		Elements of Mathematical Logic		Elements of Mathematical Logic	3	
		Symbols of propositional logic: atomic propositions p, q, r, \cdots , logical connectors \neg and \rightarrow , and parantheses ().		A valuation functions (truth assignments) A valuation v is a function defined for all propositional formulas, with values in {T,F} such that: -v(p) is defined for any atomic proposition p. $-v(\neg \alpha) = \begin{cases} T & \text{if } v(\alpha) = F \\ F & \text{if } v(\alpha) = T \end{cases}$ $-v(\alpha \rightarrow \beta) = \begin{cases} F & \text{if } v(\alpha) = T \text{si } v(\beta) = F \\ T & \text{otherwise} \end{cases}$		
Elements of Mathematical Logic November 10, 2005 - Propositional calculus - Predicate calculus - Decision procedures - Resolution theorem proving		Formulas of propositional logic: – any atomic proposition is a formula – if α is a formula, then ($\neg \alpha$) is a formula – if α and β are formulas, then ($\alpha \rightarrow \beta$) is	a formula.			
		Other known operators can be introduced as shorthands: $ - (\alpha \land \beta) \stackrel{\text{def}}{=} (\neg(\alpha \rightarrow (\neg \beta))) - (\alpha \lor \beta) \stackrel{\text{def}}{=} ((\neg \alpha) \rightarrow \beta) - (\alpha \leftrightarrow \beta) \stackrel{\text{def}}{=} ((\alpha \rightarrow \beta) \land (\beta \rightarrow \alpha)) Simplified notation: without redundant parantheses;$		An <i>interpretation</i> = a valuation for the atomic propositions of a formula An intrepretation <i>satisfies</i> a formula if the latter is evaluated to T (we say that the interpretation is a <i>model</i> for that formula). <i>valid</i> formula (<i>tautology</i>): true in all interpretations <i>satisfiable</i> formula: true in at least one interpretation		
Formal Verification. Lecture 6	Marius Minea	precedence order defined as: $\neg, \land, \lor, \rightarrow, \leftrightarrow;$ Formal Verification. Lecture 6	→ is right-associative Marius Minea	unsatisfiable formula (<i>contradiction</i>): false	in any interpretation Marius Minea	

Elements of Mathematical Logic	4	Elements of Mathematical Logic	5	Elements of Mathematical Logic Deduction	6						
Syntactic and semantic approach Semantic approach: based on logical implication (logical truth) $H \models \varphi$ A set of formulas H implies a formula φ if any truth function that satisfies H (i.e., all formulas in H) also satisfies φ .		Axioms and deduction rules Axion schemes for propositional logic: A1: $(\alpha \rightarrow (\beta \rightarrow \alpha))$ A2: $(((\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow (((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))))$ A3: $(((\neg\beta) \rightarrow (\neg \alpha)) \rightarrow (((\neg\beta) \rightarrow \alpha) \rightarrow \beta)))$ (called schemes (schemata) because axioms are obtained substituting particular formulas of propositional logic)		Let <i>H</i> be a set of formulas. We call <i>deduction</i> from <i>H</i> a sequence of formulas A_1, A_2, \dots, A_n , such that: 1. A_i is an axiom, or 2. A_i is a formula from <i>H</i> , or 3. A_i follows by MP from two previous sequence items A_j, A_k where $j < i, k < i$. We say that A_n follows from <i>H</i> (is deducible, is a consequence): $H \vdash A_n$ Example: we prove that $(\varphi \to \varphi)$							
						Syntactic approach: logical proof – based on syntactic manipulation of formulas:		We introduce a single deduction rule (<i>modus ponens, MP</i>): From the formulas φ and $\varphi \rightarrow \psi$ we can deduce ψ .		$(1) \varphi \to ((\varphi \to \varphi) \to \varphi))$ $(2) \varphi \to ((\varphi \to \varphi) \to \varphi)) \to ((\varphi \to (\varphi \to \varphi) \to (\varphi \to \varphi)))$ $(3) (\varphi \to (\varphi \to \varphi) \to (\varphi \to \varphi))$ $(4) \varphi \to (\varphi \to \varphi) \to (\varphi \to \varphi)$ $MP(\varphi \to \varphi) \to (\varphi \to \varphi)$	A1 A2
						Is a theorem provable from a set of axioms, using deducti	ion rules ?			(3) $(\varphi \rightarrow (\varphi \rightarrow \varphi) \rightarrow (\varphi \rightarrow \varphi))$ (4) $\varphi \rightarrow (\varphi \rightarrow \varphi)$ (5) $\varphi \rightarrow \varphi$	MP(1,2) A1 MP(3,4)
Formal Verification. Lecture 6	Marius Minea	Formal Verification. Lecture 6	Marius Minea	Formal Verification. Lecture 6	Marius Minea						

