Mihaiela-Ana Lupea

Andreea-Diana Mihiş

**A Computational Approach to**

Classical Logics and Circuits

Editura RISOPRINT Cluj-Napoca • 2015

**© 2015 RISOPRINT** Toate drepturile rezervate autorilor & Editurii Risoprint

*Editura RISOPRINT este recunoscută de C.N.C*.S.

*(Consiliul Național al Cercetării ştiinţifice).* www*.risoprint.ro*

**www*.cn****cs-uefiscdi.ro*

Opiniilc cxprimate în accastă cartc aparțin autorului şi nu reprezintă punctul de vedcrc

al Editurii Risoprint. Autorul işi asumă întreaga responsabilitate pentru forma şi conținutul cărţii şi se obligă să respecte toate legile privind drepturile de autor.

Toate drepturile rezervate. Tipărit în România. Nicio parte din această lucrare nu poate fi reprodusă sub nicio formă, prin niciun mijloc mecanic sau electronic, sau stocată într-o bază de date fără acordul

prealabil, în scris, al autorului. All rights reserved. Printed in Romania. No part of this publication may be reproduced or distributed in any form or by any means, or stored

in a data base or retrieval system, without the

prior written permission of the author.

Descrierea CIP a Biblioteci**i Naţionale a României LUPEA, MIHAIELA-ANA**

**A computational approach to classical logics and circuits** / Mihaiela-Ana Lupea, Andreea-Diana Mihiş. - Cluj-Napoca : Risoprint, 2015

Bibliogr.

Index ISBN 978*-9*73-53-1636-5

I. Mihiş, Andreea-Diana

004.312(*075*.8) 004.312:62.001.63(075.8)

Director editură: GHEORGHE POP Consilier editorial: VOICHIȚA-MARIA CLINCI

*Referenţi ştiinţifici: Prof. univ. dr. DOINA TĂTAR Prof. univ. dr. GABRIELA CZIBULA*

Tiparul executat la:

**S.C. ROPRINT®S.R.L.** 400 188 Cluj-Napoca. Str. Cernavodă nr. 5-9 Tel./Fax: 0264-590651 • roprint@roprint.ro

**PREFACE**

The purpose of this book is to present the logical foundations of computer science: classical logics and logic circuits.

Fundamental concepts and results of classical logics are introduced in a formal style and in an explicitly computational way. Applications to automated theorem proving in propositional logic and first-order logic are presented. The studied proof methods are: the resolution method, the semantic tableaux method and the sequent/anti-sequent calculi. The formalization of mathematical reasoning and human reasoning using propositional logic and first-order logic is also an objective of this paper.

The design of logic circuits, based on Boolean functions is an important part of this book.

The paper combines the theoretical presentation of classical logics and logic circuits with numerous examples explained and a rich base of proposed exercises.

**Chapter 1** is dedicated to *propositional logic*. The semantic issues discussed are: truth tables, validity, consistency, inconsistency, logical equivalence, logical consequence, normal forms. From a syntactic perspective propositional logic is introduced as an axiomatic (deductive) system, with the purpose of reasoning modeling

*First-order predicate logic* is the topic of chapter *2* of the paper. A Hilbert axiomatic system is used to present predicate calculus in a syntactic approach. The semantics of predicate logic is introduced in order to provide a meaning in terms of the modeled universe for each formula from the language. Normal forms, substitutions and unifiers used in predicative resolution are discussed. The method based on Herbrand's theorem is applied with the aim of reducing the problem of inconsistency of a set of predicate formulas to the problem of inconsistency in propositional logic.

**Chapter** 3 treats the s*emantic tableaux method,* a refutation proof method. The formulas are decomposed in order to determine their models. The classical approach through graphic representation using a binary tree is suggestive, but hard to implement. The new approach, using the TP function allows an easy and efficient implementation of this proof method. An automated theorem prover based on this method, including the corresponding algorithms and implementation details are presented.

**Chapter** 4 presents two complementary axiomatic systems*: the sequent* and *anti-sequent calculi. A*s syntactic and direct proof methods, they are used to check

3

the validity/derivability and non-validity/non-derivability in propositional logic and first-order logic.

The topic of **chapter** 5 is *resolution,* a syntactic and refutation proof method, very efficient and easily to implement. Resolution is introduced as an axiomatic (formal) system and as a procedure. In order to increase the efficiency of the resolution process, the strategies (deletion, set-of-support, unit preference, level saturation, linear) and the refinements (lock, semantic, linear-ordered) of resolution are studied.

**In chapter 6** examples of reasoning modeling in mathematics and daily life, using propositional logic and predicate logic, are presented. An approach of the program verification task using resolution in first-order logic is also described.

**Chapter** 7 introduces the basic concepts of *Boolean algebras* and *Boolean function*s and presents three *simplification methods* of Boolean functions. Veitch Karnaugh diagrams graphical method, Quine's analytical method and Moisil's algebraic method are described and applied to numerous examples.

Chapter 8 deals with *logic circuits.* Defined on Boolean algebras, Boolean functions are very useful in the design of logic circuits. Some of the combinational circuits used in the hardware of computers (comparator, adder, subtractor, encoder, decoder) are presented.

By its content, this book is usefull to all those interested in classical logics and logic circuits, fundamental areas of computer science. Professionals in computer science are offered a theoretical basis in the applicative direction of building automated proof systems used in mathematics, software engineering, intelligent agents, robotics, natural language, artificial vision.

We wish to acknowledge our deep appreciation to Prof. Dr. Doina Tătar for many valuable scientific discussions and guidance during all the years of study and research in the field of classical logics. Special thanks for all her constructive comments made during the preparation of this paper.

Cluj-Napoca

2015

Mihaiela-Ana Lupea

Andreea-Diana Mihiş

A Computational Approach to Classical Logics and Circuits

**CONTENTS**

..........

.................

.....*..*.

...........

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

1.5.

Formal

...............

............. .......

**Introduction** ............ **1. Propositional log**ic ....

..*..*........ 1.1. Syntax ......... 1.2. Semantics of propositional logic... 1.3. Logical equivalences .......*...*.*.*. 1.4. Normal forms in propositional logic .........*..*

............. Formal (axiomatic) system of propositional logic ......

........... 1.6. The theorem of deduction and its reverse .......... 1.7. Properties of propositional logic ........... 1.8. Decision problems and proof methods........*...*

1.9. Exercises ..........*.*.. 2. First-order logic .......

............. 35 2.1. The axiomatic (formal) system of first-order logic.. 2.2. Transformation of natural language sentences into predicate formulas... 41 2.3. The semantics of first-order (predicate) logic.... 2.4. Logical equivalences in predicate logic .... 2.5. Normal forms in first-order logic........... 2.6. Substitutions and unification... *2.7.* Herbrand – based procedure............ 2.8. Exercises ........*.....*..

.............61 **3. Semantic tableaux method** ...........

............. 67 3.1. A Classical Approach to the Semantic Tableaux Method............*.*.......*.*.*.*67 3.2. A New Approach to the Semantic Tableaux Method

..............81 3.3. An Automated Theorem Prover for Propositional Logic, based on the

Semantic Tableaux Method....

....*..*........ 86 3.4. Considerations for implementation ........

........... 3.5. Exercises .. **4. Sequent and Anti-Sequent calcul**I...

.......*..*.. 4.1. The sequent calculus .................

.........*...*..*97* 4.2. The anti-sequent calculus ......*..*

...............106 4.3. Exercises .........

115 **5. Resolution proof method..**

.......... **118** 5.1. Resolution method for propositional logic.

............ 118 5.2 Strategies for propositional resolution ..........

....*..*....... 122 5.3 Lock resolution ........*.*..

............126 5.4 Linear resolution

........ 131

..........

*..89*

**!**

**!**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

1.........

.........

Contents

.

.

.

.

.

.

2

.000

O

. O

OO00.00

.........*..*..

193

5.5 Resolution in first-order (predicate) logic ......

.............136 5.6 Semantic resolution......

.............143 *5.7* Exercises ......*..*.....*..*...*...*

151 ***6*. Reasoning modeling and program verification.**.....

...*..*....... 155 6.1. Common-sense reasoning modeling using propositional logic.............. 155 6.2. Reasoning modeling using predicate logic .......

.........*..*..*.*.......... 162 6.3. Program verification using predicate logic .........

............. 178 *7.* **Boolean algebras and Boolean functions** ............

............ 184 7.1. Boolean algebras .........*...*

...........184 *7.*2. Boolean functions ......*..*... ..............*.*

...............186 7.3. Canonical forms of Boolean functions...........

188 7.4. Simplification of Boolean functions ..

............. *7.5*. Veitch-Karnaugh diagrams method ..

196 7*.*6. Quine-Mc'Clusky's method...

..............208 *7.7.* Moisil's simplification method .........*...*

.............215 7.8. Application ..........*..*

............217 *7.9*. DCF versus CCF in simplification ....

..........218 7.10. Exercises ......

.............221 **8. Logic circuits** .............*..*

............ *224* 8.1. Basic concepts.........

...*....*.....224 8.2. Examples of useful combinational circuits .....

.............233 8.3. Exercises ....................

.............244 **Index** ........*.*..*...*..*...*...

............. *2*48 **Bibliography**.........

............. 252

.............

............ ........*..*...

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

.

.

.

.

.

.

.

.

.

.

**+**

**+**

**+**

A Computational Approach to Classical Logics and Circuits

**INTRODUCTION**

The science that studies the principles of reasoning and valid inference is called *logic*. Philosophical logic and mathematical logic are commonly associated with *deductive reasoning*, which determines whether the truth of a conclusion can be determined for an inference rule, based solely on the truth of the set of premises. In antiquity, earlier philosophers introduced *propositional logic*, but only in the 3rd century B.C. was it developed into a formal logic by the Stoics. In 322 B.C., Aristotle proposed the s*yllogism la*w that became the dominant model of correct argumentation in philosophy for more than two thousand years. This law was used later as an inference rule in symbolic logic. The philosopher Gottfried Leibniz (1646 -1716) is considered to be the founder of *symbolic logic*. His research aimed at finding a general decision procedure, but his important results were unknown to the larger logical community. Completely independent of Leibniz, two well-known mathematicians and logicians reachieved in their works the same results as Leibniz:

• George Boole: *Mathematical Analysis of Logic, giving an algebraic approach*

*to Aristotelian logic* (1847) and *The La*w*s of Thought* (1854). He applied methods from the field of symbolic algebra to logic. Augustus De Morgan: *Formal Logic (*1847). His major contribution was to the development of the theory of relations and the rise of modern symbolic, or

mathematical, logic. In 1879 Gottlob Frege proposed a formal system called *predicate o*r *first-order logic (*FOL), introducing quantified statements and proposing the notion of a *proof* in terms that are still accepted today. Propositional logic and predicate logic are called *classical logic*s and they model a valid type of deductive reasoning. The decision problems in these logics are: to check the validity of a statement and to check if a *conclusion* is derivable (inferable) from a set of statements *(axioms* and *hypothese*s). The inferential process is a monotonic one, meaning that once a conclusion is deduced (derived) from a set of hypotheses (premises), new premises will not invalidate it. Once the formal framework (syntax and semantics) of logic was developed, the efforts of researchers focused on finding efficient *proof methods* that would solve the decision problems and implement them. The proof methods are classified into *direct methods* and *refutation methods.* The direct methods build the proof of the conclusion directly from the axioms and the hypotheses using the inference rules. The refutation methods are based on the “reductio ad absurdum' principle and the idea is to show that the negation of the conclusion together with the hypotheses and the axioms lead to a contradiction.

Introduction

The invention of *truth-table,* an important tool in propositional logic, is of controversial attribution because many mathematicians-logicians (Frege, Russell, Philo, Boole, Peirce, Schröder, Łukasiewicz, Whitehead, Venn, Lewis) had ideas related to truth-tables. However, the well-known tabular structure is credited to either Ludwig Wittgenstein (1922) or Emil Post (1921). In 1936 G.K.Gentzen introduce*d natural deduction,* a direct and syntactic method, based on an axiomatic system that formalizes the deductive (inferential) process. As an improvement of this method, two complementary axiomatic systems *sequent* and *anti-sequent calculi* were developed and employed to prove the validity/derivability and non-validity*/*non-derivability respectively in classical logics. In 1965 J.A.Robinson proposed a refutation and syntactic proof method, simple and easy to implement, called *resolution.* Later, this method was refined (semantic resolution, linear resolution, lock resolution) in order to increase the efficiency. The *semantic tableaux method* introduced by R. Smullyan in 1968 is based on semantic considerations. The proof by contradiction of the conclusion consists of decomposing the conjunction of the hypotheses and the negation of the conclusion, searching for models and showing that they do not exist. *Automated Theorem Proving (ATP)* deals with the development of computer programs which show that some statement (the *conjecture*) is *a logical consequence* of a set of statements (the *axioms* and the *hypothese*s). These automated systems were used in a lot of domains such as mathematics *(EQP, Otter, Geometry Expert)*, software generation *(KIDS, AMPHIO*N), software verification *(KIV, PV*S), hardware verification *(ACL2, HOL, ANALYTICA). A*s potential fields we enumerate biology, social science, medicine, commerce. Also developed were dedicated (educational) automated theorem provers:

based on semantic tableaux method: 3TAP, PTAP, leanTAP, Cassandra;

• based on resolution method: OTTER, PCPROVE, AMPHION, Jape;

based on semantic trees + Herbrand theorem: HERBY;

• based on model elimination calulus: SETHEO; In 1938 Claude Shannon proved that a two-valued binary Boolean algebra can describe the operations of two-valued electrical switching circuits. Propositional logic is used to minimize the number of gates in a circuit, and to show the equivalence of *combinational circuits.* The Romanian mathematician Grigore Moisil invented the three-stable circuits and had important contributions in the fields of algebraic logic and differential equations. Moisil used propositional logic to minimize Boolean functions. In modern times *Boolean algebras* and *Boolean functions* are indispensable in the design of computer chips and digital circuits. A computational approach, providing methods and algorithms to solve different tasks in classical logics and circuits is the specific purpose of this book.

A Computational Approach to Classical Logics and Circuits

**1. PROPOSITIONAL LOGIC**

In this chapter the propositional logic is presented, first from a semantic perspective and then as an axiomatic system. This formalism is useful in reasoning modeling and the design of logic circuits. We used as references the following papers (1, 2, 9, 16, 19, 23, *37,* 38, 40, 42, 43, 47, 48, 59, 60, 61, 62, 63, 64).

***1.1. Syntax***

The s*ynta*x introduces the entities used to define well-formed propositional formulas.

• E*p = Var \_ propos U{F,T}uConnectives* U{(,)} is the v*ocabulary;*

*Var \_ propos = {p,q,7,*...} is a finite *set of propositional variables; Connectives =* {-(negation), 1 (conjunction), *v (disjunction),*

*implication), 4(equivalence*)*}:* The negation is a unary connective and all the others are binary connectives. The decreasing order of precedence of the connectives is as follows:

7,1,V, , .

*Fp* is the *set of well-formed formulas* built using the propositional variables, the connectives and the parentheses (to avoid ambiguity). example: *(p + 9*)^*(rva → p*as is a propositional formula.

*1*.*2. Semantics of propositional logic*

Logical propositions are models of propositional assertions from natural language, which can be “true” or “false”.

