AI 2009 - Handout 2
First-Order Logic

Gianluigi Bellin

March 3, 2009

1 Introduction

A *proposition* is an entity that can be true or false; according to the German philosopher Gottlob Frege, it is the *thought* expressed by a sentence. In the propositional calculus we consider *atomic* sentences as units which cannot be further analysed: "*it rains*", "*water freezes at 0 degrees Celsius at the sea level*", "*it snows*", "*there is ice*". There is an atomic sentence which is always false, denoted by $\bot$. Moreover sentences are built from atoms though *logical connectives not*, *and*, *or*, *implies*, according to the grammar

$$A \ := \ p \mid \bot \mid \neg A \mid A_0 \wedge A_1 \mid A_0 \vee A_1 \mid A_0 \rightarrow A_1$$

The only semantic property of such sentences is their truth value. Therefore the *interpretation* of a propositional language on a set **Atoms** of atomic sentences is a *valuation* $\mathcal{V} : \mathbf{Atoms} \rightarrow \{T, F\}$, i.e., a truth-value assignment to the atoms, which is then extended to non-atomic sentences by the familiar truth-tables for propositional connectives. A sentence is *logically valid* (e.g., $A \vee \neg A$) if it has value T for all assignments to the atoms.

A key property of propositional semantics is *compositionality*: in order to know the truth value of a non-atomic formula, we only need to look at the truth values of its subformulas.

1.1 First Order Predicate Calculus

In *First Order Predicate Calculus* a deeper logical analysis of language is presented than in the propositional case, breaking sentences into their components, e.g., *names* and *predicates*, in such a way that *the semantic value of sentences results from the semantic value of these components.* Given a set of names, e.g, **Names** = {*Romeo, Juliet* }, and a set of predicates, e.g., **Predicates** = "{__ *loves* __}", a first-order language $\mathcal{L}$ = (**Names**, **Predicates**) is interpreted by giving a *domain of discourse* D together with a fixed assignment of an element of D to each name and of a relation on D to each predicate. Thus an interpretation for our language $\mathcal{L}$ could be a set D of people, two of whom are called Romeo and Juliet, and such that there is a well-defined loving relation between pairs of elements of D (no ambiguous feelings, please!). Now the *atomic sentence Romeo loves Juliet* is *true in the interpretation* if the two elements of D called Romeo and Juliet are indeed in the loving relation and *false* otherwise.

Notice that the interpretation of a first-order language gives a truth value to a *sentence*, not to a *predicate*: here "__ *loves* __" is a *binary* predicate, "*Romeo loves* __" and "__ *loves Juliet*" are *unary* predicates and *Romeo loves Juliet* is a *closed atomic sentence*: it is closed because the "holes" of the predicate have been filled with names; it is atomic because it contains no logical symbols.

We can present exactly the same ideas in an abbreviated and more mathematical notation as follows. We have a language $\mathcal{L} = (\{R, J\}, \{L(x, y)\})$ and an interpretation $\mathcal{M} = (D, \{R_{\mathcal{M}}, J_{\mathcal{M}}\}, \{L_{\mathcal{M}}\})$ where D is our domain of discourse, $R_{\mathcal{M}}, J_{\mathcal{M}} \in D$ are the individuals named R and J, respectively, and $L_{\mathcal{M}} : D \times D \to \{T, F\}$ is the binary relation on D. With this formalism we clearly see how to find other interpretations of our language: for instance, we can let $D = \mathbf{N}$, the set of non-negative integers, $R_{\mathcal{M}} = 0 = J_{\mathcal{M}}$ [1] and the relation $L_{\mathcal{M}} =$ "$\leq''$.

As in the propositional calculus, a (closed) first-order formula A in the language $\mathcal{L}$ = (**Names**, **Predicates**) is *valid* if for *any* interpretation $\mathcal{M} = (D, \mathbf{Names}_{\mathcal{M}}, \mathbf{Predicates}_{\mathcal{M}})$ is *true in* $\mathcal{M}$.

1.2 Quantifiers

But the real power of the predicate calculus comes from the fact that it allows us to express sentences about *all* the elements or about *some* element of our intended domain without actually *naming* each one of them. Let us use *free variables* instead of "holes" as place-holders, and write *x loves y* instead of __ *loves* __ . Now we can use the expressions "*for all x*" (in symbols $\forall x$) and "*for some x*" ($\exists x$) to *bind* the free occurrences of the variable x that occur in the expression to the right (the *scope* of the quantifier "$\forall x$" and "$\exists x$"). For instance, we can define new unary predicates such as *for all x, x loves y*,

[1]We do not need to assume that R and J denote different individuals: in the language we do not have an equality predicate to say whether or not $R_{\mathcal{M}} = J_{\mathcal{M}}$.

for some x, x loves y, *for all y, x loves y*, *for some y, x loves y* and then the sentences

1.	*for all x, for all y, x loves y* ;	(*everybody loves everybody*)
2.	*for some x, for some y, x loves y*;	(*somebody loves somebody*)
3.	*for all x, for some y, x loves y*;	(?)
4.	*for all y, for some x, x loves y*;	(*everybody is loved by someone*)
5.	*for some x, for all y, x loves y*;	(?)
6.	*for some y, for all x, x loves y.*	(?)

Moreover, "*for all y, Romeo loves y*" is also a sentence, as the quantifier "*for all y*" binds the only free variable in the unary predicate "*Romeo loves y*".

Notice that the meaning of the expression "*for all y, Romeo loves y*" does not change if we say "*for all z, Romeo loves z*", i.e., $\forall y.L(R, y)$ and $\forall z.L(R, y)[z/y]$ have the same meaning. This holds because

- we replace the *bound variable* y with the variable z *both in the quantifying expression* "$\forall y$" *and in its scope* "$L(R, y)$", and also because
- z *does not already occur free in* $L(R, y)$.