Elements of Mathematical Logic	7	Elements of Mathematical Logic Soundness and Completeness	8	Elements of Mathematical Logic	9	
Deduction theorem		establish the correspondence between the syntactic on deduction, and the semantic approach, based on	approach, based truth values.	First-order languages		
Let <i>H</i> be a set of formulas and α, β two formulas. Then $H \vdash \alpha \rightarrow \beta$ if and only if $H \cup \{\alpha\} \vdash \beta$.		Soundness: If <i>H</i> is a set of formulas, and α is a formula such that $H \vdash \alpha$, then $H \models \alpha$. (Any thorem in propositional logic is a tautology) Completeness: If <i>H</i> is a set of formulas, and α is a formula such that $H \models \alpha$, then $H \vdash \alpha$. (Any tautology is a theorem). Proof: based on the following notions and auxiliary results: A set of formulas <i>H</i> is <i>inconsistent</i> if there is a formula α such that $H \vdash \alpha$ and $H \vdash \neg \alpha$.		The symbols of a first-order language are: - parantheses () - logical connectors \neg and \rightarrow - the quantifier \forall (universal quantifier) - a set of identifiers v_0, v_1, \cdots for variables - a (possibly empty) set of symbols for constants - for any $n \ge 1$ a set of <i>n</i> -ary function symbols (of <i>n</i> arguments) - for any $n \ge 1$ a set of <i>n</i> -ary predicates (relations)		
 used as additional inference rule to simplify proofs Other corollaries: if H ⊢ α and H ⊢ α → β, then H ⊢ β if G ⊆ H and G ⊢ α, then H ⊢ α if H ⊢ G and G ⊢ α, then H ⊢ α 						
		Any consistent set of formulas can be extended to sistent set (adding any other formula makes it incon	a <i>maximally con-</i> sistent)	eirst-order languages with equality: Contain = as special addition to the above.		
		A set of formulas is <i>consistent</i> if and only if it is <i>satisfiable</i> .				
Formal Verification. Lecture 6	Marius Minea	Formal Verification. Lecture 6	Marius Minea	Formal Verification. Lecture 6	Marius Minea	

Elements of Mathematical Logic	10	Elements of Mathematical Logic Interpretations and valuations	11	Elements of Mathematical Logic 12		
First-order terms and formulas		An <i>interpretation (structure)</i> I for the predicate language \mathcal{L} consists of:		Define: variable x can be substituted with term t in $\forall y\varphi$ if:		
Terms of a first-order language (defined by structural induction) - any variable symbol v_n - any constant symbol c - $f(t_1, \dots, t_n)$, if f is an n -ary function symbol and t_1, \dots, t_n are te	erms	- a nonempty set U called the <i>universe</i> or the <i>domain</i> of I (the set of values which the variables can take) - for any constant symbol c, a value $c_I \in U$ - for any <i>n</i> -ary function symbol f, a function $f: U^n \to U$ - for any <i>n</i> -ary predicate symbol P, a subset $P_I \subseteq U^n$.		- <i>x</i> does not appear free in φ or - <i>y</i> does not appear in <i>t</i> and <i>x</i> can be substituted with <i>t</i> in φ A1 : $(\alpha \to (\beta \to \alpha))$ A2 : $((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)))$ A3 : $(((\neg \beta) \to (\neg \gamma)) \to (((\neg \beta) \to \alpha) \to \beta))$		
(Well-formed) Formulas of a first-order language: - $P(t_1, \dots, t_n)$, where P is an n -ary predicate and t_1, \dots, t_n are term - $t_1 = t_2$, where t_1 and t_2 are terms (for languages with equality) - $\neg \alpha$, where α is a formula - $\alpha \rightarrow \beta$, where α, β are formulas - $\forall v_n \varphi$ where v_n is a variable and φ is a formula	ms	Let <i>I</i> be an interpretation with universe <i>U</i> for <i>L</i> , and let <i>V</i> be the set of all variable symbols from <i>L</i> . A <i>valuation</i> is a function $s: V \to U$. Extending the valuation <i>s</i> to terms and formulas we obtain a true function (valuation) for all formulas in <i>L</i> . We write $I \models s(\varphi)$ or $I \models \varphi$ if the valuation <i>s</i> evaluates formula φ to true in the interpretation <i>I</i> . Define: $I \models s(\forall x \varphi)$ if $I \models s_{x \leftarrow d}(\varphi)$ for any $d \in U$, where $s_{x \leftarrow d}$ is the valuation $s_{x \leftarrow d}(v) = \begin{cases} d & \text{if variable } v \text{ is } x \\ s(v) & \text{ for any other variable } v \end{cases}$ Denote $I \models \varphi$ (<i>I</i> is a <i>model</i> for φ) if $I \models s(\varphi)$ for any valuation <i>s</i> .	set uth [s] I.	A4: $(\langle \forall x (\alpha \rightarrow \beta) \rightarrow (\forall x \alpha \rightarrow \forall x \beta))$ A5: $(\langle \forall x (\alpha \rightarrow \alpha) \rightarrow (\forall x \alpha \rightarrow \forall x \beta))$ A5: $(\langle x \alpha \rightarrow \alpha \alpha [x \leftarrow t])$, if <i>x</i> can be substituted with <i>t</i> in α A6: $(\alpha \rightarrow \forall x \alpha)$ if <i>x</i> does not appear free in α For equality, we also add A7: $x = x$ A8: $x = y \rightarrow \alpha = \beta$ where β is obtained from α by replacing arbitrarily many occurrences of <i>x</i> with <i>y</i> .		

