Chapter 2:

Proof Rules for Predicate Logic

2.1 Introduction

Mathematical activity can be classified mainly as ‘‘proving’’, ‘‘solving’’, or ‘‘simplifying’’. Techniques for solving heavily depend on the structure of the formulae under consideration and will be discussed in many special lectures on systems of linear equations, differential equations, or integral equations. Solving, however, appears also as a subtask in proving, namely when proving formulae involving the existential quantifier. In many cases, the proof of an existential formula finally amounts to solving some formulae. We will discuss in this chapter the transition from proving to solving, we will then *not* go into the details of actually solving. ‘‘Simplification’’ is often addressed as ‘‘computation’’ and most of the times refers to ‘‘replacing equals by equals’’. Simplification, however, can be viewed in a much broader context, which will partly be covered in this chapter in some rules for proving ‘‘by symbolic computation’’. The emphasis of this chapter is being put on an introduction of rules for proving in predicate logic. These rules should be helpful for both checking the correctness of given proofs and for generating correct proofs on one’s own.

2.1.1 Proof Situations and Proofs

A *proof situation* consists of

- a formula to be proved (the ‘‘goal formula’’) together with a couple of free variables called the ‘‘proof variables’’
- and a knowledge base.

The *proof problem* consists of *showing* that the goal formula, under the assumptions in the knowledge base, is true for all values of the proof variables.

A *proof* starting from a given proof situation consists of a sequence of (algorithmic) steps that reduces, by certain ‘‘reasoning rules’’ the proof situation to (hopefully) simpler proof situations (solution situations, simplification situations) until one arrives at situations for which an answer is known. The reasoning rules also tell us how, from the answers of the intermediate situations, the answer to the initial proof problem may be obtained.

We do not give a formal definition of ‘‘proof’’, it should be understood that a proof should not only tell whether or not the goal follows from the assumptions but it should also *convince* the reader (listener) by giving arguments *why* the goal is true whenever the assumptions hold. (‘‘Truth’’ of a formula here refers to the semantics of expressions introduced in Chapter 1.)

Example of a proof situation in which the answer ‘‘proved’’ is trivially known: Any proof situation in which the goal formula can be obtained from one of the formulae in the knowledge base by substitution, i.e. the goal formula is an instance of a formula in the knowledge base. The easiest case is when the goal is equal to one of the formulae in the knowledge base.

Here, we do not discuss in which sense the reasoning steps in a proof must be algorithmic. However, in each of the reasoning steps which we introduce and train below, it should be clear that they can be ‘‘checked’’ by computers. (The *Theorema* system is a system that does not only allow to check reasoning steps algorithmically but also to produce such steps algorithmically to a certain extent.)

2.1.1.1 Conventions on Free Variables

In proof situations we assume the goal and all formulae in the knowledge base as *closed*, i.e. not containing free variables. Often the proof variables as introduced above are not mentioned explicitely. In this case, we assume all free variables bound by a universal quantifier, i.e. we prove the universal closure of the goal. Analogously, we consider the universal closure of the formulae in the knowledge base. In the description of the proof rules, we therefore assume that neither goal nor knowledge base contain free variables.

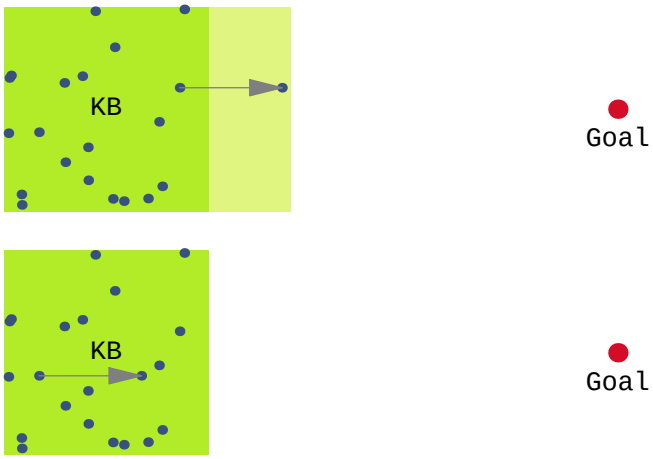
2.1.2 Intuitive Notion: The ‘‘Distance’’ between the Goal and the Knowledge Base

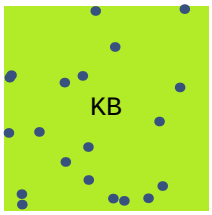
On an informal level, it might be helpful to view the process of ‘‘proving’’ as ‘‘reducing the distance between the goal and the knowledge base’’, i.e. the distance between the goal and the knowledge base is used as a measure for the ‘‘simplicity of a proof situation’’. Notice, however, that there is no exact notion of ‘‘distance’’ in this context, it is meant only as a metaphor, through which some of the characteristics of proving can be explained.

A proof is finished, when the goal is contained in the knowledge base, i.e. the distance between goal and knowledge base is zero, or the goal is an instance of a formula in the knowledge base, i.e. the goal is *very close* to a formula in the knowledge base, i.e. the distance between goal and knowledge base is "small". At the beginning of a proof, the goal is neither contained in the knowledge base nor is the goal already very close to a formula in the knowledge base, i.e. there is a "big" distance between goal and knowledge base—otherwise the proof is trivial! The aim of each individual proof step should therefore be to reduce the distance between goal and knowledge base.

Now think of each formula as a point in the plane and the knowledge base as the area containing all formulae it is composed of. Each reasoning step modifies the goal or the knowledge base.

The distance between the goal and the knowledge base can be reduced by either *moving the goal towards the knowledge base* or by *expanding the knowledge base towards the goal*. Similarly, the reasoning rules that we will study below, can be subdivided into rules for *goal reduction* and rules for *knowledge base expansion*. More generally, reasoning proceeds by alternating rounds of "goal reductions" (working backwards from the goals) and "knowledge base expansions" (working forward from known facts). Now suppose, we only apply reasoning rules, which move the goal closer to the knowledge base or which move formulae in the knowledge base closer to the goal.





The above pictures should indicate that an expansion of the knowledge base *can* reduce the distance between goal and knowledge base (see picture 1) but does *not necessarily* do so (see picture 2). *Reduction of the goal* (picture 3) on the other hand is likely to always reduce the distance between the goal and the knowledge base. Therefore, as a *general principle*, it is in most of the examples better to *first reduce the goal* and only enter a round of expanding the knowledge base when goal reduction is no longer possible. After knowledge base expansion a reduction of the goal might again be possible.

2.1.3 Notation in Reasoning Rules

When explaining reasoning rules in the subsequent sections, upper case letters like ‘F’, ‘G’, etc. normally will denote formulae, lower case letters like ‘s’, ‘t’, etc. will denote terms, and Greek letters ‘ξ’, ‘η’, etc. will denote variables. If we consider universally quantified formula of the form

$$\underset{\xi,\eta,\ldots}{\forall} F$$

in the knowledge base, we always assume tacitly that ξ, η, … are all the distinct free variables in F so that

$$\underset{\xi,\eta,\ldots}{\forall} F$$

is closed.

2.2 The Role of Propositional Logic in Predicate Logic Proving

2.2.1 Elementary Parts of Formulae

For the propositional expansion of knowledge bases and other reasoning techniques involving propositional connectives, the notion of the "elementary parts" of formulae is important: The elementary parts of a formula are determined by going from outside to inside and considering subformulae that start with a quantifier or a predicate constant as "black boxes" (= the elementary parts).

