Intelligent Agents

First Order Logic

Ute Schmid

Cognitive Systems, Applied Computer Science, Bamberg University

last change: 19. Mai 2015

Reasoning and Inference

- Reasoning is the process of using facts and inference rules to produce conclusions.
- A **knowledge-based system** typically consists of two components: knowledge representation + inference.
- Knowledge representation is domain-dependent and must be acquired and formalized for each new domain of interest (knowledge engineering).
- Inference mechanisms are independent of a concrete domain but they rely on specific formats of knowledge representation.
- *First order logic* as representation formalism and *resolution* as inference mechanism are the foundation of many knowledge-based systems
  (e.g. expert systems, question-answering systems).

Outline

- Different Types of Reasoning (Deduction, Induction, Abduction)
- Introduction of First-Order Logic (FOL)
- Basic Idea of the resolution calculus (to be continued)

Basic Types of Inference: Deduction

(Charles Peirce)

- Deduction: Derive a conclusion from given axioms ("knowledge") and facts ("observations").

Example

(axiom)	*All humans are mortal.*
(fact/premise)	*Socrates is a human.*
(conclusion)	*Therefore, it follows that Socrates is mortal.*

- The conclusion can be derived by applying the *modus ponens* inference rule (Aristotelian/propositional logic).

Basic Types of Inference: Induction

- Induction: Derive a general rule (axiom) from background knowledge and observations.

Example

(background knowledge)	*Socrates is a human.*
(observation/example)	*Socrates is mortal.*
(generalization)	*Therefore, I hypothesize that all humans are mortal.*

- Induction means to infer (unsure) generalized knowledge from example observations.
- Induction is *the* inference mechanism for learning! (see lesson on Machine Learning)
- Analogy is a special kind of induction.

Basic Types of Inference: Abduction

- Abduction: From a known axiom (theory) and some observation, derive a premise.

Example

(theory)	*All humans are mortal.*
(observation)	*Socrates is mortal.*
(diagnosis)	*Therefore,*
	Socrates must have been a human.

- Abduction is typical for diagnostic systems/expert systems.
  (*It is also the preferred reasoning method of Sherlock Holmes.*)
- Simple medical diagnosis:
  If one has the flue, one has moderate fewer.
  Patient X has moderate fewer.
  Therefore, he has the flue.

Deduction

- Deductive inference is also called theorem proving or logic inference.
- Deduction is truth preserving:
  If the premises (axioms and facts) are true, then the conclusion (also called theorem) is true.
  (given sound and complete inference rules!)
- Note:
  The truth of the premises is typically not the "logical truth" (premises in general are not tautologies).
- In deduction we infer special theorems from a more general theory.
  Deduction is "knowledge extraction" while induction is "construction of (hypothetical) knowledge".

Deduction Calculus

- To perform deductive inference on a machine, a *calculus* is needed:
  A calculus is a set of syntactical rewrite-rules defined for some language. These rules must be *sound and complete* with respect to the task they should perform (to a model).

  Soundness and completeness are **semantic properties**!
- We will focus on the **resolution calculus** for first order logic (FOL).
  - Syntax of FOL (language for knowledge representation + definition of calculus)
  - Semantics of FOL (meaning of FOL sentences + properties of calculus)

(Basic knowledge in propositional and first order logic is presumed.)

Syntax of FOL/Terms

Inductive definition:

- Terms:
  - A variable $v \in V$ is a term.
  - If f is a function symbol with arity n and $t_1 \ldots t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.
    (including constant symbols as 0-ary function symbols)
  - That are all terms.

Syntax of FOL/Formulas

Inductive definition:

- Formulas:
  - if P is a predicate symbol with arity n and $t_1 \ldots t_n$ are terms, then $P(t_1, \ldots, t_n)$ is a formula.
    (atomic formula)
  - For all formulas F and G, $\neg F$, $F \wedge G$, $F \vee G$, $F \rightarrow G$ and $F \leftrightarrow G$ are formula.
    (connectives *"not"*, *"and"*, *"or"*, *"implies"*, *"equivalent"*)
  - If v is a variable and F is a formula, then $\exists v\ F$ and $\forall v\ F$ are formulas.
    (existential and universal quantifier, *"exists"*, *"for all"*)
  - That are all formulas.

Remarks on Syntax of FOL

- Formula are constructed over terms.
  Never confuse this categories!
- Additionally, parentheses can be used to group sub-expressions.
- Expressions which obey the given inductive definition are called *well-formed formulas* (wffs).
  The closure "that are all terms/formulas" is necessary to exclude all other kinds of (not well-formed) expressions.
- We refer to atomic formulas also as " *atoms*". Positive and negated atoms (P, $\neg P$) are called *positive/negative literals*.

