First-Order Logic

Part 2: Reasoning
JKU Course "Logic"

Wolfgang Schreiner   Wolfgang Windsteiger

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
Wolfgang.Schreiner@risc.jku.at / Wolfgang.Windsteiger@risc.jku.at

Winter Semester 2019 / July 24, 2019

Abstract

These lecture notes discuss the methodology of reasoning in (first-order) predicate logic. We assume that the readers are familiar with Part 1 of the lecture notes "First-Order Logic" about syntax and semantics of first-order predicate logic. This course is given in the bachelor program "Computer Science" of the Johannes Kepler University Linz.

Contents

1 Motivation

Our aim is to find out whether a certain statement G is generally true or at least true under certain assumptions $F_1, \dots, F_n$. “Truth” is meant here in the sense of semantics, see Part 1, in other words, we want to know whether $\models G$ or $F_1, \dots, F_n \models G$. According to our definitions of “$\models$” this would require to check *all possible models*, and since there are infinitely many models, this is not an option. Even if we fix one model, i.e., we fix the meaning of all constants, all function symbols, and all predicate symbols, there would still be potentially infinitely many choices as soon we work over an infinite domain and we have quantifiers, see the semantics of quantifier formulas in Part 1.

The idea is now to develop a *finite inferencing mechanism*, a so-called *proof calculus*, and *instead of checking* $F_1, \dots, F_n \models G$ according to the semantics of predicate logic *we derive* the sequent $F_1, \dots, F_n \vdash G$ in the proof calculus. A crucial property of a proof calculus is its *soundness*, meaning that if $F_1, \dots, F_n \vdash G$ can be derived then $F_1, \dots, F_n \models G$ holds. In other words, everything that can be derived in a sound proof calculus is actually true. The opposite direction, if $F_1, \dots, F_n \models G$ then $F_1, \dots, F_n \vdash G$ can be derived, is called *completeness*. In other words, in a complete proof calculus, every true statement can be formally derived. We call a derivation of $F_1, \dots, F_n \vdash G$ a *proof* of G based on the assumptions $F_1, \dots, F_n$. Then G is called a *theorem*, whereas we call it a *conjecture* as long as we do not have a proof.

It will be interesting to observe that the inference rules are purely *syntactic*, they only depend on the syntactic structure of $F_1, \dots, F_n$ and G and they do not involve any semantics. As a consequence, the correctness of a formal proof in a proof calculus can easily be *checked*, even by a computer.

2 Proofs and Proof Search

We view a proof of a statement G as a *series of logical arguments* that ascertain that G is true (under certain assumptions $F_1, \dots, F_n$). Hence, a proof can be understood as a chain of *proof situations*, where a single proof situation captures the “status” during a proof, and the transition from one situation to another corresponds to one logical argument. Pictorially speaking[1], one might view a proof as something of the kind

$$\boxed{\text{initial situation}} \longrightarrow \boxed{\sigma_1} \longrightarrow \boxed{\sigma_2} \longrightarrow \cdots \longrightarrow \boxed{\sigma_n} \longrightarrow \boxed{\text{final situation}}$$

Such a chain of proof situations has two natural interpretations.

Forward interpretation: A proof starts from an initial situation, then progresses step-by-step via further situations $\sigma_1, \sigma_2, \dots, \sigma_n$, until it finally arrives at the final situation $F_1, \dots, F_n \vdash G$.

[1] This picture should only give a rough idea, we will soon see that the situation will, in fact, be a little more tricky.

Backward interpretation: A proof starts from the final situation $F_1, \ldots, F_n \vdash G$ to be proved, then progresses step-by-step by reducing the proof situation to $\sigma_n, \ldots, \sigma_2, \sigma_1$, until it finally arrives at the initial situation.

When writing down a proof it is mostly a matter of taste whether one prefers the forward or the backward style.