We first illustrate the process of determining the elementary parts of formulae in a couple of examples:

Example

$$x \in A \Rightarrow (x \in A \vee x \in B) \ .$$

The elementary parts ("black boxes" which we do not any more decompose) are

$$x \in A$$

and

$$x \in B \ .$$

The formula now writes as

$$\mathcal{A} \Rightarrow (\mathcal{A} \vee \mathcal{B}) \ .$$

Example

$$(A \bigcap B = \emptyset) \Leftrightarrow \neg \underset{x}{\exists} (x \in A \wedge x \in B)$$

The elementary parts are

$$A \bigcap B = \emptyset$$

and

$$\underset{x}{\exists} (x \in A \wedge x \in B)$$

and the formula writes as

$$\mathcal{A} \Leftrightarrow \neg\, \mathcal{B}$$

2.2.2 Truth Tables

Propositional expansion of knowledge bases and other reasoning techniques involving propositional connectives are based on *truth tables* that describe the behavior of the propositional connectives as functions on the truth values "true" and "false". We assume the truth tables for propositional connectives ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$) are known, they can be looked up in the lecture notes Analysis or Linear Algebra.

2.2.3 Propositional Consequences, Propositional Tautologies, Propositional Equivalences

A formula U is a *propositional consequence* of formulae F, G, … iff U is true whenever F, G, … are true "only because of the meaning of the propositional connectives". More precisely, for finding out whether or not U is a propositional consequence of F, G, …, the formulae are decomposed into their elementary parts and then it is checked whether or not, for all assignments of the possible two truth values "true" and "false" to the elementary parts, U obtains the truth value "true" whenever F, G, … obtain the truth value "true". For this check, the truth tables for the propositional connectives are applied.

A formula U is a *propositional tautology* iff U is (always) true iff U obtains the truth value "true" for all assignments of truth values to the elementary parts of U.

Formulae U and V are *propositionally equivalent* iff U is a propositional consequence of V and V is a propositional consequence of U.

It is clear that formulae U and V are propositionally equivalent iff, for any assignment of truth values to the elementary parts of U and V, they obtain identical truth values.

Also, U and V are propositionally equivalent iff $U \Leftrightarrow V$ is a propositional tautology.

Example

The formula

$$(x \in A \wedge x \in C) \vee (x \in B \wedge x \in C)$$

is a propositional consequence of the formulae

$x \in A \vee x \in B$

and

$x \in C$.

In order to check this we determine the elementary parts of the formulae and then look to the truth tables

$x \in A$	$x \in B$	$x \in C$	$x \in A \vee x \in B$	$(x \in A \wedge x \in C) \vee (x \in B \wedge x \in C)$
true	true	**true**	**true**	**true**
true	true	□	true	□
true	□	**true**	**true**	**true**
true	□	□	true	□
□	true	**true**	**true**	**true**
□	true	□	true	□
□	□	true	□	□
□	□	□	□	□

(We write '□' instead of 'false' in order to identify the positions of the 'true' in the tables more easily.)

One sees that for all combinations of truth values, for which $x \in C$ and $x \in A \vee x \in B$ are both true, also $(x \in A \wedge x \in C) \vee (x \in B \wedge x \in C)$ is true. Hence,

$(x \in A \wedge x \in C) \vee (x \in B \wedge x \in C)$

is a propositional consequence of $x \in C$ and $x \in A \vee x \in B$.

Example

The formula

$((x \in A \vee x \in B) \wedge x \in C) \Leftrightarrow ((x \in A \wedge x \in C) \vee (x \in B \wedge x \in C))$

is a tautology (which is also called "propositional distributivity of $\wedge$ over $\vee$"). In fact, when we check the truth tables

$x \in A$	$x \in B$	$x \in C$	$(x \in A \vee x \in B) \wedge x \in C$	$(x \in A \wedge x \in C) \vee (x \in B \wedge x \in C)$
true	true	true	**true**	**true**
true	true	□	□	□
true	□	true	**true**	**true**
true	□	□	□	□
□	true	true	**true**	**true**
□	true	□	□	□
□	□	true	□	□
□	□	□	□	□

we see that the columns of $((x \in A \vee x \in B) \wedge x \in C)$ and $((x \in A \wedge x \in C) \vee (x \in B \wedge x \in C))$ are identical and, hence, the column of

$$((x \in A \vee x \in B) \wedge x \in C) \Leftrightarrow ((x \in A \wedge x \in C) \vee (x \in B \wedge x \in C))$$

would consist of all 'true'. Hence, we also see that

$$((x \in A \vee x \in B) \wedge x \in C)$$

is a propositional consequence of

$$((x \in A \wedge x \in C) \vee (x \in B \wedge x \in C))$$

and that, in fact,

$$((x \in A \vee x \in B) \wedge x \in C)$$

and

$$((x \in A \wedge x \in C) \vee (x \in B \wedge x \in C))$$

are propositionally equivalent.

2.2.4 Pure Propositional Logic Proof Parts

At rare occasions, it may happen that an entire part of a proof can be established by just looking to the elementary propositional parts of the goal formula and the relevant formulae in the knowledge base and by establishing that the goal is a propositional consequence of the relevant formulae in the knowledge base. There are basically two methods for establishing that a goal formula is a propositional consequence of certain formulae in the knowledge base:

- the truth table method,

- the "natural deduction method": this is the method that results from applying the proof techniques for the propositional connectives described below that reduce proof situations to other proof situations until one arrives at trivial proof situations; we will illustrate this method in the examples. In most practical situations, the natural deduction method is applied. In contrast to the truth table method, a proof by natural deduction gives a "logical argumentation" *why* the goal formula is a consequence of the formulae in the knowledge base.

Example

The formula

$$(x \in A \wedge x \in C) \vee (x \in B \wedge x \in C)$$

is a propositional consequence of the formulae

$$x \in A \vee x \in B$$

and

$$x \in C \ .$$

This is so because if we know $x \in A \vee x \in B$ then we know that *either* $x \in A$ *or* $x \in B$ holds. We distinguish the two cases:

- $x \in A$: Together with $x \in C$ we then know $x \in A \wedge x \in C$ and therefore also $(x \in A \wedge x \in C) \vee (x \in B \wedge x \in C)$.
- $x \in B$: Together with $x \in C$ we then know $x \in B \wedge x \in C$ and therefore also $(x \in A \wedge x \in C) \vee (x \in B \wedge x \in C)$.

2.2.5 Propositional Proof Rules for Structuring General Proofs in a Natural Way

The more important role of the proof rules for propositional connectives is in general predicate logic proofs for structuring the main part in the proof into subproofs, for which quantifier rules are then necessary. The proof rules for the connectives, which reduce proof situations to other proofs situations, yield the "natural deduction style" proofs. The name "natural deduction" stems from the fact that, actually, a very natural way of understanding the connectives is by explaining their role in proofs, i.e. in the reduction of proof situations to other proof situations.

We will explain this important role of propositional logic within predicate logic in the examples. In fact, most of the proofs that occur in practice, use propositional logic for structuring proofs before one applies (general and special)

quantifier rules. Hence, although propositional logic is trivial as a stand-alone proof technique, it is of essential importance as a structuring tool for full predicate logic proofs.

Roughly, the natural deduction rules for connectives contain (at least) one proof rule for each propositional connective occurring as outermost symbol in the goal or in a formula in the knowledge base. In addition to those there are natural deduction rules for the quantifiers '∀' and '∃', again both for the quantifier occurring as outermost symbol in the goal or in a formula in the knowledge base.

2.3 Natural Deduction Proof Rules for Propositional Connectives

2.3.1 Rules for Knowledge Base Expansion

2.3.1.1 Rule ("Propositional Consequence")

If

F,
G,
…

are in the knowledge base and U is a propositional consequence of F, G, … then

U

can be added to the knowledge base.

2.3.1.2 Rule ("Propositional Tautology")

If

U

is a propositional tautology then

U

can be added to the knowledge base.

2.3.1.3 Rule (“Case Distinction on Disjunction in Knowledge Base”)

If the proof goal has the form

F

and the knowledge base contains the closed formula

$G \vee H$

then prove

F

under the additional assumption

G

and prove

F

under the additional assumption

H .

Analogously for disjunctions with more alternatives.

Usually, one announces the application of this rule in a concrete proof with a phrase similar to
“Now we distinguish two cases.
Case 1: Assume G …
Case 2: Now assume H …”.