Indeed, substituting the bound variable y with z in the predicate $\forall y.L(z, y)$ yields the sentence $\forall z.L(z, z)$ (*everybody loves oneself*), which has certainly a different meaning from that of the property "*z loves everybody*"!

1.2.1 Restricted Quantifiers

In natural language, quantifiers usually come with a specification of their *intended domain*. For instance, in the case considered above the itended domain of the quantifiers is a set of *people*. But we may talk with different intended domains: *every person loves some cat* refers to people and to cats. How shall we formalize this? Writing $P(x)$ for *x is a person* and $C(y)$ for *y is a cat*, we have:

$$\forall x.P(x) \to \exists y.C(y) \land L(x, y).$$

The general rule is that we restrict a universal quantifier using an *implication* and an existential quantifier using a *conjunction*. Indeed if we restrict a universal quantifier by a *conjunction* we put too strong a condition: indeed

$$\forall x.P(x) \land \exists y.C(y) \land L(x, y)$$

means that everything is a person, including cats. If we restrict an existential quantifier with an implication, the condition is too weak: the sentence

$$\forall x.P(x) \to \exists y.(C(y) \to L(x, y))$$

is true even if there are no cats in the domain of discourse, which is not what is meant by "*every person loves some cat*".

1.2.2 Compositionality

There is a difficulty in the semantics of quantifiers[2]: in the propositional case, the truth values of a complex formula depend only on its *subformulas*. But what are the subformulas of a quantified expression $\forall x.A(x)$? Given a domain of discourse D and an interpretation $\mathcal{M} = (D, A_{\mathcal{M}})$, we need to give a propositional value to the expression $A(x)$, assuming that the variable x ranges over the domain of discourse D; before we regarded a variable as a *place-holder*, now we regard it as a "*variable name*".

One way to clarify this is to define assignments of elements of D to the variables $\sigma : \mathbf{Vars} \to D$. In this approach we say that $\forall x.A(x)$ is true in the given interpretation $\mathcal{M}$ with the assignment σ if *for all assignments* σ' that differ from σ only for the value given to the variable x, we have that $A(x)$ is true in $\mathcal{M}$ with the assignment σ'. This works, but it is a bit strange to explain what it means to be true *for all elements of the domain* by quantifying over *all functions* assigning elements of D to variables!

Another approach is to extend our language $\mathcal{L}$ with a set A_D of "temporary names" a_d (or *parameters*), one a_d for each element d of the domain of discourse D. Then we may say that $\forall x.A(x)$ is true in the given interpretation $\mathcal{M}$ if for all $a \in P_D$ we have that $A[a/x]$ is true in $\mathcal{M}$.The requirement of compositionality is satisfied as we assume that for each $a \in P_D$ the expression $A[a/x]$ is a subformula of $\forall x.A(x)$. We follow this approach in the following sections.

2 First Order Predicate Calculus: syntax and semantics

A First Order Language $\mathcal{L} = (\mathbf{Pred}, \mathbf{Constants})$ has a sequence $\mathbf{Pred} = \{P_1^{n_1}(x_1, \ldots, x_{n_1}), \ldots P_k^{n_k}(x_1, \ldots, x_{n_k})\}$ of predicate symbols and a sequence $\mathbf{Constants} = \{c_1, \ldots, c_m\}$. Every first order language has an infinite set of free variables v_0, v_1, ... v_i, ... A *term* t is either a *free variable* v_i (an *open term*) or a *constant* c_j (a *closed term*).

Let P^n be a n-places predicate letter. Formulas are defined by the following grammar:

$$A \ := \ P^n(t_1, \ldots, t_n) \mid \bot \mid \neg A \mid A_0 \wedge A_1 \mid A_0 \vee A_1 \mid A_0 \to A_1 \mid \forall x.A \mid \exists x.A$$

The **free variables** $\mathcal{FV}(A)$ of a formula A are defined by induction on the definition of a formula: the free variables of *atomic formula* $P^n(t_1, \ldots, t_n)$ are

[2]The considerations in this section can be omitted in a first reading.

the free variables that occur in the terms t_i. For instance, $\mathcal{FV}(P^2(c, y)) = \{y\}$. Then

- $\mathcal{FV}(\bot) = \{\ \}$;
- $\mathcal{FV}(A_0 \wedge A_1) = \mathcal{FV}(A_0 \vee A_1) = \mathcal{FV}(A_0 \to A_1) = \mathcal{FV}(A_0) \cup \mathcal{FV}(A_1)$;
- $\mathcal{FV}(\forall x.A) = \mathcal{FV}(\exists x.A) = \mathcal{FV}(A) \smallsetminus \{x\}$.

A formula that has no free variables is *closed.*

Example: Let $\mathcal{L}$ have **Pred** = ($\{L^2\}$ and **Constants** = $\{R, J\}$. Here $L^2(R, J) \wedge \forall x.\neg L(R, x) \to \neg L(J, x)$ is a closed formula, which we may regard as the formalization of the sentence "*Romeo loves Juliet and Romeo dislikes only those who are disliked by Juliet.* The first conjunct is closed because the two "holes" (arguments) of the predicate $L^2(x, y)$ are filled with two constants, namely R, J; the second conjunct is closed because the variable x in $\neg L(R, x) \to \neg L(J, x)$ is bound by the quantifier $\forall x$.

2.1 Substitution

We indicate the result of substituting a term t for a free variable x in a formula $A(x)$ by $A[t/x]$. Here it is understood that no free variable of t become bound in the substitution: this is always possible if we rename the bound variables occurring in A with *new* symbols for variables, i.e., different from all those already occurring in A or in t. Indeed, replacing the term y for the variable z in $\forall y.L(z, y)$ would give an expression with a completely different meaning! Thus when we write $(\forall y.L(z, y))[y/z]$, we actually mean the following process: *first* we rename the bound variable y with a new one, say x, and obtain $\forall x.L(z, x)$ and *then* we replace y for z in the resulting expression, obtaining $\forall x.L(y, x)$, which has the meaning intended by the expression $(\forall y.L(z, y))[y/z]$.

