Foundations of Data and Knowledge Systems

EPCL Basic Training Camp 2012

Part One

Thomas Eiter and Reinhard Pichler

Institut für Informationssysteme
Technische Universität Wien

19 December, 2012

Outline

Course overview

- Focus: Foundations of Rule-based Query Answering
- Syntax of First-Order Predicate Logic
- Some Fragments of First-Order Predicate Logic
- Fundamentals of Classical Model Theory
- Declarative Semantics of Rule Languages
- Operational Semantics of Rule Languages
- Complexity and Expressive Power

Literature

Basic reading

This course is mainly based on the following article:
François Bry, Norbert Eisinger, Thomas Eiter, Tim Furche, Georg Gottlob, Clemens Ley, Benedikt Linse, Reinhard Pichler, Fang Wei:
Foundations of Rule-Based Query Answering. Reasoning Web 2007, Lecture Notes in Computer Science 4636: pp. 1 – 153, Springer (2007).

Literature

Basic reading

This course is mainly based on the following article:
François Bry, Norbert Eisinger, Thomas Eiter, Tim Furche, Georg Gottlob, Clemens Ley, Benedikt Linse, Reinhard Pichler, Fang Wei:
Foundations of Rule-Based Query Answering. Reasoning Web 2007, Lecture Notes in Computer Science 4636: pp. 1 – 153, Springer (2007).

Further references

Further references will be provided as we go along, e.g.:
Alexander Leitsch: *The Resolution Calculus*, Texts in Theoretical Computer Science, Springer-Verlag Berlin, Heidelberg, New York, 1997.

Outline

Query Languages and Logic

Motivation

- Foundations of query languages mostly stem from logic (and complexity theory)
- New query languages with new syntactical constructs and concepts depart from classical logic but keep a logical flavour.
- Typical strengths of this logical flavour are:
  - compound queries using connectives such as "and" and "or"
  - rules expressed as implications
  - declarative semantics reminiscent of Tarski's model semantics
  - query optimisation based on equivalences of logical formulas

What are Query Languages?

Tentative Definitions

1. What are . . . their purposes of use?
   selecting and retrieving data from "information systems"
2. What are . . . their programming paradigms?
   declarative, hence mostly related to logic
3. What are . . . their major representatives?
   SQL, Datalog (relational data),
   XPath, XQuery (XML data),
   SPARQL (RDF data, OWL ontologies)
4. What are . . . their research issues?
   declarative semantics, procedural semantics, complexity and expressive power, implementations, optimisation, etc.

What are Query Languages?

Tentative Definitions

1. What are ... their purposes of use?
   selecting and retrieving data from "information systems"
2. What are ... their programming paradigms?
   declarative, hence mostly related to logic
3. What are ... their major representatives?
   SQL, Datalog (relational data),
   XPath, XQuery (XML data),
   SPARQL (RDF data, OWL ontologies)
4. What are ... their research issues?
   declarative semantics, procedural semantics, complexity and expressive power, implementations, optimisation, etc.

Logic vs. Logics

The development of logic(s)

- starting in antiquity: logic as an activity of philosophy aimed at analysing rational reasoning.
- late 19th century: parts of logic were mathematically formalised.
- early 20th century: logic used as a tool in a (not fully successful) attempt to overcome a foundational crisis of mathematics.
- logic in computer science: Today, logic provides the foundations in many areas of computer science, such as knowledge representation, database theory, programming languages, and query languages.
- Key features of logic: the use of formal languages for representing statements (which may be true or false) and the quest for computable reasoning about those statements.
- Logic vs. logics: "Logic" is the name of the scientific discipline investigating such formal languages for statements, but any of those languages is also called "a logic"

Logic vs. Logics

The development of logic(s)

- starting in antiquity: logic as an activity of philosophy aimed at analysing rational reasoning.
- late 19th century: parts of logic were mathematically formalised.
- early 20th century: logic used as a tool in a (not fully successful) attempt to overcome a foundational crisis of mathematics.
- logic in computer science: Today, logic provides the foundations in many areas of computer science, such as knowledge representation, database theory, programming languages, and query languages.
- Key features of logic: the use of formal languages for representing statements (which may be true or false) and the quest for computable reasoning about those statements.
- Logic vs. logics: "Logic" is the name of the scientific discipline investigating such formal languages for statements, but any of those languages is also called "a logic" – logic investigates logics.

Outline

Symbols

Symbols in First-Order Predicate Logic

First-order predicate logic is not just a single formal language, because some of its symbols may depend on the intended application.

- The symbols common to all languages of first-order predicate logic are called logical symbols.
- The symbols that are specified in order to determine a specific language are called the signature (or vocabulary) of that language.

Symbols

Symbols in First-Order Predicate Logic

First-order predicate logic is not just a single formal language, because some of its symbols may depend on the intended application.

- The symbols common to all languages of first-order predicate logic are called logical symbols.
- The symbols that are specified in order to determine a specific language are called the signature (or vocabulary) of that language.

Definition (Signature)

A signature or vocabulary for first-order predicate logic is a pair $\mathcal{L} = \left(\{Fun^n_{\mathcal{L}}\}_{n\in\mathbb{N}}, \{Rel^n_{\mathcal{L}}\}_{n\in\mathbb{N}}\right)$ of two families of computably enumerable symbol sets, called n-ary function symbols of $\mathcal{L}$ and n-ary relation symbols or predicate symbols of $\mathcal{L}$.

The 0-ary function symbols are called constants of $\mathcal{L}$. The 0-ary relation symbols are called propositional relation symbols of $\mathcal{L}$.

Logical Symbols

Definition (Logical Symbols)

The logical symbols of first-order predicate logic are:

symbol class		symbols	name
punctuation symbols		,) (	
connectives	0-ary	$\bot$	falsity symbol
		$\top$	truth symbol
	1-ary	$\neg$	negation symbol
	2-ary	$\wedge$	conjunction symbol
		$\vee$	disjunction symbol
		$\Rightarrow$	implication symbol
quantifiers		$\forall$	universal quantifier
		$\exists$	existential quantifier
variables		$u\ v\ w\ x\ y\ z\ \ldots$ (possibly subscripted)	

The set of variables is infinite and computably enumerable.

Terms and Atoms

Definition ($\mathcal{L}$-term)

Let $\mathcal{L}$ be a signature. Terms are defined inductively:

1. Each variable x is an $\mathcal{L}$-term.
2. Each constant c of $\mathcal{L}$ is an $\mathcal{L}$-term.
3. For each $n \geq 1$, if f is an n-ary function symbol of $\mathcal{L}$ and $t_1, \ldots, t_n$ are $\mathcal{L}$-terms, then $f(t_1, \ldots, t_n)$ is an $\mathcal{L}$-term.

