Logical and Theoretical Foundations of Computer Science

LaTFoCS

Pamela Fleischmann

fpa@informatik.uni-kiel.de

Winter Semester 2019

Kiel University
Dependable Systems Group

Problems with Automated Checks

now we have analogous definitions to the propositional logic - can we again automate the check for validity?

Problems with Automated Checks

now we have analogous definitions to the propositional logic - can we again automate the check for validity?

- all-quantifier and infinite universe means infinitely many checks . . .

Problems with Automated Checks

now we have analogous definitions to the propositional logic - can we again automate the check for validity?

- all-quantifier and infinite universe means infinitely many checks . . .
- for semantic entailment we have to check infinitely many models

Problems with Automated Checks

now we have analogous definitions to the propositional logic - can we again automate the check for validity?

- all-quantifier and infinite universe means infinitely many checks . . .
- for semantic entailment we have to check infinitely many models

automation is not possible in general!

Quantifier Equivalence Example

Consider $A = \{1, 2\}$, $P^{\mathscr{M}} = \{1\}$, and $Q^{\mathscr{M}} = \{2\}$

- is $\mathscr{M}$ a model of $\forall x P(x) \to \forall x Q(x)$?

Quantifier Equivalence Example

Consider $A = \{1, 2\}$, $P^{\mathscr{M}} = \{1\}$, and $Q^{\mathscr{M}} = \{2\}$

- is $\mathscr{M}$ a model of $\forall x P(x) \to \forall x Q(x)$?

$$
\begin{aligned}
&\mathscr{M} \models \forall x P(x) \to \forall x Q(x)\\
\text{iff } &\mathscr{M} \models \forall x P(x) \text{ implies } \mathscr{M} \models \forall x Q(x)\\
\text{iff } &\text{for all } a \in A (\mathscr{M} \models_{\ell[x \mapsto a]} P(x)) \text{ implies}\\
&\text{for all } b \in A (\mathscr{M} \models_{\ell[x \mapsto b]} Q(x))
\end{aligned}
$$

Quantifier Equivalence Example

Consider $A = \{1, 2\}$, $P^{\mathscr{M}} = \{1\}$, and $Q^{\mathscr{M}} = \{2\}$

- is $\mathscr{M}$ a model of $\forall x P(x) \to \forall x Q(x)$?

$$\begin{aligned} &\mathscr{M} \models \forall x P(x) \to \forall x Q(x) \\ \text{iff } &\mathscr{M} \models \forall x P(x) \text{ implies } \mathscr{M} \models \forall x Q(x) \\ \text{iff } &\text{for all } a \in A (\mathscr{M} \models_{\ell[x \mapsto a]} P(x)) \text{ implies} \\ &\text{for all } b \in A (\mathscr{M} \models_{\ell[x \mapsto b]} Q(x)) \end{aligned}$$

an implication is true if either the premise is false or both the premise and the conclusion are true

- premise only for $a = 1$ true but the conclusion isn't true for $b = 1$!

Undecidability of Predicate Logic

Decision Problem for Predicate Logic

Definition

Given a predicate logic formula φ, decide whether $\models \varphi$ holds.

Decision Problem for Predicate Logic

Definition

Given a predicate logic formula φ, decide whether $\models \varphi$ holds.

- Can we write a computer programme answering this question correctly?

Decision Problem for Predicate Logic

Definition

Given a predicate logic formula φ, decide whether $\models \varphi$ holds.

- Can we write a computer programme answering this question correctly?
- we cannot and we will prove this

Problem Reduction

- if we take an unsolvable problem . . .

Problem Reduction

- if we take an unsolvable problem …
- and reduce it to our problem (each instance is mapped to one of our problem) …

Problem Reduction

- if we take an unsolvable problem . . .
- and reduce it to our problem (each instance is mapped to one of our problem) . . .
- can our problem be solvable?

Problem Reduction

- if we take an unsolvable problem . . .
- and reduce it to our problem (each instance is mapped to one of our problem) . . .
- can our problem be solvable?
- it cannot, using the rules $(\neg i)$ and $(\neg e)$

Post Correspondence Problem (PCP)

Definition

Given $(s_1, t_1), \ldots, (s_k, t_k) \in \{0,1\}^* \times \{0,1\}^*$ (binary strings), decide whether there exists a sequence of indices $i_1, \ldots, i_n$ such that

$$s_{i_1} \ldots s_{i_n} = t_{i_1} \ldots t_{i_n}.$$

Post Correspondence Problem (PCP)

Definition

Given $(s_1, t_1), \ldots, (s_k, t_k) \in \{0,1\}^* \times \{0,1\}^*$ (binary strings), decide whether there exists a sequence of indices $i_1, \ldots, i_n$ such that

$$s_{i_1} \ldots s_{i_n} = t_{i_1} \ldots t_{i_n}.$$

Notice that you are allowed to take a tuple more than once!