Consider the example "*for all y, Romeo loves y*". This is a *closed formula*, thus a *sentence*, the result of substituting the name R for the variable x in "*for all y, x loves y*". Such a substitution is possible because R is a closed term and cannot be "captured" by the quantifier "*for all y*". In symbols we write $(\forall y.L(x, y))[R/x] = \forall y.L(x, y)[R/x] = \forall y.L(R, y)$.

2.2 Semantics

Definition. Given a first order language $\mathcal{L}$ = (**Pred**, **Names**), where **Pred** = $\{P_1^{n_1}, \ldots P_k^{n_k}\}$ and **Constants** = $\{c_1, \ldots, c_m\}$, an *interpretation* $\mathcal{M}$ =

$(D, \mathbf{Pred}_{\mathcal{M}}, \mathbf{Constants}_{\mathcal{M}})$ of $\mathcal{L}$ consists of a nonempty domain D, of a sequence of relations $\{(P_1^{n_1})_{\mathcal{M}}, \ldots, (P_k^{n_k})_{\mathcal{M}}\}$ among the elements of D and of a sequence $\{(c_1)_{\mathcal{M}}, \ldots, (c_m)_{\mathcal{M}}\}$ of elements of D, that are assigned to the predicates and of the constants of the language $\mathcal{L}$ as indicated. (End of definition.)

We are interested only in the interpretation of *sentences*, i.e., of closed formulas. Consider the sentence "*Romeo loves everybody*"; we look for an interpretation $\mathcal{M}$ of the language $\mathcal{L}$ with $\mathbf{Pred} = \{L^2\}$ and $\mathbf{Constants} = \{R, J\}$. Given *any* domain D, we know how to interpret the predicate "L^2", as a relation $L^2_{\mathcal{L}} : D \times D \to \{T, F\}$, and we know how to interpret the constants "R" and "G", as individuals $R_{\mathcal{M}}$ and $G_{\mathcal{M}}$ of the domain D. To interpret "$\forall x.L^2(R, x)$" we need to say that for all individuals $d \in D$, the pair $\langle R_{\mathcal{M}}, d\rangle$ belongs to the relation $L^2_{\mathcal{M}}$ that interprets L^2. If we had in our formal language a name a_d for each element $d \in D$, then we could say that $\forall x.L^2(R, x)$ is true in $\mathcal{M}$ if and only if for all individuals $d \in D$, the atomic sentence $L^2(R, a_d)$ is true in $\mathcal{M}$; in other words, we could consider only *closed atomic formulas* in the basic case. This leads us to the following definition.

Definition. Given a first order language $\mathcal{L} = (\mathbf{Pred}, \mathbf{Constants})$ and an interpretation $\mathcal{M} = (D, \mathbf{Pred}_{\mathcal{M}}, \mathbf{Constants}_{\mathcal{M}})$ of $\mathcal{L}$ with domain of discourse D, extend the language $\mathcal{L}$ with a set of *names* (or *parameters*) $\mathcal{A} = \{a_d \mid d \in D\}$, one for each element of the domain D. Let A be a formula of $\mathcal{L} + \mathcal{A}$ containing only constants and parameters but not free variables; we define what it means for such a formula to be *true in* $\mathcal{M}$ (in symbols, $\mathcal{M} \models A$) by induction on the definition of A:

(i) if $A = P^n(t_1, \ldots, t_n)$ is a *atomic formula* then

$$\mathcal{M} \models A \quad \text{iff} \quad \langle (t_1)_{\mathcal{M}}, \ldots, (t_n)_{\mathcal{M}}\rangle \in P^n_{\mathcal{M}};$$

(ii) $\mathcal{M} \models (\neg B)$ iff **not** $\mathcal{M} \models B$;

(iii) $\mathcal{M} \models (B \wedge C)$ iff $\mathcal{M} \models B$ **and** $\mathcal{M} \models C$;

(iv) $\mathcal{M} \models (B \vee C)$ iff $\mathcal{M} \models B$ **or** $\mathcal{M} \models C$;

(v) $\mathcal{M} \models (B \to C)$ iff $\mathcal{M} \models B$ **implies** $\mathcal{M} \models C$;

(vi) $\mathcal{M} \models (\exists v_i.B)$ iff **there exist** a $d \in D$ such that $\mathcal{M} \models B[a_d/v_i]$;

$\mathcal{M} \models (\forall v_i.B)$ iff **for all** $d \in D$, $\mathcal{M} \models B[a_d/v_i]$.

A *closed formula* A of a language $\mathcal{L}$ is *logically valid* if it is true in all interpretations $\mathcal{M}$ of $\mathcal{L}$, i.e., for all (appropriate) $\mathcal{M}$, $\mathcal{M} \models A$. (End of definition.)

3 Proving in First Order Logic

We consider here proof-systems for First Order Logic. We start from an extension of the Semantic Tableaux procedure for the Propositional Calculus and show that it is sound and complete for the semantics of First Order Logic. Equivalent methods are Natural Deduction systems and the Resolution Method, which is at the foundations of the programming language `PROLOG`.

3.1 Semantic Tableaux Procedure

We extend the *"semantic tableaux"* procedure to classical *first-order predicate* logic: as in the propositional case given two sets Γ, Δ of *closed formulas* (first order *sentences*) in the language $\mathcal{L} = (\mathbf{Pred}, \mathbf{Names})$, the procedure inverts the logical rules of Gentzen's sequent calculus in such a way that every formula in Γ, Δ is eventually analyzed. But in the first order case we have *three* possibilities:

1. the procedure stops, yielding a *closed tree* that represents a derivation of $\Gamma \Rightarrow \Delta$ in Gentzen's sequent calculus if $(\bigwedge \Gamma) \to (\bigvee \Delta)$ is *logically valid*;

2. the procedure finds a *finite open branch* in the tree and stops, from which a *finite countermodel* $\mathcal{M}$ can be extracted such that $\mathcal{M} \models (\bigwedge \Gamma)$ and $\mathcal{M} \not\models (\bigvee \Delta)$

3. the procedure does not stop.