The aim of the semantics is to give a meaning (to assign a truth value) to the propositional formulas. The s*e****mantic domain*** is the set of *truth values: {F* (false), *T* (true)}, which satisfy the relations: *F =T, T = F.* New connectives î *(nand"*), (*nor*”), o (“x*or*”) are introduced using the following definitions:

*pTq:=-(p^9), ptq*:=-*(pvq),*

*peq*=-*(249)* These new connectives are used in the design of logic circuits.

Propositional Logic

The semantics of the connectives are provided by the following *truth tables:*

*pa p pna pva p+q pq p*la *pa p*o*q* [TITI FI Τ Ι Τ Ι Τ Ι Τ Ι FI FI F I*TF F F T* I *F F T* I *F T*

*FT T* |

*T*

*TF* 1 *FFT F F* L *T IT Τ* Ι *F*

**Remarks:**

• A conjunction i*s true* exactly when both its operands are true. As a

generalization, the conjunction *p*i^ *p2* A*..*.A*p*n *is true e*xactly when all its n operands ar*e true.* A disjunction *(“inclusive or*”) *is false* only when both its operands ar*e false.* As a generalization, the conjunction p*i v p2 V...V Pn is false* only when all its n operands ar*e false.* The implication *p →q is false* only when the hypothesis *p* is true and the conclusion *q is false (true* cannot impl*y false).* The equivalence *p q* is true only when *p* and q have the same truth value. The connective — *("exclusive o*r”) is the negation of equivalence and it is true

only when one operand is *true* and the other one *is false.* **Definition 1.1.** An *interpretation* of a formula *U(P1, P2,...,Pn) e Fp* is a function *i:{P1, P2,...,Pn}*+*{F,T}* which can be extended to *i: Fp → {F,T}* using the following relations:

*i(p*) = *mi(p) i(p^9)=i(p) ^i(9) i(pv q)=i(p) vi(q) i(p+9)=i(p) →i(9)*

*i(p 49)=i(p) Hi(9)* Interpretations assign truth values to propositional variables and using the semantics of the connectives evaluate formulas assigning truth values to them. The semantics is compositional, meaning that the truth value of a formula is obtained from the truth values of its subformulas. **Definition 1.2. *(*semantic concepts)** Let *U(P1, P2,...,P*n) be a propositional formula. 1. An interpretation *i* which evaluates the formula *U* as *true,*

*i:{P1,..*.,n} + *{T,F*} such that *i(U)=T,* is called a *model of U.*

luates the formular ū

as true

A Computational Approach to Classical Logics and Circuits

2. An interpretation *i* which evaluates the formula *U* as *false,*

*i:{P1,*...*,Pn*} + *{T,F}* such that *i(U)=F*, is called an *anti-model of U.* 3. A *formula U* is called *consistent (satisfiable*) if it has at least one model:

*3i*:{*P...., Pn} + {T,F*} such that *i(U)=T.* 4. The *formula U* is called *valid (tautology*) and we use the notation: F*U,* if *U*

is evaluated as true in all its interpretations: V*i:{P1,*...*,P*n} → *{T,F}, i(U)=T.*

All 2" interpretations of *U are models of U.* 5. The *formula U* is called *inconsistent (unsatisfiable)* if *U* does not have any

model, thus *U* is evaluated as false in all its interpretations: *Vi:{P...., P*n} →*{T,F}, i(U)= F.* All 2" interpretations of *U* are

*anti-models of U.* 6. The *formula U* is called *contingent i*f *U* is consistent, but it is not valid. A

contingent formula has at least one model and at least one anti-model. **Remarks:**

• In order to evaluate a propositional formula *U(P1, P2,...,Pn*) in all its 2"

interpretations, a truth table is built. The table has 2” lines (rows), corresponding to all interpretations and the first n columns are filled with all the possible assignments of *true* and *false* values to the n propositional variables. The column of *U* is obtained using the semantics and the

precedence of the connectives.

• If the truth table (column) of *U* contains only“*T*”, then *U* is *a tautology.*

• If the truth table (column) of *U* contains only“*F*”, then *U* is an *inconsistent*

*formula.* For a propositional formula, its *models* are the interpretations (the table’s rows) which evaluate the formula as *true* and its *anti-models* are the interpretations

(the table's rows) which evaluate the formula as *false.* The notion of *logical consequence* captures the essence of logical thinking and it is a generalization of the *tautolog*y notion, as we shall see in the following definition. **Definition 1.3.** The formula *V e Fp* is a *logical consequence* of the formula *U eFp*, notation: *U*FV , if all the models of *U* are also models of V:

*Vi:Fp +{T,F*} such that *i(U)=T*, we have that *i*(*V*)=*T.* Definition 1.4. The formulas *U e F*p and *V*e *F*p ar*e logically equivalent,* notation: *U* =V , if *U* and V have identical truth tables: *Vi: Fp →{T,F}* we have that *i(U)=i(V*).

Propositional Logic

T

Note that “E” and “=” are meta-symbols used to express logical relations between propositional formulas. **Example 1.1.** Build the truth tables of the following formulas:

*U(p,q,r)=(pvg)*^*(rv p) V(p,q,r)= (par) v(qar) v (q^p)* W*(p,q,r)=(p* † (*p^q)) vr*

*Z(p,q,r) = p^((-9 vr*)*q*) In order to evaluate *U*, the formula is decomposed in two subformulas: *pva* and *rvp*, which are evaluated and then we apply the conjunction between their corresponding truth tables (columns). In the same manner the columns of V,W,Z are built.

*pla r*i*pva rvp | U(p,q,r*) | V*(p,q,r*) W*(p,q,r*) | *Z(p,q,r*) i*TTTT T T*L *т* |i2|T|T|F| I l *т*

is |T|F|T| |i4|T|F|F

*is FTT* 1:0|F|T|F|

*in FFT T* 1 |is|F|FF| *I*

*F*

*T* I *F*

• *UV*,W and Z have 23 = 8 interpretations: 1*1, 12,...,ig.*

• The formula *U(p,q,r)* is contingent, having four models: *;,iz,is, i*n and four

anti-models: i*z, 14, 16, ig. 11:{p,q,r} -> {T,F}, (p)=T, (9)=T, ii(r) = T* and *ii(U)=T ig:{p,q,r} →{T,F}, 16(p)= F, 16(q) =T, 16(r)= F* and i*6(U)= F* The formula W*(p,q,r*) is a tautology, all its eight interpretations are models. The formula Z*(p,q,r*) is inconsistent being evaluated a*s false* in all its eight interpretations, which are the anti-models of Z.

*U =V*, because *U* and V have identical truth tables (columns).

• *U*r*pvq*, because all the models (11*,12,15,17)* of *U* are also models of the

formula: -*pva*

A Computational Approach to Classical Logics and Circuits

*1.3. Logical equivalences*

The following logical equivalences express properties of connectives, relations between them and they are used to transform syntactically a propositional formula, preserving its meaning (semantics).

• *Simplification laws:*

*U=U*

and *U +U=T U AU=F*

and *UVU=T TAU = U*

and *FVU=U U »T =T*

and *U →F=U T →U TU*

and *F→U= U HT=U*

and *UHF=U UT*=G*U*

and *UOF=U UHU = T*

and *UOU = F Idempotency laws: U AU=U*

and *UVU=U* ***Commutative laws:***

*U* A*V =V AU*

and *UVV = V VU U*1*V*=*V*1*U*

and *U1V=VU UHV*=*V HU*

and *UOV*=*VOU Absorption laws: U* A*(U VV)=U*

and *UV (*

*UV)=U* **As*sociative laws:*** *(U*A*V*)^2*=U (V*AZ) and *(UVV*) v2*=U V (V v Z*) ***Distributive laws:*** *U^(V v* Z)=*(U^*)v*(U*AZ) and *Uv(V*AZ)=*(U VV)^(U v* Z) *De Morgan's la*ws*:*

*-(UAV*)=-*UVV*

and +*(U*VV)=*-U* ^-V *Relations between connectives:*

*U →V*=*UVV*

and *U V* =-*(U^*-V) *U →V=U H (U*A*V*)

and *UV* = *V H (*

*UV*) *UAV* =*(U* →*V*)^*(V → U*) and *UOV=-(*

*UV*)V-*(*

*VU) U HV* =*(*

*UV*) → *(U*A*V*) *UvV*=*GU*AV)

and *UV*=*GUV*V) *UVV=-U →V*

and *UWV=-(U*+-V) *LU=U*1*U*

and *U=UU UvV =(U* 1 *U*) 1 (v ^ v) and *UWV =(*

*UU*) Tiv 11) *UVV =(U*1V)*(*

*U*V) and *UWV = (U* 1 v) † (U TV)

Propositional Logic

**The duality prin**ciple: For every logical equivalence *U =V* containing only the connectives 31, , , , 47, 9, there is another logical equivalence *U'=V'*, where *U', V'* are formulas obtained from *U, V* by interchanging the truth values *(T,F*) and the connectives from the following pairs: (^,V),(1,1),(4,0). Note that some of the above laws are pairs of *dual logical equivalences. Dual connectives: (*1,V), (1, 1), (4,). *Dual truth values: (T, F*).

**Definition 1.5.** A set of connectives i*s functionally complete* if there is no truth table that cannot be expressed as a formula involving only these connectives. All the other connectives can be expressed using the connectives from such a set. The following sets of propositional connectives are functionally complete. 1.{-, ^};

2. {4,V};

3. {7, → }; 4. {0,^}; 5. {0, v}; 6. {0, → }; 7.{T };

8. {\*}; The following definitions show that from the semantic point of view, a set of formulas is equivalent to the conjunction of its elements. **Definition 1.6.** Let *U ,U2,...,U,V* be propositional formulas. 1. The set *{U,,U2....,U*,} is called *consistent* if the formula *U, ^U2 A...NU,* is

consistent:

*3i: Fp →{T,F*} such that *i(U, ^U2^....AUN*)*=T.* 2. The set *{U,,U2...,U*n} is called *inconsistent* if the formula *U, ^U2 1..NU*

is inconsistent:

*Vi:Fp → {T,F}, i(U, ^U21...Un)= F.* 3. The formula V, called *conclusion,* i*s a logical consequence* of the set

*{U,U2,...,U*n} of formulas called *premises (hypotheses, facts),* notation: *U1, U2,...,U*nEV, if V*i:Fp →{T,F*} such that *i(U, ^U21 ...^U»)=T*, we have *i(V)=1*

The models of the set of premises are also models of the conclusion. **Theorem 1.1.** Let *U,,U2,...,U,,U,V* be propositional formulas. 1. F*U if and only if -U* is inconsistent.

A formula is a tautolog*y if and only if* its negation is inconsistent. 2*. UFV if and only if FU →V if and only if* the set *{U,V*} is inconsistent.

A Computational Approach to Classical Logics and Circuits

3. *U =V if and only if EU HV*.

*U* and *V* are logically equivalent *if and only if* the formula

*U HV* is a tautology. 4. *U*1*,U2 ....Un FV if and only if*

F*U, ^U2 1... A Un →V if and only if*

the set *{U*1,*U2,...,Un,* -V} is inconsistent. The proofs of the previous assertions are simple and we shall prove only some of them. **Proofs:** 2. If *U* E*V* then F*U →V*.

If *U* EV, using the definition of logical consequence relation (Definition 1.3), then:V*i:Fp →{T,F*} such that *i(U)= T* we have that i(*V)=T.* We have to evaluate the formula *U* →V in all the possible interpretations.

There are two cases: a) and b). a*) i(U)=T* and using the hypothesis *i(V)=T,*

we have that *i(*

*U V*)*=i(U) →i(V)=1 →T=1* b) *i(U*)= F thus *i(*

*U V*)*=i(U) →i(*V)*=F→i(V)=T* From a) and b) we have that V*i: Fp →{T,F}, i(*

*U V* )*=T*, therefore F*U →V*. 3. If *U =V* then F*UV* .

If *U =V* , using the definition of logical equivalence relation (Definition 1.4):

*Vi:Fp →{T,F}, i(U)=i*(V), then *Vi: Fp →{T,F}, i(*

*U V* )=*T* and according to the definition of a tautology we have that F*U HV* 4. If the set {*U,, U2...,Un,* -V} is inconsistent then *U1, U2,...,U*n FV holds.

If the set *{U1,U2 ....U*n, -V} is inconsistent, using the definition of

inconsistency of a set of formulas: *Vi:Fp →{T,F} , i(U;^U2 1...1 Un^*-V)= *F*, then *Vi:Fp* →*{T,F*} there are two possible cases: a) and b). a) 3*k*, 1 s*k* s*n*, such that *i(UK)= F* then

*i(U, ^U2 1*.*.. AU»)= F*, thus i is an anti-model of the set of premises

of the logical consequence: *U1,U2,...,Un* EV and it is an irrelevant case. b) *Vk,*1 s*k sn, i(UK)=T* and i(-V)= *F* then

*i(U,*^*U2 1... Un*) = *T* and i(V)=*T* then

15

Propositional Logic

any model *i* of the set of premises *{U,,U2,*..*.,U*n} is also a model of the conclusion *V*, so according to the definition of logical consequence relation we have that: *U ,U2,...,Un* FV holds.

**Remark:** The assertions 1, 2 and 4 from Theorem 1.) are used in the refutation proof methods, such as *resolution* and s*emantic tableaux method* and they model the proof by contradiction (“reductio ad absurdum").

Theorem 1.2. Let S = *{U1, U2...,U*n} be a set of propositional formulas. 1. If S is a consistent set, then V*j*, 1*sjsn, S - {U*,} is a consistent set. 2. If S is a consistent set and V is a valid formula, then SU{V} is a consistent

set.

3. If S is an inconsistent set and *U*; is valid, where 15*j*sn, then the set

S-*{U*;} is inconsistent. 4. If S is an inconsistent set, then W*V e Fp*, the set SU*{*v} is inconsistent. Proofs: The proofs of the above assertions are simple, the Definition 1.6 is used. 1. If S is a consistent set then

g*i:Fp →{T,F*} such that *i(U, ^U2 1...*A*U,)= T* then *Ji: Fp →{T,F*} such that V*k*, 15*k Sn, i(Uk)=T* then *Ji: Fp →{T,*F} such that Vj, 1*5 jan,*

*i(U, ^U2 ^..^U;-/^Uj+*1^...*.^U^)=T,* therefore *Vj,* 1*5 j5n, S -* {*U*} is a consistent set. If S is an inconsistent set then *Vi: Fp →{T,F}, i(U, ^U2 A.... U)=F* then *Vi: Fp →{T,F*} and W*e Fp, i(U,^U21.... Un*^V)=*F,* therefore the set Su{v} is inconsistent.