The transition from one proof situation to the next is guided by *inference rules*, we also say *proof rules*. Inference rules are denoted by

$$\frac{S_1 \qquad \ldots \qquad S_n}{S}$$

where $S_1, \ldots, S_n$ and S are sequents, i.e., proof situations. We call $S_1, \ldots, S_n$ the *premises* and S the *conclusion* of the rule. Again, such a rule has two natural ways of reading.

Forward interpretation: If $S_1, \ldots, S_n$ can be proved, then also S can be proved.

Backward interpretation: In order to prove S, we need to prove $S_1, \ldots, S_n$.

As a very simple example, let S_1 and S be the proof situations

$$\begin{aligned} S_1 &:= \text{“the assumption } A \text{ leads to a contradiction”} \\ S &:= \text{“the statement } \neg A \text{ holds”.} \end{aligned}$$

Then the inference rule

$$\frac{S_1}{S}$$

captures the simple logical argument that

(forward interpretation) if the assumption A leads to a contradiction then $\neg A$ holds. In other words,

(backward interpretation) if we want to prove $\neg A$ then we assume A and derive a contradiction.

As another example, let S_1, S_2, and S be the proof situations

$$\begin{aligned} S_1 &:= \text{“the statement } A \text{ holds”} \\ S_2 &:= \text{“the statement } B \text{ holds”} \\ S &:= \text{“the statement } A \wedge B \text{ holds”.} \end{aligned}$$

Then the inference rule

$$\frac{S_1 \qquad S_2}{S}$$

captures the simple logical argument that

(forward interpretation) if we can prove A *and* we can prove B then we can also prove $A \wedge B$. In other words,

(backward interpretation) if we want to prove $A \wedge B$ then we have to prove both A and B.

As mentioned above, a proof is a chain of proof situations that are "connected" through inference rules. Consider the following abstract inference rules:

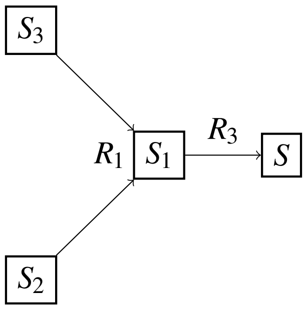
$$R_1\colon \frac{S_2 \qquad S_3}{S_1} \qquad R_2\colon \frac{}{S_4} \qquad R_3\colon \frac{S_1}{S} \qquad R_4\colon \frac{}{S_5}$$

$$R_5\colon \frac{S_4 \qquad S_5}{S_2} \qquad R_6\colon \frac{}{S_6} \qquad R_7\colon \frac{S_6}{S_3}$$

Imagine we want to *derive the sequent* S. In order to achieve this we need a chain of sequents with final sequent S. Since the only rule with conclusion S is R_3, the last segment in our chain must be

$$\boxed{S_1} \xrightarrow{R_3} \boxed{S}$$

Using rule R_1 the chain segment becomes



and we would now continue the process with S_2 and S_3. Actually, we see already in this example that, due to the possibility of multiple premises in an inference rule, our picture of a proof as a linear chain of sequents was a bit simplistic. A better picture for a proof is that of a *rooted tree* with the sequent to be derived as the root. Continuing the example from above, the proof tree looks as follows:

$$R_3\colon \frac{R_1\colon \dfrac{R_5\colon \dfrac{R_2\colon \dfrac{}{S_4} \qquad R_4\colon \dfrac{}{S_5}}{S_2} \qquad R_7\colon \dfrac{R_6\colon \dfrac{}{S_6}}{S_3}}{S_1}}{S}$$

Note that all three branches of the tree end with an “empty sequent” due to the “empty premises” in rules R_2, R_4, and R_6. These three empty sequents play the role of the “initial situation” containing trivial proof situations, which can obviously be derived. An empty sequent represents a proof status, where there is nothing to be done, which we simply *declare* to be derivable.