PCP and Domino

A visualisation of PCP is the game Domino:

- consider the tuples to be domino tiles $\begin{pmatrix} s_i \\ t_i \end{pmatrix}$
- you have each tile as often as you like
- put them next to each other such that the upper row and the lower row are identical

PCP and Domino

A visualisation of PCP is the game Domino:

- consider the tuples to be domino tiles $\begin{pmatrix} s_i \\ t_i \end{pmatrix}$
- you have each tile as often as you like
- put them next to each other such that the upper row and the lower row are identical

Example: given $\begin{pmatrix} 1 \\ 101 \end{pmatrix}, \begin{pmatrix} 10 \\ 00 \end{pmatrix}, \begin{pmatrix} 011 \\ 11 \end{pmatrix}$

PCP and Domino

A visualisation of PCP is the game Domino:

- consider the tuples to be domino tiles $\begin{pmatrix} s_i \\ t_i \end{pmatrix}$
- you have each tile as often as you like
- put them next to each other such that the upper row and the lower row are identical

Example: given $\begin{pmatrix} 1 \\ 101 \end{pmatrix}, \begin{pmatrix} 10 \\ 00 \end{pmatrix}, \begin{pmatrix} 011 \\ 11 \end{pmatrix}$

we have a solution with

$$\begin{pmatrix} 1 \\ 101 \end{pmatrix}\begin{pmatrix} 011 \\ 11 \end{pmatrix}\begin{pmatrix} 10 \\ 00 \end{pmatrix}\begin{pmatrix} 011 \\ 11 \end{pmatrix} = \begin{pmatrix} 101110011 \\ 101110011 \end{pmatrix}$$

Undecidability of Predicate Logic

Theorem

The decision problem of validity in predicate logic is undecidable.

Undecidability of Predicate Logic

Theorem

The decision problem of validity in predicate logic is undecidable.

Proof.

- plan: reducing PCP to the decision problem

Undecidability of Predicate Logic

Theorem

The decision problem of validity in predicate logic is undecidable.

Proof.

- plan: reducing PCP to the decision problem
- $C = ((s_1, t_1), \ldots, (s_k, t_k))$ instance of PCP

Undecidability of Predicate Logic

Theorem

The decision problem of validity in predicate logic is undecidable.

Proof.

- plan: reducing PCP to the decision problem
- $C = ((s_1, t_1), \ldots, (s_k, t_k))$ instance of PCP
- we need to find in finite space and time a predicate logic formula φ with

$$\models \varphi \quad \text{iff} \quad C \text{ has solution}$$

Undecidability of Predicate Logic

Theorem

The decision problem of validity in predicate logic is undecidable.

Proof.

- plan: reducing PCP to the decision problem
- $C = ((s_1, t_1), \ldots, (s_k, t_k))$ instance of PCP
- we need to find in finite space and time a predicate logic formula φ with

$$\models \varphi \quad \text{iff} \quad C \text{ has solution}$$

- idea for constructing φ: encode the bitstring in a formula

Undecidability of Predicate Logic

Theorem

The decision problem of validity in predicate logic is undecidable.

Proof.

- plan: reducing PCP to the decision problem
- $C = ((s_1, t_1), \ldots, (s_k, t_k))$ instance of PCP
- we need to find in finite space and time a predicate logic formula φ with

$$\models \varphi \quad \text{iff} \quad C \text{ has solution}$$

- idea for constructing φ: encode the bitstring in a formula
- we have constants, predicates, formulae

Proof Cont

- constant e for the empty string

Proof Cont

- constant e for the empty string
- function f_0 and f_1 (for the concatenation of 0 resp. 1)

Proof Cont

- constant e for the empty string
- function f_0 and f_1 (for the concatenation of 0 resp. 1)
- predicate P (for the existence of a sequence comparing s and t)

Proof Cont

- constant e for the empty string
- function f_0 and f_1 (for the concatenation of 0 resp. 1)
- predicate P (for the existence of a sequence comparing s and t)
- $\varphi := \varphi_1 \wedge \varphi_2 \rightarrow \varphi_3$ with

$$\varphi_1 := \bigwedge_{i \in [k]} P(f_{s_i}(e), f_{t_i}(e))$$

$$\varphi_2 := \forall v \forall w (P(v, w) \rightarrow \bigwedge_{i \in [k]} P(f_{s_i}(v), f_{t_i}(w)))$$

$$\varphi_3 := \exists z P(z, z)$$

Proof Cont

Now we have to prove: $\models \varphi$ iff C has a solution.

