Vorlesung
Grundlagen der
Künstlichen Intelligenz

Reinhard Lafrenz / Prof. A. Knoll

Robotics and Embedded Systems
Department of Informatics – I6
Technische Universität München

www6.in.tum.de
lafrenz@in.tum.de
089-289-18136
Room 03.07.055

Wintersemester 2012/13
26.11.2012
Chapter 7,8 (3rd ed.)

Propositional and First-Order Logic
From the last lecture we know

- Propositional Logic
  - Restrictions to e.g. Horn Clauses

- Proof methods:
  - Resolution
  - Forward/Backward Chaining
  - DPLL algorithm
  - WalkSAT algorithm
Hard satisfiability problems

- Consider random 3-CNF sentences (with at most 3 variables per clause) e.g.,

\[
(\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \land \\
(E \lor \neg D \lor B) \land (B \lor E \lor \neg C)
\]

Analyse “hardness“ of satisfiability problem using

\[ m = \text{number of clauses} \]
\[ n = \text{number of symbols} \]

- Hard problems seem to cluster near \( m/n = 4.3 \) (critical point)
Hard satisfiability problems

![Graph showing Pr(satisfiable) vs. Clause/symbol ratio m/n]
Hard satisfiability problems

- Median runtime for 100 satisfiable random 3-CNF sentences, $n = 50$
Inference-based agents in the wumpus world

A wumpus-world agent using propositional logic:

\[
\neg P_{1,1} \\
\neg W_{1,1} \\
B_{x,y} \iff (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y}) \\
S_{x,y} \iff (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y}) \\
W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4} \\
\neg W_{1,1} \lor \neg W_{1,2} \\
\neg W_{1,1} \lor \neg W_{1,3} \\
\ldots
\]

- 64 distinct proposition symbols (16 x P, W, B, S)
- 155 sentences
function PL-WUMPUS-AGENT( percept) returns an action
inputs: percept, a list, [stench,breeze,glitter]
static: KB, initially containing the “physics” of the wumpus world
x, y, orientation, the agent’s position (init. [1,1]) and orient. (init. right)
visited, an array indicating which squares have been visited, initially false
action, the agent’s most recent action, initially null
plan, an action sequence, initially empty
update x,y,orientation, visited based on action
if stench then TELL(KB, S_{x,y}) else TELL(KB, ¬ S_{x,y})
if breeze then TELL(KB, B_{x,y}) else TELL(KB, ¬ B_{x,y})
if glitter then action ← grab
else if plan is nonempty then action ← POP(plan)
else if for some fringe square [i,j], ASK(KB, (¬ P_{i,j} ∧ ¬ W_{i,j})) is true or
for some fringe square [i,j], ASK(KB, (P_{i,j} ∨ W_{i,j})) is false then do
plan ← A*-GRAPH-SEARCH(Route-PB([x,y], orientation, [i,j], visited))
action ← POP(plan)
else action ← a randomly chosen move
return action
Expressiveness limitation of propositional logic

- KB contains "physics" sentences for every single square
  - For every time $t$ and every location $[x,y]$:

$$L_{x,y}^t \land \text{FacingRight}^t \land \text{Forward}^t \Rightarrow L_{x+1,y}^{t+1} \land \neg L_{x,y}^{t+1}$$

- Rapid proliferation of clauses
  - Check for danger in a field:

$$OK_{x,y}^t \iff \neg P_{x,y} \land \neg (W_{x,y} \land \text{WumpusAlive}^t)$$
Pros and cons of propositional logic

😊 Propositional logic is declarative

😊 Propositional logic allows partial/disjunctive/negated information
  – (unlike most data structures and databases)

😊 Propositional logic is compositional:
  – meaning of $B_{1,1} \land P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$

😊 Meaning in propositional logic is context-independent
  – (unlike natural language, where meaning depends on context)

BUT:

😢 Propositional logic has very limited expressive power
  – (unlike natural language)
  – E.g., cannot say "pits cause breezes in adjacent squares"
    • except by writing one sentence for each square
First-order logic