Formal Verification. Lecture 6

Marius Minea

Formal Verification. Lecture 6 Marius Minea

Formal Verification. Lecture 6

Marius Minea

Sourcess and completeness Satisfiability. Applications Data Data Data Data Let <i>H</i> be a set of formulas and $\varphi = formula. We say that H implies f = \varphi. Exemute whether a propositional formulas is statistable. Context:::::::::::::::::::::::::::::::::::$	15		
Soundness and completenessProblem: Deturmine whether a propositional formula is statisfiable.Let <i>H</i> be a set of formulas and φ formula. We say that <i>H</i> implies φ Context:: senerally, complex formulas with hundreds or thousands of the prime formula is statisfiable.Function Satisfiable (list diaces containing L) (list propositional log/).For any hypothesis set <i>H</i> , and any formula φ , <i>H</i> φ iff <i>H</i> $ =\varphi$, <i>T</i> . the problem inget set <i>H</i> . $H = \varphi$ iff any diaces or pure list φ if and the prime in the prime information of the prime information of the prime information.Intercent proposition (control to set of the prime information is all diaces containing L) (list proposition) formation $H = \varphi$ if $H = \varphi$ is underdable in general.Intercent proposition (control to set of the prime information is all diaces or pure list $A = A = A = A = A = A = A = A = A = A $	hm		
Let <i>H</i> be a set of formulas and <i>q</i> a formula. We say that <i>H</i> implies <i>q</i> . (<i>H</i> = <i>y</i>) if <i>r</i> any intervation <i>I</i> , <i>H</i> = <i>H</i> implies <i>I</i> = <i>p</i> . First-order predicts calculus is sound and complete (the propositional logic). The random displexess above is different from the notion for any inporteets set <i>H</i> , and any formula <i>g</i> , <i>H</i> = <i>g</i> iff <i>H</i> = <i>p</i> . The determining the say values from mathematics the question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question whether <i>H</i> + <i>q</i> is undecidable in general. The question mathematics for system with first in the proving results from mathematics for system with first in the question method. The question function (question the proving results from mathematics for system with first in the question method. The question function (question the proving results from mathematics for system with first in the question method. The question function (question the proving results from mathematics for system with	;		
First-order predicate calculus is sound and complete (ine propositional log(). For any hypothesis set H, and any formula φ_i $H = \varphi$ iff $H = \varphi_i$. Note: The notion of completeness above is different from the notion of completeness is sufficient. The duestion whether $H + \varphi$ is undecidable in general. Humen diversation completeness is undecident for decident decident models in products based on the Davis-Puttam algorithm Terminology: und clause: composed of a single literal pure literal: appears only positively (or only negated) humen diversation. For priority registration. For priority registration completeness above is different from the notion of completeness different from the notion for the duestion whether $H + \varphi$ is undecidable in general. Humen diversation: completeness different from the notion of completeness different from the notion for station writeria. For priority registration (completentian) for priority registration (completentian) for station writeria. For station writeria. For priority registration (completentian) for station writeria. For station writeria. For priority registration (completentian) for station writeria. For priority registration (completentian) for station writeria. For priority registration (completentian) for station writeria. For priority registration (completeness different direction) for station writeria. For priority registration (completentian) for station writeria. For priority registration (completeness different direction) for station writeria. For pri			
For any hypothesis set H , and any formula q , $H \models q$ iff $H \models q$. Answer the notion of completeness above is different from the notion of completenes in	eliminate all clauses containing L eliminate ¬ L from all clauses if S is empty return TRUE elsif S contains the empty clause return FALSE until no more changes choose a literal L from S for decomposition (true/false)		
Note: The notion of completeness above is different from the notion of completeness above is different from the notion of completeness above which asks under the a set of axioms is sufficient for completeness above is sufficient for the general. The question whether $H \vdash \varphi$ is undecidable in general. Former verification (especially programs) Great variety: - for proving results from matematics - for proving results from			
$ \begin{array}{c} finite decision processes based on the Davie-Putnam algorithm TRUE decision the Davie-Putnam algorithm True decision processes based on the Davie-Putnam algorithm True decision the Davie Putnam algorithm True decisi$			