Proof Cont

Now we have to prove: $\models \varphi$ iff C has a solution. $\Rightarrow$

- assume $\models \varphi$

Proof Cont

Now we have to prove: $\models \varphi$ iff C has a solution. $\Rightarrow$

- assume $\models \varphi$
- $e^{\mathcal{M}} := \varepsilon$

Proof Cont

Now we have to prove: $\models \varphi$ iff C has a solution. $\Rightarrow$

- assume $\models \varphi$
- $e^{\mathcal{M}} := \varepsilon$
- $f_b^{\mathcal{M}}(s) = sb$ for $b \in \{0, 1\}$

Proof Cont

Now we have to prove: $\models \varphi$ iff C has a solution. $\Rightarrow$

- assume $\models \varphi$
- $e^{\mathcal{M}} := \varepsilon$
- $f_b^{\mathcal{M}}(s) = sb$ for $b \in \{0, 1\}$
- $P^{\mathcal{M}} = \{(s, t) \mid \exists (i_1, \ldots, i_m) : \ s = s_1 \ldots s_m \wedge t = t_1 \ldots t_m\}$

Proof Cont

Now we have to prove: $\models \varphi$ iff C has a solution. $\Rightarrow$

- assume $\models \varphi$
- $e^{\mathscr{M}} := \varepsilon$
- $f_b^{\mathscr{M}}(s) = sb$ for $b \in \{0,1\}$
- $P^{\mathscr{M}} = \{(s,t) | \exists (i_1, \ldots, i_m) : s = s_1 \ldots s_m \wedge t = t_1 \ldots t_m\}$
- $\models \varphi$ implies $\mathscr{M} \models \varphi$

Proof Cont

Claim: $\mathscr{M} \models \varphi$ implies $\mathscr{M} \models \varphi_1$ and $\mathscr{M} \models \varphi_2$

Proof:

Proof Cont

Claim: $\mathscr{M} \models \varphi$ implies $\mathscr{M} \models \varphi_1$ and $\mathscr{M} \models \varphi_2$

Proof:

- $(s, t) \in \mathscr{P}^{\mathscr{M}}$ implies sequence $(i_1, \ldots, i_m)$ with $s = s_1 \ldots s_{i_m}$ and $t = t_1 \ldots t_{i_m}$

Proof Cont

Claim: $\mathscr{M} \models \varphi$ implies $\mathscr{M} \models \varphi_1$ and $\mathscr{M} \models \varphi_2$

Proof:

- $(s, t) \in \mathscr{P}^{\mathscr{M}}$ implies sequence $(i_1, \ldots, i_m)$ with $s = s_1 \ldots s_{i_m}$ and $t = t_1 \ldots t_{i_m}$
- choose $(i_1, \ldots, i_m, i)$

Proof Cont

Claim: $\mathscr{M} \models \varphi$ implies $\mathscr{M} \models \varphi_1$ and $\mathscr{M} \models \varphi_2$

Proof:

- $(s, t) \in \mathscr{P}^{\mathscr{M}}$ implies sequence $(i_1, \ldots, i_m)$ with $s = s_1 \ldots s_{i_m}$ and $t = t_1 \ldots t_{i_m}$
- choose $(i_1, \ldots, i_m, i)$
- then $ss_i = s_{i_1} \ldots s_{i_m} s_i$ and $tt_i = t_{i_1} \ldots t_{i_m} t_i$

Proof Cont

Claim: $\mathscr{M} \models \varphi$ implies $\mathscr{M} \models \varphi_1$ and $\mathscr{M} \models \varphi_2$

Proof:

- $(s, t) \in \mathscr{P}^{\mathscr{M}}$ implies sequence $(i_1, \ldots, i_m)$ with $s = s_1 \ldots s_{i_m}$ and $t = t_1 \ldots t_{i_m}$
- choose $(i_1, \ldots, i_m, i)$
- then $ss_i = s_{i_1} \ldots s_{i_m} s_i$ and $tt_i = t_{i_1} \ldots t_{i_m} t_i$
- thus $\mathscr{M} \models \varphi_2$

Proof Cont

- $\mathscr{M} \models \varphi_1 \wedge \varphi_2 \rightarrow \varphi_3$ and $\mathscr{M} \models \varphi_1 \wedge \varphi_2$ imply $\mathscr{M} \models \varphi_3$

Proof Cont

- $\mathscr{M} \models \varphi_1 \wedge \varphi_2 \rightarrow \varphi_3$ and $\mathscr{M} \models \varphi_1 \wedge \varphi_2$ imply $\mathscr{M} \models \varphi_3$
- thus there is a solution for C

Proof Cont

$\Leftarrow$

- $(i_1, \ldots, i_n)$ solution for C

Proof Cont

$\Leftarrow$

- $(i_1, \ldots, i_n)$ solution for C
- to show: all models $\mathcal{M}'$ with a constant $e^{\mathcal{M}'}$, two binary functions $f_0^{\mathcal{M}'}, f_1^{\mathcal{M}'}$ and a binary predicate $P^{\mathcal{M}'}$ satisfy φ ($\mathcal{M}' \models \varphi$)