2.3.2 Rules for Universally Quantified Propositional Formulae

In fact, many rules for propositional connectives in the knowledge base do not work only when the propositional connective is the *outermost* formula symbol, but also when the propositional connective occurs inside a universal quantifier. We will only describe the rules for the quantified case, it should be obvious, how the rules specialize for the case where the universal quantifier is omitted.

2.3.2.1 Rule ("Expansion of Universally Quantified Conjunction in Knowledge Base")

If

$$\underset{\xi,\eta,\ldots}{\forall} (F \wedge G)$$

is in the knowledge base then both

$$\underset{\xi,\eta,\ldots}{\forall} F$$

$$\underset{\xi,\eta,\ldots}{\forall} G$$

can be added to the knowledge base. (Analogously for conjunctions with more than two formulae.)

2.3.2.2 Rule ("Universally Quantified Implication in Knowledge Base: Modus Ponens")

If

$$\underset{\xi,\eta,\ldots}{\forall} F \Rightarrow G$$

and

$$F_{\xi,\eta,\ldots\rightarrow s,t,\ldots}$$

(i.e. an instance of F) are in the knowledge base then also

$$G_{\xi,\eta,\ldots\rightarrow s,t,\ldots}$$

(i.e. the corresponding instance of G) and hence also

$$\underset{\alpha,\beta,\ldots}{\forall} G_{\xi,\eta,\ldots\rightarrow s,t,\ldots}$$

(where α, β, … are the free variables in s, t, …) can be added to the knowledge base.

Strategic Remark: In the phase of knowledge base expansion, applying modus ponens never is a mistake although some of the new statements generated may never be used subsequently.

2.3.2.3 Rule ("Universally Quantified Implication in Knowledge Base: Modus Ponens Special Case")

If

$$\underset{\xi,\eta,\ldots}{\forall} (F \wedge H) \Rightarrow G$$

and

$$F_{\xi,\eta,\ldots\rightarrow s,t,\ldots}$$

$$H_{\xi,\eta,\ldots\rightarrow s,t,\ldots}$$

are in the knowledge base then also

$$G_{\xi,\eta,\ldots\rightarrow s,t,\ldots}$$

(i.e. the corresponding instance of G) and hence also

$$\underset{\alpha,\beta,\ldots}{\forall} G_{\xi,\eta,\ldots\rightarrow s,t,\ldots}$$

(where α, β, … are the free variables in s, t, …) can be added to the knowledge base.

2.3.2.4 Rule ("Reflexivity of Implication")

The formula

$$\underset{\xi,\eta,\ldots}{\forall} F \Rightarrow F$$

can always be added to the knowledge base.

2.3.2.5 Rule ("Transitivity of Implication")

If

$$\underset{\xi,\eta,\ldots}{\forall} F \Rightarrow G$$

and

$$\underset{\xi,\eta,\ldots}{\forall} G \Rightarrow H$$

are in the knowledge base then

$$\underset{\xi,\eta,\ldots}{\forall} \; F \Rightarrow H$$

can be added to the knowledge base.

2.3.2.6 Rule (“Equivalence Rewriting”)

If

$$\underset{\xi,\eta,\ldots}{\forall} \; (F \Leftrightarrow G)$$

and

$$H$$

are in the knowledge base and the subformula at position p in H has the form

$$F_{\xi,\eta,\ldots \to s,t,\ldots}$$

(i.e. the subformula of H at position p is an instance of F) then also

$$H_{p \mapsto G_{\xi,\eta,\ldots \to s,t,\ldots}}$$

(i.e. H with the subformula at position p replaced by the corresponding instance of G) can be added to the knowledge base. Using this rule, *all occurrences* of instances of F can be replaced subsequently by instances of G.

Note that, if

$$\underset{\xi,\eta,\ldots}{\forall} \; (F \Leftrightarrow G)$$

is in the knowledge base then also

$$\underset{\xi,\eta,\ldots}{\forall} \; (G \Leftrightarrow F)$$

could be added to the knowledge base and, hence, equivalence rewriting can also proceed by replacing instances of G by instances of F.

The rule applies accordingly if H is the current proof goal.

2.3.2.7 Rule (“Reflexivity of Equivalence”)

The formula

$$\underset{\xi,\eta,\ldots}{\forall} F \Leftrightarrow F$$

can always be added to the knowledge base.

2.3.2.8 Rule (“Symmetry of Equivalence”)

If

$$\underset{\xi,\eta,\ldots}{\forall} F \Leftrightarrow G$$

is in the knowledge base then

$$\underset{\xi,\eta,\ldots}{\forall} G \Leftrightarrow F$$

can be added to the knowledge base.

2.3.2.9 Rule (“Transitivity of Equivalence”)

If

$$\underset{\xi,\eta,\ldots}{\forall} F \Leftrightarrow G$$

and

$$\underset{\xi,\eta,\ldots}{\forall} G \Leftrightarrow H$$

are in the knowledge base then

$$\underset{\xi,\eta,\ldots}{\forall} F \Leftrightarrow H$$

can be added to the knowledge base.

2.3.2.10 Rule (“Split Equivalence into Implication”)

If

$$\underset{\xi,\eta,\ldots}{\forall} F \Leftrightarrow G$$

is in the knowledge base then both

$$\underset{\xi,\eta,\ldots}{\forall} F \Rightarrow G$$

$$\underset{\xi,\eta,\ldots}{\forall} G \Rightarrow F$$

can be added to the knowledge base.

Note that this rule follows from already known rules! $F \Leftrightarrow G$ and $(F \Rightarrow G) \wedge (G \Rightarrow F)$ are propositionally equivalent, hence, the tautology $(F \Leftrightarrow G) \Leftrightarrow ((F \Rightarrow G) \wedge (G \Rightarrow F))$ can be added to the knowledge base. By "Equivalence Rewriting", we can replace $F \Leftrightarrow G$ by $(F \Rightarrow G) \wedge (G \Rightarrow F)$ in the knowledge base. Then use the rule "Split Conjunction" for splitting conjunctions in the knowledge base in order to derive the two desired formulae.

2.3.2.11 Rule ("Equality Rewriting in Formulae")

If

$$\underset{\xi,\eta,\ldots}{\forall} (u = v)$$

and

$$H$$

are in the knowledge base and the subterm at position p in H has the form

$$u_{\xi,\eta,\ldots \to s,t,\ldots}$$

(i.e. the subterm of H at position p is an instance of u) then also

$$H_{p \to v_{\xi,\eta,\ldots \to s,t,\ldots}}$$

(i.e. H with the subterm at position p replaced by the corresponding instance of v) can be added to the knowledge base.

Note that, if

$$\underset{\xi,\eta,\ldots}{\forall} (u = v)$$

is in the knowledge base then also

$$\underset{\xi,\eta,\ldots}{\forall} (v = u)$$

could be added to the knowledge base and, hence, equality rewriting can also proceed by replacing instances of v by instances of u.

The rule applies accordingly if H is the current proof goal.

2.3.2.12 Rule ("Equality Rewriting in Terms")

If

$$\underset{\xi,\eta,\ldots}{\forall} (u = v)$$

is in the knowledge base and the subterm at position p in the closed term h has the form

$$u_{\xi,\eta,\ldots \to s,t,\ldots}$$

(i.e. the subterm of h at position p is an instance of u) then also

$$h = h_{p \to v_{\xi,\eta,\ldots \to s,t,\ldots}}$$

can be added to the knowledge base.

2.3.2.13 Rule ("Reflexivity of Equality")

The formula

$$\underset{\xi,\eta,\ldots}{\forall} s = s$$

can always be added to the knowledge base.

2.3.2.14 Rule ("Symmetry of Equality")

If

$$\underset{\xi,\eta,\ldots}{\forall} s = t$$

is in the knowledge base then

$$\underset{\xi,\eta,\ldots}{\forall} t = s$$

can be added to the knowledge base.

2.3.2.15 Rule (“Transitivity of Equality”)

If

$$\underset{\xi,\eta,\ldots}{\forall} \quad r = s$$

and

$$\underset{\xi,\eta,\ldots}{\forall} \quad s = t$$

are in the knowledge base then

$$\underset{\xi,\eta,\ldots}{\forall} \quad r = t$$

can be added to the knowledge base.