In the third case there is no way for a mechanical procedure to produce a countermodel of $(\bigwedge \Gamma) \to (\bigvee \Delta)$[3] but we can still realize, with some intelligence, how to construct a possibly infinite countermodel $\mathcal{M}$.

Definition. (*sequent*) (i) Let $\Gamma = C_1, \ldots, C_m$ and $\Delta = D_1, \ldots, D_n$ finite sets of formulas. A formal expression $S = \Gamma \Rightarrow \Delta$ is called a *sequent*; Γ is the *antecedent* and Δ the *succedent* of the sequent S. In the semantic tableaux procedure we shall use sequents of the form

$$\overline{\mathbf{a}}; C_1, \ldots, C_m \Rightarrow D_1, \ldots, D_n$$

where $\overline{\mathbf{a}} = c_0, \ldots, c_k, a_1, \ldots, a_k$ is a list of all constants and parameters occurring in $C_1, \ldots, C_m, D_1, \ldots, D_n$. (End of definition.)

[3]This can be proved with techniques studied in Computability Theory.

axiom:
$A, \Gamma \Rightarrow \Delta, A$

$\bot$-*axiom:*
$\bot, \Gamma \Rightarrow \Delta$

logical rules

$$\frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}\ \neg\text{-R} \qquad \frac{\Gamma \Rightarrow A, \Delta}{\Gamma, \neg A \Rightarrow \Delta}\ \neg\text{-L}$$

$$\frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow\ A \wedge B, \Delta}\ \wedge\text{-R:} \qquad \frac{\Gamma, A, B \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta}\ \wedge\text{-L}$$

$$\frac{\Gamma, A \Rightarrow \Delta, B}{\Gamma \Rightarrow A \to B, \Delta}\ \to\text{-R} \qquad \frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma, B \Rightarrow \Delta}{A \to B, \Gamma \Rightarrow \Delta}\ \to\text{-L}$$

$$\frac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow\ A \vee B, \Delta}\ \vee\text{-R} \qquad \frac{A, \Gamma \Rightarrow \Delta \qquad B, \Gamma \Rightarrow \Delta}{A \vee B, \Gamma \Rightarrow \Delta}\ \vee\text{-L}$$

$$\frac{\Gamma \Rightarrow \Delta, A[a/x]}{\Gamma \Rightarrow\ \forall x.A, \Delta}\ \forall\text{-R} \qquad \frac{\Gamma, A[t/x], \forall x.A\ \Rightarrow \Delta}{\forall x.A, \Gamma \Rightarrow \Delta}\ \forall\text{-L}$$

where a is not free in Γ, Δ;

$$\frac{\Gamma \Rightarrow \Delta, A[t/x], \exists x.A}{\Gamma \Rightarrow\ \exists x.A, \Delta}\ \exists\text{-R} \qquad \frac{\Gamma, A[a/x]\ \Rightarrow \Delta}{\exists x.A, \Gamma \Rightarrow \Delta}\ \exists\text{-L}$$

where a is not free in Γ, Δ.

Table 1: First Order Classical Sequent Calculus

Table 1 presents the rules of the sequent calculus for first order logic in the most general form. Notice that in the rules $\forall$-L and $\exists$-R it is required that the parameter a occurs free only in the formula $A[a/x]$ that is being quantified over, i.e., that a does not occur in the conclusion of the inference.

In the "*Semantic Tableaux*" procedure, however, it is better to give sequent rules for $\forall$-L and $\exists$-R an equivalent, more algorithmic notation, as in Table 2. We consider here only the rules for implication, the other propositional rules being analogous, and the rules for the universal and existential quantifiers.

axiom:

$$\overline{\mathbf{a}}; A, \Gamma \Rightarrow \Delta, A$$

$\bot$-*axiom:*

$$\overline{\mathbf{a}}; \bot, \Gamma \Rightarrow \Delta$$

$$\frac{\overline{\mathbf{a}}; \Gamma, A \Rightarrow \Delta, B}{\overline{\mathbf{a}}; \Gamma \Rightarrow A \to B, \Delta}\ \to\text{-R} \qquad \frac{\overline{\mathbf{a}}; \Gamma \Rightarrow \Delta, A \qquad \overline{\mathbf{a}}; \Gamma, B \Rightarrow \Delta}{\overline{\mathbf{a}}; A \to B, \Gamma \Rightarrow \Delta}\ \to\text{-L}$$

$$\frac{\overline{\mathbf{a}}, a_{i+1}; \Gamma \Rightarrow \Delta, A(a_{i+1})}{\overline{\mathbf{a}}; \Gamma \Rightarrow \forall x.A, \Delta}\ \forall\text{-R} \qquad \frac{\overline{\mathbf{a}}; \Gamma, A(a_0), \ldots, A(a_i), \forall x.A \Rightarrow \Delta}{\overline{\mathbf{a}}; \forall x.A, \Gamma \Rightarrow \Delta}\ \forall\text{-L}$$

where $\overline{\mathbf{a}} = a_0, \ldots .a_i$;

$$\frac{\overline{\mathbf{a}}; \Gamma \Rightarrow \Delta, A(a_0), \ldots, A(a_i), \exists x.A}{\overline{\mathbf{a}}; \Gamma \Rightarrow \exists x.A, \Delta}\ \exists\text{-R} \qquad \frac{\overline{\mathbf{a}}, a_{i+1}; \Gamma, A(a_{i+1}) \Rightarrow \Delta}{\overline{\mathbf{a}}; \exists x.A, \Gamma \Rightarrow \Delta}\ \exists\text{-L}$$

where $\overline{\mathbf{a}} = a_0, \ldots .a_i$.

