AI Fundamentals: Knowledge Representation and Reasoning

Maria Simi

Knowledge Representation and Reasoning

LESSON 1: INTRODUCTIONTO KR&R – CLASSICAL LOGIC

Summary

We are introducing an additional level of complexity in the representation of states …

1. Knowledge based systems => rich representation language + inference (the ability to derive new knowledge)
2. These languages rely on classical logic => review (next week)
3. Other representation languages proposed:
   - ✓ For inherent limitations in expressivity of classical logic
   - ✓ For improving efficiency of inference

Knowledge representation and reasoning

Knowledge representation and Reasoning (KR&R) is the field of artificial Intelligence dedicated to representing information about the world in a form that a computer system can utilize to solve complex tasks.

The class of systems that derive from this approach are called **knowledge based (KB) systems/agents**.

A KB agent maintains a **knowledge base** of facts expressed in a **declarative language**, and is able to perform **automated reasoning** to solve complex tasks (accounting for System-2 kind of reasoning)

The KB is the agent's representation of the world which is responsible for intelligent behavior.

References

The classical book about KR&R is:

- ✓ Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 2004.

A more recent book is:

- ✓ F. van Harmelen, V. Lifschitz, B. Porter (eds.) *Handbook of Knowledge Representation*, Elsevier, 2008.

KR&R not so much acquisition

In this part we will deal on how knowledge is represented and reasoned about to derive new knowledge or decide actions.

- Different formalisms, different expressive power, different complexity issues
- Declarative/procedural; logic/non logic based

A separate important issue is how knowledge is acquired:

1. Hand-coded (the traditional assumption)
2. Obtained automatically by mining data and text

… and how it evolves maintaining its anchorage to the world.

Knowledge

What kind of knowledge?

The emphasis is on the relation between an agent and **facts that may be true or false in the world**:

"John knows p", *"John believes p"*, *"John desires p"*, *"John is confident that p"* … where *p* is a **proposition**, something *true* or *false*.

Contrast with **non-factual** knowledge (knowing how, when, where, a person …):

"John knows where the party is"
"John knows how to play the piano"
"John knows Bill very well"

Represented knowledge is given a **propositional account**.

Knowledge representation is about the use of formal symbolic structures to represent a **collection of propositions** believed by some agent. Not necessarily all of them.