**Theorem 1.3.** Let *R*, S be sets of propositional formulas and *U,V,*Z, be propositional formulas.

The logical consequence relation has the following properties: 1*. monotonicity:*

if *REU* and *R*SS then SE*U; 2. cut:*

if SE*V;, Vj e* {1,...*,n}, n*eN and SU{*V1*,V2,..., Vn}E*U* then SE*U;*

A Computational Approach to Classical Logics and Circuits

*3. transitivity:*

if SFU and {U} EV then SE*V*; 4. *conjunction in conclusions (right "and" );*

if SE*U* and SE*V* then S F*UAV*; *5. disjunction in premises (left "or"):*

if SU*{U*} EZ and SU{

VEZ then S*U{UVV*HEZ;

Proofs: These properties can be easily proved using the definition of logical consequence relation. 1*. monotonicity:*

If *REU*, then *Vi:Fp → {T,F*} such that *i(R) =T*, we have that *i(U)=T.* If *R*CS then *Vi:Fp + {T,F*} such that *i*(S)*=T* then *i(R)=T.* From these two relationships, we have that *Vi: Fp + {T,F*} such that

*i*(S)=*T* then *i(R)=T,* s*o i(U)=T* and thus SE*U.* ***cut:*** If SE*V, Vj* e{1,...*,n},n* e N, then *Vi: Fp → {T,F*} such that *i*(S)=*T,* we have that i(V;)=*T.* So, *Vi:Fp +{T,F}* such that i(S)=*T, Vj*e {1,...*,n},ne*N, we have that *i*(*V*,) *=T.* In other words, *Vi: Fp →{T,F*} such that i(S)=*T, i*(SU*{*V1,V2,...,Vn})=*T.* But Su{V1, V2,...,Vn} F*U*, which means that *Vi: Fp +{T,F}* such that i(SU{V1,V2...,Vn})=*T*, we have that *i(U)=T.* From the last two statements we obtain that *Vi:Fp →{T,F*} such that *i(S)=T, Vj* = {1,...*,1},n*eN, we have: *i*(SU{V1, V2,..., Vn})=*T* and that *i(U)=T.* So, *Vi: Fp →{T,F*} such that *i*(*S)=T*, we have that *i(U)=T*, which is

SE*U. 3. transitivity:*

If SE*U*, then V*i: Fp →{T,F*} such that *i*(*S*)*=T*, we have that *i(U)=T.* If *{U*}EV, then V*i: Fp →{T,F}* such that *i(U)=T'*, we have that *i(V*)=*T.* So, *Vi:Fp +{T,F*} such that i(S)=*T*, we have that *i(U)=T* and also *i(V)=T*, which is SE*V.*

Propositional Logic

*1.4. Nor****mal forms in proposition****al logic* Some of the proof methods need as input data formulas having a certain character of “normal” or “canonical form. Definition 1.7*.* 1. A *literal* is a propositional variable or its negation. (*p,-9,r*). 2*.* A *clause* is a disjunction of a finite number of literals. (examples:

*p, pvq, rvqus*). 3. A *cube* is a conjunction of a finite number of literals. (examples:

*9, P4-9, rasAp*). 4. The *empty clause,* denoted by o, is the clause without any literal and it is the

only inconsistent clause. 5. A formula is i*n disjunctive normal form (*DNF), if it is written as a disjunction

of cubes: V1 (1*-1*1,), where *li,* are literals. 6. A formula is in *conjunctive normal form* (CNF), if it is written as a

conjunction of clauses: -(*V-14,*), where *l.,* are literals.

Example 1.2.

*p* - DNF with one unit cube; *punovr* - DNF with three unit cubes;

*p^g* - DNF with one cube; *pvlqur) v GP*A - As) - DNF with three cubes.

Example 1.3.

*p* - CNF with one unit clause; *PV-vr* - CNF with one clause; *p^g* - CNF with two unit clauses;

m*p ^ (qv*-r)^*(pvry*-S) - CNF with three clauses. **Property:** Let *{l), 12,..., I*n} be a set of literals. The following assertions are equivalent: 1. The clause v";=1*1*; is a tautology. 2. The cube ^*«ll;* is inconsistent. 3. The set *{*l*l,l2,..., In*} of literals contains at least one pair of opposite literals:

*3j,k* € {1,..,n} such that *l; =-1k*

18

A Computational Approach to Classical Logics and Circuits

**Example 1.4.** 1. The *clause U = pvqur vnp* is a tautology *(UET*) since *p, -p* are

opposite literals. 2. The *cube V = 2*^*q^r^ p* is an inconsistent formula *(U =F*), because it is

a conjunction with *p, -p* as opposite literals. **Theorem 1.4.** Every propositional formula admits an equivalent conjunctive normal form (CNF) and an equivalent disjunctive normal form ( DNF). The *normalization algorithm* consists of transformations which preserve the logical equivalence and are applied to the initial formula in order to obtain the corresponding normal form. **Step 1:**

• The formulas of *“U → V*” type are replaced by the logical equivalent form

*LUVV.*

• The formulas of "*UHV*” type are replaced by the logical equivalent form

*GUVV*)^*GVVU*). **Step 2:**

• De Morgan's laws are applied: the negations are pushed in until they apply

only to propositional variables.

Multiple negations are eliminated by the reduction rule: *U=U.* Step 3:

• The distributive laws are applied.

**Theorem 1.5.** 1. A formula in CNF is a tautology if and only if all its clauses are tautologies. 2. A formula in DNF is inconsistent if and only if all its cubes are inconsistent.

**Remarks:**

• The first part of the above theorem provides a direct method to prove that a

formula is a tautology. The DNF of a propositional formula provides all the models of that formula, finding all the interpretations which evaluate, one by one, the cubes as true. The CNF of a propositional formula provides all the anti-models of that formula, finding all the interpretations which evaluate, one by one, the clauses as false.

**Dual concept**s: clause-cube, DNF-CNF.

Propositional Logic

**Example 1.5.** Write the equivalent CNF of the second axiom of propositional logic:

Az *=((U* (V → Z)) + *((*

*U* V)+(*U →* Z)).

The normalization algorithm is applied as follows:

*2* 3 4 5 6 Az =*((U +(V* → Z))+*((*

*U V* )*+(*

*U* Z ))= (replace → from the inside formulas, denoted by 2,4 and 6) *=((U*+GV v Z))+*(GUVV*)→ *GUv Z)*) =

(replace the main connective →, denoted by 3) =-(*U* →GV v Z)) v*(GUVV*)+*(-Uv Z))* =

(replace both connectives, denoted by land 5 ) = (*UVGV* v Z) G*UVV*V*U v Z*) =

(apply De Morgan's laws) *=(U ^V*^-Z) *v (U ^*-V)*VGUv*Z - DNF with 4 cubes

(apply the distributive laws) *=(UVUV-U v Z)^(UVVVGUv*Z)^*(V VŲ V-U v Z*)^

(*VV*-VV*GUvZ*)^(-Z*vUVUv*Z)^(Z*VVVGUvZ*) =*T* We have obtained a CNF with 6 clauses which are tautologies. Each clause contains at least one pair of opposite formulas (underlined). Thus, according to the previous theorem, the formula Az is a tautology.

**Example 1.6.** Write the equivalent DNF of the formula: X = 4; =-*(U* (*V →U*))

X =*-(U → (V →U*))=*(UVGV VU))=U* ^(*V VU))=U ^* ^-*U* We have obtained a CNF (with three unit clauses) or a DNF (with one cube). DNF contains an inconsistent cube, having a pair of opposite formulas, thus X is an inconsistent formula and A, (the first axiom of the propositional logic) is a tautology. **Example 1**.7*.* Using the appropriate normal form write all the models of the propositional formula: *U =(*

*p q +r*)+*(p+r)^q.* We apply the normalization algorithm:

*U=(p1q*+r) → *(p+r)*^*q=-(p^g + r) v(p+r)^q=*

=-*(p*^*g) vr*) *v (pvr)^2= (p^q*^ )v*Gpvr*)^*q=*

=*(p^q*^r)v*Gp19) vr9)* - DNF with 3 cubes The models of *U* are the interpretations which evaluate one by one the cubes of DNF as true. In a cube we assign the truth value *I* to all its literals.

20

A Computational Approach to Classical Logics and Circuits

**Cube**: *p^q*^ r provides one model assigning to all the literals *{p, q,*-} of the

cube the truth value *T.*

i*z:{p,q,r}* → *{T,F},* ;*(p)=T, 1(9)=7,1(r)= F* and *i(p^q*^ r)=*T* **Cub**e: *p^g.*

The propositional variable r is missing in the cube's expression, so the truth value of the cube does not depend on the truth value assigned to *r*: we need to assign to *p* the value *F*, to *q* the value *T*, but to r we can assign either *T* or *F* such that the cube is evaluated as true. Therefore this cube provides the following models: *iz:*{*p,q,r} →{T,F}, iz(p)=F, iz(q) =T, iz(r)=T* and iz(*-p19)=T*

i*z:{p,q,r*} *→{T,F},* i*z(p) = F, iz(q) = T, iz(r*) = *F* and iz(*p^q)=T* **Cube**: *r^g* provides the models:

1*4:{p,q,r}+{T,F}, 14(p)=T,14(q) =T,14(r)=T* and i*4(r^q) =T*

*is:{p,q,r} →{T,F}, is(p) = F,is(q) = T, i5(r) = T* and i*s(r*^*9) = T* Note that i*n = i*s, thus *U* has four distinct models is, *i2,13,14:*

*i(U)= i2(U)= iz(U)= 14(U)=T* All the other four interpretations, from the total of eight possible interpretations, evaluate the formula *U* as false, they are anti-models of *U.*

**Example 1.8.** Using the conjunctive normal form of *U = pvq→ pag* find all its anti-models.

The normalization algorithm is applied in order to obtain CNF(*U). U= pvq→ paq=-(pvq) v(p*^*9)*=*( 1-9) v(p*^*9)* apply distributivity

=(*pvp)*^*(pvq)*^*(qvp)* ^*(9vq)* =

*=T^(-pv9)^(qvp) ^T* =(*-ovq)^(-qv p)* - CNF with 2 clauses The anti-models of *U* are the interpretations which evaluate one by one the clauses of CNF as false. In a clause we assign the truth value *F* to all its literals. Clause: *pva* provides the anti-model:

į*j:{p,g} →{T,F}, (p)=T,(9)=F* , thus i*i( -p*^*q)= F* and i*;(U)= F* **Claus**e: *qv p* provides the anti-model:

*iz:*{*p,q} →{T,F}, iz(p)=F,12(q*)=*7* thus iz*(-qV p)= F* and iz*(U)=F U* has two anti-models: i, and is , which evaluate the formula as false. The total number of its interpretations is 22 = 4, therefore *U* has two models which evaluate the formula as true, and they correspond to the other two possible assignments of *T* and *F* to the variables *p* and *q.*

Propositional Logic

*1.5. Formal (axiomatic) system of propositional logic* In this section we present the formalization of propositional logic proposed in paper [62]. From a syntactic perspective propositional logic is defined using the axiomatic system: P = *({p, Fp, Ap, Rp*) where: 1. E*p = Var \_ propos U{T,F} UConnective*s U{(,)} is the v*ocabulary*

- *Var\_propos = {p1, P2*,...} is the s*et of propositional variables* - *Connective*s = {*-(negation), 1 (conjunction), V(disjunction),*

*(implication), 4(equivalence)*}- the set of *logical connectives 2. Fp* is the s*et of well-formed formulas,* which is the smallest set of formulas

satisfying the rules:

*base: Pi e Fp, i*=1,2,...; *- induction:*

if *U, V e F*p then:

*-UE Fp, U VE Fp, U V VE Fp,U →VE Fp,U V € Fp closure*: all formulas from *F*p are obtained by applying the above rules

in a finite number of steps. 3. *Ap* = {A1, A2, A3} is the set of axioms of propositional logic

*A:U → (V →U*) *A2:(U+(V* → Z)) + *((*

*U*V ) → *(*

*U* Z)) A*z :(U* →V)→GV +-*U) - modus tollens* These axioms are in fact axiomatic schemes, where *U,V, Z* are arbitrary

formulas, generating an infinite number of axioms. 4. *Rp = {mp}* is the set of *inference (deduction) rules c*ontainin**g *modus ponens***

*rule.* Notation: *U,U V* Emp *V*, with the meaning: “from the formulas *U* and *U V* we deduce (infer) V”.

**Remark:**

All the axioms obtained using the axiomatic schemes A1, A2, A3 are tautologies (valid formulas).

**Definition 1.8.** Let *U,,U2,...,U*n be formulas, called *hypothese*s and V be a formula called *conclusion. V is deducible (derivable, inferable) from U7, U2,...,U,* and we denote this by *U,,U2,...,Un*t , if there is a sequence (f*1, f2....,fm*) of formulas such that fm = V and V*i e*{1,..*., m}* we have a) or b) or c).

A Computational Approach to Classical Logics and Circuits

a*) fie Ap* (axiom); b) *fi e{U1, U2...,Un*} (hypothesis formula); c*) fins fin* F m*p fie iş <i and in <i* (formula *fi* is inferred using *modus*

*ponens* from two existing formulas) The sequence (*fi, f2,...fm*) is called the *deduction of V from* the hypotheses *U1, U2,...,Un*

**Definition 1.9.** A formula *U e Fp*, such that ØH*U* (or F*U*) is called *theorem.*

**Remark**: The theorems are the formulas deducible (inferable) only from the axioms and usin*g modus ponen*s as inference rule.

This axiomatic system (P) belongs to the group of Hilbert axiomatic systems [24], all of them having only *modus ponens* as inference rule. The formal system *(P)* presented above has as a basic binary connective *implication (*→). Other Hilbert systems change the axioms or use also *disjunction (*V) as a basic connective. All these axiomatic systems are equivalent: the axioms of such a system can be proved as theorems in all the other axiomatic systems.

**Example 1.9.** Prove that: *U,U →V*, *V* →ZEZ using the definition of deduction. *U,V,Ze Fp* We build the sequence *(f1, $2,43,f4,$5*) of formulas as follows:

*fi:U $2:U →V*

f*1, $*2 Emp V *f3* :V *fa:V* → Z

*$3,5*4 mipZ *f5:2* Note that *fi, f23f4* are initial formulas (hypotheses), and *f3, fs* are obtained using *modus ponens* rule. According to Definition 1.8 the sequence: *($1,82,83,84, 8*5) is the deduction of Z from the hypotheses *U,U →V*,V → Z.

**Example 1.10.** Prove that: *pvq,pvr, 9*.r using the definition of deduction. In order to apply *modus ponen*s the 'v' connective must be written in an equivalent form using' + '.

Propositional Logic

The sequence (*$1,$2, f3, 84, 85, 86,fy*) of formulas is built as follows:

*fi: pvq=p →q* (hypothesis) *$2:pvr*=*mp ►*r(hypothesis) *f3:*4*9* (hypothesis) *f4 :(p →9*)(*→-p*) - axiom A*3 (modus tollens)*

*f1.f*4 Fmp *---P $5:-9-→-p*

*f32f*s Fm*p - fo:-p*

*12,86* E*mp*r *f7 :r(*conclusion) Using only the hypotheses we cannot prove the conclusion, so we need to use axioms. In the deduction process modelled by the sequence *($1,$2,83, 54, f5, 56, 57*), the formulas *f1, 82, f3* are hypotheses, *f*4 is an instantiation of axiom Az and *f5, forf* are derived using *modus ponens.* According to Definition 1.8, we have proved that the deduction holds. **Example 1.11.** Prove that H*U*N →V using the definition of deduction. We transform syntactically the formula *UAV V* such that the only connectives are 7, →. The conjunction is replaced using the following logical equivalence:

*U*A*V*=(-*UV-V*)=*(*

*UTMV*) *UAV* →V=*(*

*U* V)→V We use the axioms A, and Az as follows:

fi: -*V -> (U* →-V) - an instance of axiom A *f*2:4V ->*(U*1)+(*(*

*U n* TM - axiom A*3 (modus tollens)* fi*, f*2 Emp -*(U* +4V) →V=G*UVV*) →*V =U ^V →V* The formula *UAV →V* was deduced only from the axioms and *modus ponens* inference rule, therefore it is a theorem. **Example 1.12.** Prove that E*U →U*, building the deduction of *U →U* from the axioms only. The sequence of formulas (*fi, f2, f3, 44, $5*) is built.