The decides influence of the point decides in fight data. Terminology: unit classes composed of a single literal pure literal: appears only positively (or only negated) the median point decides in products to the compositively (or only negated) the median decides in the median decides in products to the compositively (or only negated) the resolution method. Clausal form for proving results from mathematics for system verification (especially programs) Generally, implemented for higher-order logics a low types described by means of predicates a low types described by means of predicates by $Z[r(z) \lor \exists y(D(z, y)) \land Z[r(z), y) \land Z$			
Format WellGolio. Lecture 6Marias MariasFormat WellGolio. Lecture 6Marias MariasFormat WellGolio. Lecture 6Elements of Mathematical Logic16Elements of Mathematical Logic17Elements of Mathematical LogicTheorem proversTheorem proversthe resolution method. Clausal form17Elements of Mathematical LogicGreat Varieby:or porving results from mathematicsTheorem proversthe resolution method. Clausal form18Transforming to clausal form (nordinal formation in a sequence of 8 steps)Generaliv, implemented for higher-order logicsNary formula without free variables in predicate calculus can be written in clausal form in a sequence of 8 steps)Nary formula without free variables in predicate calculus can be written in clausal form in a sequence of 8 steps)Nary formula without free variables in predicate calculus can be written in clausal form in a sequence of 8 steps)Nary formula writhout free variables in predicate calculus can be written in clausal form in a sequence of 8 steps)Nary formula writhout free variables in predicate calculus can be written in clausal form in a sequence of 8 steps)Nary for (Clause) / - Var(Clause) / - Var(Claus	else return FALSE		
Elements of Mathematical Logic 27 Elements of Mathematical Logic 27 Transforming to clausal form (cont Theorem provers $\frac{1}{1} \frac{1}{1} \frac{1}$	Marius Minea		
Theorem proversTransforming to clausal form (controlTheorem proversthe resolution method. Clausal formGreat variety:- for proving results from mathematics- for proving results from mathematics- for system verification (especially programs)Generally, implemented for higher-order logics- allow types described by means of predicates- have inductive capabilitiesBasic approaches to proving:- forward chaining (derive theorems getting closer to the goal)- orbackwards chaining (generate intermediate conclusions for the given goal)- application of inference rules: controlled by <i>tactics</i> - application of inference rules: controlled by <i>tactics</i> - application of inference rules: controlled by tactics- application of inference rules: controlled by tactics	18		
Theorem proversthe resolution method. Clausal formthe resolution method. Clausal formthe resolution method. Clausal formthe resolution method. Clausal formGreat variety:- for proving results from mathematics- for proving results from mathematics- for system verification (especially programs)Generally, implemented for higher-order logics- allow types described by means of predicates- allow types described by means of predicates- have inductive capabilities- forward chaining (derive theorems getting closer to the goal)- or backwards chaining (generate intermediate conclusions for the given goal)- application of inference rules: controlled by <i>tactics</i> - application of inference rules: controlled by <i>tactics</i> the resolution method. Clausal form(4) Eliminate existential quantifiers (skolemize)For $\exists y$ within a quantifier $\forall x$, create a Skolem function in clausal form in a sequence of 8 stepsExample: start with $\forall [-P(x) \to \exists y(D(x, y) \land \neg (E(f(x), y) \lor E(x, y))] \land \neg \forall xP(x)$ (1) Eliminate all connectors except \land, \lor, \neg (2) Translate all negation inwards until they reach predicates: $\forall [P(x) \lor [Q(x, g(x)) \land \neg E(f(x), g(x)) \land \neg E(x, g(x))]) \land \neg E(x, g(x))) \land P(x) \lor \neg E(x, g(x))) \land P(x) \lor \neg E(x, g(x))) \land P(x) \lor P(x) \land P(x$.)		
Great variety:Any formula without free variables in predicate calculus can be written in clausal form in a sequence of 8 steps(F) $\exists w$ within a quantifier $\forall x$, create a $\delta kolem$ function (the value of y depends in general on the value of x) for proving results from mathematicsExample: start with $\forall x[\neg P(x) \rightarrow \exists y(D(x, y) \land \neg (E(f(x), y) \lor E(x, y))] \land \neg \forall xP(x)$ $\forall x[P(x) \lor (D(x, g(x)) \land \neg E(f(x), g(x)) \land \neg E(x, g(x)))] \land$ - allow types described by means of predicates(1) Eliminate all connectors except \land, \lor, \neg :(5) Bring to prenex normal form (all \forall quantifiers in fr- have inductive capabilities $\forall x[\neg \neg P(x) \lor \exists y(D(x, y) \land \neg (E(f(x), y) \lor E(x, y)))] \land \neg \forall xP(x)$ (5) Bring to prenex normal form (all \forall quantifiers in fr- forward chaining (derive theorems getting closer to the goal)(2) Translate all negation inwards until they reach predicates:(6) Eliminate prefix with universal quantifiers- or backwards chaining (generate intermediate conclusions for the given goal)(3) Rename variables, with unique name for each quantifier:(7) convert to conjunctive normal form $(P(x) \lor (D(x, g(x))) \land P(x) \lor (P(x) \lor (P($	(1) Eliminate evistential quantifiers (skolemize)		