Representation

A representaion is a surrogate …

first aid

women

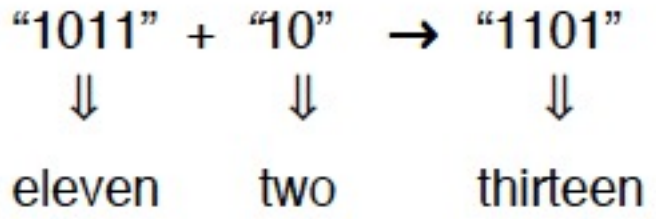
"John" → John

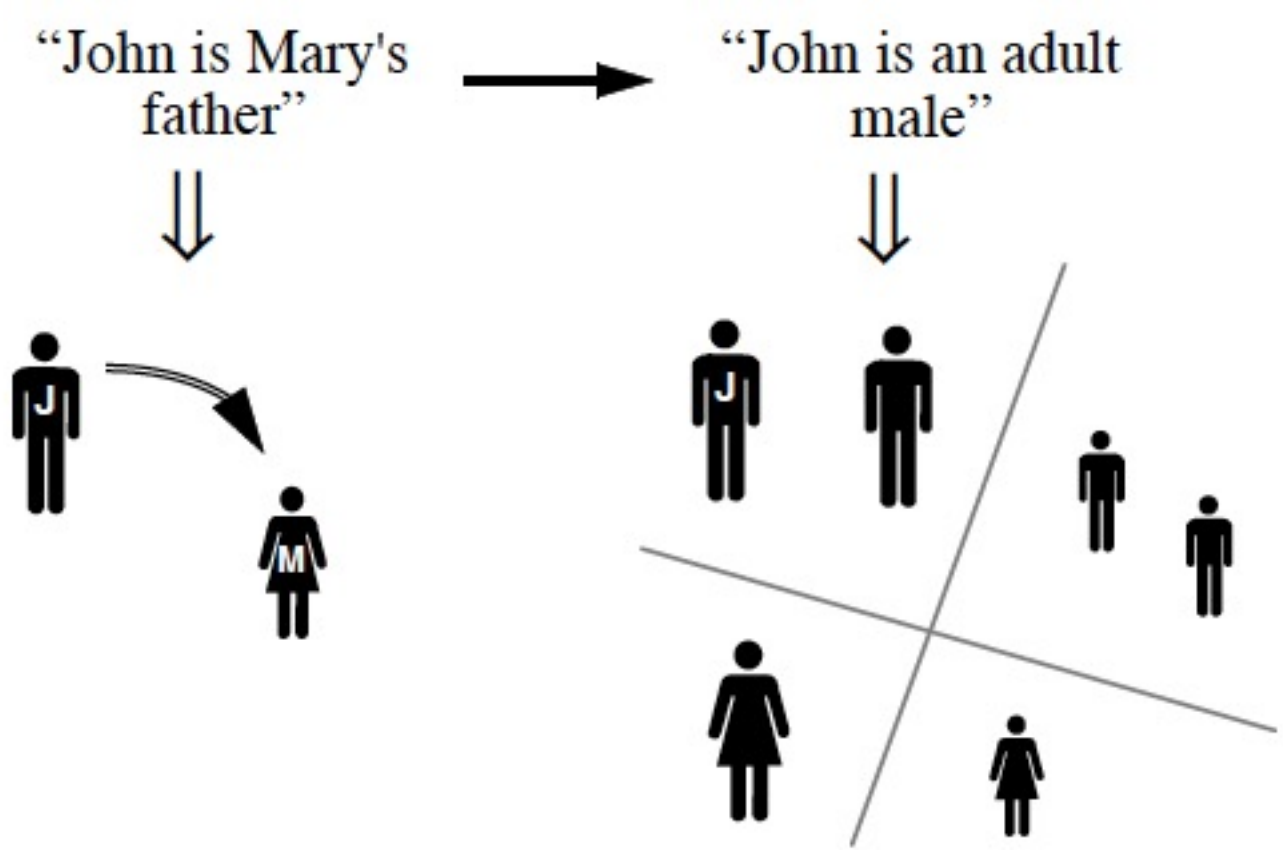
"John loves Mary" → the proposition that John loves Mary

Reasoning

Reasoning is the formal manipulation of the symbols representing a collection of beliefs to produce representations of new ones.

Analogy with arithmetic.





Logical deduction is a well know example of reasoning

Why reasoning?

We would like the system to depend on *what the system believes* and not *what was explicitly stored*. It is a question of economy of the representation. Example:

A. "*Patient x is allergic to medication m*."

B. "*Anybody allergic to medication m is also allergic to m'*."

Is it OK to prescribe *m'* for *x* ?

NO. In fact,

C. "*Patient x is allergic to medication m'*."

Usually, we need more than just DB-style retrieval of facts in the KB

Explicit vs implicit beliefs

Logical entailment (KB ⊨ α) is the fundamental operation.

Other forms of reasoning

- Not only logical entailment/deduction
- Also:
  - Limited expressivity and limited inference
  - Default reasoning
  - Probabilistic reasoning
  - **Abductive reasoning**: given a causal relation $\alpha \Rightarrow \beta$, from observing β, we can *conjecture* α. Abduction is as a way of providing an explanation.
  - **Inductive reasoning**: from specific observations to a general rule

The Knowledge Representation hypothesis

Formulated by Brian C. Smith in 1985:

> *Any mechanically embodied intelligent process will be comprised of structural ingredients that*
>
> *a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and*
>
> *b) independent of such external semantic attribution, play a formal but causal and essential role in engendering the behavior that manifests that knowledge.*

In simpler words, we want to construct A.I. systems that contain symbolic representations with two important properties:

1. We can understand these symbolic structures as propositions.
2. These symbolic structures determine the behavior of the system

Knowledge based systems have these properties.

Do you really need KR&R?