*fi:U+((UTMU)→U*) obtained from 4*:0 +*(*V →U*)

replacing V by *U →U;*

A Computational Approach to Classical Logics and Circuits *$2:(U* +(*(U → U) →U*)) + (*(U+(UTMU*)) → *(UTMU*)) obtained

from *A*2:*(U* (*V* → Z))+*((*

*U V* )*→ (U* → Z)) using the replacements: V by *U →U* and Z by *U;* $1,82 Em*p (*

*UTM(U →U*)) → *(UTMU) sz:(U+(UTMU)*) → *(UTMU*) *f4:U →(*

*U U* ) obtained from A*:U (V →U*) replacing V by *U f3, f4* F m*p UTMU fs:U →U* Therefore the sequence *(*f*1,82,83,84, 85*) is the deduction of *U →U* from the axioms, so *U →U* is a theorem.

**Example 1.13.** Prove that -*U*F*U →V* using the definition of deduction.

In the inferential process the sequence of formulas *($1,$2,53, 54,* f5) is built as follows:

*fi:U* – hypothesis formula *12:4U*+GV *U*) obtained from A*:U (V U)*

replacing *U* by *U* and V by V; f*19f*2 Fmp -V >*-0 f*3:4V -*U f4:(*-V →-*U) → (*

*U* V ) obtained from Az*:(*

*U* V)+(-V +-*U*)

replacing *U* by „V and V by -*U; $3, f*4 Fm*p U V f5:U+V* The sequence *(f1,$2,33,34, f*s) is the deduction of *U V* from *U* and the axioms.

In the inferential (deductive) process, besid*e modus ponens,* other inference rules are used. These rules can be also expressed as theorems.

*addition UFUVV*

inference rule F*U+UvV*

theorem *simplification U*A*V*E*U*

inference rule *EUWV →U*

theorem *UAV*E*D*

inference rule F*UN →V*

theorem

Propositional Logic

*conjunction modus ponens*

*modus tollens*

| *U*,*V*F*UAV*

*U,U >V*FV F*U^(U* →V) → *V*

-*V,U →V*E*U* E*VA(*

*U*V)-*U U* →VEV →*GU* F*(*

*U V*)→ *GV -U) U* →*V*,V -→ ZH*U* Z F*(U* →V)^(*V* → Z) *→ (*

*U* Z ) *U,U →V*,V →ZEZ F*U AU →V*) ^ (V → Z) → Z *UVV, U V* ZE*V v Z* F*UVV*) A*GUv*Z) +(*V v* Z)

inference rule inference rule theorem inference rule theorem inference rule theorem inference rule theorem inference rule theorem inference rule theorem

*syllogism*

*resolution*

*1.6. The theorem of deduction and its reverse* Some theorems, even if they are syntactically simple, are very difficult to be proved using only the axioms and *modus ponens,* because we need to guess the starting-point axiomatic schemes and the corresponding replacements of the arbitrary formulas *U,V,*Z. The combination of the following two theoretic results is used to help us prove at the syntactic level theorems having a special syntactic form (the connectives are mainly' ). **Theorem 1.6. Theorem of deduction:** If *Uj,...,Un-1,Un*EV , then *U1,...,Un*-1t*U*n →V.

**Theorem *1*.7. Reverse of the theorem of deduction:** If *U* 1,.*..,U*,-1*U*, →V then *U1,...,U1-1,Un*EV. Applying n times the theorem of deduction and its reverse we obtain:

*Uj,...,U-1,U*n EV if and only if *U*1*,...,U*n-1 *+ Un* →V if and only if *U*1*,...,Un-*2 +*U*n-1 *+(U*n →V) if and only if *U*F*U*2+(..*. → (U*n-1 *(*

*U* V ))...) if and only if F*U, →(U2* + ... + *(U*, →*V*)...))

A Computational Approach to Classical Logics and Circuits

Note: The application of the theorem of deduction and its reverse means that a premise (hypothesis) of a deduction can be moved from the left-hand side of 'F' to its right-hand side as a premise of a main' t' and vice-versa. **Consequences of the theorem of deduction:** 1. F*U+((*

*U* V)→*V*); 2. F*(*

*U TM*V) → ((V → Z)+*(U* → Z)) the *syllogism* law; 3. F*U →* (V → Z)) → (*V (U* → Z)) the *permutation of the premises* law; 4. F*(U*+(*V* → Z)) → *(*

*UWV* → Z)) the *reunion of the premis*es law; 5. F*(UAV* → Z) → *(U* → (*V* → Z)) the *separation of the premise*s law.

A syntactic and direct proof method based on these theorems consists of three steps: Step 1: The initial theorem (formula) to be proved is written in an equivalent form

as a deduction, applying the reverse of the theorem of deduction several times. Step 2: The deduction obtained in Step1 is proved using the axiomatic system. Step 3: Starting with the deduction proved in Step 2 we apply the theorem of

deduction several times. All the premises moved in Step 1 from right to left are

now moved from left to right in the reverse order to obtain the initial formula. **Remarks:**

• Step1 may be skipped if we can guess the starting-point deduction in order to

prove the initial formula. In Step 3, if we change the order in which the premises of the deduction are

moved from left to right, new theorems can be proved. **Example 1.14.** The consequences 1 and 2 are proved in the following. 1. We begin with the *modus ponen*s rule: *U,U →* Vmp V

Applying the theorem of deduction we obtain: *U*F*(*

*U V*)→V and we continue with another application of the same theorem:

F*U + ((U →V*)*→V*). 2. We begin with the deduction: *U,U →V*,*V* →ZEZ proved in Example 1.9. Application of the theorem of deduction *(U* is moved from left to right):

*U →V*,VŰZH*U*Z Application of the theorem of deduction (*V* + Z is moved from left to right):

*U* →VE(VZ) *(*

*U* Z ) Application of the theorem of deduction *(U →V* is moved from left to right):

F*(*

*U V* )+*(*(V → Z)+*(U →* Z))

Propositional Logic

**Example 1.15.** Using the theorem of deduction and its reverse prove:

F*(*

*p* r) → (*par →9)→ (p → 9*))

Step 1: The reverse of the theorem of deduction is applied to obtain the starting-point deduction: if *(*

*p r*)+*((par →q)+(p+q)*) then

*p+r*+*p*a*rq)(p +9*) then

*p →r,par →qtp →* then *p →r,par →q,pta*

Step 2: We prove the deduction obtained in Step 1. Using Definition 1.8 we build the sequence of formulas: (81, *82, 83, 84, 85,86):*

*fi:p* - premise (hypothesis) *$2:p →7* - premise

*f1, f2* tm*p" f3:r f4: fi^fz = p*ar (conjunction of the conclusions) *$5:p^r+q* - premise

*$4*, fs l-*mp 9* 16:9 The sequence *(f1, f2, f3,64, 65,fo*) is the deduction of a from the premises: *pr,par →9, p.*

Step 3: To the deduction *p r,par →q,pą* we apply three times the theorem of deduction.

There are 3!=6 such possibilities (to move the premises to the right-hand side of the meta-symbol F) and we prove 6 theorems: *1*1*,12,13,14,15,76* 1*. the premises are moved to the right-hand side of 'F'in the following order:*

*p, par q, pr.*

if *p +r,par →q,p* ta then

*p →r,par* →*qEp* →q then

*p*+r+*par*+q) →*(p-*— q) then E*T = (p* →r) → *((par →9*)→*(p +9*))– the theorem to be proved.

A Computational Approach to Classical Logics and Circuits

*2. the premises are moved to the right-hand side of* 'F*' in the following order:*

*P, p <r, par →q.*

if *p +r,par →9,p*t-q then

*p+r,par →9p*+q then

*p*a*r* →9*(p+r*) *(p +9*) then F*T2 =(par →9)+((p-r)+(pq))* The following theorems can be also proved: F*Tz =(p*a*r+q*) +*(p+((p+r*) q))) F*T4 = p +((par →9)→ ((pr) →q)*) E-*T3 =(p+*r) → *(p =((par →9*) ►9))

*To = p + ((pr*)*((p*a*r →q) →9*)*)*

*1.7. Pro****perties of propositional logic*** The properties of propositional logic are: *compac****tness, soundness, completenes*s*,*** *coherence, non-contradiction and decidability.* **Theorem 1.8. (compactness 1)** *An infinite set of propositional formulas has a model if and only if each of its subsets has a finite model.* **Theorem 1.9.** *Let S* =*{U,U2,..*.*,Ums...} be an infinite set of propositional formulas. 1. S is inconsistent if and only if IkeN\*, such that {U, U2,...,Uk} is*

*inconsistent.* 2. *S is consistent if and only if*

*{U1} is consistent and {U1,U2} is consistent and*

*{U,,U2,...,Um} is consistent and*

**Remarks:**

An infinite set of propositional formulas is inconsistent if and only if it has an inconsistent finite subset, therefor*e inconsistency can be proved in a finite number of steps.* An infinite set of propositional formulas is consistent if and only if all its subsets (an infinite number) are consistent, therefore *consistency cannot be proved in a finite number of steps.*

Propositional Logic

**Theorem 1.10. (compactness 2)** *A propositional formula V is a logical consequence of an infinite set of propositional formulas S (notation:* SE*V) if and only if there is a finite subset {U,,U2 ...,Un*} cS *such that U ,U2,...,Un* EV. **Theorem 1.11. Soundness theorem** *(syntactic validity implies semantic validity):*

*If EU then FU (a theorem is a tautology).*

**Theorem 1.12. Completeness theorem** *(semantic validity implies syntactic validity):*

*If EU then EU (a tautology is a theorem).*

**Theorem 1.13. Soundness and completeness for propositional logic:**

F*U if and only if EU (a formula is a theorem if and only if it is a tautology).*

og

Consequences of the last theorem are the following properties: 1. *Propositional logic is non-contradictory*: we cannot have simultaneously:

E*U* and E*U.* 2*. Propositional logic is coherent*: not every propositional formula is a theorem. 3. *Propositional logic is decidable:* always we can decide if a propositional

formula is a theorem or not. The truth table method is a decision method.

*1.8. Decision problems and proof methods* **Decision problems** in propositional logic: 1. Is a propositional formula a tautology/theorem*?*

E*V* or E*V* 2. Is a propositional formula a logical/syntactic consequence of a set of

hypotheses?

*U,...,U*n ÉV or *U...U*nFV In order to solve these two decision problems, theorem proving methods are applied. These methods will be studied in the next chapters. In the following we provide w the classifications of the studied proof methods from two perspectives: First classification: **semantic *versus* syntactic methods** 1. *semantic proof methods:*

• the truth table method;

• the semantic tableaux method;

• CNF- conjunctive normal form.

30

A Computational Approach to Classical Logics and Circuits

2. ***synta****ctic proof methods:*

• the definition of deduction;

• the theorem of deduction and its reverse;

• the resolution method;

• the sequent calculus method. Second classification: **direct v*ersus* refutation methods** 1. *direct methods:* they use directly the formula to be proved:

• the truth table method;

• the CNF- conjunctive normal form;

• the definition of deduction;

• the theorem of deduction and its reverse;

• the sequent calculus method.

2*. refutation methods:* they model the "reductio ad absurdum" (proof by

contradiction) using the negation of the formula to be proved:

• the semantic tableaux method;

• the resolution method.

***1.9. Exercises*** Exercise 1.1. Check the following properties for \*('nor*'), 7 *('nand'*) and @ ('xo*r*') connectives using the truth table method. 1. associativity of 't'connective:

*p*t *(q* T r) =*(p*T*q)* fr; 2. associativity of 'I'connective:

pt*(a*t*r)*=*(pq)* r; 3. associativity of 'o' connective:

*po (*

*gr)=(*

*pqO*r; 4. distribution of '1' connective over 't connective:

*pî (q + r) = (p* † q*) (p*îr); 5. distribution of 'I'connective over '1' connective:

*pt (a* Tr)=*(p+q*) † (*ptr*); 6. De Morgan's laws for ' t' and '1':

*(p+q*)=*-p7-9* and (pî*q*)=*-p*t-q; *7. p*t *(qvr)*=(*pTq)^*(p Tr) and pt *(qar)=(p+q) v(p*tr). 8. *pt (q*Tp)*= F* and *p*† *(q + p)*=*T;*

Propositional Logic

**Exercise 1.2.** Using the truth table method decide what kind of formula (consistent, inconsistent, tautology, contingent) is *U,,je* {1,2,...,8}. Write all the models and anti-models of *Uj, j*e {1,2,...,8}. 1*. Uı =q^p^r -pv-(gar); 2. U2 =-pv-(q^r) →q^-P;* 3. *U3 =-^(-qvr) →qv-pvr*; 4. *UA=*-*(pvq) vr* →*-p*vG*qvr*); 5. *Uz = -pvGqv*-r)*→9*^*-P*; 6. *U6 =-pvc-*^-r) →9^*-p*ar; *7. Un = p → (9 Ar) vq1p*; 8. *Ug =(pvq)^2 + paqur.*

Exercise 1.3. Using the truth table method, check if the following logical consequences hold: 1. *p +9(p*+1)=*(*

*p q ^r)*; 2*. p+q*F(*q + r*) →*(p*+r); 3. *p +*(*q*r)F(*p+q) →(p*+r); 4. *p+*1+*(q+r*)+*((pvq) ►r*); *5. p qFGP9)→q;*

*6*. *p +9*1=(*q 1) (Pqar)*; *7. p +9*1=(*9 → r*) + *(p →qvr)*; 8. *r →(9-→ p*)F(r +*9*)→ *(r + p).*

**Exercise 1.4.** Prove that the following formulas are tautologies using the truth table method. 1. the left-distribution of over '1': *(p +(qar*))+((*p →9)^(p* +r)); 2. the permutation of the premises law: *(p*-> (*q+r*)) → *(9(*

*p* r )); 3. the reunion of the premises law:*(p → (*

*g r*)) → *(p1q*+r); 4. the separation of the premises law: *(p^q ►r*) + *(p +(q + r*)); 5. the 'cut law:*(p →9)^(p^q ►*r) → *(p +r*); 6. the left-distribution of'*v*'over' +*: pv (q* + r) + *((pvq) + (pvr*)); *7*. the syllogism law: *(p+q)^(q*+r)*(p*+*r)*; 8. the left-distribution of' +'over'v?*:(p+(qvr*))+*((p →9) v(*

*p r)*).

**Exercise 1.5.**

Transform the formulas *Uj, je* {1,2,...,8} into their equivalent conjunctive and disjunctive normal forms. Using one of these forms prove that *Uj, j*e {1,2,...,8} are valid formulas in propositional logic. 1. *U*1 =*(p → (9 4r*))+*((p+9) H (P + r*)); 2*. U2 =(*

*p q)^(*

*pq* +r)+*(p*r); 3. *Uz =(p1q+r)* → *(P +*(*9-→*r));

A Computational Approach to Classical Logics and Circuits

4. *U4 =(p+(qvr*))+((*p+q)v(p+r*)); 5*. Ug =(pv (q + r*)) =*((pvq) + (pv*r)); 6*. U6=(p*+(*q*►*r*))+(*(p+q)(p+r*)); *7. U, =(p+ (q*^*r*))+(*(p +9)^(p+r*)); 8. *Ug =pv*(*a*r)*+((pvq*) → *(pvr)*).