- Whereas propositional logic assumes the world contains facts,

- First-Order Logic (like natural language) assumes the world contains
  - **Objects**: people, houses, numbers, colors, baseball games, …
  - **Relations**: red, round, prime, brother of, bigger than, part of, comes between, …
  - **Functions**: father of, best friend, one more than, plus, …
Models for FOL: Example
Syntax of FOL: Basic elements

- **Constants:** KingJohn, 2, TUM,...
- **Predicates:** Brother, >,...
- **Functions:** Sqrt, LeftLegOf,...
- **Variables:** x, y, a, b,...
- **Connectives:** ¬, ⇒, ∧, ∨, ⇔
- **Equality:** =
- **Quantifiers:** ∀, ∃
Atomic sentences

Atomic sentence = \textit{predicate} \((\textit{term}_1,\ldots,\textit{term}_n)\)
\hspace{1cm} or \(\textit{term}_1 = \textit{term}_2\)

Term = \textit{function} \((\textit{term}_1,\ldots,\textit{term}_n)\)
\hspace{1cm} or \textit{constant} or \textit{variable}

Examples:
- \textit{Brother}(\textit{KingJohn},\textit{RichardTheLionheart})
- \(> \ (\text{Length(LeftLegOf(\textit{Richard}))},\ 
\hspace{1cm} \text{Length(LeftLegOf(\textit{KingJohn}}))))\)
Complex sentences

- Complex sentences are made from atomic sentences using connectives

\[ \neg S, S_1 \land S_2, S_1 \lor S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2, \]

E.g. \( \text{Sibling(KingJohn, Richard)} \Rightarrow \text{Sibling(Richard, KingJohn)} \)
\( >(1,2) \lor \leq (1,2) \)
\( >(1,2) \land \neg >(1,2) \)
First-Order-Logic: Syntax in BNF

\[
\text{Sentence} \rightarrow \text{AtomicSentence} \mid \text{ComplexSentence}
\]

\[
\text{AtomicSentence} \rightarrow \text{Predicate} \mid \text{Predicate(Term, \ldots)} \mid \text{Term} = \text{Term}
\]

\[
\text{ComplexSentence} \rightarrow (\text{Sentence}) \mid [\text{Sentence}]
\]

\[
\quad \rightarrow \text{Sentence}
\]

\[
\quad \rightarrow \text{Sentence} \land \text{Sentence}
\]

\[
\quad \rightarrow \text{Sentence} \lor \text{Sentence}
\]

\[
\quad \rightarrow \text{Sentence} \Rightarrow \text{Sentence}
\]

\[
\quad \rightarrow \text{Sentence} \Leftrightarrow \text{Sentence}
\]

\[
\quad \rightarrow \text{Quantifier Variable, \ldots Sentence}
\]

\[
\text{Term} \rightarrow \text{Function(Term, \ldots)}
\]

\[
\quad \rightarrow \text{Constant}
\]

\[
\quad \rightarrow \text{Variable}
\]

\[
\text{Quantifier} \rightarrow \forall \mid \exists
\]

\[
\text{Constant} \rightarrow A \mid X_1 \mid John \mid \ldots
\]

\[
\text{Variable} \rightarrow a \mid x \mid s \mid \ldots
\]

\[
\text{Predicate} \rightarrow \text{True} \mid \text{False} \mid \text{After} \mid \text{Loves} \mid \text{Raining} \mid \ldots
\]

\[
\text{Function} \rightarrow \text{Mother} \mid \text{LeftLeg} \mid \ldots
\]

Operator Precedence: $\neg$, $\Rightarrow$, $\iff$, $\land$, $\lor$, $\equiv$, $=$
Truth in first-order logic

- Sentences are true (a model) or false with respect to an interpretation.

- Interpretation specifies referents for:
  - Constant symbols $\rightarrow$ objects
  - Predicate symbols $\rightarrow$ relations
  - Function symbols $\rightarrow$ functional relations

- An atomic sentence $\text{predicate}(\text{term}_1, \ldots, \text{term}_n)$ is true \textbf{iff} the objects referred to by $\text{term}_1, \ldots, \text{term}_n$ are in the relation referred to by $\text{predicate}$.
Universal quantification

- $\forall <\text{variables}> \enspace <\text{sentence}>$

Everyone at TUM is smart:
$\forall x \: \text{At}(x, \text{TUM}) \implies \text{Smart}(x)$