Proof Cont

$\Leftarrow$

- $(i_1, \ldots, i_n)$ solution for C
- to show: all models $\mathcal{M}'$ with a constant $e^{\mathcal{M}'}$, two binary functions $f_0^{\mathcal{M}'}, f_1^{\mathcal{M}'}$ and a binary predicate $P^{\mathcal{M}'}$ satisfy φ ($\mathcal{M}' \models \varphi$)
- if $\mathcal{M}' \not\models \varphi_1$ or $\mathcal{M}' \not\models \varphi_2$, the claim is true due to the implication

Proof Cont

$\Leftarrow$

- $(i_1, \ldots, i_n)$ solution for C
- to show: all models $\mathscr{M}'$ with a constant $e^{\mathscr{M}'}$, two binary functions $f_0^{\mathscr{M}'}, f_1^{\mathscr{M}'}$ and a binary predicate $P^{\mathscr{M}'}$ satisfy φ ($\mathscr{M}' \models \varphi$)
- if $\mathscr{M}' \not\models \varphi_1$ or $\mathscr{M}' \not\models \varphi_2$, the claim is true due to the implication
- assume $\mathscr{M}' \models \varphi_1 \wedge \varphi_2$

Proof Cont

$\Leftarrow$

- $(i_1, \ldots, i_n)$ solution for C
- to show: all models $\mathscr{M}'$ with a constant $e^{\mathscr{M}'}$, two binary functions $f_0^{\mathscr{M}'}, f_1^{\mathscr{M}'}$ and a binary predicate $P^{\mathscr{M}'}$ satisfy φ ($\mathscr{M}' \models \varphi$)
- if $\mathscr{M}' \not\models \varphi_1$ or $\mathscr{M}' \not\models \varphi_2$, the claim is true due to the implication
- assume $\mathscr{M}' \models \varphi_1 \wedge \varphi_2$
- we need to verify $\mathscr{M} \models \varphi_3$

Proof Cont

- idea: interpreting finite, binary strings in the domain of values of $\mathscr{A}'$ (like interpreter do for one programming language in another)

Proof Cont

- idea: interpreting finite, binary strings in the domain of values of $\mathscr{A}'$ (like interpreter do for one programming language in another)
- define

$$\begin{aligned} \text{interpret}(\varepsilon) &= e^{\mathscr{M}'} \\ \text{interpret}(s0) &= f_0^{\mathscr{M}'}(\text{interpret}(s)) \\ \text{interpret}(s1) &= f_1^{\mathscr{M}'}(\text{interpret}(s)). \end{aligned}$$

Proof Cont

- by $\mathscr{M}' \models \varphi_1$ we have for all $i \in [k]$

$$(\text{interpret}(s_i), \text{interpret}(t_i)) \in P^{\mathscr{M}'}$$

Proof Cont

- by $\mathscr{M}' \models \varphi_1$ we have for all $i \in [k]$

$$(\text{interpret}(s_i), \text{interpret}(t_i)) \in P^{\mathscr{M}'}$$

- by $\mathscr{M}' \models \varphi_2$ we have for all $i \in [k]$

$$(\text{interpret}(ss_i), \text{interpret}(tt_i)) \in P^{\mathscr{M}'}$$

Proof Cont

- by $\mathcal{M}' \models \varphi_1$ we have for all $i \in [k]$

$$(\text{interpret}(s_i), \text{interpret}(t_i)) \in P^{\mathcal{M}'}$$

- by $\mathcal{M}' \models \varphi_2$ we have for all $i \in [k]$

$$(\text{interpret}(ss_i), \text{interpret}(tt_i)) \in P^{\mathcal{M}'}$$

- recursive application leads to

$$(\text{interpret}(s_{i_1} \ldots s_{i_n}), \text{interpret}(t_{i_1} \ldots t_{i_n})) \in P^{\mathcal{M}'}$$

Proof Cont

- $s_{i_1} \ldots s_{i_n}$ and $t_{i_1} \ldots t_{i_n}$ are solution of C

Proof Cont

- $s_{i_1} \dots s_{i_n}$ and $t_{i_1} \dots t_{i_n}$ are solution of C
- $\rightsquigarrow$ they are equal

Proof Cont

- $s_{i_1} \dots s_{i_n}$ and $t_{i_1} \dots t_{i_n}$ are solution of C
- $\rightsquigarrow$ they are equal
- $\rightsquigarrow$ interpret($s_{i_1} \dots s_{i_n}$), and interpret($t_{i_1} \dots t_{i_n}$) are equal

Proof Cont

- $s_{i_1} \dots s_{i_n}$ and $t_{i_1} \dots t_{i_n}$ are solution of C
- $\rightsquigarrow$ they are equal
- $\rightsquigarrow$ interpret($s_{i_1} \dots s_{i_n}$), and interpret($t_{i_1} \dots t_{i_n}$) are equal
- $\mathscr{M}' \models \exists z P(z, z)$ $\square$

Undecidability of Satisfiability

- consequence of the definitional clause $\mathscr{M} \models_{\ell} \neg\varphi$:

Undecidability of Satisfiability

- consequence of the definitional clause $\mathscr{M} \models_\ell \neg\varphi$:

Theorem

φ unsatisfiable iff $\neg\varphi$ valid

Undecidability of Satisfiability

- consequence of the definitional clause $\mathscr{M} \models_{\ell} \neg\varphi$:

Theorem

φ unsatisfiable iff $\neg\varphi$ valid

Corollary

Satisfiability is not decidable.