Exercise 1.6. Using the appropriate normal form write all the models of the following formulas: 1. *U1 =(pvqr*)*(p +r*)*^q;* 2*. U2 = (pvg) vr*-^-(*qar*);

*Uz =(p1q+r*) + *(pr)^q;* 4. *U4* = *(pvq)*^-*} → p^qur;* 5. *Ug = pv (9^ 5r) → p^q^or;* 6. *U*g =*(pvg+7)*+(*9+r)*^*p; 7. U, =(qvrp)* →*(p*+r)*^q;* 8. *Ug* =*(qur + p*)*+(p+r)^q.* Exercise 1.7. Using the appropriate normal form, prove that the following formulas are inconsistent: 1. *U=(p+(q+r)*^-*((p+q*)+*(p*+r)); *2. U2 =(pvq)*^-*69+p*); 3. *Uz = (p+9)^(p^q+r)^(p^r*); 4. *U4 = (p → (q vr))^(p-9)^-(p+r)*); 5. *Us = p^(9+r)^((p^q^-(par*)); 6. *U6 =(p+(9+r)*)*^(p^q^*-*r)*; *7. U, =(p→ (q+r*)^*-(9+(p*+r)); 8. *Ug =(p1q+r)*^-*(p+(*

*g r)*).

**Exercise 1.8.** Write all the anti-models of the following formulas using CNF. 1. *U =(qar+p) + (p+r*)^*9; 2. U, =(qur* + *p*)+(*p+r)*^*9;* 3. *Uz =(pvq+r)+(q*+r)^*p;* 4. *U4 = PV*-(91*-r) → p^q^r* ; 5. *Ug = pv*-*(9*^ *r) → p^q*^-*r*;

Propositional Logic

6. *Ug =(p1q+r) →(p+r)^q; 7. U,*= -*(pvq) vr + p^-*(*qar)*; 8. *Ug =(pvq+r*)+*(p+r)*^*q.* Exercise 1.9. Using the definition of deduction, prove the following deductions: 1. *p +9,r →t,pvr, \*t;*

2. *p 1, pvp →9,r*+*9*; 3. *9 → p,t →r,qvt, p*h*r;* 4. *pv (q + r), pv9,-p*+r; 5. *pv-qur,q,p*t*;*

*p ovr,png,pr; 7. rv(q p),r vq,77*1*p;*

8. *P→9,9+,→t,pl-t.*

Exercise 1.10. Prove the following theorems using the theorem of deduction and its reverse. 1. F*pv (*

*gr*) *((pvq) → (pvr)*); 2. F*(p < (r +9*))*+(rvpvq);* 3. F*(p → (q + r*) + *(p^q►r*); 4. F*(pag+r) →(p+(q + r*)); 5. F*(p* +(*9+r*))+ *(9 →(*

*p* r )); 6. F(*p(9->r*))+*((p →9)+(p+*r)); 7. F*(p →9)^(p1qr*) *(*

*p r )*; 8. F*(p+9*)*+((p->r*) *→ (p+qar)*). Exercise 1.11. Using the theorem of deduction and its reverse prove that:

1. F*(p(qvr*))+*((p +9)v(*

*p* r)); 2. F*(p →9)+((rvp*) → (r *+9*)*)*; 3. F*pv (q + r)* +*((pvq) → (pvr*)); 4. F*(p+r*)+*((q + r*) *→ (pvq ►*r)); 5. F*(p+q)(*(r +*1)(par →q^1))*; 6. F*(p+r*)-*((par →9)(p9)); 7.* F*Gqvp)* → ((s →*q)* → (*s →)*); 8. *(p → (q + r*)) → *(p*(r +79)).

A Computational Approach to Classical Logics and Circuits

2. F**IRST-ORDER LOGIC**

First-order (predicate) logic was introduced by Gottlob Frege in 1879. It is an extension of propositional logic and allows reasoning about the objects of some non-empty universe and the relations among these objects. As references for the theoretical concepts and results presented in this chapter, the following papers were used [1, 2, 12, 16, 19, 20, 21, 23, 26, 37, 43, 44, 47, 54, 58, 60, 61, *6*2, 63].

2*.1. The axiomatic (formal) system of first-order logic* In predicate logic new syntactic categories: quantifiers (existential and universal), constants, variables, function symbols and predicate symbols, are introduced. We present an axiomatic (deductive) system for predicate logic as was proposed in paper [62]: Pr =(Epr*, F*p*s, Apr, Rpr*), where:

• Ep*r =Var u Const* (U"\_F;) U(U=P,) *Connectives Quantifiers* is the *vocabulary*

*Va*r is the set of variable symbols {x, y, 2,...}; the variables take different values in a specific domain *D* and they are generic terms, type definitions: *book, child, event. Const* is the set of constants {*a,b,c,.*..}; the constants take fixed values in a domain *D* and they usually specify objects names, persons names*: Paul, book\_3.*

F; = {*f*l*f:D! → D*} is the set of function symbols of arity“;” P; = {*p*l*p:D' + {T,F*}} is the set of predicate symbols of arity“*;*” and usually represent connection rules among variables and constants. *Connectives* = {7,^, V, 7, +}; *Quantifiers = { v(universal quantifier),* 3 *(existential quantifier)}* In first-order logic quantifiers refer only variables. The scope of a quantifier is the formula to which the quantifier applies. The priority of the

quantifiers is greater than the priorities of the connectives. To define the well-formed predicate formulas we need to introduce first the

concepts: term, atom, literal. *O TERM*S is the set of terms defined as follows:

- *Var TERMS; Const C TERMS;* - if *f* eFk and tp*, t2...tk € TERM*S then *f (t1,t2 ..., tx) E TERMS*

examples: x*, a, f(*x), g(x,*a), g(f(*x), y)

*U*

0

First-order Logic

O *ATOM*S is the set of atomic formulas (atoms):

- *T,FE ATOM*S - if Pe Pk and tp.*..tk E TERM*S then *P(*t*1 ...tk) E ATO*MS - examples: *T, F, P(*x, y*,a), Q(f(*x)*,a), R(a,g(f*(x), y)

where: x,y e *Var, a e Const, fe*F,ge F2, *P, R* € P3, Q € P2. *o literal -* an atom or its negation.

- examples: *P(f(x),a*, y, g(x*,b)),* -Q(x*,a, f(x)*),

where: x,y e *Var, a, b e Const, f*eFge F2, *P* € P4, Q € Pz.

• Fpis the set of well-formed predicate (first-order) formulas: - if *U, V e F*ps then

*U e Fpr, UWV e Fpr, UVV e Fpr, U*-VE *Fpr, U V e Fpr* - if *UE Fpr*, Xe *Var*, and x is not within the scope of a quantifier then

(*Vx)U(*x) *€ F*ps and (2x)*U(*.\*) e *Fps.*

• Apr = {A1, A2, A3, A*4,* A5} is the set of axioms:

*A:U →* (*V →U*) *A2:(U*+(V → Z))+*((*

*U V*) → *(U →* Z)) *A*z *: (UTM*V)(-V +-*U) (modus tollens) A*4 :(Vx*)U*(x) → *U(1)*, where *t* is a term, *(universal instantiation)* A*g :(U →V*(y)) → *(U* → (Vx)V(x)), where x is not a free variable in *U*

or V, y is free in V and y does not appear in *U.* Rpr *= {mp, univ\_gen*} is the set of inference rules: *- modus ponens* rule: *U,U* →VE mp V *- universal generalization* rule:

*U*(x) Funiv gen (Vx*)U*(x), where x is a free variable in *U.* **Remarks:** 1. *Ap C Apr, Fp C F*pr*e Rp C* Rpr and thus *Theorems p < Theoremspr*. The

theorems in propositional logic are also theorems in predicate logic. 2. In axiom A*4 (universal instantiation),* (Vx)*U*(x) says that *U*(x) holds true for

all the objects in the domain (universe), so *U* holds for any specific object in the domain. The term t used for instantiation can be a variable or a constant of

the domain. 3. The *universal generalization* rule says that if *U* holds for any arbitrary

element x of the universe, then we can conclude that (Vx)*U(*x).

36

A Computational Approach to Classical Logics and Circuits

**Definition 2.1.** 1. In a predicate formula the variables which are within the scope of a quantifier

are called *bound variables,* all the others are called *free variables.* 2. A first-order formula is called a *closed formula i*f all its variables are bound. 3. If a predicate formula contains at least one free variable, the *formula is open.* Example 2.1.

• The predicate formula (Vx)Ez)(*P(*x,*z,a)* v (y)Q(*x, f*(y))) is closed (all

variables are bound), where: x,y,z e*Var, a e Const, f €*F*, P*EP3, Q € P2. The predicate formula (Vx*)P*(x, y) *^Q(2,a)* is open, *a e Const,* x,y,z e *Var, P, Q* € P2, the variables y and z are free and x is a bound variable (universal quantified - within the scope of ).

**Definiti**on 2.2. [62] Let *U ,U2,...,U,V* be first-order formulas, *U,,U2,...,U,* are the hypotheses. *V is deducible (inferable, derivable) from U,,U2,...,Un*, notation: *U*1*,U 2..., U*n FV, if there is a sequence of formulas *(f1, f2..... f*m) such that fm = V and V*i*e {1,..., *m}* we have a) or b) or c) or d). 1. *fie A*p (axiom of predicate logic); *2. f; €{U1,U2,...,Un*} (hypothesis formula); 3. f*ine fin* Emi*p fii <i and iz <i* (formula *fi* is inferred, using *modus ponens*

rule, from two formulas that are already in the sequence); 4. *fi*t univ*\_gen fi, j<i* (formula *fi* is obtained using the *universal*

*generalization* rule from a formula that exists already in the sequence). The sequence (81, *82,..., fm*) is called the *deduction* of V from *U1,U2,...,Un.* **Definition** 2.3. A formula *U e Fp*r, such that ØE*U* (notation: F*U*) is called *a theorem.*

**Ma*r*k:**

**Remark:**

The theorems are the formulas deducible from the axioms, using modus ponens and the universal generalization rule.

In the inferential (deduction) process the inference rules specific to propositional logic *(modus ponens, addition, simplification, modus tollens, syllogism)* and the rules specific to predicate logic *(universal instantiation, universal generalization, existential instantiation, existential generalization*) are used.

First-order Logic

*universal instantiation*

Inference rules for quantifiers (*V*x*)U*(x) Funiv\_*insi U(t), t* is a term (variable, constant of the domain) *U*(x) Funiv gen (V*x)U*(x), x is a free variable in *U*

*univer*s*al generalization existential instantiation existential generalization*

(3*x)U*(x) Fexist*\_inst U(c)*, c is a constant of the domain

*U(D*) Fexist pen (x)U(x), *t* is a variable or a constant of the domain, x must not appear free in *U*

**Remarks:** 1. The *existential instantiation rule s*ays that if *U* holds for some element of the

universe, then we can give that element a name such as c. When selecting symbols, one must select them one at a time and must not use a symbol that has

already been selected within the same reasoning/proof. 2. The *existential generalization rule s*ays that if there is some element *t* in the

universe and t has the property *U*, then there exists something in the universe

that has the property *U.* Example 2.2. Using the definition of deduction, prove that the formula

(Vx)(P(x)^Q(x)) + (*Hx)P*(x) is a theorem. We build the sequence *(*$1*,$2,53, 54, f*s) of formulas as follows:

fi:(Wx)(*P*(x)*^Q*(x)) → *P*(y)^*2*(y) - A4 axiom where y is the term

used for the *instantiation of the universal* quantified variable x *$2:P*(y)^&(y) → *P*(y) – theorem (from propositional logic)

*f*19f2 t-s*yllog*ism\_ ru*le $3* = (Wx)(P(*x)^*Q(x)) → *P*(y*) f4 :*((Wx)(P(x*)^*Q(x)) *→ P*(y)) → ((Wx)*(P*(x*)^*Q(x)) + *(1x)P*(x))

- Ag axiom *f*3, f*4* F mp *f*s = (Vx)(P(x)^Q(x)) + (Vx)P(x*) modus ponen*s is applied *(*$1*,52, 53, 54, f*s) is the deduction (proof) of the theorem

(Vx)(P(x*)^Q*(x)) → (Hx)*P(*x).

**Example 2.3.** Using the definition of deduction, prove that:

(Vy)(Vz)(*P*(y) *VQ*(z)) (Vx)(*P*(*x) v* Q(x)) The sequence (5*1,52,53, 54, 55, 5*6) of predicate formulas is obtained:

fi :(Vy)(Hz)(*P*(y) v Q(z)) – hypothesis

3*8*

A Computational Approach to Classical Logics and Circuits

*$2*:(Wy)(Hz)*(P*(y) vQ(z)) → (Hz)(P(x) *v&(*z)) - A4 axiom where x is

the term used for the *instantiation of the universal* quantified variable y

f*1,32* Fm*p f3 =* (Hz)(*P*(x) v Q*(z)) - modus ponen*s is applied *f4 :(Hz)(P*(x) *v Q(*z)) → P(x*) VQ*(x) - A4 axiom where x is the term

used for the *instantiation of the universal* quantified variable z *$3, f4* Fm*p f5 = P*(*x) v Q(*x*) - modus ponens* is applied

fs Funiv *gen f6* =(*V*x)*(P*(x*) vQ(*x)) *- universal generalization* is applied *(8*1*, 82, 53, 54, 55, f*ő) is the deduction (the proof) of (Wx)*(P*(x*) v*Q(x)) from the hypothesis (Vy)()(P(y) vQ(z)). Example 2.4. Prove that the formula (Vx)*(P*(x) VQ(x)) is derivable from the formula (Vy)(Hz)*(*P(y) v Q(z)), using the definition of deduction. We have to prove that: (Wy)(Hz)(*P(y) V*Q(z))F(Wx)*(P*(x) VQ(x)) The sequence *(*5*1,52, f3,54, 55,* 56) of predicate formulas is build as follows:

*f*1 :(Vy)(V2)(*P*(y*) VQ(*z)) – hypothesis *$2* :(Vy)(Vz)(*P*(y) v Q(z))+ (Vz)(*P*(x*)* v Q(z)) - A4 axiom where x

is the term used for the *instantiation of the universal* quantified variable y

f*i, f2* Fm*p fz* =(Hz)*(P*(x)vQ*(z)) - modus ponens* is applied *f4* :(*Hz)(P(x) vQ*(z)) *→ P*(x*)* VQ(x) - A4 axiom where x is the term

used for the *instantiation of the universal* quantified variable z *f3, f4* Fm*p fs = P*(x) VQ(x*) modus ponens* is applied

fs Funiv *gen fo* = (*V*x)*(P(x*) *v* Q(x*)) universal generalization* is applied *(51,52, 53, 54, 55, fő*) is the deduction (the proof) of (Vx)*(P*(x*) v* Q(x)) from (Vy)(Vz)*(P*(y) v *Q(z*)).