2.3.2.16 Notation for Rewriting Chains

Equivalence and equality rewriting together with reflexivity, symmetry, and transitivity of equivalence and equality plays an important role in the expansion of knowledge bases (and in the reduction of goals). By transitivity, rewriting steps can be combined in chains of which only the first and last formula (term) are interesting as “the result”. Basically, also modus ponens and reflexivity and transitivity of implications can be used in this way only that only one direction is valid.

Such symbolic computation reasoning chains are presented in various forms. Here, we introduce one form: The following notation

F_0

$\downarrow$ by (label 1) with $x, y, \ldots \to s_1, t_1, \ldots$

F_1

$\downarrow$ by (label 2) with $x, y, \ldots \to s_2, t_2, \ldots$

F_2

$\downarrow$ by (label 3) with $x, y, \ldots \to s_3, t_3, \ldots$

...

$\downarrow$ by (label k) with $x, y, \ldots \to s_k, t_k, \ldots$

F_k

means that

- F_0 is in the knowledge base,
- (label 1) is an implication or equivalence or equality by which, with the substitution $x, y, \ldots \to s_1, t_1, \ldots, F_1$ can be derived by modus ponens or equivalence rewriting or equality rewriting,
- F_1 can, hence, be added to the knowledge base,
- (label 2) is an implication or equivalence or equality by which, with the substitution $x, y, \ldots \to s_2, t_2, \ldots, F_2$ can be derived by modus ponens or equivalence rewriting or equality rewriting,
- F_2 can, hence, be added to the knowledge base,

....

- F_k can, hence, be added to the knowledge base.

Note that this does *not* just mean that

$F_0 \Rightarrow F_1$

$F_1 \Rightarrow F_2$

...

is in the knowledge base! Therefore, it is not a good idea to write these chains in the following way

$F_0 \Rightarrow F_1 \Rightarrow F_2 \ldots$

except if the meaning of this notation is completely understood! In particular, it is very dangerous to describe reasoning chains by saying something like:

Now we know,

$F_0 \Rightarrow F_1 \Rightarrow \ldots$

because what one wants to say is

“We know

F_0

and from this

F_1

...

can be concluded by modus ponens, equivalence rewriting or equality rewriting.'' !

Often, the substitutions used in the symbolic computation steps are not indicated explicitly. Also, the positions at which the replacements are done are almost never given explicitly although only the explicit indication of the position would make a given symbolic computation step unique.

2.3.3 Rules for Goal Reduction

2.3.3.1 Rule (“Proof by Contradiction”)

If the proof goal has the form

$\neg F$

where F is a closed formula, add

F

to the knowledge base and try to prove a contradiction, i.e. try to prove a formula of the form

$G \wedge \neg G$.

Strategic Remark about Contradictions: Of course, the advice of proving negations by contradictions does not help much because the question is which formula G can be found such that

$G \wedge \neg G$

can be proved. However, often, there is already a formula G in the knowledge bases such that $\neg G$ is conjectured to be a good candidate for being provable from F. Thus, when we derive $\neg G$ and we have G in the knowledge base then the contradiction $G \wedge \neg G$ is proven. Proofs by contradiction are also called “indirect proofs”.

In principle, the proof technique of proving by contradiction can be applied in just every proof problem: In order to prove a closed formula

F

assume

$\neg F$

and derive a contraction.

However, is it advisable to attempt, first, a "direct proof" before one attempts an "indirect proof" because, sometimes, direct proofs give more information than indirect proofs. It is also worth mentioning that "real life proving" as a fundamental part of mathematical exploration is typically applied in situations where it is not known whether or not a given formula F is true or false (under a given knowledge base). Hence, the basic exploration cycle for analyzing the truth of a "conjectured" formula F by proving is as follows:

(1) try to prove F (in which case F is true),

(2) assume $\neg F$ and try to prove a contradiction (in which case F is also true)

(3) try to prove $\neg F$ (in which case F is false),

(4) assume F and try to prove a contradiction (in which case F is also false),

go back to (1).

Going a couple of times through this basic exploration cycle is not a standstill: The insight gained from a failing proof in one of the phases (1)−(4) may well help to be successful in a later phase!

2.3.3.2 Rule ("Equivalence Rewriting")

If the proof goal has the form

F

and we have

$F \Leftrightarrow G$

in the knowledge base then try to prove

G .

Strategic Remark: In particular, when the proof goal is

$\neg F$

there are some useful equivalences to be applied for rewriting the negated proof goal, e.g.

$$\neg (F \wedge G) \Leftrightarrow \neg F \vee \neg G$$

$$\neg (F \vee G) \Leftrightarrow \neg F \wedge \neg G$$

$$\neg \underset{x}{\forall} F \Leftrightarrow \underset{x}{\exists} \neg F$$

$$\neg \underset{x}{\exists} F \Leftrightarrow \underset{x}{\forall} \neg F$$

Rewriting the negated proof goal may then allow a direct proof instead of an indirect proof.

2.3.3.3 Rule (“Decomposition of Conjunctions”)

If the proof goal has the form

$$F \wedge G \wedge \ldots$$

where F, G, … are closed formulae then prove

$$F$$

and prove

$$G$$

and prove

…

under the given knowledge base.

2.3.3.4 Rule (“Splitting Disjunctions I”)

If the proof goal has the form

$$F \vee G \vee \ldots$$

where F, G, … are closed formulae then prove

$$F$$

or prove

G

or prove

…

under the given knowledge base.

2.3.3.5 Rule (“Deduction Rule”)

If the proof goal has the form

$F \Rightarrow G$

where F and G are closed formulae then prove

G

under the additional assumption

F .

2.3.3.6 Rule (“Contraposition”)

Use the equivalence

$(F \Rightarrow G) \Leftrightarrow (\neg G \Rightarrow \neg F)$.

If the proof goal has the form

$F \Rightarrow G$

where F and G are closed formulae then prove

$\neg F$

under the additional assumption

$\neg G$.

(This rule, essentially, is a version of the “proof by contradiction” rule. Proofs by contradiction and proofs by contraposition are both also called “indirect proofs”.)

2.3.3.7 Rule (“Contradiction”)

Use the equivalence

$$(F \Rightarrow G) \Leftrightarrow (\neg F \vee G) \quad .$$

If the proof goal has the form

$$F \Rightarrow G$$

where F and G are closed formulae then assume

$$F$$

$$\neg G$$

and derive a contradiction.

2.3.3.8 Rule (“Splitting Disjunctions II”)

Use the equivalence

$$(F \vee G \vee H \vee \ldots) \Leftrightarrow (\neg (G \vee H \vee \ldots) \Rightarrow F) \Leftrightarrow (\neg G \wedge \neg H \wedge \ldots \Rightarrow F) \quad .$$

in order to prove the goal

$$F \vee G \vee H \vee \ldots$$

assume

$$\neg G$$

$$\neg H$$

$$\neg \ldots$$

and prove

$$F \quad .$$

2.3.3.9 Rule (“Prove Both Directions”)

If the proof goal has the form

$$F \Leftrightarrow G$$

then prove

G

under the additional assumption

F

and prove

F

under the additional assumption

G .

2.3.3.10 Rule (“Prove Multi Equivalence”)

If the proof goal has the form

$F \Leftrightarrow G \Leftrightarrow \ldots \Leftrightarrow H$

then prove

$F \Rightarrow G$

and prove

$G \Rightarrow \ldots$

$\ldots$

and prove

$H \Rightarrow F$.

Remark: Proof goals of the form $F \Leftrightarrow G \Leftrightarrow \ldots \Leftrightarrow H$ are commonly formulated in the following way: “The following are equivalent:

- F
- G
- $\ldots$
- H .

Before applying the rule above the formulae may be re-ordered. Transitivity of implication justifies this rule.