- for system verification (especially programs)Example: start withOtherwise, choose a new Skolem constant.Generally, implemented for higher-order logics $\forall x [\neg P(x) \rightarrow \exists y(D(x, y) \land \neg (E(f(x), y) \lor E(x, y))] \land \neg \forall xP(x)$ $\forall x [P(x) \lor (D(x, g(x)) \land \neg E(f(x, g(x)) \land \neg E(x, g(x)))] \land$ - allow types described by means of predicates(1) Eliminate all connectors except \land, \lor, \neg :(5) Bring to prenex normal form (all \forall quantifiers in fr- have inductive capabilities $\forall x [\neg \neg P(x) \lor \exists y(D(x, y) \land \neg (E(f(x), y) \lor E(x, y)))] \land \neg \forall xP(x)$ $\forall x [(P(x) \lor (D(x, g(x)) \land \neg E(f(x), g(x)) \land \neg E(x, g(x)))] \land$ Basic approaches to proving:(2) Translate all negation inwards until they reach predicates:(6) Eliminate prefix with universal quantifiers- or backwards chaining (generate intermediate conclusions for the given goal)(3) Rename variables, with unique name for each quantifier:(7) convert to conjunctive normal form- application of inference rules: controlled by <i>tactics</i> $\forall x [P(x) \lor \exists y(D(x, y) \land \neg E(f(x), y) \land \neg E(x, y)]) \land \exists z \neg P(z)$ (8) Eliminate \land and write disjunctions as separate clain	For $\exists y$ within a quantifier $\forall x$, create a <i>Skolem function</i> $y = g(x)$ (the value of y depends in general on the value of x). Otherwise, choose a new <i>Skolem constant</i> . $\forall x[P(x) \lor (D(x,g(x)) \land \neg E(f(x),g(x)) \land \neg E(x,g(x)))] \land \neg P(a)$ (5) Bring to prenex normal form (all \forall quantifiers in front) $\forall x([P(x) \lor (D(x,g(x)) \land \neg E(f(x),g(x)) \land \neg E(x,g(x)))] \land \neg P(a))$ (6) Eliminate prefix with universal quantifiers $[P(x) \lor (D(x,g(x)) \land \neg E(f(x),g(x)) \land \neg E(x,g(x)))] \land \neg P(a)$		
- allow types described by means of predicates(1) Eliminate all connectors except \land, \lor, \neg :(5) Bring to prenex normal form (all \forall quantifiers in fr $\forall x[\neg \neg P(x) \lor \exists y(D(x,y) \land \neg (E(f(x),y) \lor E(x,y)))] \land \neg \forall xP(x)$ - have inductive capabilities(1) Eliminate all connectors except \land, \lor, \neg :(5) Bring to prenex normal form (all \forall quantifiers in fr $\forall x[P(x) \lor \exists y(D(x,y) \land \neg (E(f(x),y) \lor E(x,y)))] \land \neg \forall xP(x)$ Basic approaches to proving: - forward chaining (derive theorems getting closer to the goal) - or backwards chaining (generate intermediate conclusions for the given goal) - application of inference rules: controlled by <i>tactics</i> (2) Translate all negation inwards until they reach predicates: $\forall x[P(x) \lor \exists y(D(x,y) \land \neg E(f(x),y) \land \neg E(x,y))] \land \exists x \neg P(x)$ (6) Eliminate prefix with universal quantifiers $[P(x) \lor (D(x,g(x)) \land \neg E(f(x),g(x)) \land \neg E(x,g(x)))] \land \neg E(x,g(x)))$ - application of inference rules: controlled by <i>tactics</i> (3) Rename variables, with unique name for each quantifier: $\forall x[P(x) \lor \exists y(D(x,y) \land \neg E(f(x),y) \land \neg E(x,y))] \land \exists z \neg P(z)$ (7) convert to conjunctive normal form $(P(x) \lor D(x,g(x))) \land (P(x) \lor E(f(x),g(x))) \land (P(x) \lor E(f(x),g($			
Basic approaches to proving:(2) Translate all negation inwards until they reach predicates:(6) Eliminate prefix with universal quantifiers- forward chaining (derive theorems getting closer to the goal) $\forall x [P(x) \lor \exists y(D(x, y) \land \neg E(f(x), y)) \land \neg E(x, y)]) \land \exists x \neg P(x)$ $[P(x) \lor (D(x, g(x)) \land \neg E(f(x), g(x)) \land \neg E(x, g(x)))] \land \neg E(x, g(x))) \land \neg E(x, g(x)) \land \neg E(x, g(x))) \land \neg E(x, g(x))) \land \neg E(x, g(x)) \land \neg E(x, g(x))) \land \neg E(x, g(x)) \land \neg E(x, g(x))) \land \neg E$			
(3) Rename variables, with unique name for each quantifier: - application of inference rules: controlled by <i>tactics</i> (3) Rename variables, with unique name for each quantifier: - application of inference rules: controlled by <i>tactics</i> (3) Rename variables, with unique name for each quantifier: $\forall x[P(x) \lor \exists y(D(x,y) \land \neg E(f(x),y)) \land \exists z \neg P(z)$ (4) Eliminate \land and write disjunctions as separate claiming to the second s			
	'(a)		
Formal Verification. Lecture 6 Marius Minea Formal Verification. Lecture 6 Marius Minea Formal Verification. Lecture 6	P(a) (a) (a) (a) (a) (a) (a) (a)		