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

- Roughly speaking, equivalent to the conjunction of instantiations of $P$

\[
\begin{align*}
\text{At}(&\text{KingJohn}, \text{TUM}) \implies \text{Smart(KingJohn)} \\
\land \quad &\text{At}(&\text{Richard}, \text{TUM}) \implies \text{Smart(Richard)} \\
\land &\quad \text{At}(&\text{TUM}, \text{TUM}) \implies \text{Smart(TUM)} \\
\land \quad &\ldots
\end{align*}
\]
A common mistake to avoid

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

- Common mistake: using $\land$ as the main connective with $\forall$:
  $\forall x \; At(x, \text{TUM}) \land \text{Smart}(x)$
  means “Everyone is at TUM and everyone is smart”
Existential quantification

- $\exists<\text{variables}> <\text{sentence}>$

- Someone at TUM is smart:
  - $\exists x \; \text{At}(x, \text{TUM}) \land \text{Smart}(x)$

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

- Roughly speaking, equivalent to the disjunction of instantiations of $P$

  $\text{At}(\text{KingJohn}, \text{TUM}) \land \text{Smart}(\text{KingJohn})$
  $\lor \; \text{At}(\text{Richard}, \text{TUM}) \land \text{Smart}(\text{Richard})$
  $\lor \; \text{At}(\text{TUM}, \text{TUM}) \land \text{Smart}(\text{TUM})$
  $\lor \ldots$
Another common mistake to avoid

- Typically, $\land$ is the main connective with $\exists$

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

  $$\exists x \text{ At}(x, \text{TUM}) \Rightarrow \text{Smart}(x)$$

  is true if there is anyone who is not at TUM!
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 \( \forall y \exists x \)
- \( \exists x \forall y \text{ Loves}(x,y) \)
  - “There is a person who loves everyone in the world”
- \( \forall y \exists x \text{ Loves}(x,y) \)
  - “Everyone in the world is loved by at least one person”

- **Quantifier duality**: each can be expressed using the other
- \( \forall x \text{ Likes}(x,\text{IceCream}) \) \( \iff \exists x \neg \text{ Likes}(x,\text{IceCream}) \)
- \( \exists x \text{ Likes}(x,\text{Broccoli}) \) \( \iff \neg \forall x \neg \text{ Likes}(x,\text{Broccoli}) \)
## De Morgan Rules

<table>
<thead>
<tr>
<th>Quantified</th>
<th>Not quantified</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x \neg P \equiv \neg \exists x P$</td>
<td>$\neg (P \lor \neg Q) \equiv \neg P \land Q$</td>
</tr>
<tr>
<td>$\neg \forall x P \equiv \exists x \neg P$</td>
<td>$\neg (P \land Q) \equiv \neg P \lor \neg Q$</td>
</tr>
<tr>
<td>$\forall x P \equiv \neg \exists x \neg P$</td>
<td>$P \land Q \equiv \neg (\neg P \lor \neg Q)$</td>
</tr>
<tr>
<td>$\exists x P \equiv \neg \forall x \neg P$</td>
<td>$P \lor Q \equiv \neg (\neg P \land \neg Q)$</td>
</tr>
</tbody>
</table>
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 $Parent$:

$$
\forall x,y \ Sibling(x,y) \iff [\neg (x = y) \land \exists m,f \neg (m = f) \land \\
\quad Parent(m,x) \land Parent(f,x) \land Parent(m,y) \land Parent(f,y)]
$$
Possible models

- Language with 2 constant symbols and 1 binary relation

- Up to 6 objects: 137.506.194.466 possibilities
Using FOL

The kinship domain:

- Brothers are siblings
  \[ \forall x, y \; \text{Brother}(x, y) \iff \text{Sibling}(x, y) \]

- One's mother is one's female parent
  \[ \forall m, c \; \text{Mother}(c) = m \iff (\text{Female}(m) \land \text{Parent}(m, c)) \]

- “Sibling” is symmetric
  \[ \forall x, y \; \text{Sibling}(x, y) \iff \text{Sibling}(y, x) \]
Using FOL – defining exact semantics