Competing approaches:

- **Procedural approach**: knowledge is embedded in programs.
- **Connectionist approach:** avoids symbolic representation and reasoning, and instead model intelligent behavior by computing with networks of weighted links between artificial “neurons.”

Advantages of KB systems

1. They try solving open-ended tasks, not pre-compiled kind of behavior for specific tasks
2. **Separation of knowledge and "inference engine".**
3. **Extensibility**: we can extend the existing behavior by simply adding new propositions. Knowledge is modular, the reasoning mechanism does not change.
4. **Understandability**: the system can be understood at the **knowledge level**.
   - ✓ **Debugging**: we can debug faulty behavior by changing erroneous beliefs.
   - ✓ **Accountability**: the system can explain and justify current behavior in terms of the knowledge used.

The interplay between representation and reasoning

Representation and **Reasoning** are intimately connected In A.I. research.

1. The representation scheme must be expressive enough to describe many aspects of complex worlds with symbolic structures.
2. The reasoning mechanism needs to ensure that reasoning can be performed efficiently enough.

There is a tradeoff between these two concerns.

The **fundamental tradeoff in knowledge representation and reasoning**: *the more expressive the representation language, the more complex is the reasoning*. We need to find the best compromise.

[Seminal paper in bibliography:

A fundamental tradeoff in knowledge representation and reasoning, by Levesque and Brachman.]

The fundamental trade-off

Expressivity of representation language:

- Does not concern what *can be said* but what *may be left unsaid*: it is related to the possibility of expressing uncertainty and incomplete information.

Complexity of inference:

- The computational cost of deciding entailment: $\mathrm{KB} \models \alpha$

The more expressive the language, the more complex is reasoning.

- The case of databases: only positive and concrete facts, no general rules, no inference (only retrieval). Closed World Assumption.
- The case of FOL

Applications

Much of AI involves building systems that are **knowledge-based:** their ability derives [in part] from reasoning over explicitly represented knowledge. Typical KB systems are:

- expert systems
- decision support systems
- diagnostic systems
- planning

Some systems are KB, to a certain extent

- language understanding, machine reading, QA, personal digital assistents …
- game-playing, vision, etc.

Some systems are KB to a much lesser extent

- speech, motor control, image processing etc.

KR&R today has many applications outside AI:

- Bio-medicine, Engineering, Business and commerce, Databases, Software engineering

Current research question: how much of intelligent behavior is knowledge-based?

Knowledge representation and classical logic

A REVIEW OF PROP AND FOL - SAT ALGORITHMS – THE RESOLUTION INFERENCE METHOD.

The role of “classical logic”

Sound logic inference (classical logic and entailment) is the place to start.

$KB \models \alpha$

By “classical logic” we mean

- ✓ Propositional calculus (PROP)
- ✓ First Order predicate Logic (FOL)

We can understand KB systems at two different levels [A. Newell].

Knowledge level: representation language and its semantics; expressive adequacy (what can be expressed), what can be inferred.

Symbol level: computational aspects, efficiency of encoding, data structures and efficiency of reasoning procedures, including their complexity.

The tools of symbolic logic seem especially suited for the knowledge level.

What you need to know about logic

- ✓ Truth, satisfiability and validity
- ✓ Logical entailment KB $\models \alpha$
- ✓ Logical deduction KB $\vdash \alpha$
- ✓ Soundness and Completeness of a formal inference system
- ✓ Proof by refutation
- ✓ Transformation in clausal form
- ✓ Basic inference techniques and their decidability/complexity

Truth, satisfiability, validity

- A sentence is **true** (or false) with respect to an interpretation.
- An **interpretation** (possible world) assigns truth values to atomic formulas. Truth values of compound formulas follow as a consequence. An interpretation that makes a set of formulas true, is called a **model**.
- A formula is **satisfiable** if there is at least one interpretation that makes it true. **Unsatisfiable** if it is false in all interpretations.
- A formula is **valid** (a **tautology** in PROP) if it is true in all interpretations.
- **Note**:
  1. The negation of a satisfiable formula can be satisfiable or unsatisfiable.
  2. The negation of a valid formula is unsatisfiable
  3. The negation of an unsatisfiable formula is valid

Logical entailment, equivalence

- A set of sentences KB [logically] **entails** a sentence α iff any model of KB is also a model of α. We also say that α is a **logical consequence** of KB.