Remarks on Syntax of FOL cont.

- A variable which is in the scope of a quantifier is called *bound*, otherwise it is is called *free*.

$$P(x) \vee \forall y \exists z \; Q(y, z)$$

  x is free and y and z are bound.
  A formula without free variables is called *sentence*.
- Propositional logic is a special case of FOL:
  use only 0-ary predicate symbols (then there are no terms, no variable and no quantifiers) or just forbid variables and quantifiers (use only grounded formulas).

Semantics of FOL

- Syntax defines how to form well-formed expressions. Semantics gives *meaning* to expressions.
- Expressions are *interpreted* using an interpretation function $\mathcal{I}$ over a domain (set of objects) $\mathcal{U}$. A pair $\mathcal{A} = (\mathcal{U}, \mathcal{I})$ is called a *structure* (or algebra).
- Function symbols are interpreted as functions and predicate symbols as predicates/relations over a domain.

Example Interpretation

$(\exists x\ \text{on}(x, A) \rightarrow \neg\text{clear}(A)) \wedge \exists y\ \text{clear}(\text{topof}(y))$

- **Constant symbol** 'A' is interpreted as a block in a blocksworld $\mathcal{B}$.
- **Function symbol** 'topof(x)' is interpreted as a function $t : \mathcal{B} \rightarrow \mathcal{B}$ which returns the block which is lying on top of block x or $\perp$ if no block is lying on x.
- **Predicate symbol** 'clear(x)' is interpreted as 1-ary relation $C \subseteq \mathcal{B}$ which holds if no other block is lying on block x.
- Predicate symbol 'on(x,y)' is interpreted as binary relation $O \subseteq \mathcal{B} \times \mathcal{B}$ which holds if a block x is lying immediately on top of a block y.

Example Illustration

Possible interpretations for the formulas:

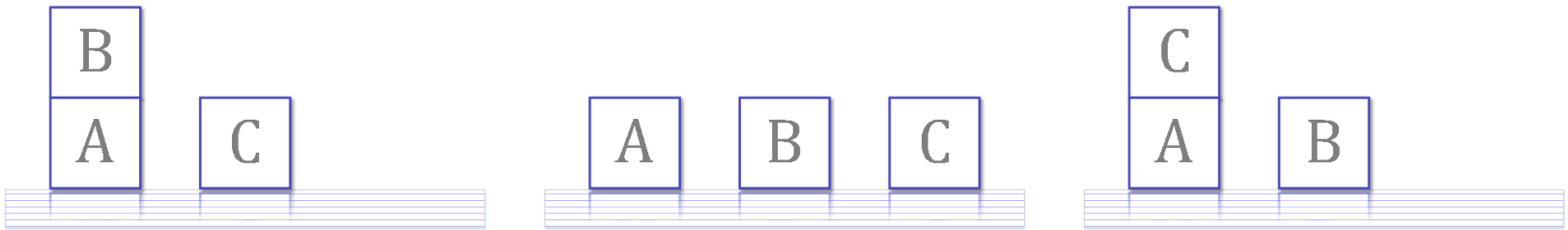
clear(B)

clear (B) $\wedge$ clear(C)

clear(B) $\vee$ clear(A)

on(A, Table) $\rightarrow$ clear(B)

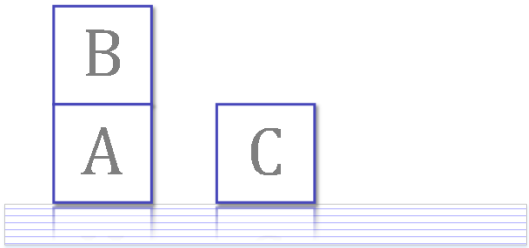
clear(B) $\wedge$ clear(C) $\rightarrow$ on(A, Table)



Intuition about Semantics

Natural language sentence $\xrightarrow{\text{describes}}$ *situation in the world*

"Block B lies on block A"



FOL formula $\xrightarrow{\text{is interpreted by}}$ *a structure*

Blocksworld = $(\mathcal{B}, \mathcal{I})$ with $\mathcal{B}$ as set of blocks and $\mathcal{I}$ as defined above.

Propositional Logic vs. FOL

- In propositional logic, formula are interpreted *directly* by truth values.
- In FOL we first must map the components of a formula stepwise into a structure.
- Constant symbols are mapped to objects in the domain. For interpreted functions, their result can be determined within the structure (evaluation of grounded terms). The result is again an object in the domain.
- Formula over terms are mapped to truth values.

Now the difference between terms and formula should be clear: terms evaluate to values, formula to truth values!

Propositional Logic vs. FOL cont.