2.3.3.11 Rule ("Symbolic Computation Goal Reduction")

If the proof goal has the form

$F \Leftrightarrow G$

where F and G are closed formulae then try to reduce the goal

G

by *equivalence symbolic computation* to

F .

Strategic Remark: Try to establish a sequence

$G \Leftrightarrow H$

$H \Leftrightarrow \ldots$

$\ldots$

$\ldots \Leftrightarrow F$,

which, by transitivity and symmetry of equivalence, proves $F \Leftrightarrow G$. Of course, one can also start with F and establish a sequence of equivalences ending up in G.

2.3.3.12 Rule ("Symbolic Computation Equality Proving")

If the proof goal has the form

$s = t$

where s and t are closed terms then try to transform

s

to

t

by *symbolic computation using equalities* in the knowledge base.

2.3.3.13 Rule ("Equality Proving by Simplification")

If the proof goal has the form

$$s = t$$

where s and t are closed terms then simplify

$$s$$

to

$$\tilde{s}$$

and simplify

$$t$$

to

$$\tilde{t}$$

and then prove

$$\tilde{s} = \tilde{t} \ .$$

2.3.3.14 Rule ("Goal Reduction using Implications")

If the proof goal has the form

$$G_{\xi,\eta,\ldots\to s,t,\ldots}$$

and we have a formula

$$\underset{\xi,\eta,\ldots}{\forall} F \Rightarrow G$$

in the knowledge base then it is sufficient to prove

$$F_{\xi,\eta,\ldots\to s,t,\ldots}$$

Important Remark: If any of the variables $\xi, \eta, \ldots$ does *not occur* in G, say ζ, then the substitution $\xi, \eta, \ldots \to s, t, \ldots$ would *not* substitute a term for ζ, and therefore leave ζ free in $F_{\xi,\eta,\ldots\to s,t,\ldots}$. In this situation, the goal

$$G_{\xi,\eta,\ldots\to s,t,\ldots}$$

can only be reduced to

$$\exists_{\zeta} F_{\xi,\eta,\ldots\to s,t,\ldots} \quad .$$

In this version, the rule introduces a quantifier, hence, it transforms the proof situation into a generally more complicated proof situation! Use this rule with caution.

Example: Suppose we know

$$\forall_{x,y,z} (x \le y \wedge y \le z) \Rightarrow x \le z \quad \text{(Trans)}$$

Then, for proving

$$a \le b$$

it is sufficient to prove

$$\exists_{y} (a \le y \wedge y \le b) \quad .$$

The presentation of the successful proof would typically proceed by first presenting a proof for

$$a \le y_0 \quad \text{and}$$

$$y_0 \le b$$

for some (well chosen) y_0 and then derive from this

$$a \le b$$

by knowledge base expansion using Modus Ponens for (Trans). However, the goal reduction introducing the $\exists$-quantifier represents the "idea" to search for a y_0 .

2.4 Natural Deduction Rules for Quantified Formulae

2.4.1 Rules for Knowledge Base Expansion

2.4.1.1 Rule ("Instanciation of Universal Quantifier")

If

$$\underset{\xi,\eta,\ldots}{\forall} F$$

is in the knowledge base then also

$$F_{\xi,\eta,\ldots\to s,t,\ldots}$$

and hence also

$$\underset{\alpha,\beta,\ldots}{\forall} F_{\xi,\eta,\ldots\to s,t,\ldots}$$

(where α, β, … are the free variables in s, t, …) can be added to the knowledge base.

Strategic Remark: This rule is fundamental but strategically not very useful for reasoning because infinitely many new formulae could be generated in the knowledge base by this rule without making any contribution to achieving a specified goal. Therefore, special cases of this rule (in dependence on the structure of F) are more useful, see Section 2.3.2 above. In many cases, the proof goal can give an idea on how to choose the terms s, t, … appropriately in order to being able to continue goal reduction.

An instanciation step is usually announced in a proof by saying:

Since we know

$$\underset{\xi,\eta,\ldots}{\forall} F$$

we know in particular

$$F_{\xi,\eta,\ldots\to s,t,\ldots}$$

2.4.1.2 Rule (“Skolem Constants”)

If

$$\underset{\xi,\eta,\ldots}{\exists} F$$

is in the knowledge base (where ξ, η, … are the only free variables in F) then you can add

$$F_{\xi,\eta,\ldots\to \mathrm{x0},\mathrm{y0},\ldots}$$

to the knowledge base, where $\mathrm{x0}, \mathrm{y0}, \ldots$ are *new object constants* (i.e. object constants that do neither occur in the knowledge base nor in the goal formula). $\mathrm{x0}, \mathrm{y0}, \ldots$ are called "Skolem constants".

Application of this rule is often announced in the following way: We know

$$\underset{\xi,\eta,\ldots}{\exists} F$$

Therefore we can choose $\mathrm{x0}, \mathrm{y0}, \ldots$ such that

$$F_{\xi,\eta,\ldots\to \mathrm{x0},\mathrm{y0},\ldots} \quad .$$

2.4.1.3 Rule ("Skolem Functions")

If

$$\underset{\alpha,\beta,\ldots}{\forall} \underset{\xi,\eta,\ldots}{\exists} F$$

is in the knowledge base (where $\alpha, \beta, \xi, \eta, \ldots$ are the only free variables in F) then you can add

$$\underset{\alpha,\beta,\ldots}{\forall} F_{\xi,\eta,\ldots\to \mathrm{x0}[\alpha,\beta,\ldots],\mathrm{y0}[\alpha,\beta,\ldots],\ldots}$$

to the knowledge base, where $\mathrm{x0}, \mathrm{y0}, \ldots$ are *new function constants* (i.e. function constants that do neither occur in the knowledge base nor in the goal formula). $\mathrm{x0}, \mathrm{y0}, \ldots$ are called "Skolem function constants".

Application of this rule is often announced in the following way: We know

$$\underset{\alpha,\beta,\ldots}{\forall} \underset{\xi,\eta,\ldots}{\exists} F$$

Therefore we can choose functions $\mathrm{x0}, \mathrm{y0}, \ldots$ such that

$$\underset{\alpha,\beta,\ldots}{\forall} F_{\xi,\eta,\ldots\to \mathrm{x0}[\alpha,\beta,\ldots],\mathrm{y0}[\alpha,\beta,\ldots],\ldots} \quad .$$

2.4.2 Rules for Goal Reduction

2.4.2.1 Rule ("Arbitrary But Fixed")

If the proof goal is a formula

$$\forall_{\xi,\eta,\ldots} F$$

in which ξ, η, … are the free variables of F, then prove

$$F_{\xi,\eta,\ldots\to\xi_0,\eta_0,\ldots}$$

where ξ_0, η_0, … have to be *new object constants*, i.e. object constants that do not occur in the knowledge base and neither in the goal formula.

One often announces the application of this proof technique by saying:

We take arbitrary but fixed ξ_0, η_0, … and prove

$$F_{\xi,\eta,\ldots\to\xi_0,\eta_0,\ldots}$$

(In this formulation, "fixed" expresses the fact that the symbols 'ξ_0' etc. are object constants and "arbitrary" expresses the fact that the symbols 'ξ_0' etc. have to be new symbols that do not occur anywhere else in the goal and the knowledge base.)

Explanation

The proof technique "arbitrary but fixed" is the most important elementary proof technique that bridges quantifier-free with quantifier proving. It is important that one understands, intuitively, why this technique works. Therefore, we give here an intuitive explanation of the correctness of the technique:

Assume one succeeds to prove that

$$F_{\xi,\eta,\ldots\to\xi_0,\eta_0,\ldots}$$

for new constant symbols 'ξ_0' etc. Then it is clear that no specific knowledge on the objects denoted by 'ξ_0' etc. went into the proof because 'ξ_0' etc. are symbols that do not occur anywhere in the knowledge base. Hence, it would be possible to give the proof, without any change, for any specific object constants 'c', etc. denoting specific objects. Hence, the proof could be repeated, without any change, for all objects in the domain of discourse. Hence,

$$\forall_{\xi,\eta,\ldots} F$$

is true. Note that it is important that the new symbols 'ξ_0' etc. are constants: Thereby

$$F_{\xi,\eta,\ldots\to\xi_0,\eta_0,\ldots}$$

becomes a formula without free variables that can then be treated with other proof techniques, i.e. can be decomposed in parts according to the proof techniques to be discussed earlier.