  We write: $KB \models \alpha$

- Alternative definition:

  $KB \models \alpha \; iff \; M(KB) \subseteq M(\alpha)$ where M stands for the set of models

- **Logical equivalence**:

  $\alpha \equiv \beta \; iff \; \alpha \models \beta$ e $\beta \models \alpha$

Logical deduction

- A deductive system is defined by a set of axioms and inference rules. Axioms can be logical or part of the agent's KB.
- A **proof** is a sequence of formulas, starting from the axioms, where each formula can be obtained from previous ones by application of inference rules.

  We write: $KB \vdash \alpha$ when there is a proof of α from KB
- Examples of inference rules:

$$\frac{\alpha,\ \beta}{\alpha \wedge \beta} \qquad \frac{\alpha}{\alpha \vee \beta} \qquad \frac{\alpha,\ \alpha \Rightarrow \beta}{\beta}$$

Connecting deduction and entailment

- **Soundness**

  if KB $\vdash \alpha$ then KB $\models \alpha$

  If there is a proof of α from the axioms in KB, then KB entails α.

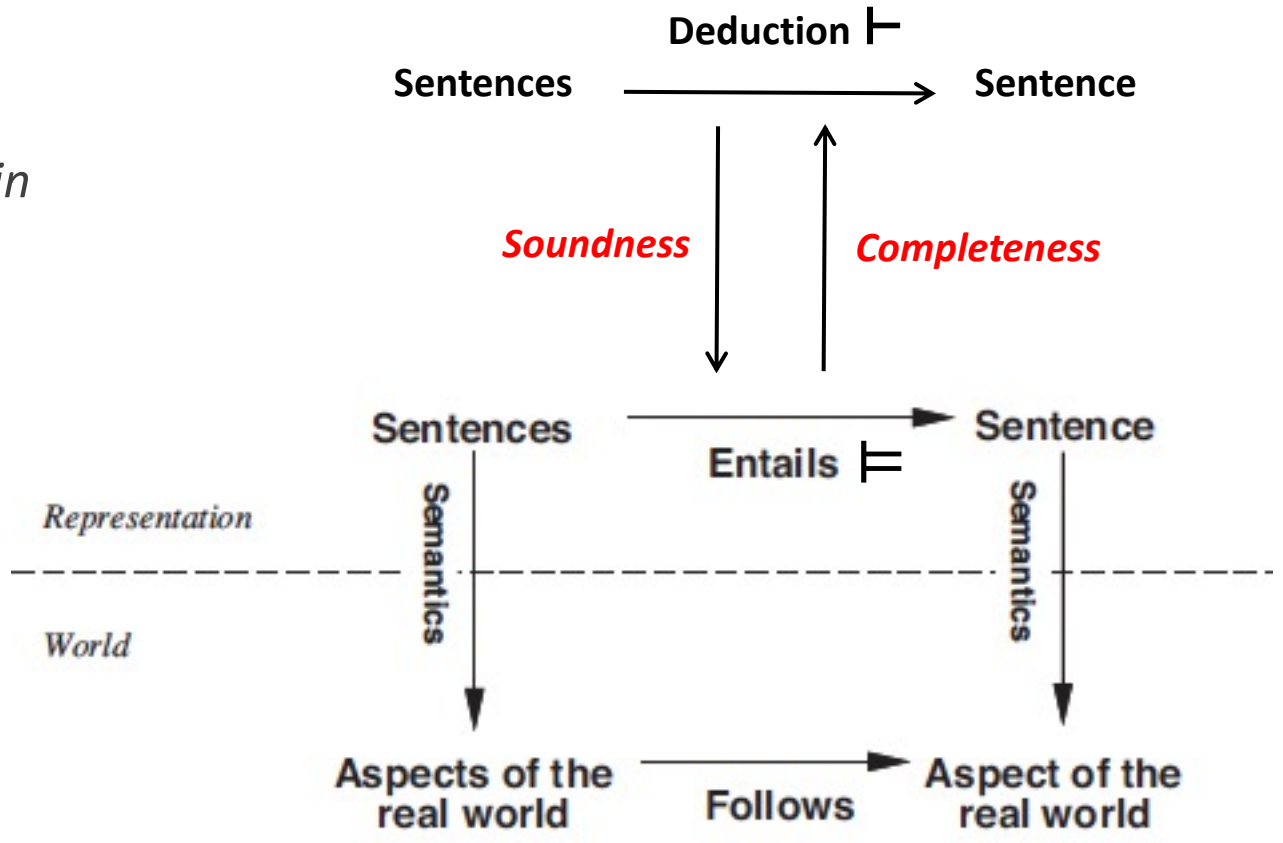
  Each inference rule preserves truth.