Terms and Atoms

Definition ($\mathcal{L}$-term)

Let $\mathcal{L}$ be a signature. Terms are defined inductively:

1. Each variable x is an $\mathcal{L}$-term.
2. Each constant c of $\mathcal{L}$ is an $\mathcal{L}$-term.
3. For each $n \geq 1$, if f is an n-ary function symbol of $\mathcal{L}$ and $t_1, \ldots, t_n$ are $\mathcal{L}$-terms, then $f(t_1, \ldots, t_n)$ is an $\mathcal{L}$-term.

Definition ($\mathcal{L}$-atom)

Let $\mathcal{L}$ be a signature.

For $n \in \mathbb{N}$, if p is an n-ary relation symbol of $\mathcal{L}$ and $t_1, \ldots, t_n$ are $\mathcal{L}$-terms, then $p(t_1, \ldots, t_n)$ is an $\mathcal{L}$-atom or atomic $\mathcal{L}$-formula.

For $n = 0$, the atom may be written $p()$ or p and is called a propositional $\mathcal{L}$-atom.

Formulas

Definition ($\mathcal{L}$-formula)

Let $\mathcal{L}$ be a signature. Formulas are defined inductively:

1. Each $\mathcal{L}$-atom is an $\mathcal{L}$-formula. (atoms)
2. $\bot$ and $\top$ are $\mathcal{L}$-formulas. (0-ary connectives)
3. If φ is an $\mathcal{L}$-formula, then
   $\neg\varphi$ is an $\mathcal{L}$-formula. (1-ary connectives)
4. If φ and ψ are $\mathcal{L}$-formulas, then $(\varphi \wedge \psi)$ and
   $(\varphi \vee \psi)$ and $(\varphi \Rightarrow \psi)$ are $\mathcal{L}$-formulas. (2-ary connectives)
5. If x is a variable and φ is an $\mathcal{L}$-formula, then
   $\forall x \varphi$ and $\exists x \varphi$ are $\mathcal{L}$-formulas. (quantifiers)

Formulas

Definition ($\mathcal{L}$-formula)

Let $\mathcal{L}$ be a signature. Formulas are defined inductively:

1. Each $\mathcal{L}$-atom is an $\mathcal{L}$-formula. (atoms)
2. $\bot$ and $\top$ are $\mathcal{L}$-formulas. (0-ary connectives)
3. If φ is an $\mathcal{L}$-formula, then $\neg\varphi$ is an $\mathcal{L}$-formula. (1-ary connectives)
4. If φ and ψ are $\mathcal{L}$-formulas, then $(\varphi \wedge \psi)$ and $(\varphi \vee \psi)$ and $(\varphi \Rightarrow \psi)$ are $\mathcal{L}$-formulas. (2-ary connectives)
5. If x is a variable and φ is an $\mathcal{L}$-formula, then $\forall x\varphi$ and $\exists x\varphi$ are $\mathcal{L}$-formulas. (quantifiers)

Remark

In most cases the signature $\mathcal{L}$ is clear from context, and we simply speak of terms, atoms, and formulas without the prefix "$\mathcal{L}$-".

Notational Conventions

Symbols

In particular, if no signature is specified, one usually assumes the conventions:

$p, q, r, \ldots$	are relation symbols with appropriate arities.
$f, g, h, \ldots$	are function symbols with appropriate arities $\neq 0$.
$a, b, c, \ldots$	are constants, i.e., function symbols with arity 0.

Notational Conventions

Symbols

In particular, if no signature is specified, one usually assumes the conventions:

$p, q, r, \ldots$	are relation symbols with appropriate arities.
$f, g, h, \ldots$	are function symbols with appropriate arities $\neq 0$.
$a, b, c, \ldots$	are constants, i.e., function symbols with arity 0.

Use of Parentheses

Unique Parsing of Terms and Formulas. Since formulas constructed with a binary connective are enclosed by parentheses, any term or formula has an unambiguous syntactical structure.

Precendence of Operators. For the sake of readability this strict syntax definition can be relaxed by the convention that $\wedge$ takes precedence over $\vee$ and both of them take precedence over $\Rightarrow$.

Example. $q(a) \vee q(b) \wedge r(b) \Rightarrow p(a, f(a, b))$ is a shorthand for the fully parenthesised form $((q(a) \vee (q(b) \wedge r(b))) \Rightarrow p(a, f(a, b)))$.

Variables in Formulas

Example (Bound/free variable)

Let φ be $\big(\forall x[\exists x p(x) \wedge q(x)] \Rightarrow [r(x) \vee \forall x s(x)]\big)$. The x in $p(x)$ is bound in φ by $\exists x$. The x in $q(x)$ is bound in φ by the first $\forall x$. The x in $r(x)$ is free in φ. The x in $s(x)$ is bound in φ by the last $\forall x$.

Let φ' be $\forall x\big([\exists x p(x) \wedge q(x)] \Rightarrow [r(x) \vee \forall x s(x)]\big)$. Here both the x in $q(x)$ and the x in $r(x)$ are bound in φ' by the first $\forall x$.

Variables in Formulas

Example (Bound/free variable)

Let φ be $\big(\forall x[\exists x\, p(x) \land q(x)] \Rightarrow [r(x) \lor \forall x\, s(x)]\big)$. The x in $p(x)$ is bound in φ by $\exists x$. The x in $q(x)$ is bound in φ by the first $\forall x$. The x in $r(x)$ is free in φ. The x in $s(x)$ is bound in φ by the last $\forall x$.

Let φ' be $\forall x\big([\exists x\, p(x) \land q(x)] \Rightarrow [r(x) \lor \forall x\, s(x)]\big)$. Here both the x in $q(x)$ and the x in $r(x)$ are bound in φ' by the first $\forall x$.

Definition (Rectified formula)

A formula φ is rectified, if for each occurrence Qx of a quantifier for a variable x, there is neither any free occurrence of x in φ nor any other occurrence of a quantifier for the same variable x.

Variables in Formulas

Example (Bound/free variable)

Let φ be $(\forall x[\exists x p(x) \wedge q(x)] \Rightarrow [r(x) \vee \forall x s(x)])$. The x in $p(x)$ is bound in φ by $\exists x$. The x in $q(x)$ is bound in φ by the first $\forall x$. The x in $r(x)$ is free in φ. The x in $s(x)$ is bound in φ by the last $\forall x$.