It is also important that symbols 'ξ_0' etc. are really new, i.e. they do not occur anywhere in the knowledge base and neither in F. If we use symbols 'c', 'd', etc. for which there is already knowledge available in the knowledge base, the proof of

$$F_{\xi,\eta,\ldots\to c,c,\ldots}$$

might work only because of this knowledge and, maybe, would not work without this knowledge. Thus, we could not rely on the fact that the proof could be repeated for arbitrary constants denoting arbitry objects.

The proof technique "arbitrary but fixed" is very important in situations where formulae containing free variables are nested and the proof process for the individual quantifiers must be carefully distinguished. Sometimes, however, only the outermost universal quantifier has to be treated by the "arbitrary but fixed". In such cases, it is unusual to really introduce new constants by the "arbitrary but fixed" technique. Rather, one just uses the variable names 'ξ', etc. as new constants and one may say, for example,

"Let now ξ, η, … be arbitrary but fixed and prove F."

This is alright as long as everybody understands that, from now on, 'ξ', etc. are constants and one does not use, in the proof of F, any knowledge about 'ξ', etc. that may perhaps appear somewhere in the knowledge base.

2.4.2.2 Rule ("Find Appropriate Terms")

If the proof goal is

$$\exists_{\xi,\eta,\ldots} F$$

where ξ, η, … are the only free variables in F then it suffices to *find terms* s, t, … such that

$$F_{\xi,\eta,\ldots\to s,t,\ldots}$$

can be proved.

2.4.3 Case Distinction

Case distinctions do usually not appear stand-alone but inside predicates or logical connectives, i.e.

$$p\left[t, \begin{cases} t_1 & \Leftarrow & F_1 \\ t_2 & \Leftarrow & F_2 \\ t_3 & \Leftarrow & \text{otherwise} \end{cases}\right] \quad \text{or} \quad C\left[H, \begin{cases} G_1 & \Leftarrow & F_1 \\ G_2 & \Leftarrow & F_2 \\ G_3 & \Leftarrow & \text{otherwise} \end{cases}\right]$$

where p is some predicate constant and C is some logical connective. The predicate constant or the logical connective move inside the case distinction so that the above formulae stand for

$$\begin{cases} p[t, t_1] & \Leftarrow & F_1 \\ p[t, t_2] & \Leftarrow & F_2 \\ p[t, t_3] & \Leftarrow & \text{otherwise} \end{cases} \quad \text{or} \quad \begin{cases} C[H, G_1] & \Leftarrow & F_1 \\ C[H, G_2] & \Leftarrow & F_2 \\ C[H, G_3] & \Leftarrow & \text{otherwise} \end{cases} \quad \text{, respectively.}$$

The convention is that a case distinction

$$\begin{cases} G_1 & \Leftarrow & F_1 \\ G_2 & \Leftarrow & F_2 \\ G_3 & \Leftarrow & \text{otherwise} \end{cases} \tag{1}$$

is an abbreviation for the formula

$$F_1 \Rightarrow G_1 \ \wedge\ \neg F_1 \wedge F_2 \Rightarrow G_2 \ \wedge\ \neg F_1 \wedge \neg F_2 \Rightarrow G_3 \ . \tag{2}$$

(Analogously proceed for case statements with less or more than three cases.)

Case distinctions are most commonly used in definitions of functions or predicates, such as

$$f[x] := \begin{cases} t_1 & \Leftarrow & F_1 \\ t_2 & \Leftarrow & F_2 \\ t_3 & \Leftarrow & \text{otherwise} \end{cases} \quad \text{or} \quad H :\Leftrightarrow \begin{cases} G_1 & \Leftarrow & F_1 \\ G_2 & \Leftarrow & F_2 \\ G_3 & \Leftarrow & \text{otherwise} \end{cases}$$

By the rules above, these definitions mean

$$\begin{cases} f[x] := t_1 & \Leftarrow & F_1 \\ f[x] := t_2 & \Leftarrow & F_2 \\ f[x] := t_3 & \Leftarrow & \text{otherwise} \end{cases} \quad \text{or} \quad \begin{cases} H :\Leftrightarrow G_1 & \Leftarrow & F_1 \\ H :\Leftrightarrow G_2 & \Leftarrow & F_2 \\ H :\Leftrightarrow G_3 & \Leftarrow & \text{otherwise} \end{cases}$$

The rules for handling case distinctions are based on replacing of a case distinction of the form (1) by a formula of the form (2) and then apply the appropriate rule for a conjunction.

2.4.3.1 Rule ("Expansion of Case Distinction")

If a case distinction

$$\begin{cases} G_1 & \Leftarrow & F_1 \\ G_2 & \Leftarrow & F_2 \\ G_3 & \Leftarrow & \text{otherwise} \end{cases}$$

is in the knowledge base then

$$F_1 \Rightarrow G_1$$

$$\neg F_1 \wedge F_2 \Rightarrow G_2$$

$$\neg F_1 \wedge \neg F_2 \Rightarrow G_3$$

can be added to the knowledge base.

2.4.3.2 Rule ("Reduction of Case Distinction")

The rule for reduction of case distinctions in the proof goal combines the rule for reducing conjunctions with the rule for reducing implications.

For proving a case distinction

$$\begin{cases} G_1 & \Leftarrow & F_1 \\ G_2 & \Leftarrow & F_2 \\ G_3 & \Leftarrow & \text{otherwise} \end{cases}$$

we split the proof into cases:

$$\text{Prove } G_1 \text{ under the assumption } F_1$$

$$\text{Prove } G_2 \text{ under the assumption } \neg F_1 \wedge F_2$$

$$\text{Prove } G_3 \text{ under the assumption } \neg F_1 \wedge \neg F_2$$

Example:

Let for all $x \in \mathbb{R}$

$$|x| := \begin{cases} x & \Leftarrow \quad x \geq 0 \\ -x & \Leftarrow \quad \text{otherwise} \end{cases}$$

Prove

$$\forall_{x,y \in \mathbb{R}} |x + y| \leq |x| + |y|$$

Let x_0, y_0 arbitrary but fixed and show $|x_0 + y_0| \leq |x_0| + |y_0|$. We substitute the definition for absolute value on the left hand side and move the predicate symbol "$\leq$" into the case distinction and have to prove

$$\begin{cases} x_0 + y_0 \leq |x_0| + |y_0| & \Leftarrow \quad x_0 + y_0 \geq 0 \\ -(x_0 + y_0) \leq |x_0| + |y_0| & \Leftarrow \quad x_0 + y_0 < 0 \end{cases}$$

First we assume $x_0 + y_0 \geq 0$ and show $x_0 + y_0 \leq |x_0| + |y_0|$. Analogously, we substitute the definition on the right hand side for x_0 and have to prove

$$\begin{cases} x_0 + y_0 \leq x_0 + |y_0| & \Leftarrow \quad x_0 \geq 0 \\ x_0 + y_0 \leq -x_0 + |y_0| & \Leftarrow \quad x_0 < 0 \end{cases}$$

First we assume $x_0 \geq 0$ and show $x_0 + y_0 \leq x_0 + |y_0|$. We substitute the definition on the right hand side for y_0 and have to prove

$$\begin{cases} x_0 + y_0 \leq x_0 + y_0 & \Leftarrow \quad y_0 \geq 0 \\ x_0 + y_0 \leq -x_0 + (-y_0) & \Leftarrow \quad y_0 < 0 \end{cases}$$

First we assume $y_0 \geq 0$ and show $x_0 + y_0 \leq x_0 + y_0$.
Now we assume $y_0 < 0$ and show $x_0 + y_0 \leq x_0 + (-y_0)$.