Proof Trees More formally, a *formal proof* is a finite tree, where

1. every node is a sequent,
2. if $S_1, \ldots, S_n$ are the children nodes of a node S, then there must be an inference rule of the form $\dfrac{S_1 \;\ldots\; S_n}{S}$.

A *formal proof of* R is a proof tree with root R.

Special case: A *leaf node* in a tree is a node that has no children, hence

for every leaf S in the tree there must be a rule $\dfrac{}{S}$ with empty premises.

Proof Search A proof tree is a tree whose structure is governed by inference rules in the sense that every relation between a node and its children nodes must be justified by an inference rule. If we need to construct a proof of R, we can start with a trivial tree containing only one node, namely the root node R, and starting from there construct an admissible tree using appropriate inference rules as the guiding principle.

```
Input: $R$
Output: $P$ such that $P$ is a formal proof of $R$.

$P$ := tree containing only the root node $R$
$Q$ := $\{R\}$ // the leaves of the tree, i.e., the still “open” proof situations
while $Q \neq \emptyset$
    choose $T \in Q$ and choose a rule $\frac{S_1 \;\ldots\; S_n}{S}$ such that $S = T$
    replace $T$ in $Q$ by $S_1, \ldots, S_n$
    add $S_1, \ldots, S_n$ as children nodes of $T$ in $P$
return $P$
```

Depending on

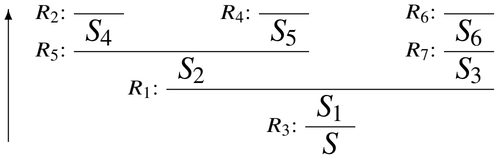
1. the rules and
2. the choice of the rule in the loop,

the procedure might not terminate. Moreover, it is not guaranteed that in every pass of the loop there actually exists an appropriate rule to choose. If, e.g., for $T \in Q$ there is no rule that fits, then the procedure will not be able to generate a complete proof tree, since T will then be a leaf node in P without a justifying rule, violating condition 2 for proof trees.

> Proving is the art of selecting the “right” rule applications to elaborate, from the desired goal as a root, a complete proof tree.

Proof Generation vs Proof Presentation Following the proof search procedure just described, it is clear that a proof tree is generated from root to the leaves. When presenting a proof, however, it is typically *not the proof tree* that is shown, but it is the “chain of logical arguments” that is encoded as that tree. Hence, the presentation of a proof requires the translation of the proof tree into some form of natural language. We observe two different styles of proof presentation, a *backward style* and a *forward style*.

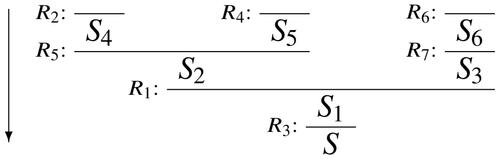
Backward style proof presentation: In this style the presentation of the proof follows the way it was generated, i.e., we start with sequent to be proved, then work backwards applying inference rules from bottom to top (backward interpretation, see above).



Presentation: In order to prove S, by R_3, we have to prove S_1. For this, by R_1, we

1. prove S_2: by R_5 we have to prove S_4 and S_5, which are guaranteed by R_2 and R_4, respectively. Now we still have to
2. prove S_3: by R_7 it is sufficient to prove S_6, which we know from R_6. QED (“quod erat demonstrandum”, “what was to be proved”).

Forward style proof presentation: In this style the presentation of the proof starts from known facts and works forward, applying rules from top to bottom (forward interpretation, see above), until the sequent to be proved is reached.



Presentation: We know S_4 and S_5 can be proved, hence by R_5, S_2 can be proved. Furthermore we know that S_6 can be proved, hence by R_7, also S_3 can be proved. Together with S_2, by R_1, we know that S_1 can be proved, and therefore, by R_3, also S. QED.