Let φ' be $\forall x([\exists x p(x) \wedge q(x)] \Rightarrow [r(x) \vee \forall x s(x)])$. Here both the x in $q(x)$ and the x in $r(x)$ are bound in φ' by the first $\forall x$.

Definition (Rectified formula)

A formula φ is rectified, if for each occurrence Qx of a quantifier for a variable x, there is neither any free occurrence of x in φ nor any other occurrence of a quantifier for the same variable x.

Remark

Any formula can be rectified by consistently renaming its quantified variables. E.g., the above φ can be rectified to $(\forall u[\exists v p(v) \wedge q(u)] \Rightarrow [r(x) \vee \forall w s(w)])$.

Ground and Propositional Case

Definition (Ground term or formula, closed formula)

A ground term is a term containing no variable.
A ground formula is a formula containing no variable.
A closed formula or sentence is a formula containing no free variable.

Ground and Propositional Case

Definition (Ground term or formula, closed formula)

A ground term is a term containing no variable.
A ground formula is a formula containing no variable.
A closed formula or sentence is a formula containing no free variable.

Definition (Propositional formula)

A propositional formula is a formula containing no quantifier and no relation symbol of arity > 0.

Ground and Propositional Case

Definition (Ground term or formula, closed formula)

A ground term is a term containing no variable.
A ground formula is a formula containing no variable.
A closed formula or sentence is a formula containing no free variable.

Definition (Propositional formula)

A propositional formula is a formula containing no quantifier and no relation symbol of arity > 0.

Ground vs. Propositional

Obviously, each propositional formula is ground. Conversely, every ground formula can be regarded as propositional in a broader sense:

Let $\mathcal{L}$ be an arbitrary signature and let $\mathcal{L}'$ be a new signature defining each ground $\mathcal{L}$-atom as a 0-ary relation "symbol" of $\mathcal{L}'$. Then each ground $\mathcal{L}$-formula can be considered as a propositional $\mathcal{L}'$-formula.

Outline

Semantics of First-Order Predicate Logic

Classical Tarski Model Theory

- Goal: attributation of meaning to terms and formulas
- Principle of a Tarski-style semantics: The interpretation of a compound term and the truth value of a compound formula are defined recursively over the structure of the term or formula.
- Advantage of this approach: recursive definition makes things simple; well-defined, finite, and restricted computation scope.
- Disadvantage of this approach: allowing for any kind of sets for interpreting terms makes it apparently incomputable.

Semantics of First-Order Predicate Logic

Definition (Variable assignment)

Let D be a nonempty set. A variable assignment in D is a function V mapping each variable to an element of D. We denote the image of x under V by x^V.

Semantics of First-Order Predicate Logic

Definition (Variable assignment)

Let D be a nonempty set. A variable assignment in D is a function V mapping each variable to an element of D. We denote the image of x under V by x^V.

Definition ($\mathcal{L}$-Interpretation)

Let $\mathcal{L}$ be a signature. An $\mathcal{L}$-interpretation is a triple $\mathcal{I} = (D, I, V)$ where

- D is a nonempty set called the domain or universe (of discourse) of $\mathcal{I}$.
  Notation: $dom(\mathcal{I}) := D$.
- I is a function defined on the symbols of $\mathcal{L}$ mapping
  - each n-ary function symbol f to an n-ary function $f^I : D^n \to D$.
    For $n = 0$ this means $f^I \in D$.
  - each n-ary relation symbol p to an n-ary relation $p^I \subseteq D^n$.
    For $n = 0$ this means either $p^I = \emptyset$ or $p^I = \{()\}$.

  Notation: $f^{\mathcal{I}} := f^I$ and $p^{\mathcal{I}} := p^I$.
- V is a variable assignment in D. **Notation:** $x^{\mathcal{I}} := x^V$.

Value of Terms

Definition

The value of a term t in an interpretation $\mathcal{I}$, denoted $t^{\mathcal{I}}$, is an element of $dom(\mathcal{I})$ and inductively defined:

1. If t is a variable or a constant, then $t^{\mathcal{I}}$ is defined as above.
2. If t is a compound term $f(t_1, \ldots, t_n)$, then $t^{\mathcal{I}}$ is defined as $f^{\mathcal{I}}(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}})$

Value of Terms

Definition

The value of a term t in an interpretation $\mathcal{I}$, denoted $t^{\mathcal{I}}$, is an element of $dom(\mathcal{I})$ and inductively defined:

1. If t is a variable or a constant, then $t^{\mathcal{I}}$ is defined as above.
2. If t is a compound term $f(t_1, \ldots, t_n)$, then $t^{\mathcal{I}}$ is defined as $f^{\mathcal{I}}(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}})$

Notation

Let V be a variable assignment in D, $x \in V$, and $d \in D$. Then $V[x \mapsto d]$ is the variable assignment which, for every variable z, is defined as follows:

$$z^{V[x \mapsto d]} = \begin{cases} d & \text{if } z = x \\ z^V & \text{if } z \neq x \end{cases}$$

Let $\mathcal{I} = (D, I, V)$ be an interpretation. Then $\mathcal{I}[x \mapsto d] := (D, I, V[x \mapsto d])$.

Value of Formulas

Definition (Tarski, model relationship)

Let $\mathcal{I}$ be an interpretation and φ a formula. The relationship $\mathcal{I} \models \varphi$, pronounced *"$\mathcal{I}$ is a model of φ"* or *"$\mathcal{I}$ satisfies φ"* or *"φ is true in $\mathcal{I}$"*, and its negation $\mathcal{I} \not\models \varphi$, pronounced *"$\mathcal{I}$ falsifies φ"* or *"φ is false in $\mathcal{I}$"*, are defined inductively:

$$
\begin{array}{lll}
\mathcal{I} \models p(t_1, \ldots, t_n) & \text{iff} & (t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}}) \in p^{\mathcal{I}} \qquad (n\text{-ary } p,\ n \geq 1) \\
\mathcal{I} \models p & \text{iff} & () \in p^{\mathcal{I}} \qquad (0\text{-ary } p) \\
\mathcal{I} \not\models \bot & & \\
\mathcal{I} \models \top & & \\
\mathcal{I} \models \neg\psi & \text{iff} & \mathcal{I} \not\models \psi \\
\mathcal{I} \models (\psi_1 \wedge \psi_2) & \text{iff} & \mathcal{I} \models \psi_1 \text{ and } \mathcal{I} \models \psi_2 \\
\mathcal{I} \models (\psi_1 \vee \psi_2) & \text{iff} & \mathcal{I} \models \psi_1 \text{ or } \mathcal{I} \models \psi_2 \\
\mathcal{I} \models (\psi_1 \Rightarrow \psi_2) & \text{iff} & \mathcal{I} \not\models \psi_1 \text{ or } \mathcal{I} \models \psi_2 \\
\mathcal{I} \models \forall x\, \psi & \text{iff} & \mathcal{I}[x \mapsto d] \models \psi \text{ for each } d \in D \\
\mathcal{I} \models \exists x\, \psi & \text{iff} & \mathcal{I}[x \mapsto d] \models \psi \text{ for at least one } d \in D
\end{array}
$$

For a set S of formulas, $\mathcal{I} \models S$ iff $\mathcal{I} \models \varphi$ for each $\varphi \in S$.