Table 2: Semantic Tableaux procedure, modified rules (Breadth-first search).

Recall the motivations of the "semantic tableaux" procedure: we try to *falsify* a sequent

$$S: \qquad \overline{\mathbf{a}}; \Gamma \Rightarrow \Delta.$$

This means that we try to find an interpretation $\mathcal{M}$ such that the *antecedent* Γ is true in $\mathcal{M}$ and the *succedent* Δ is false in $\mathcal{M}$: such an interpretation is called a *countermodel.* We proceed *from bottom up* by inverting the rules of the sequent calculus.

Thus in the case of the rule for implication to the right ($\to$-R) $A \to B$ occurs on the right hand side of the $\Rightarrow$ symbol (in the *succedent*) of the sequent-conclusion and we are trying to make $A \to B$ *false*: this means that we have to make A *true* and B *false* at the same time; therefore in the sequent-premise of the rule we write A to the left (in the *antecedent*) and B to the right (in the *succedent*) of $\Rightarrow$.

In the case of the rule for implication to the left ($\to$-L) $A \to B$ occurs in the *antecedent* of the sequent-conclusion and we are trying to make $A \to B$ *true*: this can be done *in two ways*, one, by making A *false* (no matter what truth value B may take) second, by making B *true* (whatever value A may take). Thus the procedure branches and we have a rule with *two* sequent-premises: on the left sequent-premise we write A in the *succedent* and on the right sequent-premise B in the *antecedent.*

However, in first order logic, to specify an interpretation means first of all to find a set of individuals D, our domain of discourse, which must be non-empty. Thus we regard the constants c_j and the parameters a_i occurring free in $S : \Gamma \Rightarrow \Delta$ as *names* for the individuals of our potential domain D. Since D must be nonempty, if there are no constants in S, then we must put at least one parameter a_0 in $\overline{\mathbf{a}}$.

On one hand, in the case of the rule for universal quantification to the right ($\forall$-R) a formula $\forall x.A$ occurs in the *succedent* of the sequent-conclusion and we are trying to make $\forall x.A$ *false*: this means that there must be an element d in the domain D, denoted by a parameter a_d, such that $A[a_d/x]$ is *false*, and indeed the existence of such a d will suffice to make $\forall x.A$ false. In the sequent a finite set of constants and names occurs, which are collected in the sequence $\overline{\mathbf{a}} = c_0, \ldots, c_k, a_0, \ldots, a_i$. But we have no reason to assume that the particular d making $A[a_d/x]$ *false* is the same as any of the elements named in $\overline{\mathbf{a}}$. Therefore we introduce a *new parameter* a_{i+1} in $\overline{\mathbf{a}}$ and in the succedent of the sequent-premise we replace $\forall x.A$ with $A[a_{i+1}/x]$.

Similarly, in the case of the rule for existential quantification to the left ($\exists$-L) we want to verify a formula $\exists x.A$ in the antecedent of the sequent-conclusion: for the same reasons as above we must introduce a new parameter a_{i+1} to the list $\overline{\mathbf{a}}$ and replace $\exists x.A$ with $A[a_{i+1}/x]$ in the antecedent of the sequent-premise.

On the other hand, in the case of the rule for universal quantification to the left ($\forall$-L) a formula $\forall x.A$ occurs in the *antecedent* of the sequent-conclusion and we are trying to make $\forall x.A$ *true*: since for all d in the domain D and for all terms t denoting d, $A[t/x]$ must be *true*, we must have $A[c_j/x]$ and $A[a_i/x]$ true for all $c_j, a_i \in \overline{\mathbf{a}}$, i.e., the property denoted by A must be true of all the individuals we have named so far; but, moreover, in must be true also of *all the individuals we shall name in the future*, if our procedure forces us to introduce new parameters at a $\forall$-R or $\exists$-L stage. Therefore we shall *add* $A[c_j/x]$ *and* $A[a_k/x]$ *for all* c_j *and* $a_k \in \overline{\mathbf{a}}$ *to the antecedent of the sequent premise, and in it we do not erase* $\forall x.A$, for future use[4]

Similarly, in an $\exists$-R inference a formula $\exists x.A$ occurs in the *succedent* of the sequent-conclusion and must be *false*: thus we must make $A[c_j/x]$ and $A[a_k/x]$ false for all c_j, a_i considered so far, and also for the names a_ℓ that shall be considered in the future. Therefore we add $A[c_j/x]$ and $A[a_k/x]$ for

[4]It is clear that we could improve the procedure by requiring that $A[c_j/x]$ and $A[a_k/x]$ are added only for those c_j and a_k which have not been substituted earlier in the procedure when analyzing $\forall x.A$.

all c_j and $a_k \in \overline{\mathbf{a}}$ to the succedent of the sequent premise without erasing $\exists x.A$.

Suppose at some stage we find that the same formula A occurs *in the antecedent and in the succedent* of a sequent

$$\overline{\mathbf{a}}; C_1, \ldots, C_m, A \Rightarrow D_1, \ldots, D_n, A.$$

This means that the search for a countermodel is over, at least for the subroutine of the semantic tableaux procedure that was running along the branch under consideration. This means also that the parameters a_i in $\overline{\mathbf{a}}$ cannot be regarded as denoting individuals of a countermodel: *thus if all branches in the procedure are closed and we have a proof-tree in the sequent calculus, the parameters are simply a particular kind of free variables*, which do not appear in the closed formulas in the sequent-root of the derivation.