- **Completeness**

  if KB $\models \alpha$ then KB $\vdash \alpha$

  If KB entails α then it is possible to construct a proof of α from the axioms in KB.

  The inference system ***as a whole*** has enough rules.



Connecting *entailment* and *satisfiability*

- Proof by **refutation** is based on the following *meta-theorem*

  $KB \models \alpha$ *iff* $KB \cup \{\neg\alpha\}$ is unsatisfiable

- This means can be used in wo ways:
  1. Entailment can be formulated as a **satisfiability problem (SAT)**: we can use SAT algorithms for checking entailment.
  2. If $KB \cup \{\neg\alpha\}$ derives a contradiction, and the proof system is sound, then $KB \models \alpha$

Review of PROP fundamentals (see AIMA)

- ✓ Syntax and semantics
- ✓ Transformation in clausal forms
- ✓ Algorithms for satisfiability (DPLL, Walk-SAT)
- ✓ The resolution method

[AIMA cap. 7]

PROP syntax and semantics

$$
\begin{aligned}
\mathit{Sentence} &\rightarrow \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} &\rightarrow \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} &\rightarrow (\ \mathit{Sentence}\ ) \mid [\ \mathit{Sentence}\ ] \\
&\mid \neg\ \mathit{Sentence} \\
&\mid \mathit{Sentence} \wedge \mathit{Sentence} \\
&\mid \mathit{Sentence} \vee \mathit{Sentence} \\
&\mid \mathit{Sentence} \Rightarrow \mathit{Sentence} \\
&\mid \mathit{Sentence} \Leftrightarrow \mathit{Sentence}
\end{aligned}
$$

OPERATOR PRECEDENCE : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

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*

Strategies for computing entailment in PROP

1. **Model checking (reduce to SAT problem)**
   - ✓ Check satisfiability using different heuristics (DPLL)
   - ✓ Use a local and incomplete search method (Walk-SAT)
   - ✓ SAT is decidable and NP-complete for PROP
2. **Deduction** (uses a proof system and deduction strategies)
   - ✓ The **resolution method** uses only one inference rule, the **resolution rule**
   - ✓ Resolution strategies for searching more efficiently

Clausal form (PROP)

- Conjunctive normal form (a conjunction of disjunctions of atomic formulas).
- Any PROP formula can be converted in an equivalent set of clauses. Each conjunct is a **clause**, a disjunction of **literals** (positive o negative atoms).

$$\{a, b\} \quad \{a, \neg b, c\} \quad \{\neg c, d\}$$

- Suppose:

$$G = (((a \wedge b) \vee (\neg a \wedge \neg b)) \wedge \neg c) \vee d$$

The classical algorithm (also in AIMA tools) produces:

$$((a \vee \neg a) \wedge (b \vee \neg a) \wedge (a \vee \neg b) \wedge (b \vee \neg b) \wedge \neg c)) \vee d$$

$$(a \vee \neg a \vee d) \wedge (b \vee \neg a \vee d) \wedge (a \vee \neg b \vee d) \wedge (b \vee \neg b \vee d) \wedge (\neg c \vee d)$$

$$\{a, \neg a, d\} \ \{b, \neg a, d\} \ \{a, \neg b, d\} \ \{b, \neg b, d\} \ \{\neg c, d\}$$

DPLL (Davis, Putman, Lovemann, Loveland)

It requires a formula in clausal form.