Undecidability of Provability

Theorem (Not proven here)

$\models \varphi$ *iff* $\vdash \varphi$

Undecidability of Provability

Theorem (Not proven here)

$\models \varphi$ *iff* $\vdash \varphi$

Corollary

Provability is undecidable.

Expressiveness of Predicate Logic

Propositional Logic vs. Predicate Logic

- predicate logic much more expressive than propositional logic

Propositional Logic vs. Predicate Logic

- predicate logic much more expressive than propositional logic
- neither validity, nor satisfiability, nor provability are decidable

Interpretation as Directed Graphs

- in reality we often have only binary relations: software models, design standards, execution models for hardware and software

Interpretation as Directed Graphs

- in reality we often have only binary relations: software models, design standards, execution models for hardware and software
- elements of A can be visualised as nodes, the ones of $R^{\mathcal{M}}$ as edges

Interpretation as Directed Graphs

- in reality we often have only binary relations: software models, design standards, execution models for hardware and software
- elements of A can be visualised as nodes, the ones of $R^{\mathscr{M}}$ as edges
- a question in software models for the automotive industry is for instance: is the node for braking always reachable?

Example

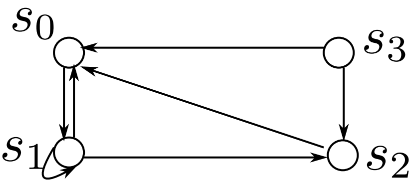
Consider

- $A = \{s_0, s_1, s_2, s_3\}$ and
- $R^{\mathscr{M}} = \{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$

Example

Consider

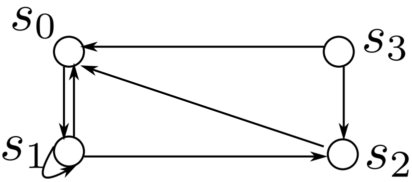
- $A = \{s_0, s_1, s_2, s_3\}$ and
- $R^{\mathscr{M}} = \{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$



Example

Consider

- $A = \{s_0, s_1, s_2, s_3\}$ and
- $R^{\mathscr{M}} = \{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$



If s_3 is the *braking state*, we are not able to brake if we are in before in the states s_0, s_1, or s_2!

Reachability Problem

Definition

Given a directed graph $G = (V, E)$ and nodes $u, v \in V$, decide whether there exists a path from u to v.

Reachability Problem

Definition

Given a directed graph $G = (V, E)$ and nodes $u, v \in V$, decide whether there exists a path from u to v.

- Can we solve this problem with predicate logic?

Reachability Problem

Definition

Given a directed graph $G = (V, E)$ and nodes $u, v \in V$, decide whether there exists a path from u to v.

- Can we solve this problem with predicate logic?
- idea: constructing a formula φ for G, u, v such that φ is satisfiable iff there exists a path from u to v

Reachability Problem

Definition

Given a directed graph $G = (V, E)$ and nodes $u, v \in V$, decide whether there exists a path from u to v.

- Can we solve this problem with predicate logic?
- idea: constructing a formula φ for G, u, v such that φ is satisfiable iff there exists a path from u to v
- unfortunately we are not (φ would be infinitely large)

Reachability Problem

Definition

Given a directed graph $G = (V, E)$ and nodes $u, v \in V$, decide whether there exists a path from u to v.

- Can we solve this problem with predicate logic?
- idea: constructing a formula φ for G, u, v such that φ is satisfiable iff there exists a path from u to v
- unfortunately we are not (φ would be infinitely large)
- we need two other important results before we prove this

Compactness Theorem

Theorem

If all finite subsets of a set of predicat logic sentences Γ *are satisfiable, then* Γ *is as well.*

Compactness Theorem

Theorem

If all finite subsets of a set of predicat logic sentences Γ are satisfiable, then Γ is as well.

Proof:

- Assumption: Γ is not satisfiable

Compactness Theorem

Theorem

If all finite subsets of a set of predicat logic sentences Γ are satisfiable, then Γ is as well.

Proof:

- Assumption: Γ is not satisfiable
- $\rightsquigarrow \Gamma \models \bot$ (no model in which all $\varphi \in \Gamma$ are `true`)

Compactness Theorem

Theorem

If all finite subsets of a set of predicat logic sentences Γ are satisfiable, then Γ is as well.