Example

Signature:

function symbols: 0-ary a, b 1-ary f

relation symbols: 1-ary p, q 2-ary r

Example

Signature:

function symbols: 0-ary a, b 1-ary f

relation symbols: 1-ary p, q 2-ary r

Formula:

$$\varphi = q(a) \ \wedge \ r(a,b) \ \wedge \ \neg r(f(a),b) \ \wedge \ \forall x \Big(p(x) \Rightarrow r(x, f(x)) \Big)$$

Example

Signature:

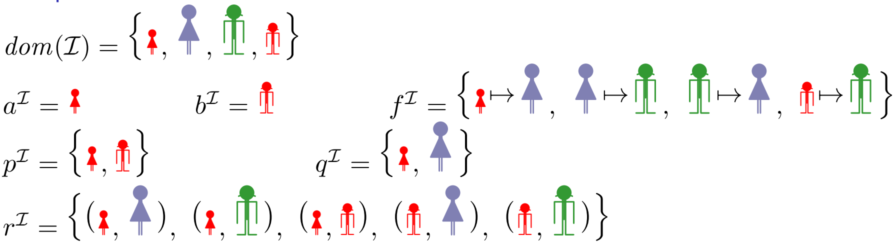
function symbols: 0-ary a, b 1-ary f

relation symbols: 1-ary p, q 2-ary r

Formula:

$$\varphi = q(a) \ \wedge \ r(a,b) \ \wedge \ \neg r(f(a),b) \ \wedge \ \forall x\Big(p(x) \Rightarrow r(x, f(x))\Big)$$

Interpretation $\mathcal{I}$:



Example

Signature:

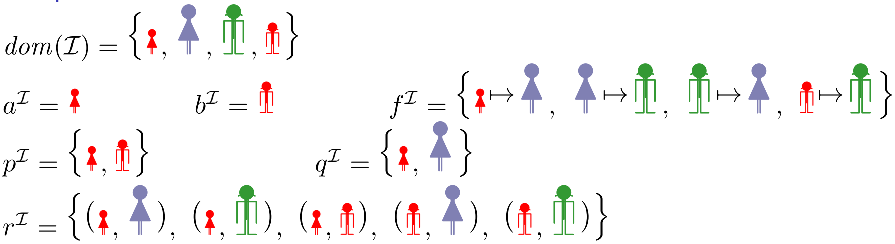
function symbols: 0-ary a, b 1-ary f

relation symbols: 1-ary p, q 2-ary r

Formula:

$$\varphi = q(a) \;\wedge\; r(a,b) \;\wedge\; \neg r(f(a),b) \;\wedge\; \forall x\Big(p(x) \Rightarrow r(x, f(x))\Big)$$

Interpretation $\mathcal{I}$:



Model relationship:

We can check that $\mathcal{I} \models \varphi$ holds.

Semantic Properties, Entailment, Logical Equivalence

Semantic Properties. A formula is

valid	iff	it is satisfied in each interpretation	$p \vee \neg p$
satisfiable	iff	it is satisfied in at least one interpretation	p
falsifiable	iff	it is falsified in at least one interpretation	p
unsatisfiable	iff	it is falsified in each interpretation	$p \wedge \neg p$

Semantic Properties, Entailment, Logical Equivalence

Semantic Properties. A formula is

valid	iff	it is satisfied in each interpretation	$p \vee \neg p$
satisfiable	iff	it is satisfied in at least one interpretation	p
falsifiable	iff	it is falsified in at least one interpretation	p
unsatisfiable	iff	it is falsified in each interpretation	$p \wedge \neg p$

Entailment, Logical Equivalence. For formulas φ and ψ

$\varphi \models \psi$	iff	for each interpretation $\mathcal{I}$: if $\mathcal{I} \models \varphi$ then $\mathcal{I} \models \psi$	$(p \wedge q) \models (p \vee q)$
$\varphi \models\!\!\mid \psi$	iff	$\varphi \models \psi$ and $\psi \models \varphi$	$(p \wedge q) \models\!\!\mid (q \wedge p)$

Semantic Properties, Entailment, Logical Equivalence

Semantic Properties. A formula is

valid	iff	it is satisfied in each interpretation	$p \vee \neg p$
satisfiable	iff	it is satisfied in at least one interpretation	p
falsifiable	iff	it is falsified in at least one interpretation	p
unsatisfiable	iff	it is falsified in each interpretation	$p \wedge \neg p$

Entailment, Logical Equivalence. For formulas φ and ψ

$\varphi \models \psi$	iff	for each interpretation $\mathcal{I}$: if $\mathcal{I} \models \varphi$ then $\mathcal{I} \models \psi$	$(p \wedge q) \models (p \vee q)$
$\varphi \models\!\!\!| \psi$	iff	$\varphi \models \psi$ and $\psi \models \varphi$	$(p \wedge q) \models\!\!\!| (q \wedge p)$

Inter-translatability: Being able to determine one of validity, unsatisfiability, or entailment, is sufficient to determine all of them:

φ is valid	iff $\neg\varphi$ is unsatisfiable	iff $\top \models \varphi$.
φ is unsatisfiable	iff $\neg\varphi$ is valid	iff $\varphi \models \bot$.
$\varphi \models \psi$	iff $(\varphi \Rightarrow \psi)$ is valid	iff $(\varphi \wedge \neg\psi)$ is unsatisfiable.

Calculi, Proof Systems

Motivation

- Entailment $\varphi \models \psi$ formalises the concept of logical consequence.
- A major concern in logic is the development of calculi, also called proof systems, which formalise the notion of deductive inference.

Calculi, Proof Systems

Motivation

- Entailment $\varphi \models \psi$ formalises the concept of logical consequence.
- A major concern in logic is the development of calculi, also called proof systems, which formalise the notion of deductive inference.