**Theorem 2.1. Theorem of deduction** [20] Let X be a set of predicate formulas and *U, V* predicate formulas. If XU*{U*} EV then XH*U →V*.

**Theorem 2.*2*. Refutation theorem** [20] Let X be a set of predicate formulas and *U* a predicate formula. If XU*{U*} is inconsistent then XH*U.*

The last theorem is used in proof methods such as: *resolution* and *semantic tableaux method,* called *refutation proof methods* and they model śreductio ad absurdum' (proof by contradiction).

First-order Logic

**Example 2.5.** Using the theorem of deduction, prove that the formula

(Vx)(A(x) → *B*(x))+((Vx)A(x) → (Vx*)B*(x) is a theorem.

Step1: We apply the reverse of the theorem of deduction: If (Vx)(A(x) *→ B*(x)) + *(*(Hx)A(x) → (V*x)B*(x)) then

(Vx)(A(x) *→ B*(x))+((Vx)A(x) → (V*x)B*(x)) then (Vx)(A(x) + *B*(x)), (Vx)A(x) +(Vx)*B*(x)

Step2: We prove that (Vx)(A(x) *B*(x)*),* (Vx)A(x)F(V*x)B*(x) building the sequence *(f1, 52*,..*.,fg*) of predicate formulas:

fi :(Vx)( A(x) → *B*(x)) - hypothesis *f2 :(* x)(A(x) + *B*(x)) → (Aly) → *B*(y)) - axiom A4, x is instantiated with

the term y *f1, f*2 Enu*p f3* = A(y) *→ B(y*) *f4 :*(Vx)A(x) - hypothesis *fs :*(Vx)A(x) → A(y) - axiom A4, x is instantiated with the term y

*f49fs t*m*p f*o = A(y) *$3,8*6 F m*p fn = B*(y)

*f*t Funiv\_g*en fo* = (*Vx) B(x) (*f1*, f2...*., f*x*) is the deduction of (*Vx) B*(x) from the hypothesis (Vx)(A(x) → *B(*x)) and (Vx)A(x).

Step3: Using the deduction proved in Step2 and applying twice the theorem of deduction we obtain:

If (Vx)(A(x) *→ B*(x)),(Vx)A(x)F(Vx)*B*(x) then

(Vx)(A(x) → *B*(x)) F (Vx*)*A(x) → (Vx)*B*(x) then E (Vx)(A(x) *B(*x)) + *(*(Vx) A(x) → (Vx)*B*(x))

Thus we have proved that the initial formula is a theorem.

*40*

A Computational Approach to Classical Logics and Circuits

**on**

*2.2. Transformation of natural language sentences into predicate* ***formulas***

In the following examples, sentences from natural language are translated into first-order language. We explain how the quantifiers, the variables, the constants, the function symbols and the predicate symbols are used in predicate formulas in order to introduce objects and to express properties and relations among objects. 1. All Computer Science (CS) students are smart.

(Vx)(CS*\_student(*x) *→ smart(*x*)*) - C*S\_student* and *smart* are unary predicate symbols, expressing

properties of the person x. 2. There is someone who studies at Babes-Bolyai University *(BBU)* and is smart.

(3x)*(student \_BBU(*x) ^ *smart(*x)) - *student\_BBU* and *smart* are unary predicate symbols, expressing

properties of the person x. 3. Every child loves anyone who gives the child any present.

(Vx)(Vy) (Vz*)(child(x)^ present(y*) *A give*s(z, x,y) *→ lov*es(x,z)) - *child , present* are unary predicates symbols;

*love*s is a binary predicate symbol *(love*s(x, y) is true if x *lov*es y); *giv*es is a ternary predicate symbol (*gives*(z, x, y) is true if z *giv*es to x

the *present y*). 4. John's relatives, except Mary, live in Cluj, some of them like opera music, but

all of them like to dance. (Vx)*(relative(John, x) → likes \_to\_dance*(x)^ *Gequal(x, Mary)*

*→ lives*(x*, Cluj)))* ^ (3y)*(relative(John, y) ^ likes \_ opera*(y)) *John, Mary, Cluj* are constants; *- likes\_to\_dance,likes\_opera* are unary predicate symbols;

*relative, lives, equal* are binary predicate symbols; *relative* is a symmetric binary relation, thus in a reasoning process involving this relation we have to add the formula: (Vx)(Vy)*(relative* (x,y) → *relative (*y,x)); the predicate *equal* is defined by the following axioms: *(Vx) equal(x,*x) – reflexivity; (Vx)(Wy)*(equal(*x, y) *→ equal(y*,x)) – symmetry; (Hx)(Vy) (Vz*)(equal(*x, y) *A equal(y,*z) *→ equal(x,z*)) – transitivity.

First-order Logic

5.

If x and y are nonnegative integers and x is greater than y, then xis greater than yż. (Vx)(Wy)(*nonneg*(x) A *nonneg*(y*) ^ greater*(x, y) → *greater(square(*x*),square*(y)) - function symbol: *square* e F1, *square*(x) = x2; - predicate symbols: *nonneg* e Pl*, nonneg*(x):"x >0" and *greater €* P2,

*greater*(x, y):"x >y".

6. In a plane if a line x is parallel to a constant line *d* then all the lines

perpendicular to x are also perpendicular to *d.*

(Vx*)(parallel (x,d*) → (Vy)*(perpendicular(y*,x) *→ perpendicular(y,d*)))

*perpendicular* and *parallel a*re binary predicate symbols corresponding to the geometric relations. In a reasoning process involving these relations we have to add the formulas expressing the properties of reflexivity (for *parallel ),* symmetry (for *parallel* and *perpendicular),* transitivity (for *parallel).*

*7.* The axioms which define the natural numbers:

aj. Every natural number has a unique immediate successor.

existence: (Vx)(Ey*)equal(y, successor*(x))

uniqueness: (Vx)(Vy)(Vz)(*equal(y, successor(*x)) A *equal(z, successor*(x)) *→ equal(y,z*))

az. The number 0 is not the immediate successor of a natural number.

(3x)*egal(0, succesor*(x)), 0 is a constant.

az. Every natural number, except 0, has a unique immediate successor.

existence: (Vx)Ey) (*equal(*0,x) *1 equal(y, predecessor*(x)))

uniqueness: (Vx)(Vy)(Vz)*(equal(y, predecessor*(x*)) ^ equal(z, predecessor*(x)) -> *equal(*y,z))

- unary functions: *successor, predecessor;* - binary predicate: *equal* is reflexive, symmetric and transitive.

The following formulas express the equality of the successors and the predecessors of two equal numbers:

(Vx)(Vy)(*equal(*x,y) *→ equal(successor*(x*), successor*(y))) (Vx)(Vy)(*equal(x,* y) *→ equal(predecessor*(x)*, predecessor*(y)))

A Computational Approach to Classical Logics and Circuits

2.*3. The semantics of first-order (predicate) logic* The semantics of predicate logic realize the connection between the constant symbols, the function symbols, the predicate symbols and the real constants, functions, predicates from the modeled universe. Also, a meaning for each formula from the language, in terms of the modeled universe, is provided. **Definition 2.4.** An *interpretation* for a language *L* of predicate logic is a pair *I =< D,m>*, where: 1. *D* is a nonempty set called the domain of interpretation. 2*. in* is a function that assigns:

- a fixed value *m(c) e D* t*o* the constant *c*. - a function *m(f):D*" *+ D* to each *n*-ary function symbol *f;*

- a predicate *m(P):D" →{T,F*} to each n-ary predicate symbol *P.* **Notation**s: *I =< D,m*> be an interpretation.

• 1*1| = D* is the domain of *I, Va*r is the set of variables.

• *1* X | is *m*(X) where X is a predicate symbol or a function symbol.

*As(I*) is the set of assignment functions for variables over the domain of interpretation *1.*

An assignment function *fa e As(I*) is defined as follows: *fa: Var +1*.

• [*fa], = {*f*a's fa'e As(I*) and *fa*'(y) = *fa*(y), for every y + x}. **Definition 2.5.** Let *U* and V be first-order formulas, A a formula free of the variable x, *1* an interpretation and *fa e As(I*) an assignment function. The evaluation function v'co is defined inductively as follows: 1. V'fa(x) = *fa*(x), x € *Var ;* 2. vf*a*(c)= 1|c!,ce *Const* ; 3. vf*a (f(t*1*,62....st*n) = 118|(fa(11), vfa(*1*2),...,vfa*ltn),ge Fn,*n>0; 4. vš*a (P(1,62 ....yty) =* 1|P|(vfalt,), vša(t2 ),...,.va*ln)), P*EP,,n>0 5. VS*A* (L*U*)=-vf*a(U)*; 6. VA*CU* WV)= vš*a(U*) Avfa(V); *7.* V*YAKUV*V)=vš*a(U*) v Vfa(V); 8. Va (UTMV)= vša*(U)*→vfall); 9. vfac(3x)A(x)*) =T* if and only if vfa (A(x)) =*T* for a function *fa'*e [*f*a], 10. vfall(Hx)A(x)) =T if and only if vfa (A(x))=*T* for any function *fa'e* [*f*a]x

First-order Logic

**Definition 2.6.** 1. A first-order formula *U is satisfiable (consistent)* if there exists an

interpretation *1* and an assignment function *fa e As(I*) such that vfa*(U)=T.*

Otherwise, the formula is called *unsatisfiable (inconsistent).* 2. A first-order formula *U is true under the interpretation 1* if for any

assignment function *fa e As(I), vfa (U)=T,* notation: F*, U*, and I is called

*model of U.* 3. A first-order formula *U is false under the interpretation 1* if for any

assignment function *fa e As(I*), VF*O(U)= F*, and *I* is called *anti-model o*f *U.* 4. A first-order formula *U is valid (tautology)* if *U* is true under all the possible

interpretations, notation: =*U.* 5. The first-order formulas *U* and V ar*e logically equivalent* if Vfa*(U*)=Vfa(*V)*

for any interpretation *1* and any assignment function *fa*, notation: *U=V.* 6. A set of first-order formulas S *logically implies* the first-order formula V if

all the models of the set S (the models of the conjunction of all formulas from S) are also models of the formula V. We say that V is a *logical*

*consequence* of the set S, notation: SEV. *7. A set* of first-order formulas is *consistent* if the conjunction of all its formulas

has at least one model. A *set* of formulas i*s inconsistent* if the conjunction of

all its formulas does not have a model.

**Remarks:** 1. The evaluation of a closed formula *U* depends only on the interpretation in

which we want to evaluate it, notation: v' *(U* 2. Any first-order (predicate) formula has an infinite number of interpretations. **Example 2.6.** Build a model and an anti-model for the closed predicate formula:

*U*=(Vx)*(P*(x) VQ(x)) - (Vx*)P*(x) v (Vx*)*Q(x) Let us consider the interpretation 11 *=< Di,m>*, where:

*De* = N (the set of natural numbers) *m(P*):N *{T,F},m(P*)(x)="x:2" *m(Q*):N→*{T,F},m*(Q)(x) ="x:3". V'(*U*) = v'!((Vx)(*P(x) v Q(*x))) → v'!((H*x)P*(x)\(Wx)*Q*(x))

= v'((Vx)*(P(*x) VQ(x))) →v!!(*(Vx)P*(x)) v v'!((Vx*)Q*(x)) = (Vx)ren (x:2 v x:3) → (Vx)ren (x:2) v (Vx) sen(x:3) *= F - FVF= F → F=T.*

44

A Computational Approach to Classical Logics and Circuits

*=T, U* is evaluated as true under the interpretation *I*, which is a model of *U.* Let us consider the interpretation *12 =< D2,m1 >*, where:

*D2* = {4,9} - the domain of interpretation; *m(P)*:{4,9} → *{T,F},m(P*)(x)="x:2" *m(Q*):{4,9} + *{T,F},m(Q*)(x)="x:3". To evaluate the formula *U* under the interpretation *1*2, with the finite domain *D2 =* {4,9}, the universally quantified subformulas are replaced by the conjunction of their instances for x =4 and x=9. *vl?(U*)=v??((Vx)(P(x) *V*Q(x))) → v!?*((Vx)P*(x) v(Vx*)Q*(x))

= yłz ((Hx)*(P(*x)vQ(x))) → v!2 *(*(Wx*)*P(x)) v v'2 ((Vx)*Q*(x)) =(4:2 v 4:3)^(9*:2* v 9:3) +(4:219:2) v(4:319:3) *= (T v F)^(FVT) → (T AF) V(FAT)=T AT → FvF.*

*= T* → *F=F In* evaluates the formula U as false, *I*, is an anti-model of *U* and thus *U i*s *not a valid formula, it is a contingent one (1*, is a model of *U*).

Example 2.7*.* Evaluate the open formula *U(*z) under the interpretations *I*, and *12*

*U(*z) = (3x)Ey*)P(f*(x, y), z) 1. *1*1 *=< D1, mi >,D=*Z (the set of integer numbers),

*mi (*)(x,y) = (x + y)2 and *m (*P)(x, y):"x = y". Because *U(*z) is an open formula, its evaluation depends on the assignment of values (integers) to the free variable z*, fa e As(1*1*),* where:

vy*a (U(z*)) = (3x)xez (Ey) yez "(x + y)2 *= fa(*z)"={

- foc)\_ST, if *fa(z) is a square*

| F, otherwise *U(*z) is a consistent formula, but it is not a true formula under the

interpretation I, therefore *I, is not a model of the formula. 2. 12 =< D2, m2 >, D=*Z (the set of integer numbers),

*m2(f)(a,b) = a* +b and *m2*(*P)(a,b):"a =b"*. V*a(U(*z)) = (3x) xez (By) yez"x + y = *fa*(z)"=*T, Vfa(z)* € Z. The formula *U(z*) is true under the interpretation *1*2 : every integer number *fa*(z) is the sum of two integer numbers x and y. Fi*, U*(z), therefore *12* is a model of *U(*z).

First-order Logic

*2.4. Logical equivalences in predicate logic*

• *Expansion laws* (Vx)A(x)=(Vx)A(x) *^ A(t*) – the universal quantifier is an infinitary conjunction, (3x) A(x) = (3x) A(x) v *Alt*) – the existential quantifier is an infinitary disjunction, where *t* is a term which does not contain x.

•

*DeMorgan's infinitary law*s (9x)A(x) = (Vx-A(x)

(Vx)A(x) = (3x)–A(x)

• *Quantifiers interchanging laws* (9x)Ey) A(x, y) = (Sy)(3x)A(x, y)

(Vx)(Vy)A(x, y) = (Vy) (Vx)A(x, y) **Remark**: (2x)(Vy)*B*(x, y) (Vy)(x*)B*(x, y) Quantifiers of the same type commute, but quantifiers of different type do not commute.

• *The extraction of quantifiers in front of the formula law*s Av (3x*)*B(x) = (3x)(A*v B(*x))

Av (Vx*)B(*x) = (x)(*Av B*(x)) *A*^(2x)B(x) = (3x)( A *B(*x))

*A*^(*Vx)B*(x) =(Vx)(A^ *B*(x)) where A does not contain x as a free variable.

(Ex) A(x) *v B* = (3x)(A(x) *v B*)

(Vx*)A*(x) *v B =* (Vx)(A(x) *v B)* (Ex)A(x) ^ *B* = (3x)(A(x*) AB)*

(Vx)A(x*) A B =*(Vx)(A(x) ^ *B)* where B does not contain x as a free variable.