Proof:

- Assumption: Γ is not satisfiable
- $\rightsquigarrow \Gamma \models \bot$ (no model in which all $\varphi \in \Gamma$ are $\texttt{true}$)
- completeness $\rightsquigarrow \Gamma \vdash \bot$ valid

Theorem

If all finite subsets of a set of predicat logic sentences Γ are satisfiable, then Γ is as well.

Proof:

- Assumption: Γ is not satisfiable
- $\rightsquigarrow \Gamma \models \bot$ (no model in which all $\varphi \in \Gamma$ are `true`)
- completeness $\rightsquigarrow \Gamma \vdash \bot$ valid
- $\rightsquigarrow$ this sequent has a proof in natural deduction

Compactness Theorem

Theorem

If all finite subsets of a set of predicat logic sentences Γ are satisfiable, then Γ is as well.

Proof:

- Assumption: Γ is not satisfiable
- $\rightsquigarrow \Gamma \models \bot$ (no model in which all $\varphi \in \Gamma$ are `true`)
- completeness $\rightsquigarrow \Gamma \vdash \bot$ valid
- $\rightsquigarrow$ this sequent has a proof in natural deduction
- proofs are finite $\rightsquigarrow$ only finitely many premises Δ from Γ are used

Compactness Theorem

Theorem

If all finite subsets of a set of predicat logic sentences Γ are satisfiable, then Γ is as well.

Proof:

- Assumption: Γ is not satisfiable
- $\rightsquigarrow \Gamma \models \bot$ (no model in which all $\varphi \in \Gamma$ are `true`)
- completeness $\rightsquigarrow \Gamma \vdash \bot$ valid
- $\rightsquigarrow$ this sequent has a proof in natural deduction
- proofs are finite $\rightsquigarrow$ only finitely many premises Δ from Γ are used
- $\rightsquigarrow \Delta \vdash \bot$

Compactness Theorem

Theorem

If all finite subsets of a set of predicat logic sentences Γ are satisfiable, then Γ is as well.

Proof:

- Assumption: Γ is not satisfiable
- $\rightsquigarrow \Gamma \models \perp$ (no model in which all $\varphi \in \Gamma$ are `true`)
- completeness $\rightsquigarrow \Gamma \vdash \perp$ valid
- $\rightsquigarrow$ this sequent has a proof in natural deduction
- proofs are finite $\rightsquigarrow$ only finitely many premises Δ from Γ are used
- $\rightsquigarrow \Delta \vdash \perp$
- $\rightsquigarrow \Delta \models \perp$ (soundness)

Compactness Theorem

Theorem

If all finite subsets of a set of predicat logic sentences Γ are satisfiable, then Γ is as well.

Proof:

- Assumption: Γ is not satisfiable
- $\rightsquigarrow \Gamma \models \bot$ (no model in which all $\varphi \in \Gamma$ are `true`)
- completeness $\rightsquigarrow \Gamma \vdash \bot$ valid
- $\rightsquigarrow$ this sequent has a proof in natural deduction
- proofs are finite $\rightsquigarrow$ only finitely many premises Δ from Γ are used
- $\rightsquigarrow \Delta \vdash \bot$
- $\rightsquigarrow \Delta \models \bot$ (soundness)
- not all finite subsets are consistent

Löwenheim-Skolem Theorem

Theorem

Let ψ be a predicate logic sentence such that for all $n \in \mathbb{N}$ there exists a model of ψ with at least n element. Then ψ has a model with infinitely many elements.

Löwenheim-Skolem Theorem

Theorem

Let ψ be a predicate logic sentence such that for all $n \in \mathbb{N}$ there exists a model of ψ with at least n element. Then ψ has a model with infinitely many elements.

Proof:

- $\varphi_n = \exists x_1 \ldots \exists x_n (\bigwedge_{1 \leq i < j \leq n} \neg(x_i = x_j))$ specifies the existence of at least n elements

Löwenheim-Skolem Theorem

Theorem

Let ψ be a predicate logic sentence such that for all $n \in \mathbb{N}$ there exists a model of ψ with at least n element. Then ψ has a model with infinitely many elements.

Proof:

- $\varphi_n = \exists x_1 \ldots \exists x_n (\bigwedge_{1 \leq i < j \leq n} \neg(x_i = x_j))$ specifies the existence of at least n elements
- $\Gamma := \{\psi\} \cup \{\varphi_n \mid n \in \mathbb{N}\}$, Δ finite subset of Γ

Löwenheim-Skolem Theorem

Theorem

Let ψ be a predicate logic sentence such that for all $n \in \mathbb{N}$ there exists a model of ψ with at least n element. Then ψ has a model with infinitely many elements.

Proof:

- $\varphi_n = \exists x_1 \ldots \exists x_n (\bigwedge_{1 \leq i < j \leq n} \neg(x_i = x_j))$ specifies the existence of at least n elements
- $\Gamma := \{\psi\} \cup \{\varphi_n | \, n \in \mathbb{N}\}$, Δ finite subset of Γ
- choose $k \in \mathbb{N}$ such that for all $n \leq k$, $\varphi_n \in \Delta$

Löwenheim-Skolem Theorem

Theorem

Let ψ be a predicate logic sentence such that for all $n \in \mathbb{N}$ there exists a model of ψ with at least n element. Then ψ has a model with infinitely many elements.

