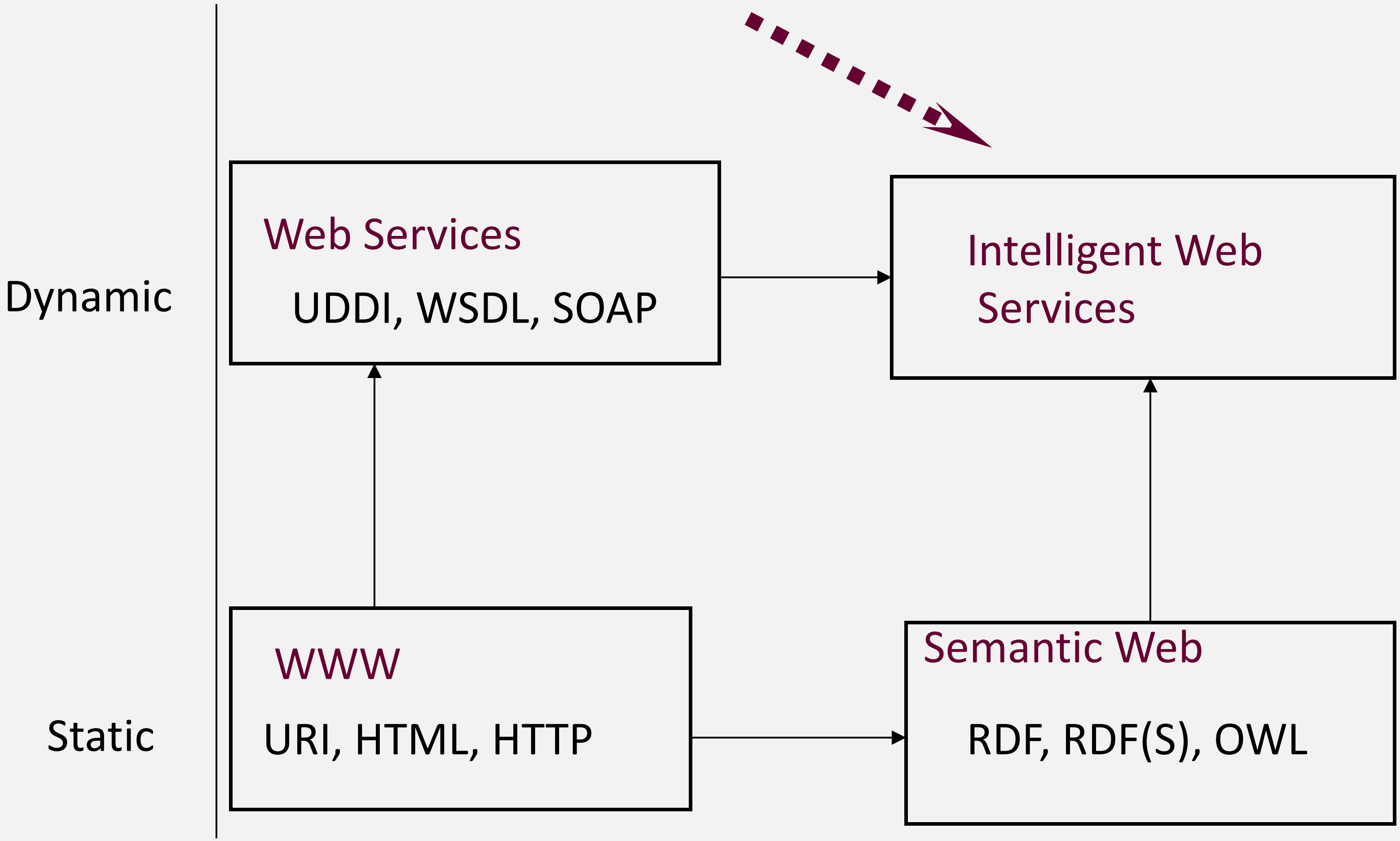
- Self-contained, self-describing, modular applications that can be published, located, and invoked across the Web.
- Perform functions, which can be anything from simple requests to complicated business processes. …
- once deployed, can be discovered and invoked by other services

Web Services

- Web Services connect computers and devices with each other using the Internet to exchange data and combine data in new ways.
- The key to Web Services is on-the-fly software creation through composition of loosely coupled, reusable software components.
- Software composition can be reduced to theorem proving – finding a proof that there exists a composition that meets the specifications

Semantic Web Service

Realizing the full potential of web services



Semantic Web Services

- Semantic Web Services combine Semantic Web and Web Service Technology
- Automating Web Service Discovery, Composition, and Invocation needed to the technology scalable

Summary

- The semantic web
  - Relies on machine-interpretable semantics of data
  - Makes use of formal ontologies with precise semantics
  - Needs KR languages such as RDF, and OWL, and tools that make use of these languages