Example 1. Consider the sequent S : $a_0; B \to \forall xA \Rightarrow \forall x.B \to A$. The procedure starts by inverting a $\to$-L rule, trying to verify $B \to \forall xA$: on the left branch we try to falsify both B and $\forall x.B \to A$ and thus invert a $\forall$-R, creating a new parameter a and replacing $B \to A[a/x]$ for $\forall x.B \to A$ in the succedent; however, after inverting a $\to$-R, we find a B on both sides in the sequent-premise and the search ends in this branch.

On the right branch we have the sequent $a_0; \forall xA \Rightarrow \forall x.B \to A$ and we try to falsify $\forall x.B \to A$: we invert a $\forall$-R creating the new parameter a_1 yielding $B \to A[a_1/x]$ in the succedent of the sequent-premise. The substitution $A[a_1/x]$ required by the application of a $\forall$-L and an inversion of $\to$-R yield $A[a_1/x]$ in both sides of the sequent-premise. Thus the procedure terminates, and the sequent S is valid – and proved by the derivation tree under consideration.

$$\dfrac{\dfrac{\dfrac{a_0, a; B \;\Rightarrow\; B, A[a/x]}{a_0, a; \;\Rightarrow\; B, \mathbf{B} \to \mathbf{A}[\mathbf{a}/\mathbf{x}]}\;\to\text{-R}}{a_0; \;\Rightarrow\; \forall \mathbf{x}.\mathbf{B} \to \mathbf{A}, B}\;\forall\text{-R} \qquad \dfrac{\dfrac{\dfrac{a_0, a_1; A[a_1/x], \forall x.A, B \;\Rightarrow\; A[a_1/x]}{a_0, a_1; A[a_1/x], \forall x.A \;\Rightarrow\; \mathbf{B} \to \mathbf{A}[\mathbf{a_1}/\mathbf{x}]}\;\to\text{-R}}{a_0, a_1; \forall \mathbf{x}.\mathbf{A} \;\Rightarrow\; B \to A[a_1/x]}\;\forall\text{-L}}{a_0; \forall x.A \;\Rightarrow\; \forall \mathbf{x}.\mathbf{B} \to \mathbf{A}}\;\forall\text{-R}}{a_0; \mathbf{B} \to \forall \mathbf{x}\mathbf{A} \Rightarrow \forall x.(B \to A)}\;\to\text{-L}$$

(End of Example 1.)

Suppose now that the procedure stops at some branch with an *open* sequent

$$\overline{\mathbf{a}};\ \overline{\mathbf{p}},\ \overline{\forall x.A}\ \Rightarrow \overline{\mathbf{q}},\ \overline{\exists y.B}$$

where we have $p_i \neq q_j$ for all atomic formulas $p_i \in \overline{\mathbf{p}}$ and all $q_j \in \overline{\mathbf{q}}$ and, moreover, all non-atomic formulas in $\overline{\forall x.A}$ and in $\overline{\exists y.B}$ have already been instantiated with all the terms in $\overline{\mathbf{a}}$. Then we can construct a countermodel by using the *terms* in $\overline{\mathbf{a}}$ as the individuals of D and by defining the interpretation of the predicate letters through the atoms in $\overline{\mathbf{p}}$, as it can be seen in the following example.

Example 2. Let $L(x, y)$ and $x = y$ be the only predicate letter, let R and J be the only constants of our language, and consider the sequent S

$$R, J; L(J, R), \exists y.\neg L(y, R), \forall x.L(x, J) \Rightarrow \forall z.L(z, R) \to z = J.$$

expressing the sentence "*if Juliet loves Romeo, Romeo has ennemies and everybody loves Juliet, then only Juliet loves Romeo*".

If we apply the procedure to S, it terminates: since the quantifiers in the left formula $\exists y.\neg L(y, R)$ [and in the right formula $\forall z.L(z, R) \to z = J$] do not occur under the scope of an universal [an existential] quantifier, the generation of new names stops after one step. In more detail, in order to make $\exists y.\neg L(y, R)$ *true*, we name an individual a_1 as Romeo's ennemy; in order to make $\forall z.L(z, R) \to z = J$ *false*, we name another individual a_2 who loves Romeo without identifying a_2 with Juliet. Thus now we have a situation in which *Juliet loves Romeo, Romeo has ennemies and everybody loves Juliet, but there is another individual that loves Romeo and is different from Juliet* which contradicts the sentence in question.

The formal countermodel is built by taking $\mathcal{M} = (D, L_{\mathcal{M}}, =_{\mathcal{M}})$ where $D = \{\{R, J, a_1, a_2\}$ and $L_{\mathcal{M}}$ satisfies $\overline{L(t, J)}$, $L(a_2, R)$ and $L(J, R)$. No identification is made in $\mathcal{M}$ between different names.[5]

$$\begin{array}{cl}
R, J, a_1, a_2; \overline{L(t, J)}, \forall x.L(x, J), L(a_2, R), L(J, R) \Rightarrow a_2 = J, L(a_1, R) & \\
\hline
R, J, a_1, a_2; \neg\mathbf{L}(\mathbf{a_1}, \mathbf{R}), \overline{L(t, J)}, \forall x.L(x, J), L(a_2, R), L(J, R) \Rightarrow \ a_2 = J & \neg\text{-L} \\
\hline
R, J, a_1, a_2; \neg L(a_1, R), \overline{L(t, J)}, \forall x.L(x, J), L(J, R) \ \Rightarrow \mathbf{L}(\mathbf{a_2}, \mathbf{R}) \to \mathbf{a_2} = \mathbf{J} & \to\text{-R} \\
\hline
R, J, a_1, a_2; \forall \mathbf{x}.\mathbf{L}(\mathbf{x}, \mathbf{J}), \neg L(a_1, R), L(J, R) \Rightarrow L(a_2, R) \to a_2 = J & \forall\text{-L} \\
\hline
R, J, a_1; \forall x.L(x, J), \neg L(a_1, R), L(J, R) \Rightarrow \forall \mathbf{z}.\mathbf{L}(\mathbf{z}, \mathbf{R}) \to \mathbf{z} = \mathbf{J} & \forall\text{-R} \\
\hline
R, J; \exists \mathbf{y}.\neg\mathbf{L}(\mathbf{y}, \mathbf{R}), \forall x.L(x, J), L(J, R), \Rightarrow \forall z.L(z, R) \to z = J & \exists\text{-L}
\end{array}$$