Mapping of formula to truth values:

- To assign a truth value, first, the arguments of a predicate symbol are interpreted, afterwards it is determined whether the corresponding relation holds in the structure.
- If all atomic formula are associated with truth values, more complex formulas can be interpreted as known over the definition of the junctors.
- For an existentially quantified formula there needs to be at least one element in the domain for which the formula is true. For universally quantified formulas, the formula must be true for all domain objects.

Semantics of the Connectives

A	B	$\neg$ A	A $\wedge$ B	A $\vee$ B	A $\rightarrow$ B	A $\leftrightarrow$ B
0	0	1	0	0	1	1
0	1		0	1	1	0
1	0	0	0	1	0	0
1	1		1	1	1	1

$\neg$ binds stronger than
$\wedge$ and $\vee$ which bind stronger than
$\rightarrow$ and $\leftrightarrow$

Model, Satisfiability, Validity

- If the interpretation of a formula F with respect to a structure $\mathcal{A}$ results in the truth value “true”, the structure is called a model. We write $\mathcal{A} \models F$.
- If every structure for F is a model, we call F valid and write $\models F$.
- If there exists at least one model for F, we call F satisfiable.

Remark:
In general, structures can be defined over domains which are arbitrary sets. That is, the sets can contain lots of irrelevant objects. There is a canonic way to construct structures over a so called *Herbrand Universe*.

Illustration

- $\forall x\ y\ (p(x) \wedge p(y)) \rightarrow (p(x) \vee p(y))$ is valid

  e.g. *For all humans it holds that if two persons are rich then at least one of them is rich.*

  e.g. *For all natural numbers holds that if both numbers are even then at least one of them is even.*

- $\exists x\ (s(x) \wedge v(x, L))$ is satisfiable

  e.g. *It exists a natural number smaller than 17.*

  e.g. *It exists a human being who is a student and who likes logic.*

Semantical Entailment

- A formula G is called logical consequence (or entailment) of a set of formula $F = \{F_1 \ldots F_n\}$, if each model of F is also a model of G.
  Note:
  We write $\mathcal{A} \models G$ to denote the "*model relation*" and also $F \models G$ to denote the "*entailment relation*".
- The following propositions are equivalent:
  1. G is a logical consequence of F.
  2. $(\wedge_{i=1}^{n} F_i) \rightarrow G$ is valid (tautology).
  3. $(\wedge_{i=1}^{n} F_i) \wedge \neg G$ is not satisfiable (a contradiction).

  This relation between logical consequences and syntactical expressions can be exploited for syntactical proofs. We write $F \vdash G$ if formula G can be derived from the set of formulas F.

Resolution Calculus

- The resolution calculus consists of a single rule (and does not possess any axioms).
- Resolution is defined for clauses (each formula is a disjunction of positive and negative literals).
- All formulas must hold: conjunction of clauses.
- Proof by contradiction, exploiting the equivalence given above. If

$$(\bigwedge_{i=1}^{n} F_i) \wedge \neg G$$

is not satisfiable, then “false” (the empty clause) can be derived:

$$[(\bigwedge_{i=1}^{n} F_i) \wedge \neg G] \vdash \square$$

Resolution Calculus cont.

Resolution rule in propositional logic:

$$(P \vee P_1 \vee \ldots P_n) \wedge (\neg P \vee Q_1 \vee \ldots Q_m) \vdash (P_1 \vee \ldots P_n \vee Q_1 \vee \ldots Q_m)$$

Resolution rule for clauses:

$$[(L \vee C_1) \wedge (\neg L \vee C_2)]\sigma \vdash [C_1 \vee C_2]\sigma$$

(σ is a substitution of variables such that L is identical in both parts of the conjunction)

The general idea is to cut out a literal which appears positive in one disjunction and negative in the other.

Resolution in Propositional Logic

Example

Theory:

All humans are mortal. $F_1 = \text{Human} \rightarrow \text{Mortal}$
Socrates is a human. $F_2 = \text{Human}$

Query:

Socrates is mortal: $G = \text{Mortal}$

To prove $F_1 \wedge F_2 \wedge \neg G \vdash \square$, we need the following resolution steps:

1. $\text{Human} \rightarrow \text{Mortal} \equiv \neg\text{Human} \vee \text{Mortal}$
2. $\text{Human} \wedge [\neg\text{Human} \vee \text{Mortal}] \wedge \neg\text{Mortal}$
3. $\vdash \text{Mortal} \wedge \neg\text{Mortal}$
4. $\vdash \square$.