Now we assume $x_0 < 0$ and show $-x_0 + y_0 \leq x_0 + |y_0|$. …

Now we assume $x_0 + y_0 < 0$ and show $-(x_0 + y_0) \leq |x_0| + |y_0|$. …

The presentation of the proof above would typically be given as follows:

Case 1: $x_0 + y_0 \geq 0,\ x_0 \geq 0,\ y_0 \geq 0$: …
Case 2: $x_0 + y_0 \geq 0,\ x_0 \geq 0,\ y_0 < 0$: …
…

2.5 Theory-Specific Proof Rules and Strategies

2.5.1 Set Theory

Set theory is an extension of predicate logic that is frequently used in every-day mathematics. Intuitively, a "set" is a collection of objects. For some set M and an object s the most interesting property to study is whether s is a member of M or not . As long as we are speaking about sets containing only finitely many objects (whatever this means …) this question is easy to answer by checking finitely many cases. However, we want to handle also collections of infinitely many objects (whatever this means …). It is desirable to define a set by a characteristic property of its elements, i.e. the set of all x satisfying the property P written as $\{x \mid P\}$. If one continues beyond this point on an intuitive level, one quickly runs into paradoxa like Russell's paradox (see lecture notes Algorithmic Methods 1) or the barber's paradox (see lecture notes Linear Algebra).

Set theory can, however, be introduced in an axiomatic fashion. Roughly speaking, set theory introduces a binary predicate '$\in$' (membership) and certain axioms that characterize the behaviour of membership. In addition, certain new function and predicate symbols are introduced through explicit definitions in terms of membership. There are several axiomatizations of set theory (Zermelo-Frankel set theory (ZF), von Neuman-Gödel-Bernays set theory (NGB), or type theory of Russell and Whitehead to name a few), which differ in the way how to avoid the well-known paradoxa, but which, on the other hand, do not affect the way how set theory is used in every-day mathematics (at least, the author is not aware of concrete situations in "usual mathematics", where it would make a difference).

We show the flavour of axiomatic set theory based on axioms of ZF. Most of the axioms deal with the existence of certain sets that are characterized by the way how membership in them can be decided, e.g. there is for each formula P with free variable x and for each set M (where M and S do not occur in P) an axiom of the form

$$\underset{S}{\exists!}\ \underset{x}{\forall}\ x \in S \Leftrightarrow (x \in M \wedge P) \quad .$$

Since, by the above axiom—the axiom scheme of separation—, we know the existence of such a set S we can define

$$\mathrm{set}[M, P] := \underset{S}{\exists!}\left(\underset{x}{\forall}\ x \in S \Leftrightarrow (x \in M \wedge P)\right) \quad . \tag{3}$$

$\mathrm{set}[M, P]$ is commonly written as

$$\{x \in M \mid P\}$$

and it should be noted that the language construct $\{x \in M \mid \ldots\}$ again *binds* the variable x in the expression and, thus, should be considered a *quantifier*. From the definition of $\mathrm{set}[M, P]$ we know

$$\underset{x}{\forall}\, x \in \mathrm{set}[M, P] \Leftrightarrow (x \in M \wedge P)$$

or, using the notation just introduced

$$\underset{y}{\forall}\, y \in \{x \in M \mid P\} \Leftrightarrow (y \in M \bigwedge P_{x \to y}) \ , \tag{4}$$

respectively. Note that we renamed the bound variable in the universal quantifier in order to distinguish it from the bound variable in the inner quantifier. Without this renaming formula (4) would read as

$$\underset{x}{\forall}\, x \in \{x \in M \mid P\} \Leftrightarrow (x \in M \wedge P) \ ,$$

which is not wrong but confusing, at least for the author. Formula (4) also tells exactly how to check membership for a set defined in this way. The ZF axiom system contains the necessary axioms that guarantee the existence of certain sets and thereby justify to introduce certain sets implicitely like in (3). The axiom of extensionality, i.e.

$$A = B \ \Leftrightarrow \ \underset{x}{\forall}\, x \in A \Leftrightarrow x \in B$$

tells when two sets are equal. A predicate '$\subseteq$' is introduced explicitely:

$$A \subseteq B :\Leftrightarrow \ \underset{x}{\forall}\, x \in A \Rightarrow x \in B \ .$$

The construction of sets of the form $\{x \mid x \in M \wedge P\}$ is allowed by the axioms of separation, the construction of sets of different shape must be claimed by axioms. In the sequel, we will introduce certain set constructions or list the rules how to test for membership in various set constructions. All definitions and all membership rules are justified by the underlying axioms of ZF.

$$\emptyset := \{x \mid x \neq x\}$$

$$A \bigcap B := \{x \mid x \in A \wedge x \in B\}$$

$$\bigcap A := \left\{x \,\middle|\, \underset{y \in A}{\forall}\, x \in y\right\}$$

$$A \bigcup B := \{x \mid x \in A \vee x \in B\}$$

$$\bigcup A := \left\{x \,\middle|\, \exists_{y \in A}\, x \in y\right\}$$

$$\mathcal{P}[A] := \{x \mid x \subseteq A\}$$

$$\{\} := \emptyset$$

$$\{a\} := \{x \mid x = a\}$$

$$\{a, b\} := \{a\} \cup \{b\} = \{x \mid x = a \vee x = b\}$$

$$\{a_1, \ldots, a_n\} := \{a_1\} \cup \ldots \cup \{a_n\} = \{x \mid x = a_1 \vee \ldots \vee x = a_n\} = \left\{x \,\middle|\, \exists_{i=1,\ldots,n}\, x = a_i\right\}$$

$$\left\{\mathfrak{T}_x \mid_{x \in M} \mathfrak{C}_x\right\} := \left\{y \,\middle|\, \exists_{x \in M}\, \mathfrak{C}_x \bigwedge y = \mathfrak{T}_x\right\}$$

For a term A_y with free variable y it is often convenient to use the abbreviations

$$\bigcap_{x \in M} A_x \quad \text{for} \quad \bigcap \left\{A_x \mid_{x \in M}\right\}$$

$$\bigcup_{x \in M} A_x \quad \text{for} \quad \bigcup \left\{A_x \mid_{x \in M}\right\}$$

These abbreviations must again be viewed as quantifiers binding x. Using these abbreviations, it can be shown that

$$a \in \bigcap_{x \in M} A_x \Leftrightarrow \forall_{x \in M}\, a \in A_x$$

$$a \in \bigcup_{x \in M} A_x \Leftrightarrow \exists_{x \in M}\, a \in A_x \quad .$$

In many cases proving with sets reduces to standard predicate logic proving using the definitions and conventions above.

2.5.2 Natural Numbers

2.5.2.1 Induction Proofs for Universally Quantified Goals over the Natural Numbers

Natural numbers allow one special proof technique, namely induction. The principle of proving by induction is derived from the induction axioms for natural numbers, i.e. for each formula A_x with free variable x a formula of the form

$$\left(A_{x\to 0} \bigwedge \underset{n\in\mathbb{N}}{\forall} A_{x\to n} \Rightarrow A_{x\to n+1}\right) \Rightarrow \underset{x\in\mathbb{N}}{\forall} A_x$$

where n does not occur in A. An induction axiom can be used to *reduce* a proof goal (see "Goal Reduction using Implications")

$$\underset{x\in\mathbb{N}}{\forall} A_x$$

to proving

$$A_{x\to 0} \tag{5}$$

$$\underset{n\in\mathbb{N}}{\forall} A_{x\to n} \Rightarrow A_{x\to n+1} \tag{6}$$

Formula (5) is called the induction base, for proving (6) one assumes

$$A_{x\to n_0} \tag{7}$$

and proves

$$A_{x\to n_0+1} \tag{8}$$

for arbitrary but fixed $n_0 \in \mathbb{N}$. Assumption (7) is called *induction hypothesis*, the proof goal (8) is called *induction step*. Intuitively, it should be clear that after having proved (5) and (6) the formula really holds *for all natural numbers*: From (6) we know in particular

$$A_{x\to 0} \Rightarrow A_{x\to 1}$$

and from this, together with (5) by Modus Ponens,

$$A_{x\to 1} \quad ,$$

by this, together with the instance of (6)

$$A_{x\to 1} \Rightarrow A_{x\to 2}$$

by Modus Ponens

$$A_{x\to 2}$$

etc. and we can continue application of Modus Ponens together with appropriate instances of (6) in order to derive $A_{x\to n}$ for all natural numbers n, i.e. A_x is true for each assignment for x, i.e. $\underset{x\in\mathbb{N}}{\forall} A_x$ is true according to the semantics of the $\forall$-quantifier.