Finally, suppose that the procedure does not terminate on one branch. It should be clear now that for this to happen an expression of the form $\forall x.\exists y.A(x, y)$ must occur in the antecedent [or an expression $\exists x.\forall y.A(x, y)$ must occur in the succedent] of some sequent in the branch and also that

[5]We use the abbreviation $\overline{L(t, J)}$ for the sequence $L(R, J)$, $L(J, J)$, $L(a_1, J)$, $L(a_2, J)$ (i.e., the result of substituting each member of the list (R, J, a_1, a_2) for t in $L(t, J)$.

such an expression must be recurrently evaluated, so that new parameters will be periodically introduced to substitute y in $A(x, y)$.

Example 3. An example of this situation is given by the sequent

$$a_0; \forall x.\exists y.A(x,y) \;\Rightarrow\; \exists y.\forall x.A(x,y)$$

$$\begin{array}{cl}
\vdots & \\
a_0, a_1, a_2 a_3, a_4, a_5, a_6; \forall x.\exists y.A(x,y), A(a_1,a_3), A(a_2,a_4), A(a_0,a_1) \Rightarrow & \\
\Rightarrow \exists y.\forall x.A(x,y), A(a_5,a_1), A(a_6,a_2), A(a_2,a_0) & \\
\hline
a_0, a_1, a_2, a_3, a_4; \forall x.\exists y.A(x,y), A(a_1,a_3), A(a_2,a_4), A(a_0,a_1) \Rightarrow & \forall\text{-R twice} \\
\Rightarrow \forall \mathbf{x}.\mathbf{A}(\mathbf{x},\mathbf{a_1}), \forall \mathbf{x}.\mathbf{A}(\mathbf{x},\mathbf{a_2}), \exists y.\forall x.A(x,y), A(a_2,a_0) & \\
\hline
a_0, a_1, a_2; \exists \mathbf{y}.\mathbf{A}(\mathbf{a_1},\mathbf{y}), \exists \mathbf{y}.\mathbf{A}(\mathbf{a_2},\mathbf{y}), \forall x.\exists y.A(x,y), A(a_0,a_1) \Rightarrow & \exists\text{-L twice} \\
\Rightarrow \forall x.A(x,a_1), \forall x.A(x,a_2), \exists y.\forall x.A(x,y), A(a_2,a_0) & \\
\hline
a_0, a_1, a_2; \exists y.A(a_1,y), \exists y.A(a_2,y), \forall x.\exists y.A(x,y), A(a_0,a_1) \Rightarrow \exists \mathbf{y}.\forall \mathbf{x}.\mathbf{A}(\mathbf{x},\mathbf{y}), A(a_2,a_0) & \exists\text{-R} \\
\hline
a_0, a_1, a_2; \forall \mathbf{x}.\exists \mathbf{y}.\mathbf{A}(\mathbf{x},\mathbf{y}), A(a_0,a_1) \Rightarrow \exists y.\forall x.A(x,y), A(a_2,a_0) & \forall\text{-L} \\
\hline
a_0, a_1; \forall x.\exists y.A(x,y), A(a_0,a_1) \;\Rightarrow\; \forall \mathbf{x}.\mathbf{A}(\mathbf{x},\mathbf{a_0}), \exists y.\forall x.A(x,y) & \forall\text{-R} \\
\hline
a_0; \exists \mathbf{y}.\mathbf{A}(\mathbf{a_0},\mathbf{y}), \forall x.\exists y.A(x,y) \;\Rightarrow\; \forall x.A(x,a_0), \exists y.\forall x.A(x,y) & \exists\text{-L} \\
\hline
a_0; \exists y.A(a_0,y), \forall x.\exists y.A(x,y) \;\Rightarrow\; \exists \mathbf{y}.\forall \mathbf{x}.\mathbf{A}(\mathbf{x},\mathbf{y}) & \exists\text{-R} \\
\hline
a_0; \forall \mathbf{x}.\exists \mathbf{y}.\mathbf{A}(\mathbf{x},\mathbf{y}) \;\Rightarrow\; \exists y.\forall x.A(x,y) & \forall\text{-L}
\end{array}$$

With some insight it is clear that we shall never have the same formula in the antecedent and in the consequent of any sequent: indeed the order of the quantifiers is such that in the antecedent we only have atomic formulas $A(a_i, a_j)$ where $i < j$ and in the succedent only atomic formulas of the form $A(a_j, a_i)$ with $j > i$. Thus we can define an interpretation $\mathcal{M} = (D, A_{\mathcal{M}})$ where $D = \{a_i \mid i \in \mathbf{N}\}$ and $A_{\mathcal{M}} = \{\langle a_i, a_j \rangle \mid i < j\}$. But there is more: we can give an interpretation in the language or arithmetic by setting $\mathcal{M} = (\mathbf{N}, <)$. If the universe of discourse is that of natural numbers and $A(x, y)$ means $m < n$, then our sentence means "*if for every natural number m there is a greater natural number n, then there is a number greatest than all*" which is clearly false. It should also be noticed that the use of infinite models to refute our sentence is unnecessary: other models are given by two totally self-centered individuals loving only themselves, or two people desperately in love with each other to the point of not loving themselves anymore...