Backward versus forward, which is better? We will not give a definite answer to this question, both styles have their advantages and disadvantages. Which method to choose also depends on the goals that we want to achieve by showing a certain proof. If we want to convince the audience that a certain statement is true, then forward style is good, because for many people the forward interpretation of inference rules better reflects the “logical flow of an argument”.

If, on the other hand, we want the audience to learn, how they could do their own proofs, then backward style is good, because it corresponds exactly to the way how the proof was constructed. In forward style, it is often unclear, why we start with certain facts (S_4, S_5, and S_6 in the example above), and the direction, in which we go, is only justified at the end, when we somehow "magically" or "luckily" exactly reach the desired statement.

3 Inference Rules and Rule Matching

Inference rules are given as schematic "patterns" of the form

$$\textit{name:} \quad \frac{K_1 \ldots \vdash G_1 \qquad \ldots \qquad K_n \ldots \vdash G_n}{K \ldots \vdash G}$$

where $n \in \mathbb{N}_0$. There can be several premises, including the special case $n = 0$ meaning there are *no premises* at all, but there is always *exactly one* conclusion. The premises and conclusions are "patterns" of sequents with schematic *variables*, namely

sequence variables (e.g. $K \ldots$) that match an arbitrary sequence of formulas,

formula variables (e.g. F, G, etc.) that match an arbitrary formula, and

term variables (e.g. t, u, etc.) that match an arbitrary term.

As an example,

$$K \ldots \vdash G_1 \land G_2$$

matches a sequent with arbitrarily many assumptions, whose goal is an arbitrary conjunction, i.e., a formula with "$\land$" as the *outermost symbol* and two subformulas G_1 and G_2. Note that the order of assumptions does *not* matter, meaning that, e.g.,

$$K \ldots, F_1 \land F_2 \vdash G$$

matches as soon as a conjunction occurs among the assumptions. Moreover, multiple occurrences of the same variable denote the *same* expression all over the inference rule, meaning that, e.g.,

$$K \ldots, F \land G \vdash G$$

matches if, among the assumptions, there is a conjunction whose second subformula is identical to the goal. In the proof search procedure, inference rules have to be "matched" against concrete proof situations. This means, that in the line

- choose $T \in Q$ and choose a rule $\dfrac{S_1 \ \ldots \ S_n}{S}$ such that $S = T$

in the proof search procedure, we need to replace $S = T$ by "S matches T", i.e., there is a substitution for all variables in S that makes S equal to T. Then, in the subsequent lines in the procedure

- replace T in Q by $S_1, \ldots, S_n$
- add $S_1, \ldots, S_n$ as children nodes of T in P

this substitution must of course be applied to the sequents $S_1, \ldots, S_n$ as well.

As an example, let

- a be a constant,
- f, g unary function symbols,
- p, q, r, s unary predicate symbols, and
- t a schematic term variable,

and consider the following inference rules:

$$R_1\colon \frac{\vdash r(t) \qquad \vdash s(g(t))}{\vdash r(f(t))} \qquad R_2\colon \frac{}{\vdash p(a)} \qquad R_3\colon \frac{\vdash r(f(t))}{\vdash s(t)} \qquad R_4\colon \frac{}{\vdash q(a)}$$

$$R_5\colon \frac{\vdash p(t) \qquad \vdash q(t)}{\vdash r(t)} \qquad R_6\colon \frac{}{\vdash s(f(a))} \qquad R_7\colon \frac{\vdash s(f(t))}{\vdash s(g(t))}$$

We want to *derive* $\vdash s(a)$, hence, we start with $\vdash s(a)$ as the root of the tree and $Q = \{\vdash s(a)\}$. We choose $T := \vdash s(a) \in Q$ and the only rule that matches is R_3 with the substitution $t \mapsto a$, and we have to

- replace $\vdash s(a)$ in Q by $\vdash r(f(a))$
- add $\vdash r(f(a))$ as child node of T in P

Note that $\vdash r(f(a))$ is the result of applying $t \mapsto a$ to the conclusion $\vdash r(f(t))$ of R_3. Thus,

$$Q = \{\vdash r(f(a))\} \qquad\qquad P = \frac{\vdash r(f(a))}{\vdash s(a)}$$

We choose $T := \vdash r(f(a)) \in Q$ and now

- R_1 matches with $t \mapsto a$ and
- R_5 matches with $t \mapsto f(a)$.