Here we considered the set of natural numbers to start with 0. However, for some applications it can be convenient to consider 1 the smallest natural

number. $\mathbb{N}_0$ is in this case commonly written for $\mathbb{N} \cup \{0\}$. The induction axioms are available with respective modification, i.e. the induction base is then

$$A_{x\to 1} \quad .$$

In general, using the induction base

$$A_{x\to s}$$

and induction hypothesis and induction step as usual (with the additional assumption that the arbitrary but fixed n_0 satisfies $n_0 \geq s$) can be used for proving

$$\forall_{\substack{x\in\mathbb{N}\\ x\geq s}} A_x \quad .$$

2.5.2.2 Comment on Presentation of Inference Chains

It can often be observed that an argument as above is presented in the following way (see also Section 2.3.2.16):

"We know $A_{x\to 0} \Rightarrow A_{x\to 1} \Rightarrow A_{x\to 2} \Rightarrow \ldots \Rightarrow A_{x\to n}$ "

Note that "⇒" in this case is *not* the propositional connective "implication" but it is just a symbol abbreviating the word "therefore" (or similar). In fact, it is not that we only know $A_{x\to 0} \Rightarrow A_{x\to 1} \Rightarrow A_{x\to 2} \Rightarrow \ldots \Rightarrow A_{x\to n}$ but we know *all of* $A_{x\to 0}$, $A_{x\to 1}$, $A_{x\to 2}$, …, $A_{x\to n}$. Consider alternatively,

$$\text{"We know } A_{x\to 0} \underset{\uparrow\, A_{x\to 0}\Rightarrow A_{x\to 1}}{\Longrightarrow} A_{x\to 1} \underset{\uparrow\, A_{x\to 1}\Rightarrow A_{x\to 2}}{\Longrightarrow} A_{x\to 2} \underset{\uparrow\, A_{x\to 2}\Rightarrow A_{x\to 3}}{\Longrightarrow} \ldots \underset{\uparrow\, A_{x\to n-1}\Rightarrow A_{x\to n}}{\Longrightarrow} A_{x\to n} \text{ "} ,$$

which is now completely confusing, because the "⇒" in the top line is "therefore" and the "⇒" in the underscript is "implication". One could live with

$$\text{"We know } A_{x\to 0} \underset{\substack{\uparrow\\ \text{by an instance}\\ \text{of (6)}}}{\Longrightarrow} A_{x\to 1} \underset{\substack{\uparrow\\ \text{by an instance}\\ \text{of (6)}}}{\Longrightarrow} A_{x\to 2} \underset{\substack{\uparrow\\ \text{by an instance}\\ \text{of (6)}}}{\Longrightarrow} \ldots \underset{\substack{\uparrow\\ \text{by an instance}\\ \text{of (6)}}}{\Longrightarrow} A_{x\to n} \text{ "} ,$$

and a convention that "$\underset{\uparrow}{\Rightarrow}$" always means "therefore, by …" instead of "implication". Whatever notation you use, always follow the principle:

Save the time of the reader!

i.e. mathematical notation should help the reader to understand the text rather than making it more difficult to follow the ideas.

2.5.2.3 Complete Induction

The principle of *complete induction* is a consequence of the induction axioms given in the previous section. The axiom reads

$$\left(\underset{n\in\mathbb{N}}{\forall} \left(\underset{\substack{m\in\mathbb{N} \\ m<n}}{\forall} A_{x\to m} \right) \Rightarrow A_{x\to n} \right) \Rightarrow \underset{x\in\mathbb{N}}{\forall} A_x$$

where m and n do not occur in A and x is a free variable in A. By this axiom, the proof of

$$\underset{x\in\mathbb{N}}{\forall} A_x$$

reduces to proving

$$\left(\underset{\substack{m\in\mathbb{N} \\ m<n_0}}{\forall} A_{x\to m} \right) \Rightarrow A_{x\to n_0}$$

for arbitrary but fixed n_0 . At this point, typically, a case distinction is made:
Case 1 (Induction base): $n_0 = 0$. We have to show

$$A_{x\to 0}$$

Case 2: $n_0 \geq 1$. We assume (induction hypothesis)

$$\underset{\substack{m\in\mathbb{N} \\ m<n_0}}{\forall} A_{x\to m}$$

and show (induction step)

$$A_{x\to n_0} \quad .$$

2.5.3 Tuples

Similar to the domain of natural numbers, we can use an induction principle for tuples. In a first formulation, an induction principle for tuples allows to prove a formula

$$\underset{\text{is-tuple}[x]}{\underset{x}{\forall}} A_x$$

by proving

$$A_{x \to \langle\rangle} \tag{9}$$

$$\forall_{\substack{\tau \\ \text{is-tuple}[\tau]}} \forall_{y} A_{x \to \tau} \Rightarrow A_{x \to \tau \frown y} \quad \text{or} \quad \forall_{\substack{\tau \\ \text{is-tuple}[\tau]}} \forall_{y} A_{x \to \tau} \Rightarrow A_{x \to y \smile \tau} \tag{10}$$

Formula (9) is called the induction base, for proving (10) one assumes

$$A_{x \to \tau_0} \tag{11}$$

and proves

$$A_{x \to \tau_0 \frown y_0} \quad \text{or} \quad A_{x \to y_0 \smile \tau_0} \tag{12}$$

for arbitrary but fixed y_0 and an arbitrary but fixed tuple τ_0, where $y \smile \tau$ and $\tau \frown y$ stand for "the object y prepended/appended to the tuple τ", respectively. Assumption (11) is again called *induction hypothesis*, the proof goal (12) is again called *induction step*. Whether one chooses to prepend or append an object in the induction step depends in most of the cases on the structure of the formula or on additional knowledge in the knowledge base.

Similar to natural numbers, there is also a *complete induction* for tuples. In order to prove

$$\forall_{\substack{x \\ \text{is-tuple}[x] \wedge |x| \geq L}} A_x$$

it is sufficient to prove

$$\forall_{\substack{x \\ \text{is-tuple}[x] \wedge |x| = L}} A_x$$

$$\forall_{l > L} \forall_{\substack{x \\ \text{is-tuple}[x] \wedge |x| < l}} A_x \Rightarrow \forall_{\substack{x \\ \text{is-tuple}[x] \wedge |x| = l}} A_x$$

In this case, the induction base is then to prove

$$A_{x \to \tau_0}$$

for an arbitrary but fixed tuple τ_0 of length L. As the induction hypothesis, we assume (for arbitrary but fixed $l_1 > L$)

$$\forall_{\substack{x \\ \text{is-tuple}[x] \wedge |x| < l_1}} A_x$$

and show in the induction step

$A_{x \to \tau_1}$

for an arbitrary but fixed tuple τ_1 of length l_1.

2.5.4 Polynomials

Since we saw (see lecture notes Algorithmic Methods 1) that polynomials can be represented by (coefficient) tuples, induction over tuples is a commonly used prove technique for polynomials. Since the length of the tuple corresponds to the degree of the polynomial, tuple induction is in this context often referred to as "induction over the degree of the polynomial": First prove the formula for all polynomials of degree 0, then assume the formula for all polynomials of degree n and prove the formual for all polynomials of degree $n+1$, or in the case of complete induction: show the formula for all polynomials of degree n assuming the induction hypothesis for all polynomials of degree less than n.

2.5.4.1 Organization