**• *Distributive laws*** (3x)(A(x) *v B*(x)) = (3x) A(x) v (Ex)*B*(x) distribution of "?" over " *y "* (Vr)(A(x) ^ *B(*x)) = (Vx) A(x) ^ (Vx*)B*(x) distribution of "V" over ””

The above laws are pairs of dual equivalences. "V" and "?" are dual quantifiers. **Remark**: The distribution of

"}” over "A", "over” v”, ”I” over” →", "V" over ” + ” do not provide valid distributive laws. These combinations of quantifiers and connectives express only semi-distributive laws as follows:

Semi-distributivity of "3" over "A":

F (3x)(A(x) *^ B*(x)) -> (3x)A(x)^(3x*)B*(x) and #(3x) A(x) ^ (3x*)B*(x) + (2x)(A(x*) ^ B*(x))

Semi-distributivity of "*W*"over"V":

F(Vx) A(x*) v (Vx)B*(x) → (Vx)(A(x) *v B*(x)) and #(Vx)(A(x*) v B*(x)) + (Wx)A(x) v (*Vx)B*(x))

A Computational Approach to Classical Logics and Circuits

Semi-distributivity of "3" over "-":

F((Ex)A(x) + (2x*)B*(x)) + (2x)( A(x) *+ B(*x)) and

# (Ex)(A(x*) →* B(x)) + ((Ex)A(x) + (2x*)B*(x)) Semi-distributivity of "V" over " +":

F(Vx)(A(x) *→ B*(x)) + ((Vx)A(x) + (Vx*)B*(x)) and *#(*(Vx*)*A(x) + (*x)B*(x)) → (Vx)(A(x) + *B*(x))

**mi**

Example 2.8. Prove that the universal and existential quantifiers do not commute We have to prove that the logical equivalence:

(3x)(Vy*)L*(x, y) = (Vy)(3x)*L*(x, y) does not hold. We choose the interpretation *I =<D,m>*, where: *- D* is the set of all persons in the world *- m(L):DxD →{T,F), m(L*)(x, y) =*"x love*s y'' Under the interpretation *I*, the formula *U,* = (Ex)(Vy)*L*(x, y) has the meaning: *"There exists a person who loves all persons."*

The formula *U2* = (y)(Ex)*L*(x, y) has the meaning: *”All persons are loved by at least one person.*”, under the same interpretation *I.* These two natural language statements are not equivalent, so *U, #U*2, but note that

*U*F*U*2.

Example 2.9. Prove that the formula *U =* (Ex)A(x)^(2x*)B*(x) + (2x)(A(x) ^ *B*(x)) is not valid. Let us consider the interpretation *I =<D,m>*, where *D* is the set of all straight lines belonging to a plan *P.* Let *dep* be a constant object (line) from the domain of interpretation.

*M(A):D →{T,F}, m(A*)(x)="xl*d*";

*m(B):D →{T,F},m(B*)(x)="x || d"; v*'(U)*= v' ((3x)A(x)^(3x)B(x)) →v'((Ex)(A(x) ^ *B*(x))) =

= v'*(*(Ex)A(x))^ v' ((Ex*)B*(x)) →v'((Ex)(A(x) *\ B*(x))) = = (3x)xed(x Id)^(Ex)zed (x||*d*) + (Ex)xD(x1*d* 1x|| d)=

*=TAT → F=T + F=F. U* is evaluated as false under the interpretation *1, so I* is an anti-model of *U.* We conclude that *U is not a valid formula,* but *it is consistent,* having as a model.

First-order Logic

*J=<Di, mi >*, where:

*D*i - the set of all persons living in Cluj-Napoca. *m(A):D →{T,F},m*(A)(x):"x owns a car"; *m(B):D. → {T,F},m; (B*)(x):"x has blue eyes*”;*

'*(U*)=v"((3x)A(x)^(3x)B(x)) +v'(5x)(A(x*)* ^ B(x))) =

= p'((Ex)A(x))^(Ex)*B*(x)) →v" ((Ex)(A(x*) ^ B*(x))) = = (3x) xe D, "x owns a car"^(3x) xeD, "x has blue eyes" →

→ (3x)xed, "x owns a car *and* x has blue eyes" *= T AT →T=T +T=T. U* is evaluated as true under the interpretation *J, J* is a model of *U.*

**owns a car**

Theorem 2.3. [58] **Soundness and completeness theorem st**ates the equivalence between "*logical consequence*" and "s*yntactic consequence*” concepts. Let *U ....U*n-1*,U,V* be first-order formulas. 1. completeness: if U1*,...,Un-1,U*, EV then *U1 ..., Un-U*nEV. 2. soundness: if *U.,*.*..,Un-1,U*, EV then *U1,...,U-1,Un*E*V*.

A particular case of this theorem is the following result: **“A formula is a tautology if and only if it is a theorem in first-order l**ogic.”

**Theorem 2.4. (A. Church, 193**6) [13] The problem of the validity of a first-order formula is *undecidable,* but it is *semi-decidable.* If a procedure *Proc* is used to check the validity of a first-order formula *U* we have the following cases: 1. if *U* is a valid formula, then *Proc* ends with the corresponding answer. 2. if the formula *U* is not valid, then *Pro*c ends with the corresponding answer or

*Proc* may never stop. The above result can be improved for fragments of first-order logic as follows:

**Theorem** 2.5. [17] There is a decision procedure to check the validity of the formulas belonging to the following classes of formulas with the prefix of the prenex form: 1. v\*\*\*:(Hx] ...(Vx,WC5y;...(Jym)*,m,n* 20; 2. v\*v\* :(Vx)...(Vx,W3y)(W21)...(W*zm), m,n* 20; 3. V\*3=v\* :(Vxz)...(Vx,)(3y})(3y2)(U21)...(Uzm)*, m,n*20.

A Computational Approach to Classical Logics and Circuits

The proof methods used in predicate logic:

semantic tableaux method (semantic and refutation); resolution (syntactic and refutation);

sequent/anti-sequent calculi (syntactic and direct);

• definition of deduction - axiomatic system (syntactic and direct)

• Herbrand-based procedure (syntactic and refutation) 2.*5. Normal forms in first-order logic* The normal forms of predicate formulas are used as input data in proof methods such as *resolution* and *Herbrand-based procedure.* **Definition** 2.7*.* A predicate formula *U* is in ***prenex normal form*** if it has the form: *(2*x1)...*(Q*nXn)M, where *Lisi* = 1,.*.., n* are quantifiers, and M is quantifier-free. The sequence *(Q1*x)...(n\*n) is called the *prefix* of the formula *U*, and M is called the *matrix o*f the formula *U. A* predicate formula is in *conjunctive prene*x *normal form* if it is in prenex normal form and the matrix is in CNF. **Theorem** 2.6. [20] A predicate formula admits a logical equivalent conjunctive prenex normal form. The prenex normal form is obtained by applying transformations which preserve the logical equivalence, according to the following algorithm: Step 1: The connectives " +" and” 4” are replaced using the connectives

"–,1,*V”.* Step 2: The bound variables are renamed such that they will be distinct. Step 3: Application of infinitary DeMorgan's laws. Step 4: The extraction of quantifiers in front of the formula laws are applied. Step 5: The matrix is transformed into CNF using DeMorgan's laws and the

distributive laws.

**Remarks:**

After the Step 4 we obtain the prenex normal form which is not unique. If the formula obtained after the second step contains n distinct and independent groups of quantifiers, these groups can be extracted in an arbitrary order, therefore there exist n! prenex normal forms, logically equivalent to the initial formula. A conjunctive prenex normal form is obtained after Step 5.

First-order Logic

**Definition 2.8.** Let *U* be a first-order formula, and *UP = (Q1*x1)...(Qxx*n)M* be one of its conjunctive prenex normal form. 1. A formula in *Skolem normal form,* denoted by *U* corresponds to *U* and it is

obtained as follows: For each existential quantifier *Q,* from the prefix we apply the transformation:

If *Q*, is the left-most universal quantifier in the prefix, then we introduce a new constant *a*, and we replace in M all the occurrences of x, by *a. (Qr*x,) is deleted from the prefix. If *ls, ....,ls*,15S<...<Sm<r, are all the universal quantifiers at the left side of Q, , then we introduce a new *m*-place function symbol, *f,* and we replace in M all occurrences of x, by *f* (x5, ..,\*). (*Qr*x, ) is deleted from the prefix.

The constants and functions used to replace the existential quantified variables are called *Skolem constants* and S*kolem functions.* The prefix of the formula *US* contains only universal quantifiers, and the matrix is in conjunctive normal

form. 2. A formula in *clausal normal form* denoted by *Uo* corresponds to *U* and it is

obtained by deleting the prefix of *US.*

**Remarks:** The transformations used in the Skolemization process do not preserve the logical equivalence but preserve the inconsistency according to the following theorem.

**Theorem** 2.7*.* Let *U1,U2,...,U,V* be first-order formulas. 1. V is inconsistent if and only if *y*p is inconsistent, if and only if *v* is

inconsistent, if and only if yc is inconsistent. 2. The set *{U,U2,...,U*n} is inconsistent if and only if the set *{U^,U2,...,U,9}*

is inconsistent. **Example 2.10.** Transform into prenex normal form and Skolem normal form the formula:

*U=*-((Wx)(P(x) →Q(x)) + ((W*x)P*(x) +(*V*x*)*Q(x)))

50

A Computational Approach to Classical Logics and Circuits

1

2 *U*=-((Vx)(*P*(x) Q(x))+((V*x)P*(x)->(Vx*)*Q(x))) Step 1: replace the innermost " +"connectives, denoted by 1 and 3

*U=*-((x)(\_P(x*) VQ*(x))+((W*x)P(x) v(V*x*)*Q(x))) Step 1: replace ” + ” connective

*U*=(((Vx)(-*P*(x*) V*Q(x))) ((Vx*)P*(x) v(Vx*)*Q(x)) Step 2: rename the bound variables such that they will be distinct

*U*=(((Vx)(-*P*(x) VQ(x))) v ((Wy)*P*(*y) v*(V2*)*Q(z)) Step 3: apply DeMorgan's laws

*U*=-(-((Vx)(-P(x) VQ(x))) ((*Vy)P*(y) v(V2*)*Q(z)) Step 3: apply DeMorgan's laws

*U* = (Vx)(-*P*(x) VQ(x))^-6-(Vy)*P(y) v* (Vz*)Q(*z)) Step 4: extract the quantifiers in front of the formula

*U* = (3x)(Vx)(Vy)((~P(x) v Q(x)*) ^* P(*y)^*-Q*(*z*)) =U = U,” U* = (Wx)(=z)(Wy)<(—P(*x) V*Q(x)*)* ^ *P*(y*)^*-Q(z)*) =U2 = Uz” U* = (Wx)(Vy)(=z)((\_*P*(x*) v Q*(x)*)^ P*(y)^-Q(z)) *= Uz =U3”*

*U ,U2, U*3 are three prenex forms of the initial formula *U.* Because the formula *U* contains 3 distinct and independent bound variables, there exists 3!= 6 prenex normal forms logically equivalent to *U.* Note that the matrix is in CNF, and we will not apply Step 5.

*UP,U2, U*3 are prenex normal forms. After the Skolemization process, applied to the formulas *U;,U2”,Uz'* we obtain:

*U*\* = (Ux)(Wy)(*(*-P(*x) v* Q*(x))* ^ P(y)^-Q(a)), [

z a],

*a* is Skolem constant *U2S* = (4x)(Vy)<(—*P*(x*) v Q*(x)) ^ P(y) ^-*0*1*F*(x))), [z+ f(x)],

*s* is a unary Skolem function *U3*S = (4x)(Wy)(=*P(x) V*Q(x)*) ^* P(y) ^\_*Q(g*(x, y))), [z+ g(x,y)],

*g* is a binary Skolem function The clausal normal forms are obtained by eliminating the universal quantifiers.

*UC* = *(-P*(x) *v Q*(x)*) ^* P(y) ^-Q(*a) U2*° = *(~P*(x) VQ(x)*) ^* P(y) ^ \_Q*(f*(x*)) U*3° = *(–*P(x) VQ(x)*) ^ P*(y) ^\_Q(8(x, y))

First-order Logic

**Example 2.11.** Transform into prenex normal form and Skolem normal form the formula:

*U = (*3x)(Vy)*P*(x, y) v (32)(-Q(2) *v (Vu)(St)R(z,u,t)) UP* = (3x)(Wy)(=z)(Vu)(*31)(P*(x,y) v-Q(z)v *R(2,0,1)) US =* (Wy)(*Vu)(P*(*a*, y) V-*Q(f*(y*)*) *v R(f(y*),*u, g(y,u*))), where:

[xra],[2 + *f*(y)],[t + g(y,*u*)], *a* - Skolem constant,

*fi*g - Skolem functions

*UC = P*(*a*, y) V-*Of(*y)) *v R(f*(y)*,u,* g(y,*u*))

2*.6. Substitutions and unification* In this section we introduce *substitutions* and *unification,* used in predicate resolution (see chapter 5). Definition 2.9. *A substitution* is a mapping from the set of variables *Var* into the set of terms: *TERMS.* We denote by 0=[x] + 1,.*..,*xklk], a substitution, representing a finite set of replacements of variables, where x1,..., \*\* are distinct variables, *1*1*....lk* are terms, such that *Vi* = 1,...,*kit;* # x, and x; is not a subterm of *li. dom(O)* = {x},...,xk] is called the *domain of 8.*

We use Greek letters: *0,8,0,,0,*2 to stand for substitutions. The empty substitution is denoted by *E.*

**Definition 2.10.** The result of applying the substitution 0=(x +4*1, ...*, xket*t*] to a formula is defined recursively as follows: 1. *O*(x;*) = t*;, x*; e dom(0);* 2*. 0(x)*= x, x*Ę dom(0);* 3. O*(C)=C*, C-constant; 4. *of (t..., tx)) = f (O(tj),..., 0(tn)), f e*fni 5. *(p(*t*j...g*i*n)*) *= p(O(t*j)*,..., 0(*t*1)), p*e Pn; 6. *B(-U*)=*=0(U*); *7. O(*

*U V*)*= O(U). A(V*),o € {V,1, +, 6}

is obtained by replacing simultaneously each

Note: An instance *(U) o*f *U* occurrence of x; in U by t*;.*

A Computational Approach to Classical Logics and Circuits

**Definition 2.11.** The *composition* of two substitutions *0,1* = [X] + 11,..., X*K* + tk] and *0*2 = [y1 + $1,..., Yn+s,] is defined as: *0 = 0,02* = [x; *+ 02 (*43)|x*; e dom(Q*), x*;* # 02(t;)]U[y; + sily; *e dom(O2)\dom(*0)]

**Propertie**s: Let *0,0,1,0,2,0*, be substitutions.

• *E0 = Dε = 0; E* - empty substitution;

• *(0203*) *= (0,02)*83 associativity property;

• *0,02 +020*1, the composition of two substitutions is not commutative Example 2.12. We prove that the composition of two substitutions is not commutative

*Q* =[x+ *f(*y), y*t f(a),z*+u], *O2* =[y f g(*a), u +* 2,vt *f(f(a)*)] 24 = *0,02* = [x + *O2(f*(y)), y *2(f(a)),z + O2(u*)]U[

u z,v*t f(f(a*))]= =[x+ *f*(g(a)), y *= f(a*), z = *2,u+* 2,v*t f(f(a)*)]=

= [X + *f(g(a)*), yt *f(a),u*t z,v*t f(f(a)*)] 2x *= 020* =[y-g*(a)*,vt *f(f(a)*),x+ *f*(y),z ] 2012, so the composition of two substitutions is not commutative in general. We compute for *U = p(u*,v,x,y,z) the instances 2 (*U*) and 1*(U)*:

4*(U)= p(z, f(f(a)), f(g(a)), f*(*a)*, z)

*22 (U)= p(u, f(f(a))*, *f*(y), g*(a),u*) Definition 2.1*2*. 1. A substitution *@* is a *unifier* of the terms t, and tz if *(*t*i) = 0(t2*).