Using R_5 would give us

$$Q = \{\vdash p(f(a)), \vdash q(f(a))\},$$

and we could not choose any rule that would fit in the next pass of the loop. Therefore, we choose R_1 with the substitution $t \mapsto a$ and we get

$$Q = \{\vdash r(a), \vdash s(g(a))\} \qquad\qquad P = \frac{\dfrac{\vdash r(a) \qquad \vdash s(g(a))}{\vdash r(f(a))}}{\vdash s(a)}$$

We choose $T := \ \vdash r(a) \in Q$. Now, R_5 matches with $t \mapsto a$ and we get

$$Q = \{\, \vdash p(a),\ \vdash q(a),\ \vdash s(g(a))\} \qquad P = \dfrac{\dfrac{\dfrac{\vdash p(a) \qquad \vdash q(a)}{\vdash r(a)} \qquad \vdash s(g(a))}{\vdash r(f(a))}}{\vdash s(a)}$$

We choose $T := \ \vdash p(a) \in Q$ such that R_2 matches (without any substitution) and, since R_2 has empty premises, we get

$$Q = \{\, \vdash q(a),\ \vdash s(g(a))\} \qquad P = \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash p(a)} \qquad \vdash q(a)}{\vdash r(a)} \qquad \vdash s(g(a))}{\vdash r(f(a))}}{\vdash s(a)}$$

We choose $T := \ \vdash q(a) \in Q$ such that R_4 matches (without any substitution) and, since R_4 has empty premises, we get

$$Q = \{\, \vdash s(g(a))\} \qquad P = \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash p(a)} \qquad \dfrac{}{\vdash q(a)}}{\vdash r(a)} \qquad \vdash s(g(a))}{\vdash r(f(a))}}{\vdash s(a)}$$

We choose $T := \ \vdash s(g(a)) \in Q$. Now, R_7 matches with $t \mapsto a$ and we get

$$Q = \{\, \vdash s(f(a))\} \qquad P = \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash p(a)} \qquad \dfrac{}{\vdash q(a)}}{\vdash r(a)} \qquad \dfrac{\vdash s(f(a))}{\vdash s(g(a))}}{\vdash r(f(a))}}{\vdash s(a)}$$

We choose $T := \ \vdash s(f(a)) \in Q$ such that R_6 matches (without any substitution) and, since R_6 has empty premises, we get

$$Q = \emptyset \qquad P = \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash p(a)} \qquad \dfrac{}{\vdash q(a)}}{\vdash r(a)} \qquad \dfrac{\dfrac{}{\vdash s(f(a))}}{\vdash s(g(a))}}{\vdash r(f(a))}}{\vdash s(a)}$$

and the while-loop terminates. Note, that in the final pass rule R_3 would also match but, of course, we will always prefer a rule with empty premises, because this “closes a branch”. In general, we prefer rules with fewer premises over those with more premises, because with fewer branches we expect less work, although this is not true in general. The final proof tree is now

$$R_3\!:\ \dfrac{R_1\!:\ \dfrac{R_5\!:\ \dfrac{R_2\!:\ \dfrac{}{\vdash p(a)} \qquad R_4\!:\ \dfrac{}{\vdash q(a)}}{\vdash r(a)} \qquad R_7\!:\ \dfrac{R_6\!:\ \dfrac{}{\vdash s(f(a))}}{\vdash s(g(a))}}{\vdash r(f(a))}}{\vdash s(a)}$$

4 The Inference Rules of First-Order Logic

For the envisioned proof calculus for predicate logic, one could give a (minimal) set of inference rules for first-order predicate logic, which can be shown to be *sound* and *complete*, i.e.,

1. every formula, which has a formal proof, is also semantically true and
2. every semantically true formula has a formal proof.