It enumerates, with a depth first strategy, all interpretations, **looking for a model**.

It uses three strategies:

1. Anticipated control:
   - if one clause is false backtrack;
   - if one literal is true the clause is satisfied
2. Pure symbols heuristics
   - Assign first **pure symbols** (which appear everywhere with the same sign)
3. Unit clauses heuristics
   - Assign first **unit clauses** (only one literal)

DPLL at a glance

- A complete algorithm for checking satisfiability of a formula s.

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

  clauses ← the set of clauses in the CNF representation of s
  symbols ← a list of the proposition symbols in s
  return DPLL(*clauses*, *symbols*, { })

---

function DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

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

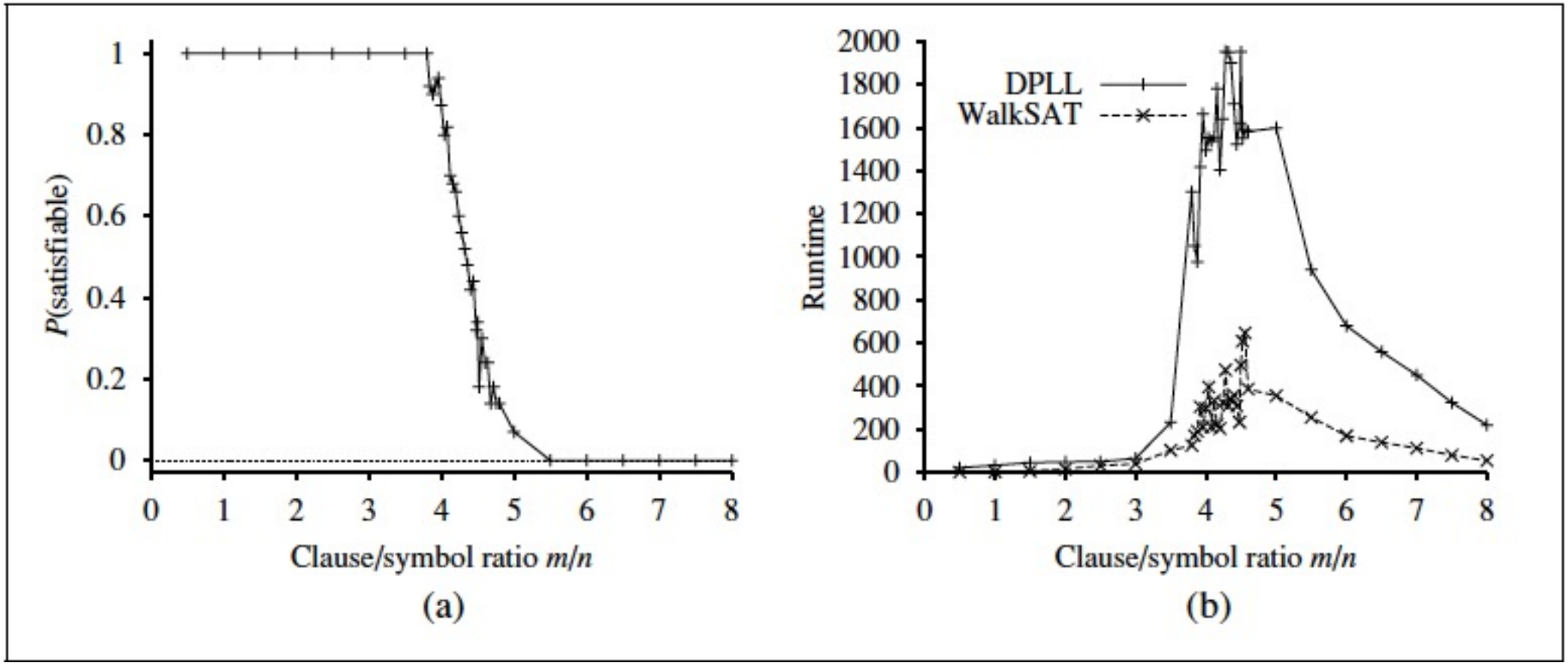
Walk-SAT at a glance

- Walk-SAT is one (of many) **local search** methods.
- To be used for SAT problems where solutions exist and are evenly distributed.
- Not complete. Very effective for large problems.