Elements of Mathematical Logic

19

Resolution principle

Consider two clauses, written as sets of disjunctive terms. Consider first the case of propositional formulas Call resolvent of two clauses C_1 , C_2 with respect to literal l (for which $l \in C_1$, $(\neg l) \in C_2$): $\operatorname{rez}_l(C_1, C_2) = (C_1 \setminus \{l\}) \cup (C_2 \setminus \{\neg l\})$. Example: $\operatorname{rez}_p(\{p, q, r\}, \{\neg p, s\}) = \{q, r, s\}$. $(p \lor q \lor r) \land (\neg p \lor s) \rightarrow (q \lor r \lor s)$ Proposition: C_1, C_2 is satisfiable iff $\operatorname{rez}_l(C_1, C_2)$ is satisfiable.

We determine the satisfiability of a formula in conjunctive normal form by repeatedly adding resolvents, and trying to derive the empty clause.

Term unification

For predicate calculus, proceed likewise; but instead of a literal l and its negation, $\neg l$ consider the negation $\neg l'$ of a literal l' that can be *unified* with it.

Two literals can be unified if there is a term substitution for the occurring variables that makes the literals identical.

Example: P(a, x, y) and P(z, f(z), b) can be unified to P(a, f(a), b).

To unify two literals: successively unify terms on same argument position (for functions and predicates) until the same literal is obtained, or unification becomes impossible (symbols of different functions, or unification of x with a term containing x).

Formal Verification. Lecture 6

Marius Minea

Formal Verification. Lecture 6

Elements of Mathematical Logic

Marius Minea