Proof:

- $\varphi_n = \exists x_1 \ldots \exists x_n (\bigwedge_{1 \le i < j \le n} \neg(x_i = x_j))$ specifies the existence of at least n elements
- $\Gamma := \{\psi\} \cup \{\varphi_n \mid n \in \mathbb{N}\}$, Δ finite subset of Γ
- choose $k \in \mathbb{N}$ such that for all $n \le k$, $\varphi_n \in \Delta$
- $\{\psi, \varphi_k\}$ satisfiable by assumption $\rightsquigarrow$ Δ satisfiable

Löwenheim-Skolem Theorem

Theorem

Let ψ be a predicate logic sentence such that for all $n \in \mathbb{N}$ there exists a model of ψ with at least n element. Then ψ has a model with infinitely many elements.

Proof:

- $\varphi_n = \exists x_1 \ldots \exists x_n (\bigwedge_{1 \leq i < j \leq n} \neg(x_i = x_j))$ specifies the existence of at least n elements
- $\Gamma := \{\psi\} \cup \{\varphi_n | \, n \in \mathbb{N}\}$, Δ finite subset of Γ
- choose $k \in \mathbb{N}$ such that for all $n \leq k$, $\varphi_n \in \Delta$
- $\{\psi, \varphi_k\}$ satisfiable by assumption $\rightsquigarrow$ Δ satisfiable
- compactness theorem $\rightsquigarrow$ Γ satisfiable by some model $\mathcal{M}$

Löwenheim-Skolem Theorem

Theorem

Let ψ be a predicate logic sentence such that for all $n \in \mathbb{N}$ there exists a model of ψ with at least n element. Then ψ has a model with infinitely many elements.

Proof:

- $\varphi_n = \exists x_1 \ldots \exists x_n (\bigwedge_{1 \leq i < j \leq n} \neg(x_i = x_j))$ specifies the existence of at least n elements
- $\Gamma := \{\psi\} \cup \{\varphi_n | \, n \in \mathbb{N}\}$, Δ finite subset of Γ
- choose $k \in \mathbb{N}$ such that for all $n \leq k$, $\varphi_n \in \Delta$
- $\{\psi, \varphi_k\}$ satisfiable by assumption $\rightsquigarrow$ Δ satisfiable
- compactness theorem $\rightsquigarrow$ Γ satisfiable by some model $\mathscr{M}$
- $\rightsquigarrow \mathscr{M} \models \psi$

Löwenheim-Skolem Theorem

Theorem

Let ψ be a predicate logic sentence such that for all $n \in \mathbb{N}$ there exists a model of ψ with at least n element. Then ψ has a model with infinitely many elements.

Proof:

- $\varphi_n = \exists x_1 \ldots \exists x_n (\bigwedge_{1 \leq i < j \leq n} \neg(x_i = x_j))$ specifies the existence of at least n elements
- $\Gamma := \{\psi\} \cup \{\varphi_n | \, n \in \mathbb{N}\}$, Δ finite subset of Γ
- choose $k \in \mathbb{N}$ such that for all $n \leq k$, $\varphi_n \in \Delta$
- $\{\psi, \varphi_k\}$ satisfiable by assumption $\rightsquigarrow$ Δ satisfiable
- compactness theorem $\rightsquigarrow$ Γ satisfiable by some model $\mathcal{M}$
- $\rightsquigarrow$ $\mathcal{M} \models \psi$
- $\mathcal{M}$ satisfies all φ_n $\rightsquigarrow$ is has to be infinite

Reachability Problem and Predicate Logic

Theorem

Reachability is not expressible in predicate logic.

Reachability Problem and Predicate Logic

Theorem

Reachability is not expressible in predicate logic.

Proof:

- Supposition: φ formula expressing the path-existence from u to v in G

Reachability Problem and Predicate Logic

Theorem

Reachability is not expressible in predicate logic.

Proof:

- Supposition: φ formula expressing the path-existence from u to v in G
- c, c' constants

Reachability Problem and Predicate Logic

Theorem

Reachability is not expressible in predicate logic.

Proof:

- Supposition: φ formula expressing the path-existence from u to v in G
- c, c' constants
- φ_n formula expressing the existence of a path from c to c' of length n:

Reachability Problem and Predicate Logic

Theorem

Reachability is not expressible in predicate logic.

Proof:

- Supposition: φ formula expressing the path-existence from u to v in G
- c, c' constants
- φ_n formula expressing the existence of a path from c to c' of length n:
  - $\varphi_0 := c = c'$

Reachability Problem and Predicate Logic

Theorem

Reachability is not expressible in predicate logic.