Definition (Calculus)

- A calculus defines derivation rules, with which formulas can be derived from formulas by purely syntactic operations.
- The derivability relationship $\varphi \vdash \psi$ for a calculus holds iff there is a finite sequence of applications of derivation rules of the calculus, which applied to φ result in ψ.
- Ideally, derivability should mirror entailment: a calculus is called sound iff whenever $\varphi \vdash \psi$ then $\varphi \models \psi$ and complete iff whenever $\varphi \models \psi$ then $\varphi \vdash \psi$.

Important Results about Tarski Model Theory

Theorem (Gödel, completeness theorem)

There exist calculi for first-order predicate logic such that $S \vdash \varphi$ iff $S \models \varphi$ for any set S of closed formulas and any closed formula φ.

Important Results about Tarski Model Theory

Theorem (Gödel, completeness theorem)

There exist calculi for first-order predicate logic such that $S \vdash \varphi$ *iff* $S \models \varphi$ *for any set* S *of closed formulas and any closed formula* φ*.*

Theorem (Church-Turing, undecidability theorem)

For signatures with a non-propositional relation symbol and a relation or function symbol of arity ≥ 2*, satisfiability is undecidable.*

Important Results about Tarski Model Theory

Theorem (Gödel, completeness theorem)

There exist calculi for first-order predicate logic such that $S \vdash \varphi$ iff $S \models \varphi$ for any set S of closed formulas and any closed formula φ.

Theorem (Church-Turing, undecidability theorem)

For signatures with a non-propositional relation symbol and a relation or function symbol of arity ≥ 2, satisfiability is undecidable.

Theorem (Gödel-Malcev, finiteness or compactness theorem)

Let S be an infinite set of closed formulas. If every finite subset of S is satisfiable, then S is satisfiable.

Remark. Proofs to be provided in part two of this lecture.

Outline

Equality

Motivation

- In many applications, we want to express equality: For this purpose, let the signature $\mathcal{L}$ contain a special 2-ary relation symbol $\doteq$ for equality.
- The relation symbol $\doteq$ shall indeed be *interpreted* as equality: we consider normal interpretations (and thus treat equality as a built-in predicate).
- Alternatively, we can add equality axioms to the formula: this is fine for many purposes; but it does not exclude non-normal models!

Equality

Motivation

- In many applications, we want to express equality: For this purpose, let the signature $\mathcal{L}$ contain a special 2-ary relation symbol $\doteq$ for equality.
- The relation symbol $\doteq$ shall indeed be *interpreted* as equality: we consider normal interpretations (and thus treat equality as a built-in predicate).
- Alternatively, we can add equality axioms to the formula: this is fine for many purposes; but it does not exclude non-normal models!

Definition (Normal interpretation)

An interpretation $\mathcal{I}$ is normal, iff it interprets the relation symbol $\doteq$ with the equality relation on its domain, i.e., $\doteq^{\mathcal{I}}$ is the relation $\{(d, d) \mid d \in dom(\mathcal{I})\}$.
For formulas or sets of formulas φ and ψ, we write:
$\mathcal{I} \models_{=} \varphi$ iff $\mathcal{I}$ is normal and $\mathcal{I} \models \varphi$.
$\varphi \models_{=} \psi$ iff for each normal interpretation $\mathcal{I}$: if $\mathcal{I} \models_{=} \varphi$ then $\mathcal{I} \models_{=} \psi$.

Equality Axioms

Definition (Equality axioms)

Given a signature $\mathcal{L}$ with 2-ary relation symbol $\doteq$, the set $EQ_{\mathcal{L}}$ of equality axioms for $\mathcal{L}$ consists of the formulas:

- $\forall x \; x \doteq x$ (reflexivity of $\doteq$)
- $\forall x \forall y (x \doteq y \Rightarrow y \doteq x)$ (symmetry of $\doteq$)
- $\forall x \forall y \forall z ((x \doteq y \land y \doteq z) \Rightarrow x \doteq z)$ (transitivity of $\doteq$)
- for each n-ary function symbol f, $n > 0$ (substitution axiom for f)
  $\forall x_1 \dots x_n \forall x'_1 \dots x'_n ((x_1 \doteq x'_1 \land \dots \land x_n \doteq x'_n) \Rightarrow$
  $f(x_1, \dots, x_n) \doteq f(x'_1, \dots, x'_n))$
- for each n-ary relation symbol p, $n > 0$ (substitution axiom for p)
  $\forall x_1 \dots x_n \forall x'_1 \dots x'_n ((x_1 \doteq x'_1 \land \dots \land x_n \doteq x'_n \land p(x_1, \dots, x_n)) \Rightarrow$
  $p(x'_1, \dots, x'_n))$

Theorem (Equality axioms)

- *For each interpretation* $\mathcal{I}$*, if* $\mathcal{I}$ *is normal then* $\mathcal{I} \models EQ_{\mathcal{L}}$.
- *For each interpretation* $\mathcal{I}$ *with* $\mathcal{I} \models EQ_{\mathcal{L}}$ *there is a normal interpretation* $\mathcal{I}_{=}$ *such that for each formula* φ*:* $\mathcal{I} \models \varphi$ *iff* $\mathcal{I}_{=} \models_{=} \varphi$.
- *For each set* S *of formulas and formula* φ*:* $EQ_{\mathcal{L}} \cup S \models \varphi$ *iff* $S \models_{=} \varphi$.

Theorem (Equality axioms)

- *For each interpretation $\mathcal{I}$, if $\mathcal{I}$ is normal then $\mathcal{I} \models EQ_{\mathcal{L}}$.*
- *For each interpretation $\mathcal{I}$ with $\mathcal{I} \models EQ_{\mathcal{L}}$ there is a normal interpretation $\mathcal{I}_=$ such that for each formula φ: $\mathcal{I} \models \varphi$ iff $\mathcal{I}_= \models_= \varphi$.*
- *For each set S of formulas and formula φ: $EQ_{\mathcal{L}} \cup S \models \varphi$ iff $S \models_= \varphi$.*

Corollary (Finiteness or compactness theorem with equality)

Let S be an infinite set of closed formulas with equality. If every finite subset of S has a normal model, then S has a normal model.