The term *e(t*) is called the *co****mmon instan****ce* of the unified terms. 2. *A unifier of a set {U,...,U*,} of formulas is a substitution such that

*(U1) =... = O(Un).*

**Example 2.13.** Let x, y be variables, *a, b* constants, *f*,g function symbols and *P,Q* predicate symbols

• the terms *f*(x,x) and *f(a,b*) cannot be unified because they do not have a

common instance if *a,b* are distinct constants.

P(x) and P*(f*(x*)*) are not unifiable because x is a subterm of f(x). the atoms *P*(x) and *Q(a*) are not unifiable because they do not have the same predicate symbol. the atoms *P*(x, y, z) and *P(a,b*) are not unifiable because the predicate symbols have different arity.

First-order Logic

*t1 = f(x,b), t2 = f(a,* y) have a common instance *f(a,b) = 0(t*1) *= 0(t2*), where 0 =[x 4 *a*, y *b)* is a unifier for these two terms. tj = g(g(x)) and t2 = g(y) have many unifiers:

*M*y = [y+ g(x)], the common instance: g(g(x)) *H2* = (x+ 5, y = g(5)], the common instance: g(g(5)).

*M* is more general than *H2.* **Definition 2.13.** A *most general unifier ( mgu*) is a unifier *H* such that any other unifier *0* can be obtained from *u* by a further substitution 2, *0 = Ma. Algorithm for computing the mgu of two literals [6*2]: input: 1, = P(*t*1, *2*61, ...*,*11,) and *12 = P2(t2,,t2, ...,1*2,*.*) two literals **output*:*** *mgu(11,12*) or the message"*1,12* are not unifiable” **begin**

if (*P P ) //* the predicate symbols are different

**then** write “*l, l,* are not unifiable”; exit; **end\_if** if (*n #k)*

**then** write “*4,12* are not unifiable”; exit; **end\_if** *0:=£; //* initialization with empty substitution while *(O(1) 0(12*))

find the terms corresponding to the outermost function symbols or variables

that are different in 0*(*2*1), 0(12*) and denote them by t; and t2. if (neither one of t, and t2 is a variable or one is a subterm of the other one)

**then w**rite “Z, and l, are not unifiable”; exit; **end\_if** if *(*t, is a variable) *||* 2 is the unifier of the terms t*y, t*y in the current iteration

**then** 2:=[t1 + 2];

**else** i:=*[t*2 + til; **end if**

*0:=02;* if (*O* is not a substitution)

**then w**rite “l, and *l*, are not unifiable”; exit;

**end if**

**end\_while**

**write** “l*,* and la are unifiable and **end**

i*s mgulli, 12)”*

*54*

*A* Computational Approach to Classical Logics and Circuits

**Remark:** The most general unifier of two literals may be not unique: if during an iteration, the terms that must be unified t, and t2 are both variables we can replace t by tz or *1*2 by tj*.*

**Example 2.14.** Find the most general unifier of the literals l, and l2.

*l1 = P(a*,x*, f*(g(y))), *12 = P*(y*, f(z), f(z)*), where x, y, z *e Var, a € Const, f*ige Fi, *Pe* Pz. The algorithm presented before is applied. In each iteration, the terms that are unified are underlined.

*0:= E*

*0(4) = P(a,x, f*(g(y))) and *0(12) = P*(*y, f(z), f*(z)), first iteration:

2:=[y+ a]; *0:=02*=[yta];

0*(L) = P(a*,x*, f*(g(*a*))) and *0(12) = P(a, f(z), f (*z)) second iteration:

2:=[x+ *f*(z)]; *0:=01* =[y+a][x+ f()]=[y+ a,x*+ f(*x)];

*0(*1*1) = P(a, f(z), f*(g(*a*))) and *(12)=P(a, f(z), f (*z)) third iteration:

2:=[z + g*(a*)] *0:=02=* =[yt *a*,x*+ f(*z)][z + g(a)]=[y+ *a*,x+ *f*(g(*a)*), z = g(*a*)] *= mgu(li,1*2) *l,* and *l*are unifiable and their common instance is: 0*(11)=0(12) = psa, f*(g(*a)*), f(g(*a*))).

**Example 2.15.** Check if the literals l, and 12 are unifiable:

*= Q*(*x,a, f(*x, y)) and 1*2 = Q(b, y, f(2,*c)), where x,y,z *e Var, a,b,c e Const, f* e F2, Qe Pz. The algorithm for computing the *mgu* of two literals is applied:

*0:=x* ; *O(l) =* Q(x*,a, f*(x,y)) and *6(12) = Q(b, y, f(*z*,*c));

First-order Logic

first iteration:

2:=[x+ b), unifier of the terms x(variable) and *b* (constant); *0:=02* =[x+*b*];

*0(4)= Q(b,a, f(b*,y)) and *0(12)= Q(b*, y*, f(z,c));* second iteration:

a =[y+ a), unifier of the terms y (variable) and *a* constant) *0:= 82 0:*=[r+b][yt*a*]=[x+*b*, yra],

*O(13) = Q(b,a, f(b,a)*) and *(12) = Q(b,a, f(2,c*)); third iteration:

i =[z*+ b*], unifier of the terms z (variable) and *b(*constant) *0:=02 0:*=[x+ *b,*yt a][z + *b*]=[x*+byt a,z+ b*]

*0(4)= Q(b,a, f(b,a)*) and *0(12)= Q(b,a, f(b,c*)); fourth iteration:

The terms *a* and c, which are distinct constants, are not unifiable, so the terms *f(b,a)* and *f*(*2,*c) corresponding to the third argument of the literals

are not unifiable. The conclusion is that th*e literals land l, are not unifiable.*

*2.7. Herbrand – based procedure Herbrand-based procedure* [12] is a refutation proof method used to solve the decision problem in first-order logic. **Definition** 2.14. 1. A *ground term o*r a *ground atom* are free of variables. 2. A formula is called *ground formula* if it does not contain any variable or quantifier. 3. AS is a *ground instance* of the quantifier-free formula A, if A8 is obtained by replacing all free variables from A by ground terms. **Example 2.16.**

• *a, f(a), g(f(a),b*)-ground terms, where:

- *a,b* - constants, *f,* g - function symbols

• *P(a), Q(f(a),b)* – ground atoms, where:

- *a,b* - constants, *f* - function symbol, *P,*Q - predicate symbols

*A8 = P(a)* ^-*Q(b, f(a)*) is a ground instance of the formula A(x, y, z)= *P*(x)^ 2(y, *f(*z)), [x fo*r a*, y*tb,zt a*].

A Computational Approach to Classical Logics and Circuits

**Definition 2.15. [44]** Let S ={A1, A*2,.*.., An} be a set of first-order quantifier-free formulas. 1. The *Herbrand universe of S,* denoted by *Hs*, is the set of all ground terms

built using the constants and the function symbols from S as follows:

*Hs* =Vizo*H; ,* where: *H =*C = the set of all constants of S;

if *C* =0 then *Ho ={a}, a* is a new constant; *Hi+1= H; V{f(*t... *Ik) | 4*1*,..., tk € Hi, f e Fk; f - function symbol of S}.* 2. The *Herbrand base* of S, denoted by *BHs,* is the set of all ground atoms built

using the predicate symbols of S and the ground terms of *Hs.*

*BHs = {P(t1 ....tk*) 147....*the* HS,*PE*Pk*, P-predicate symbol of S}.* 3. The *Herbrand system* of S, denoted by S*HS*, is the set of all ground instances

of the formulas from S, obtained by replacing the free variables by ground terms of *Hs.*

Example *2.*17*.* 1. S; *= {P(a) v*-Q(x*), R*(y)}

*Hs =* {a} - the Herbrand universe is finite. *BHs, = {P(a),Q(a), R(a*)} - the Herbrand base

*SHs, = {P*(*a) v-Q(a), R(a*)} - the Herbrand system 2. S2 *= {P(f(x*)), -Q(z,y)}

*Hg, = {c, f(c), f(f(*c)),...} ; C - constant, the Herbrand universe is infinite *BHs, = {P{f(c*))*,Q(c,c),P(f(f(c)), Q(*c*, f(c)),Q(f(c),c),Q(f (c), f(*c)),...}

- the Herbrand base is infinite *SHs, = {P(f(*c)),-Q(c*,c), P(f (f (c)),*-Q[c*, f*(c)),-Q*(f(c),c)*,-Q*(f (c), f(c)*),...}

- the Herbrand system is infinite

**Theorem 2.8. Herbrand's theorem** 121 Let S be a set of first-order quantifier-free formulas. S admits a model if and only if the set *SH,* (the Herbrand system of S) admits a model.

**Theorem** 2.9. [12] A set S of first-order quantifier-free formulas is inconsistent *if and only if* the set *SHg* (the Herbrand system of S) is inconsistent *if and only if* there is a finite inconsistent subset of S*Hs.*

First-order Logic

**Remark:**

The problem of checking the inconsistency of a set of first-order formulas was reduced to checking the inconsistency of a set (finite or infinite) of propositional formulas according to:

S inconsistent if and only if the Herbrand system S*HS,* is inconsistent,

where Si = SC, SC is the set of clausal normal forms of the formulas from S.

Herbrand's theorem suggests a refutation proof procedure based on the following theoretical result:

*U ,U2,...,U*, E*V if and only if* S *={U,Už ..., UM,*(-1)} is inconsistent *if and only if SH,* (Herbrand system of S) is inconsistent *if and only if* there is a finite inconsistent subset of S*HS.*

*Herbrand-based algorithm:* **input:** *U,,U2 ,...,U,,V* - first-order formulas **output**: the message *U1,U2,...,U*nF*V* or *U1,U2,...,U*n*HV* **begin**

build S = *{U; ,Uị,...,UM,* (1)} build S*HS; i*f S*Hs* is finite

**then**

if *(*S*H,* is inconsistent)

**then writ**e *“Uj,U2,...,U*, EV "; exit;

**else wri**te “*U1, U2...,U,HV* ”; exit; **end if** else *// SHg* = {21,22,..., Zm..} is infinite

*k*:=1; **while** (Z1 12*2 ^.*.. A Z*k* is consistent) (\*\*)

*k:=*k+1; **end while**

write “*U1, U2,...,UnH V* "; exit; **end\_if end**

A Computational Approach to Classical Logics and Circuits

This is a semi-decision procedure which never stops if the set *SHS* is an infinite consistent set (\*\*). For implementation, the loop "while” must contain a restriction for the iterations number, to avoid the infinite loop. If we exit the loop because the maximum number of iterations was reached, we cannot decide if S*Hs* is inconsistent or not, and thus we cannot decide if *U1,U2,...,Un* EV or *U1,U2,...,U,HV*.

**e**

**ca**

**Example 2.18.** Using the Herbrand-based algorithm check the validity of the predicate formula:

*U* = (Vx)*(P*(x) = Q(x)) + ((Vx)*P*(x) → (Vx)Q(x)). Let us denote A=*4U* and we transform A into clausal normal form.

A*= U= (*(x)*(P*(x) → *Q(*x)) + *((Vx)P(*x) → (Hx*)O*(x))) =

= (Vx)(~*P*(x) *VQ*(x))^-((Vx*)P(*x) + (Vx)Q(x)) = = (Vx)—*P*(x) *V*Q(x))^*(V*x*)P(*x*)*^-(Vx*)*Q(x)) = = (Vx)(-*P*(x) v *Q*(x))^(*Vx)*P(x)^(3x)-Q(x) =

= (Vx)-P(x) VQ(x))^(Vy)*P*(y*)*^(Ez)-Q(z)= We consider two prenex normal forms:

*AP* = (z)(Vx)(Vy)((-P(x) *v Q*(x)*) ^ P*(y)^-Q*(a*))

A = (Vx)(Ez)(Wy)(\_P(x) *VQ*(x)) ^ P(y)^-Q(a)) The corresponding Skolem and clausal forms are as follows:

Ai=(Vx)(Wy)(*(~P*(x*) VQ(*x)*)^ P*(y)^-Q(a)), a - Skolem constant

A = (*-P*(x*) VQ*(x))^ P(y)^-*2(a) A1* =(*V*x)(Wy)(*(*-P(x) *VQ(x))^ P*(y)^-*Q(f*(x))), f- unary Skolem function

A = (-*P*(x) VQ(x)*) ^* P(y)^-*Q(f*(x)) We consider the following two sets of clauses:

*S*i={-*P(*x*) v Q*(x), *P*(y), -Q(*a*)} and

S2 = {-P(x) v *Q(x*), P(y), -Q*(f*(x))} “*U* is valid if and only if S*H s* is inconsistent”, where S is S, or $2.

s are

Si is used in the Herbrand-based algorithm:

*Hs = {a}, SHs, = {C*} = *-P(a) VQ(a),C2 = P(a),*C3 =-Q(a)} The Herbrand universe and Herbrand system are finite.

First-order Logic

We derive the empty clause using propositional resolution (see Chapter 5) as follows:

G ==*P*(a)v (a) C = P(a)

ca = (a) C=-9(a)

*SHS,* is an inconsistent set, therefore A is inconsistent and *U* is a valid formula.

S2 is used in the Herbrand-based algorithm:

*Hs, = {*c, *f(c), f(f(C)*),...} *SHs, ={-P(c) VQ(c), P(c), -2(f(*c)*), -P(f(c)) VQ(f(c)), P(f(c)*),

-*Of(f(c*))),...} The Herbrand universe and Herbrand system are infinite. We consider the finite set of clauses:

W *= {P{f(*c))*,-Q(f(c),—P(f(c)) VQ(f(c*))} which is a subset of *SHs, .* From *W* we can derive o as follows:

C = *-P(f*(c*)) v Q(f*(c)) C2 = *P(f(c*))

C: = Q(f(c) C =-Q(FC)

Thus, W is an inconsistent set, and S*Hs,* is inconsistent because it has an inconsistent subset. The conclusion is that A is an inconsistent formula and *U* is a tautology. If we want to avoid working with an infinite set of clauses, caused by the existence of function symbols, we must choose the Skolem form containing only Skolem constants. **Example 2.19.** Check the consistency/inconsistency of the following set of first-order quantifier free formulas:

S=*{P*(x)→ *Q*(x) *v* R(x)*,* 2(y) → R(y)*, R(a), -P(a*)}. We transform the formulas into clausal forms:

S ={-P(x) v Q(x) v *R*(x), -Q(y) *v R*(y), R*(a*)*, -P(a)} Hs =*{a} - Herbrand universe