Such inference systems can be found in the literature, e.g., *sequent calculus*, *Gentzen calculus*, or the *natural deduction calculus*. However, we are interested in proof rules that help in *practical proving* of mathematical statements and *checking of given proofs*, and we do not care about completeness of the calculus[2]. In our rule set we distinguish:

Propositional rules: closing rules, structural rules, connective rules.

Predicate logic rules: equality rules, quantifier rules.

We will see that the calculus contains for every logical connective and every standard quantifier at least one rule, where the connective or quantifier occurs as the outermost symbol in the goal or one of the assumptions. The benefit of applying these rules is that the proof gets simpler in each step in the sense that in at least one formula in one of the "open sequents" the nesting depth decreases.

4.1 Closing Rules and Structural Rules

It is important for the termination of the proof search procedure to have rules with empty premises, because only these can "close" a branch in a proof.

- If the goal is among the assumptions, the goal can be proved.

$$\text{GoalAssum: } \frac{}{K \ldots, G \vdash G}$$

- If the assumptions are contradictory, any goal can be proved.

$$\text{ContrAssum: } \frac{}{K \ldots, A, \neg A \vdash G}$$

- If the assumptions include "false", any goal can be proved.

$$\text{FalseAssum: } \frac{}{K \ldots, \bot \vdash G}$$

Furthermore, there are several rules that "rearrange formulas" within a sequent.

- Any assumption may be dropped:

[2] We do care about soundness, though, because we do not want to be able to prove false statements!

$$\text{Drop: } \frac{K \ldots \vdash G}{K \ldots, A \vdash G}$$

- Any assumption may be added, if it is also proved (the “cut-rule”):

$$\text{Cut: } \frac{K \ldots \vdash A \qquad K \ldots, A \vdash G}{K \ldots \vdash G}$$

In a proof presentation, we would then write something like:

> We have to prove G. First we prove A: .... Now we prove G with the additional assumption A.

It should be understood that the cut-rule needs “an idea” for a formula A to be introduced. Of course, the rule is *correct* for any A, but the big question is, which A would help in the proof of G. The rule itself does not tell us, which A would be useful.

- Rather than proving G, we may assume $\neg G$ and derive a contradiction (an “indirect proof”, “proof by contradiction”):

$$\text{Indirect: } \frac{K \ldots, \neg G \vdash \bot}{K \ldots \vdash G}$$

In a proof presentation, we would then write something like:

> We have to prove G. Thus we may assume $\neg G$ and derive a contradiction.

Note that this rule can *always be applied*, since the conclusion matches any sequent. Moreover, the rule introduces a negation, such that the nesting depth of the formula *increases*. Therefore, this rule should be applied with care. Typically, we use an indirect proof as a “last ressort” if nothing else appears to work successfully.

4.2 Connective Rules (Quantifier-free)

Negation

- Prove a negation as goal:

$$\text{P-}\neg\text{: } \frac{K \ldots, G \vdash \bot}{K \ldots \vdash \neg G}$$

In a proof presentation, we would then write something like:

> We have to prove $\neg G$. Thus we may assume G and derive a contradiction.

This rule is just a special case of an indirect proof: if the goal is a negation, then it is a good idea to use an indirect proof.

- Use a negation as an assumption:

$$\text{A-}\neg\text{:}\ \frac{K\ldots,\neg G \vdash A}{K\ldots,\neg A \vdash G}$$

In a proof presentation, we would then write something like:

> We know $\neg A$ and have to prove G. Thus we may assume $\neg G$ and prove A.

This rule is just a combination of an indirect proof with cut-rule and contradicting assumptions.