On the other hand, the use of infinite models is necessary in some cases. The property of being an ordering which is irreflexive (IRR $= \forall x.(x < x)$, transitive (TRANS $= \forall x \forall y \forall z.(x < y \wedge y < z) \rightarrow x < z$) and has no maximum (NM $= \forall x.\exists y.x < y$) can be expressed by the formula in the antecedent of the following sequent S

$$\forall x.\exists y.x < y, \forall x.(x < x), \forall x \forall y \forall z.(x < y \wedge y < z) \rightarrow x < z \Rightarrow \bot$$

The sequent S claims that these properties are inconsistent. But there exists an ordering which is irreflexive and transitive and has no maximum: this is again the "*less than*" relation between natural numbers, i.e., $\mathcal{M} = (\mathbf{N}, <)$ is a model of the antecedent of S. However, no *finite* structure is a model of IRR + TRANS + NM (exercise). Thus if we apply the "*semantic tableaux procedure*" to the sequent S we shall obtain an infinite branch, and all the atomic formulas in the antecedents of its sequents are of the form $m < n$ for $m, n \in \mathbf{N}$.

Two final, but important, remarks. We may doubt that if the procedure does not terminate then a countermodel can be built using the information given by the procedure. Doubts may be justified in two grounds:

1. How do we know that if the procedure does not terminate then we can find an infinite branch in the tree?

2. Even if we have an infinite branch, which must therefore contain an expression of the form $\forall x.\exists y.A(x, y)$ in the antecedent or an expression $\exists x.\forall y.A(x, y)$ in the succedent of some sequent, how do we know that such an expression will be recurrently evaluated, so that in $\forall x.\exists y.A(x, y)$ the universal quantifier can be instantiated with all the names of the infinite domain?

The answer to (1.) is *König's Lemma: every finitely branching but infinite tree contains an infinite path.* We could indeed have an infinite tree, with only finite branches of unlimimited length: but this is possible only if the tree has some vertex with infinitely many outgoing edges. The case of *binary trees* (i.e., with only two edges outgoing from each node) clearly suffices for our "*semantic tableaux*" procedure.

The answer to (2.) comes from the fact that we have applied a *breadth first* search in the formulation of our rules. Notice that in a sequent $\Gamma \Rightarrow \Delta$ where Γ and Δ are *lists*, they can be regarded as *queues* and a *breadth-first search-strategy* in our procedure is implemented simply by adding the new subformulas to the end of the queue, when inverting the sequent calculus rules as indicated above. On the other hand, a *depth-first strategy* would be implemented by defining our rules for the quantifiers as follows:

Depth-First search-strategy

$$\frac{a_0,\ldots,a_i,a_{i+1};\Gamma \Rightarrow A[a_{i+1}/x],\Delta}{a_0,\ldots,a_i;\Gamma \Rightarrow \forall x.A,\Delta}\,\forall\text{-R}$$

$$\forall\text{-L}\,\frac{a_0,\ldots,a_i;A[a_0/x],\ldots,A[a_i/x],\forall x.A,\Gamma \Rightarrow \Delta}{a_0,\ldots,a_i;\forall x.A,\Gamma \Rightarrow \Delta}$$

$$\frac{a_0,\ldots,a_i;\Gamma \Rightarrow A[a_0/x],\ldots,A[a_i/x],\exists x.A,\Delta}{a_0,\ldots,a_i;\Gamma \Rightarrow \exists x.A,\Delta}\,\exists\text{-R}$$

$$\exists\text{-R}\,\frac{a_0,\ldots,a_i,a_{i+1};A[a_{i+1}/x],\Gamma \Rightarrow \Delta}{a_0,\ldots,a_i;\exists x.A,\Gamma \Rightarrow \Delta}$$

Think what would it happen if we applied *depth first search* to a sequent of the form

$$\Rightarrow A, \exists y\forall x.x < y, \neg A$$

(exercise).

We can now extend the main theorem about the "*Semantic Tableaux*" procedure for First Order Predicate Logic.

Theorem. *The "semantic tableaux" procedure for classical first-order predicate logic given a formula A in $\mathcal{L}$,*

1. *either terminates returning a tree of sequents τ with $\Rightarrow A$ as the root, such that every sequent in τ is valid, if $\models A$;*
2. *or terminates with an* open *branch from which a* finite *countermodel can be constructed;*
3. *or does not terminate, but it contains an infinite branch containing all information needed to construct a countermodel $\mathcal{M}$ falsifying A.*

The theorem can be proved through the following Lemma and Propositions.

König's Lemma *Any infinite binary tree has an infinite path.*

Sketch of Proof. Let τ be an infinite tree with root n_0. Since τ is infinite at least one of the subtrees τ_{00} and τ_{01} beginning with the successors n_{00}, n_{01} of n_0 must be infinite, so let n_1 be the root of an infinite subtree, say τ_{01}. Now repeat the argument applied to n_1. With infinitely many choices we get an infinite path.

Proposition 1. (i) *In any finite open branch every expression of the form* $\forall x.A(x)$ *occuring in the antecedent and any expression* $\exists x.A(x)$ *occuring in the succedent] of some sequent is instantiated as* $A[a_i/x]$ *in the antecedent [in the succedent] of some sequent of the branch, for all* a_i *in the list* $\mathbf{\overline{a}} = a_0$, *. . ., a_k occurring in the leaf.*

(ii) *In any infinite open branch every expression of the form* $\forall x.A(x)$ *occuring in the antecedent* [*any expression* $\exists x.A(x)$ *occuring in the succedent*] *of some sequent, is instantiated as* $A[a_i/x]$ *in the antecedent* [*in the succedent*] *of some sequent of the branch*, for all a_i in any list $\mathbf{\overline{a}} = a_0$, . . ., a_k occurring in any sequent of the branch.

The proof of (i) is clear; (ii) follows from the fact that we have implemented a breadth-first procedure.

Proposition 2. *In every inference of the classical sequent calculus the sequent-conclusion is falsifiable if and only if at least one of the sequent-premises is falsifiable.*

Proof: By inspection of each rule.