Write the sentence
“Richard has 2 brothers, John and Geoffrey” in FOL

\( \text{Brother}(\text{John}, \text{Richard}) \land \text{Brother}(\text{Geoffrey}, \text{Richard}) \)

- Is this enough?
- What if Geoffrey = John?
Add  \( \land (\text{John} \neq \text{Geoffrey}) \)

- What if there are more brothers?
\( \text{Brother}(\text{John}, \text{Richard}) \land \text{Brother}(\text{Geoffrey}, \text{Richard}) \land (\text{John} \neq \text{Geoffrey}) \land (\forall x \text{ Brother}(x, \text{Richard}) \implies (x=\text{John} \lor x=\text{Geoffrey})) \)
Using FOL – database semantics

Reconsider set of possible models

- Unique identities (John ≠ Geoffrey is implicit)
- Closed-world assumption (no constants not in the KB)

The number of possible models is reduced to $2^4 = 16$

Database semantics are used in logic programming languages
Using FOL

The set domain:
- \( \forall s \ Set(s) \iff (s = \{\} ) \lor (\exists x, s_2 \ Set(s_2) \land s = \{x|s_2\}) \)
- \( \neg \exists x, s \ \{x|s\} = \{\} \)
- \( \forall x, s \ x \in s \iff s = \{x|s\} \)
- \( \forall x, s \ x \in s \iff [\exists y, s_2] (s = \{y|s_2\} \land (x = y \lor x \in s_2)) \)
- \( \forall s_1, s_2 \ s_1 \subseteq s_2 \iff (\forall x \ x \in s_1 \Rightarrow x \in s_2) \)
- \( \forall s_1, s_2 \ (s_1 = s_2) \iff (s_1 \subseteq s_2 \land s_2 \subseteq s_1) \)
- \( \forall x, s_1, s_2 \ x \in (s_1 \cap s_2) \iff (x \in s_1 \land x \in s_2) \)
- \( \forall x, s_1, s_2 \ x \in (s_1 \cup s_2) \iff (x \in s_1 \lor x \in s_2) \)
Interacting with FOL KBs

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

  
  \[
  \text{Tell}(KB, \text{Percept}([\text{Smell}, \text{Breeze}, \text{None}], 5))
  \]

  \[
  \text{Ask}(KB, \exists a \text{ BestAction}(a, 5))
  \]

- I.e., does the KB entail some best action at $t=5$?

- Answer: Yes, $\{a/\text{Shoot}\}$ ← substitution (binding list)

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

  \[
  S = \text{Smarter}(x,y)
  \]

  \[
  \sigma = \{x/\text{Hillary}, y/\text{Bill}\}
  \]

  \[
  S\sigma = \text{Smarter}(\text{Hillary}, \text{Bill})
  \]

- $\text{Ask}(KB, S)$ returns some/all $\sigma$ such that $KB \models \sigma$
Knowledge base for the wumpus world

- **Perception**
  - $\forall t, s, b$ Percept([s, b, Glitter], t) $\Rightarrow$ Glitter(t)

- **“Reflex”**
  - $\forall t$ Glitter(t) $\Rightarrow$ BestAction(Grab, t)
Deducing hidden properties

- $\forall x,y,a,b \ Adjacent([x,y],[a,b]) \iff [a,b] \in \{[x+1,y], [x-1,y],[x,y+1],[x,y-1]\}$

Properties of squares:
- $\forall s,t \ At(Agent,s,t) \land Breeze(t) \Rightarrow Breezy(s)$

Squares are breezy near a pit:
- $\forall s \ Breezy(s) \iff \exists r \ Adjacent(r,s) \land Pit(r)$
  - **Diagnostic** rule---infer cause from effect
    - $\forall s \ Breezy(s) \Rightarrow \exists r \ Adjacent(r,s) \land Pit(r)$
  - **Causal** rule---infer effect from cause
    - $\forall r \ Pit(r) \Rightarrow [\forall s \ Adjacent(r,s) \Rightarrow Breezy(s) ]$

Consideration of time
- $\forall t \ HaveArrow(t+1) \iff HaveArrow(t) \land \neg Action(Shoot, t))$
Knowledge engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base
Summary

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

- Increased expressive power: sufficient to define wumpus world including “hidden properties” such as “hasArrow”