Theorem (Equality axioms)

- *For each interpretation $\mathcal{I}$, if $\mathcal{I}$ is normal then $\mathcal{I} \models EQ_{\mathcal{L}}$.*
- *For each interpretation $\mathcal{I}$ with $\mathcal{I} \models EQ_{\mathcal{L}}$ there is a normal interpretation $\mathcal{I}_=$ such that for each formula φ: $\mathcal{I} \models \varphi$ iff $\mathcal{I}_= \models_= \varphi$.*
- *For each set S of formulas and formula φ: $EQ_{\mathcal{L}} \cup S \models \varphi$ iff $S \models_= \varphi$.*

Corollary (Finiteness or compactness theorem with equality)

Let S be an infinite set of closed formulas with equality. If every finite subset of S has a normal model, then S has a normal model.

Proof (sketch)

Consider the infinite set $S \cup EQ_{\mathcal{L}}$ and an arbitrary finite subset $S' \cup E'$ of $S \cup EQ_{\mathcal{L}}$ with $S' \subseteq S$ and $E' \subseteq EQ_{\mathcal{L}}$.
(1) By assumption, S' has a normal model $\mathcal{I}$. By the theorem, we conclude that $\mathcal{I}$ is a model of $S' \cup EQ_{\mathcal{L}}$ and hence of $S' \cup E'$. Hence, $S' \cup E'$ is satisfiable.
(2) Thus, by compactness, $S \cup EQ_{\mathcal{L}}$ has a model $\mathcal{I}'$. Therefore, by the theorem, there exists a normal interpretation $\mathcal{I}'_=$ with $\mathcal{I}'_= \models_= S$. □

Model Extension Theorem and Non-Normal Models

Theorem (Model extension theorem)

For each interpretation $\mathcal{I}$ and each set $D' \supseteq dom(\mathcal{I})$ there is an interpretation $\mathcal{I}'$ with $dom(\mathcal{I}') = D'$ such that for each formula φ: $\mathcal{I} \models \varphi$ iff $\mathcal{I}' \models \varphi$.

Model Extension Theorem and Non-Normal Models

Theorem (Model extension theorem)

For each interpretation $\mathcal{I}$ *and each set* $D' \supseteq dom(\mathcal{I})$ *there is an interpretation* $\mathcal{I}'$ *with* $dom(\mathcal{I}') = D'$ *such that for each formula* φ*:*
$\mathcal{I} \models \varphi$ *iff* $\mathcal{I}' \models \varphi$*.*

Proof (sketch)

Fix an arbitrary element $d \in dom(\mathcal{I})$. The idea is to let all "new" elements behave exactly like d. For this purpose, we define an auxiliary function π mapping each "new" element to d and each "old" element to itself:
$\pi : D' \to dom(\mathcal{I})$, $\pi(d') := d$ if $d' \notin dom(\mathcal{I})$, $\pi(d') := d'$ if $d' \in dom(\mathcal{I})$.
Then we define $f^{\mathcal{I}'} : D'^{\,n} \to D'$, $f^{\mathcal{I}'}(d_1, \ldots, d_n) := f^{\mathcal{I}}(\pi(d_1), \ldots, \pi(d_n))$ and $p^{\mathcal{I}'} \subseteq D'^{\,n}$, $p^{\mathcal{I}'} := \{ (d_1, \ldots, d_n) \in D'^{\,n} \mid (\pi(d_1), \ldots, \pi(d_n)) \in p^{\mathcal{I}} \}$ for all signature symbols and arities. □

Model Extension Theorem and Non-Normal Models

Corollary (Existence of non-normal models)

Every satisfiable set of formulas has non-normal models.

Model Extension Theorem and Non-Normal Models

Corollary (Existence of non-normal models)

Every satisfiable set of formulas has non-normal models.

Proof (sketch)

By the construction in the above proof, if $(d, d) \in \doteq^{\mathcal{I}}$ then $(d, d') \in \doteq^{\mathcal{I}'}$ for each $d' \in D'$ and the fixed element $d \in dom(\mathcal{I})$. Hence, if $\mathcal{I}'$ is any proper extension of a normal interpretation $\mathcal{I}$, then $\mathcal{I}'$ is not normal. □

Model Extension Theorem and Non-Normal Models

Corollary (Existence of non-normal models)

Every satisfiable set of formulas has non-normal models.

Proof (sketch)

By the construction in the above proof, if $(d, d) \in \doteq^{\mathcal{I}}$ then $(d, d') \in \doteq^{\mathcal{I}'}$ for each $d' \in D'$ and the fixed element $d \in dom(\mathcal{I})$. Hence, if $\mathcal{I}'$ is any proper extension of a normal interpretation $\mathcal{I}$, then $\mathcal{I}'$ is not normal. □

Remarks

- Every model of $EQ_{\mathcal{L}}$ interprets $\doteq$ by a congruence relation on the domain.
- The equality relation is the special case with singleton congruence classes.
- Because of the model extension theorem, there is no way to prevent models with larger congruence classes, unless equality is treated as built-in by making interpretations normal by definition.

Outline

Undecidability of First-Order Predicate Logic

Motivation

- We inspect a proof of the undecidability of (the satisfiability or validity of) first-order predicate logic.
- The proof is folklore
  - It does not make use of equality at all.
  - The first-order formula is a conjunction of Horn clauses.

Undecidability of First-Order Predicate Logic

Motivation

- We inspect a proof of the undecidability of (the satisfiability or validity of) first-order predicate logic.
- The proof is folklore
  - It does not make use of equality at all.
  - The first-order formula is a conjunction of Horn clauses.

Proof idea

We reduce (a variant of) the Halting Problem to the unsatisfiability problem:

Given a deterministic Turing machine T with halting state $\mathtt{h}$, *it is undecidable if T when starting with the empty tape eventually reaches the halting state* $\mathtt{h}$.

Turing Machines

Definition (Deterministic Turing machine)

A deterministic Turing machine (DTM) is defined as a quadruple (S, Σ, δ, q_0) with the following meaning: S is a finite set of states, Σ is a finite alphabet of symbols, δ is a transition function, and $q_0 \in S$ is the initial state. The alphabet Σ contains a special symbol $\sqcup$ called blank. The transition function δ is a map

$$\delta : \quad S \times \Sigma \quad \to \quad (S \cup \{\mathtt{h}\}) \times \Sigma \times \{\mathtt{-1,\ 0,\ +1}\},$$

where h denotes an additional state (the halting state) not occurring in S, and -1, 0, +1 denote motion directions.

We may assume w.l.o.g., that the machine never moves off the left end of the tape, i.e., $d \neq \mathtt{-1}$ whenever the cursor is on the leftmost cell; this can be easily ensured by a special symbol $\triangleright$ which marks the left end of the tape.