Suppose we had to prove that $\sqrt{2}$ is not a rational number. As a first step in our proof, we would recognize that the goal "$\sqrt{2}$ is *not* a rational number" has the form of a *negation*, thus we would apply the rule P-¬:

$$\text{P-}\neg\text{:}\ \frac{\sqrt{2} \in \mathbb{Q}, \ldots \vdash \bot}{\ldots \vdash \sqrt{2} \notin \mathbb{Q}}$$

We assume $\sqrt{2}$ *was* a rational number and derive a contradiction. We cannot continue this proof at this point, because we lack appropriate rules to carry on.

Conjunction

- Prove a conjunction as a goal:

$$\text{P-}\wedge\text{:}\ \frac{K\ldots \vdash F_1 \qquad K\ldots \vdash F_2}{K\ldots \vdash F_1 \wedge F_2}$$

In a proof presentation, we would then write something like:

> We have to prove $F_1 \wedge F_2$. First we prove F_1: .... Now we prove F_2: ....

- Use a conjunction as an assumption:

$$\text{A-}\wedge\text{:}\ \frac{K\ldots,F_1,F_2 \vdash G}{K\ldots,F_1 \wedge F_2 \vdash G}$$

In a proof presentation, we would then write something like:

> We know $F_1 \wedge F_2$, therefore we know F_1 and we know F_2.

Disjunction

- Prove a disjunction as a goal:

$$\text{P-}\vee\text{:}\ \frac{K\ldots,\neg F_1 \vdash F_2}{K\ldots \vdash F_1 \vee F_2} \qquad \text{P-}\vee\text{:}\ \frac{K\ldots,\neg F_2 \vdash F_1}{K\ldots \vdash F_1 \vee F_2}$$

In a proof presentation, we would then write something like:

> We have to prove $F_1 \vee F_2$. Thus we may assume $\neg F_1$ and prove $\neg F_2$. (or: Thus we may assume $\neg F_2$ and prove $\neg F_1$).

- Use a disjunction as an assumption ("proof by cases"):

$$\text{A-}\vee\text{:}\ \frac{K \ldots, F_1 \vdash G \qquad K \ldots, F_2 \vdash G}{K \ldots, F_1 \vee F_2 \vdash G}$$

In a proof presentation, we would then write something like:

> We know $F_1 \vee F_2$. We proceed by case distinction. Case F_1: .... Case F_2: ....

A typical example for the A-∨ rule is in proofs involving natural numbers, which frequently use a case distinction between even and odd numbers. Actually, this case distinction stems from the knowledge $even(m) \vee odd(m)$, which is available for any natural number m. More formally, written as a proof tree

$$\text{A-}\vee\text{:}\ \frac{\dfrac{P_1}{even(m) \vdash G} \qquad \dfrac{P_2}{odd(m) \vdash G}}{even(m) \vee odd(m) \vdash G}$$

and the presentation of the proof would be something like:
We already know that m is even or m is odd. Thus, we can distinguish the two cases:

1. m is even: ... (insert proof P_1 here)
2. m is odd: ... (insert proof P_2 here)

Implication

- Prove implication as a goal.

$$\text{P-}\longrightarrow\text{:}\ \frac{K \ldots, F_1 \vdash F_2}{K \ldots \vdash F_1 \longrightarrow F_2}$$

In a proof presentation, we would then write something like:

> We have to prove $F_1 \longrightarrow F_2$. Thus we may assume F_1 and prove F_2.

- Use an implication as an assumption:

$$\text{A-}\longrightarrow\text{:}\ \frac{K \ldots \vdash F_1 \qquad K \ldots, F_2 \vdash G}{K \ldots, F_1 \longrightarrow F_2 \vdash G}$$

In a proof presentation, we would then write something like:

> We know $F_1 \longrightarrow F_2$. First we prove F_1: .... Now we know F_2.