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

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

Comparison of DPLL and WalkSAT



Relation of PROP with CSP

- Logic provides a convenient language to express constraints on Boolean variables

  $Happy$ means $happy = True$ $\quad\quad$ $\neg Happy$ means $happy = False$

  $Happy \vee Sad \vee \neg Living$ is a constraint on three variables excluding one assignment

- It is possible to convert any finite domain CSP into a PROP satisfiability problem:

  1. If Y has domain $\{v_1, \ldots v_k\}$ introduce k Boolean variables $Y_1, \ldots Y_k$, with Y_i meaning $Y = v_i$
  2. Constraints: at least one of $Y_1, \ldots Y_k$ is true and if $i \neq j$ then Y_i and Y_j not both true
  3. A clause for each disallowed combination of values $(v_1, \ldots v_k)$: $\{\neg v_1 \vee \ldots \vee \neg v_k\}$

- Satisfiability (SAT) algorithms for PROP can be made more efficient than general CSP solvers. Techniques can be transferred [AIFCA, 5.2]

First Order Logic (FOL)

Review of FOL fundamentals (see AIMA)

- ✓ Syntax and semantics
- ✓ Transformation in clausal forms (including Skolemization)
- ✓ Unification
- ✓ The resolution method for FOL

Syntax

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \mid [\textit{Sentence}] \\
&\mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\mid \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
\textit{Term} &\rightarrow \textit{Function}(\textit{Term}, \ldots) \\
&\mid \textit{Constant} \\
&\mid \textit{Variable}
\end{aligned}
$$

$$
\begin{aligned}
\textit{Quantifier} &\rightarrow \forall \mid \exists \\
\textit{Constant} &\rightarrow A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} &\rightarrow a \mid x \mid s \mid \cdots \\
\textit{Predicate} &\rightarrow \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} &\rightarrow \textit{Mother} \mid \textit{LeftLeg} \mid \cdots
\end{aligned}
$$

Clausal form (FOL)

- Also in the case of FOL you can obtain a clausal form in an **effective** way.
- The transformation involves the elimination of existential quantifiers by **Skolemization**.

$\exists x\, Father(x, G)$	becomes	$Father(K, G)$	K new costant symbol
$\forall x \exists y\, Father(x, y)$	becomes	$\forall x\, Father(x, F(x))$	
	rather than	$\forall x\, Father(x, K)$	

- The transformation preserves satisfiability (not equivalence)

Resolution theorem proving [Robinson]

Given the fundamental problem KB $\models \alpha$ an equivalent problem is

KB $\cup \neg\alpha$ is unsatisfiable

This can be solved by a correct deductive system showing that $\mathrm{KB} \cup \neg\alpha \vdash \{\}$, where $\{\}$ is the empty clause meaning *False*.

Resolution by refutation is a method which is *correct* and *complete*:

1. transform the KB **in clausal form** (a conjunction of disjunction of literals)
2. add to KB the negation of the goal, converted in clausal form
3. use the **resolution rule** as **unique** inference rule.

This strategy works for PROP and FOL with different complexity results:

1. It is decidable and NP-complete for PROP
2. It is semi-decidable for FOL.

Resolution rule in PROP

Clauses are sets of literals, i.e. atomic formulas or their negation: $\{p_1, p_2, \ldots p_k\}$

Resolution rule for PROP (c_1 and c_2 are clauses):

$$\frac{c_1 \cup \{p\} \qquad c_2 \cup \{\neg p\}}{c_1 \cup c_2} \qquad (\textit{the resolvent})$$

In a resolution refutation we aim to deduce the empty clause:

$$\frac{\{p\} \qquad \{\neg p\}}{\{\,\}}$$

Unification and the resolution rule in FOL

A fundamental operation is **unification,** which computes a **substitution** that makes two expressions identical.