Computation of a Turing Machine

Configurations

Let T be a DTM (Σ, S, δ, q_0). The tape of T is divided into cells containing symbols of Σ. There is a cursor that may move along the tape. At every time instant, the current configuration of T is characterized by a tuple (q, w, σ, w'), where q denotes the state, w and w' denote the tape contents (written as string) to the left/right of the cursor and σ denotes the currently scanned symbol.

Computation of a Turing Machine

Configurations

Let T be a DTM (Σ, S, δ, q_0). The tape of T is divided into cells containing symbols of Σ. There is a cursor that may move along the tape. At every time instant, the current configuration of T is characterized by a tuple (q, w, σ, w'), where q denotes the state, w and w' denote the tape contents (written as string) to the left/right of the cursor and σ denotes the currently scanned symbol.

Initial Configuration

On input string I, the TM T is initially in configuration $(q_0, \varepsilon, \triangleright, I)$, i.e., T is in the initial state q_0, the tape contains the start symbol $\triangleright$ followed by the input string I, and the cursor points to the leftmost cell of the tape.

Computation of a Turing Machine

Configurations

Let T be a DTM (Σ, S, δ, q_0). The tape of T is divided into cells containing symbols of Σ. There is a cursor that may move along the tape. At every time instant, the current configuration of T is characterized by a tuple (q, w, σ, w'), where q denotes the state, w and w' denote the tape contents (written as string) to the left/right of the cursor and σ denotes the currently scanned symbol.

Initial Configuration

On input string I, the TM T is initially in configuration $(q_0, \varepsilon, \triangleright, I)$, i.e., T is in the initial state q_0, the tape contains the start symbol $\triangleright$ followed by the input string I, and the cursor points to the leftmost cell of the tape.

Notation

We denote a configuration (q, w, σ, w') in the format $B\Sigma^* S\Sigma^* E$, with the state symbol written in front of the currently scanned tape symbol. B and E are symbols marking the beginning and the end of the tape contents $(w\sigma w')$.

Computation Step

The transition relation for T, denoted by $\vdash_T$, is defined as follows:

1. $Bwaq\sigma w'E \vdash_T Bwq'a\sigma'w'E$, if $\delta(q,\sigma) = (q',\sigma',\text{-}1)$.
2. $Bwq\sigma w'E \vdash_T Bwq'\sigma'w'E$, if $\delta(q,\sigma) = (q',\sigma',0)$.
3. $Bwq\sigma aw'E \vdash_T Bw\sigma'q'aw'E$ and $Bwq\sigma E \vdash_T Bw\sigma'q' \sqcup E$, if $\delta(q,\sigma) = (q',\sigma',\text{+}1)$.

We write $\vdash^*_T$ to denote the reflexive and transitive closure of $\vdash_T$.

Computation Step

The transition relation for T, denoted by $\vdash_T$, is defined as follows:

1. $Bwaq\sigma w'E \vdash_T Bwq'a\sigma'w'E$, if $\delta(q,\sigma) = (q', \sigma', \text{-1})$.
2. $Bwq\sigma w'E \vdash_T Bwq'\sigma'w'E$, if $\delta(q,\sigma) = (q', \sigma', 0)$.
3. $Bwq\sigma aw'E \vdash_T Bw\sigma'q'aw'E$ and $Bwq\sigma E \vdash_T Bw\sigma'q' \sqcup E$, if $\delta(q,\sigma) = (q', \sigma', \text{+1})$.

We write $\vdash^*_T$ to denote the reflexive and transitive closure of $\vdash_T$.

Halting

T halts when it reaches the state $\mathtt{h}$, i.e., there exist values w, σ, and w', s.t. T reaches the configuration $(\mathtt{h}, w, \sigma, w')$.

That is, T halts on input I if $Bq_0 \triangleright IE \vdash^*_T Bw\mathtt{h}\sigma w'E$ for some w, σ, and w'.

Proof of the Undecidability of First-Order Predicate Logic

Encoding of TM configurations as atoms

For every state $q \in S$, let $\widehat{q}$ be a constant symbol.
For every tape symbol $a \in \Sigma$, let $\widehat{a}$ be a unary function symbol.
The constant symbols $\widehat{B}$ and $\widehat{E}$ correspond to the end-of-tape markers B and E.

A configuration $B\sigma_1 \ldots \sigma_m q \sigma_{m+1} \ldots \sigma_n E$ is represented by the atom

$$P(\widehat{\sigma_m}(\ldots \widehat{\sigma_1}(\widehat{B}) \ldots), \widehat{q}, \widehat{\sigma_{m+1}}(\ldots \widehat{\sigma_n}(\widehat{E}) \ldots))$$

(The tape to the left of the current position is represented in reversed order.)

Encoding of TM computations

Given an arbitrary TM T, we define the set Φ_T of formulas as the smallest set containing the following formulas (i.e., Horn clauses):

1. If $\delta(q,\sigma) = (q',\sigma',\text{-}1)$ then for all $a \in \Sigma$,
   $(\forall x)(\forall y)P(\widehat{a}(x),\widehat{q},\widehat{\sigma}(y)) \Rightarrow P(x,\widehat{q'},\widehat{a}(\widehat{\sigma'}(y))) \in \Phi_T$
2. If $\delta(q,\sigma) = (q',\sigma',0)$ then $(\forall x)(\forall y)P(x,\widehat{q},\widehat{\sigma}(y)) \Rightarrow P(x,\widehat{q'},\widehat{\sigma'}(y)) \in \Phi_T$
3. If $\delta(q,\sigma) = (q',\sigma',+1)$ then $(\forall x)(\forall y)P(x,\widehat{q},\widehat{\sigma}(y)) \Rightarrow P(\widehat{\sigma'}(x),\widehat{q'},y)) \in \Phi_T$
4. $(\forall x)P(x,\widehat{q},\widehat{E}) \Rightarrow P(x,\widehat{q},\widehat{\sqcup}(\widehat{E})) \in \Phi_T$.

Encoding of TM computations

Given an arbitrary TM T, we define the set Φ_T of formulas as the smallest set containing the following formulas (i.e., Horn clauses):

1. If $\delta(q,\sigma) = (q', \sigma', \text{-}1)$ then for all $a \in \Sigma$, $(\forall x)(\forall y)P(\widehat{a}(x), \widehat{q}, \widehat{\sigma}(y)) \Rightarrow P(x, \widehat{q'}, \widehat{a}(\widehat{\sigma'}(y))) \in \Phi_T$
2. If $\delta(q,\sigma) = (q', \sigma', 0)$ then $(\forall x)(\forall y)P(x, \widehat{q}, \widehat{\sigma}(y)) \Rightarrow P(x, \widehat{q'}, \widehat{\sigma'}(y)) \in \Phi_T$
3. If $\delta(q,\sigma) = (q', \sigma', +1)$ then $(\forall x)(\forall y)P(x, \widehat{q}, \widehat{\sigma}(y)) \Rightarrow P(\widehat{\sigma'}(x), \widehat{q'}, y)) \in \Phi_T$
4. $(\forall x)P(x, \widehat{q}, \widehat{E}) \Rightarrow P(x, \widehat{q}, \widehat{\sqcup}(\widehat{E})) \in \Phi_T$.