Proof:

- Supposition: φ formula expressing the path-existence from u to v in G
- c, c' constants
- φ_n formula expressing the existence of a path from c to c' of length n:
  - $\varphi_0 := c = c'$
  - $\varphi_1 := R(c, c')$

Reachability Problem and Predicate Logic

Theorem

Reachability is not expressible in predicate logic.

Proof:

- Supposition: φ formula expressing the path-existence from u to v in G
- c, c' constants
- φ_n formula expressing the existence of a path from c to c' of length n:
  - $\varphi_0 := c = c'$
  - $\varphi_1 := R(c, c')$
  - $\varphi_n := \exists x_1 \ldots \exists x_{n-1}(R(c, x_1) \wedge R(x_1, x_2) \wedge \cdots \wedge R(x_{n-1}, c'))$

Proof Cont

- $\Delta := \{\neg\varphi_i | \, i \in \mathbb{N}_0\} \cup \{\varphi[c/u, c'/v]\}$

Proof Cont

- $\Delta := \{\neg\varphi_i \mid i \in \mathbb{N}_0\} \cup \{\varphi[c/u, c'/v]\}$
- all elements of Δ are sentences

Proof Cont

- $\Delta := \{\neg\varphi_i | i \in \mathbb{N}_0\} \cup \{\varphi[c/u, c'/v]\}$
- all elements of Δ are sentences
- Δ unsatisfiable (conjunction says: no path of length i for all $i \in \mathbb{N}_0$)

Proof Cont

- $\Delta := \{\neg\varphi_i \,|\, i \in \mathbb{N}_0\} \cup \{\varphi[c/u, c'/v]\}$
- all elements of Δ are sentences
- Δ unsatisfiable (conjunction says: no path of length i for all $i \in \mathbb{N}_0$)
- $\varphi[c/u, c'/v]$ means: finite path from c to c'

Proof Cont

- $\Delta := \{\neg\varphi_i \,|\, i \in \mathbb{N}_0\} \cup \{\varphi[c/u, c'/v]\}$
- all elements of Δ are sentences
- Δ unsatisfiable (conjunction says: no path of length i for all $i \in \mathbb{N}_0$)
- $\varphi[c/u, c'/v]$ means: finite path from c to c'
- path of every finite length $\Rightarrow$ all finite subsets of Δ satisfiable

Proof Cont

- $\Delta := \{\neg\varphi_i \mid i \in \mathbb{N}_0\} \cup \{\varphi[c/u, c'/v]\}$
- all elements of Δ are sentences
- Δ unsatisfiable (conjunction says: no path of length i for all $i \in \mathbb{N}_0$)
- $\varphi[c/u, c'/v]$ means: finite path from c to c'
- path of every finite length $\Rightarrow$ all finite subsets of Δ satisfiable
- Compactness Theorem $\rightsquigarrow$ contradiction

Second-Order Logic - An Outlook

Limits of First-Order Predicate Logic

- description of first-order predicate logic:
    - propositional logic enriched by
    - functions (and constants), predicates
    - variables
    - quantifiers specifying only variables

Limits of First-Order Predicate Logic

- description of first-order predicate logic:
  - propositional logic enriched by
  - functions (and constants), predicates
  - variables
  - quantifiers specifying only variables
- reachability problem not expressible

Limits of First-Order Predicate Logic

- description of first-order predicate logic:
  - propositional logic enriched by
  - functions (and constants), predicates
  - variables
  - quantifiers specifying only variables
- reachability problem not expressible
- we don't want to develop a complete new language

Limits of First-Order Predicate Logic

- description of first-order predicate logic:
  - propositional logic enriched by
  - functions (and constants), predicates
  - variables
  - quantifiers specifying only variables
- reachability problem not expressible
- we don't want to develop a complete new language
- how can we extend the first-oder predicate logic?

Second-Order Logic

Definition

A formula φ is in existential second-order predicate logic

- φ has only elements of a first-order predicate logic formula
- φ has additionally constructs $\exists P \psi$ where P is a predicate symbol

Definition

A formula φ is in Universal Second-Order Logic iff $\neg\varphi$ is in Existential First Order Logic.

Second-Order Logic

Definition

A formula φ is in existential second-order predicate logic

- φ has only elements of a first-order predicate logic formula
- φ has additionally constructs $\exists P \psi$ where P is a predicate symbol

Definition

A formula φ is in Universal Second-Order Logic iff $\neg\varphi$ is in Existential First Order Logic.

In universal Second-Order Logic describes the reachability problem is describable.

Second-Order Predicate Logic

In the general Second-Order Predicate Logic both quantifiers quantifying predicates are allowed.

Second-Order Predicate Logic

In the general Second-Order Predicate Logic both quantifiers quantifying predicates are allowed.

- higher order logic: quantifying relations over relation over relation . . .
- keeping soundness and completness is hard (see Russel's Antimony)