Substitution: $\{x/A, y/B, z/C\}$

Unifier: $P(A, y, z)$ e $P(x, B, z)$ can be made identical with the substitution

$\tau=\{x/A, y/B, z/C\}$

$\sigma=\{x/A, y/B\}$ is also a unifier and is *more general* than τ

Unification returns the Most General Unifier (**MGU**) or FAIL

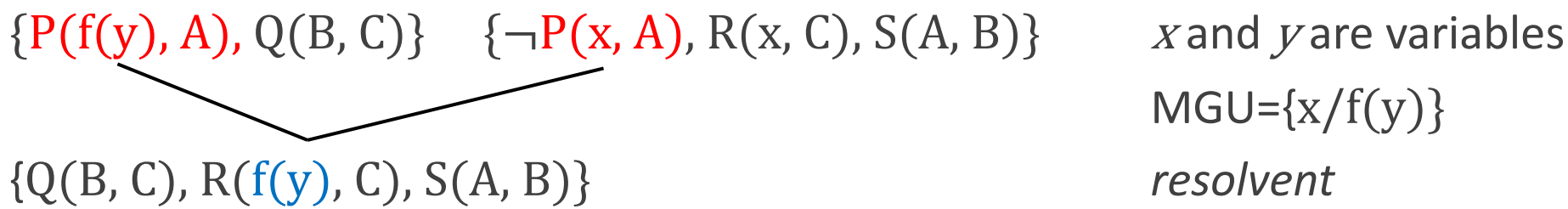
Resolution rule for FOL:

$$\frac{c_1 \cup \{p\} \qquad c_2 \cup \{\neg q\}}{[\, c_1 \cup c_2]\, \gamma}$$

and $\gamma = \text{MGU}(p, q)$ and γ is not *fail*

Resolution rule: an example

For example:



x and y are variables

MGU={x/f(y)}

resolvent

With the substitution {x/f(y)} the literal ¬P(x, A) becomes ¬P(f(y), A), and it is possible to apply the resolution rule.

Unification algorithm [Martelli, Montanari, 1982]

- Computes the MGU by means of a **rule-based** equation-rewriting system
- Initially the **working memory** (WM) contains the equality of the two expressions to be unified
- The rules modify equations in the WM
- The algorithm terminates with failure or when there are no applicable rules (**success**)
- At the end, if there is no failure, the WM contains the MGU.

Note: different from the AIMA unification algorithm but easier to understand.

The rules

1. $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n) \rightarrow s_1 = t_1, \ldots, s_n = t_n$
2. $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m) \rightarrow$ *fail* when $f \neq g$ or $n \neq m$
3. $x = x \quad \rightarrow$ remove equation
4. $t = x \rightarrow x = t$ bring variable to the left
5. $x = t$, x does not occur in $t \rightarrow$ apply $\{x / t\}$ to other equations
6. $x = t$, t is not x, x occur in $t \rightarrow$ *fail* (*occur check*)

Note: when comparing two different constants, rule 2 applies, as a special case where $n = m = 0$, and we fail.

Unification algorithm: example 1

Computing the MGU of $P(A, y, z)$ and $P(x, B, z)$

Step 3

$x = A$

$y = B$

MGU!

Unification algorithm: example 2

Computing the MGU of $P(f(x), x)$ and $P(z, z)$

Step 3

$z = f(x)$

$x = f(x)$ rule 6

FAIL!

(occurr check)

What’s next in this section

1. Knowledge engineering and Ontology engineering
2. Non monotonic reasoning
3. Reasoning about Knowledge & Beliefs.
4. Structured representations: semantic networks, objects and frames.
5. Description logics

Your turn

Please review (if you did not cover this material in IIA) :

- ✓ PROP: AIMA ch. 7
- ✓ FOL: AIMA ch. 8 (1-3)
- ✓ Inference in FOL: AIMA ch 9 (skip Logic Programming and Prolog)

References

[KRR] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 2004. (Ch. 1, 3)

[AIMA] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach* (3rd edition). Pearson Education 2010 (Ch. 7, 8, 9).

[KRR Handbook] Handbook of Knowledge Representation, F, Van Harmelen, V. Lifschitz, B. Porter, Elsevier, 2008 (Ch.1).