Proposition

For any Turing machine T, every $v, v', w, w' \in \Sigma^*$ and $q, q' \in S$ with $v = v_1, \ldots, v_r$, $v' = v'_1, \ldots, v'_{r'}$, $w = w_1, \ldots, w_s$, and $w = w'_1, \ldots, w'_{s'}$:
$BvqwE \vdash^*_T Bv'q'w'E$ *iff*
$\Phi_T \models P(\widehat{v_r}(..\widehat{v_1}(\widehat{B})..), \widehat{q}, \widehat{w_1}(..\widehat{w_s}(\widehat{E})..)) \Rightarrow P(\widehat{v'_{r'}}(..\widehat{v'_1}(\widehat{B})..), \widehat{q'}, \widehat{w'_1}(..\widehat{w'_{s'}}(\widehat{E})..))$

For any Turing machine T, $\Phi_T \cup \{P(\widehat{B}, \widehat{q_0}, \widehat{\triangleright}(\widehat{E})), (\forall x)(\forall y)\neg P(x, \widehat{h}, y)\}$ is unsatisfiable iff T, when starting with the empty tape, eventually halts.

Outline

Model Cardinalities

Motivation

We sometimes want to enforce that a formula only has models of a certain cardinality, e.g.:

- (only) infinite models
- (only) finite models
- (only) finite models with cardinality bounded by some constant
- etc.

Some of these properties cannot be expressed in first-order logic (possibly not even if we may use equality).

Theorem

Lower bounds of model cardinalities can be expressed in first-order predicate logic (even without equality).

Theorem

Lower bounds of model cardinalities can be expressed in first-order predicate logic (even without equality).

Example

All models of the following satisfiable set of formulas have domains with cardinality ≥ 3:

$$\begin{aligned}\{\quad & \exists x_1(\ \ p_1(x_1) \wedge \neg p_2(x_1) \wedge \neg p_3(x_1)),\\ & \exists x_2(\neg p_1(x_2) \wedge \ \ p_2(x_2) \wedge \neg p_3(x_2)),\\ & \exists x_3(\neg p_1(x_3) \wedge \neg p_2(x_3) \wedge \ \ p_3(x_3)) \quad \}\end{aligned}$$

Theorem

Lower bounds of model cardinalities can be expressed in first-order predicate logic (even without equality).

Example

All models of the following satisfiable set of formulas have domains with cardinality ≥ 3:

$$\begin{array}{ll} \{ & \exists x_1(\ \ p_1(x_1) \wedge \neg p_2(x_1) \wedge \neg p_3(x_1)), \\ & \exists x_2(\neg p_1(x_2) \wedge \ \ p_2(x_2) \wedge \neg p_3(x_2)), \\ & \exists x_3(\neg p_1(x_3) \wedge \neg p_2(x_3) \wedge \ \ p_3(x_3)) \quad \} \end{array}$$

Example

All models of the following satisfiable set of formulas have infinite domains:

$$\{\ \forall x\, \neg(x < x),\ \ \forall x \forall y \forall z (x < y \wedge y < z \Rightarrow x < z),\ \ \forall x \exists y\, x < y\ \}.$$

Inexpressibility without Equality

Theorem

Upper bounds of model cardinalities cannot be expressed in first-order predicate logic without equality.

Theorem

Each satisfiable set of formulas without equality has models with infinite domain.

Corollary

Finiteness cannot be expressed in first-order predicate logic without equality.

Inexpressibility without Equality

Theorem

Upper bounds of model cardinalities cannot be expressed in first-order predicate logic without equality.

Theorem

Each satisfiable set of formulas without equality has models with infinite domain.

Corollary

Finiteness cannot be expressed in first-order predicate logic without equality.

Proof (sketch)

All three results immediately follow from the model extension theorem. □

Expressibility and Inexpressibility with Equality

Theorem

Bounded finiteness can be expressed in first-order predicate logic with equality. That is, for any given natural number $k \geq 1$, the upper bound k of model cardinalities can be expressed.

Expressibility and Inexpressibility with Equality

Theorem

Bounded finiteness can be expressed in first-order predicate logic with equality. That is, for any given natural number $k \geq 1$, the upper bound k of model cardinalities can be expressed.

Example

All *normal* models of the following satisfiable formula have domains with cardinality ≤ 3: $\exists x_1 \exists x_2 \exists x_3 \forall y (y \doteq x_1 \vee y \doteq x_2 \vee y \doteq x_3)$.

Theorem

If a set of formulas with equality has arbitrarily large finite normal models, then it has an infinite normal model.

Theorem

If a set of formulas with equality has arbitrarily large finite normal models, then it has an infinite normal model.

Proof

Let S be such that for each $k \in \mathbb{N}$ there is a normal model of S whose domain has finite cardinality $> k$. We show that S has an infinite normal model.

For each $n \in \mathbb{N}$ let φ_n be the formula $\forall x_0 \ldots x_n \exists y(\neg(y \doteq x_0) \wedge \ldots \wedge \neg(y \doteq x_n))$ expressing "more than n elements". Then every finite subset of $S \cup \{\varphi_n \mid n \in \mathbb{N}\}$ has a normal model. By the finiteness/compactness theorem with equality, $S \cup \{\varphi_n \mid n \in \mathbb{N}\}$ has a normal model $\mathcal{I}$.

Obviously, $\mathcal{I}$ cannot be finite, but is also a normal model of S. □

Corollary

A satisfiable set of formulas with equality has either only finite normal models of a bounded cardinality, or infinite normal models.

Corollary

Unbounded finiteness cannot be expressed in first-order predicate logic with equality.

Corollary

A satisfiable set of formulas with equality has either only finite normal models of a bounded cardinality, or infinite normal models.

Corollary

Unbounded finiteness cannot be expressed in first-order predicate logic with equality.

Theorem (Löwenheim-Skolem)

Every satisfiable enumerable set of closed formulas has a model with a finite or infinite enumerable domain.