Resolution in FOL

Example

Theory:

All humans are mortal. $F_1 = \forall x\ \text{Human(x)} \rightarrow \text{Mortal(x)}$
Socrates is a human. $F_2 = \text{Human(S)}$

Query:

Socrates is mortal: $G = \text{Mortal(S)}$

To prove $F_1 \wedge F_2 \wedge \neg G \vdash \square$, we need the following resolution steps:

1. $\forall x\ \text{Human(x)} \rightarrow \text{Mortal(x)} \equiv \forall x\ \neg\text{Human(x)} \vee \text{Mortal(x)}$
   (substitute S for universally quantified variable x)
2. $[\text{Human(S)} \wedge [\neg\text{Human(x)} \vee \text{Mortal(x)}] \wedge \neg\text{Mortal(S)}]\{x \rightarrow S\}$
3. $\vdash \text{Mortal(S)} \wedge \neg\text{Mortal(S)}$
4. $\vdash \square$.

Resolution in FOL cont.

- Resolution was introduced by Robinson (1965) as a mechanic way to perform logical proofs.
- We need to understand:
  - Transformation of FOL formulas in clauses by applying equivalence rules.
  - Substitution and unification.

Semantic Equivalence

- Two formulas F and G are called equivalent, if for each interpretation of G and F holds that G is valid iff F is valid. We write $F \equiv G$.
- Theorem: Let be $F \equiv G$. Let H be a formula where F appears as a sub-formula. Let H' be a formula derived from H by replacing F by G. Then it holds $H \equiv H'$.
- Equivalences can be used to rewrite formulas.

Semantic Equivalence cont.

$(F \wedge F) \equiv F, (F \vee F) \equiv F$	(idempotency)
$(F \wedge G) \equiv (G \wedge F), (F \vee G) \equiv (G \vee F)$	(commutativity)
$((F \wedge G) \wedge H) \equiv (F \wedge (G \wedge H)), ((F \vee G) \vee H) \equiv (F \vee (G \vee H))$	(associativity)
$(F \wedge (F \vee G)) \equiv F, (F \vee (F \wedge G)) \equiv F$	(absorption)
$(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H)),$ $(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H))$	(distributivity)
$\neg\neg F \equiv F$	(double negation)
$\neg(F \wedge G) \equiv (\neg F \vee \neg G), \neg(F \vee G) \equiv (\neg F \wedge \neg G)$	(de Morgan)
$(F \rightarrow G) \equiv (\neg F \vee G)$	(remove implication)

$F \vee \neg F \equiv$ true	(tautology)
$F \wedge \neg F \equiv$ false	(contradiction)

Remark: This is the *tertium non datur* principle of classical logic.

Semantic Equivalence cont.

$\neg\forall x\ F \equiv \exists x\ \neg F$, $\neg\exists x\ F \equiv \forall x\ \neg F$

$(F \vee G) \equiv F$, if F tautology;
$(F \wedge G) \equiv G$, if F tautology
$(F \vee G) \equiv G$, if F contradiction;
$(F \wedge G) \equiv F$, if F contradiction

If x is not free in G it holds:
$(\forall x\ F \wedge G) \equiv \forall x\ (F \wedge G)$, $(\forall x\ F \vee G) \equiv \forall x\ (F \vee G)$,
$(\exists x\ F \wedge G) \equiv \exists x\ (F \wedge G)$, $(\exists x\ F \vee G) \equiv \exists x\ (F \vee G)$

$(\forall x\ F \wedge \forall x\ G) \equiv \forall x\ (F \wedge G)$, $(\exists x\ F \vee \exists x\ G) \equiv \exists x\ (F \vee G)$

$\forall x \forall y F \equiv \forall y \forall x F$, $\exists x \exists y F \equiv \exists y \exists x F$

Classical Logic

- Propositional logic and FOL are classical logics.
- Classical logic is bivalent and monotonic:
  There are only two truth values “true” and “false”.
  Because of the *tertium non datur*, derived conclusions cannot be changed by new facts or conclusions (vs. multi-valued and non-monotonic logics).
- In classical logic, “everything” follows from a contradiction (*ex falso quod libet*).
  A theorem can be proven by contradiction.
  In contrast, in intuitionistic logic, all proofs must be *constructive*!

Summary

- Deduction is the only type of inference where correctness of derivations (conclusions) can be guaranteed
- Given a sound and complete deduction calculus, proofs can be performed automatically (by a computer program)
- Syntax of FOL is defined as well-formed formulas which are constructed over terms
- Semantics of propositional logic is based in semantics of the connectives only
- Semantics of FOL relies on interpretation of formula in a model
- Formula can be satisfiable or valid
- Semantic entailment can be characterized by three equivalent propositions
- Resolution is a calculus based on the concept of proof by contradiction
- Formula can be rewritten syntactically, based on semantic equivalences