- Often used instead: "modus ponens" and "modus tollens"

$$\text{MP: } \frac{K \ldots, F_1, F_2 \vdash G}{K \ldots, F_1 \longrightarrow F_2, F_1 \vdash G} \qquad \text{MT: } \frac{K \ldots, \neg F_2, \neg F_1 \vdash G}{K \ldots, F_1 \longrightarrow F_2, \neg F_2 \vdash G}$$

In a proof presentation, we would then write something like:

modus ponens: We know $F_1 \longrightarrow F_2$ and we know F_1. Therefore we know F_2.

modus tollens: We know $F_1 \longrightarrow F_2$ and we know $\neg F_2$. Therefore we know $\neg F_1$.

Whether to use A-$\longrightarrow$ or modus ponens/modus tollens depends on the concrete example. If we *can* apply MP or MT, then we use them, because in the general rule, there will always be one branch to turn out as being trivial from the beginning. Still, the general rule can be helpful, because, even if neither MP nor MT can be applied, the presence of an implication among the assumptions can lead to the "idea" to proof a subgoal F_1 and then continue the proof using this new information in the knowledge base.

As an example, if we want to prove $((A \longrightarrow (B \lor C)) \land \neg C) \longrightarrow (A \longrightarrow B)$, where A, B, and C are abbreviations for complex predicate logic formulas, we develop the following proof tree[3].

$$\begin{array}{rcc}
 & \multicolumn{2}{c}{\vdash ((A \longrightarrow (B \lor C)) \land \neg C) \longrightarrow (A \longrightarrow B)} \\ \hline
\text{P-}\longrightarrow\text{:} & \multicolumn{2}{c}{(A \longrightarrow (B \lor C)) \land \neg C \vdash A \longrightarrow B} \\ \hline
\text{A-}\land\text{:} & \multicolumn{2}{c}{A \longrightarrow (B \lor C), \neg C \vdash A \longrightarrow B} \\ \hline
\text{P-}\longrightarrow\text{:} & \multicolumn{2}{c}{A \longrightarrow (B \lor C), \neg C, A \vdash B} \\ \hline
\text{MP:} & \multicolumn{2}{c}{\neg C, A, B \lor C \vdash B} \\ \hline
\text{A-}\lor\text{:} & \ldots, B \vdash B & \ldots, \neg C, C \vdash B \\ \hline
 & \text{GoalAssum:} & \text{ContrAssum:}
\end{array} \qquad \downarrow$$

Equivalence

- Prove equivalence as a goal:

$$\text{P-}\leftrightarrow\text{: } \frac{K \ldots \vdash F_1 \longrightarrow F_2 \qquad K \ldots \vdash F_2 \longrightarrow F_1}{K \ldots \vdash F_1 \leftrightarrow F_2}$$

In a proof presentation, we would then write something like:

We prove $F_1 \leftrightarrow F_2$. First we prove $F_1 \longrightarrow F_2$: ... Now we prove $F_2 \longrightarrow F_1$: ...

- Use equivalence as an assumption ("substitution")[4]:

[3] Note, that in this example we now write the tree top-down with root on top, which is convenient in practice, because then we need not know in advance how much space to reserve for the tree. Although it should be clear anyway, what is root and what are the leaves, we indicate this variant by the "↓" at the beginning.

[4] We use the notation $\Gamma[F_2/F_1]$ for the result of replacing in some formula(s) in Γ some occurrence of formula F_1 by formula F_2.

$$\text{A-}\leftrightarrow\text{:}\ \frac{K\ldots[F_2/F_1], F_1 \leftrightarrow F_2 \vdash G}{K\ldots, F_1 \leftrightarrow F_2 \vdash G} \qquad \text{A-}\leftrightarrow\text{:}\ \frac{K\ldots, F_1 \leftrightarrow F_2 \vdash G[F_2/F_1]}{K\ldots, F_1 \leftrightarrow F_2 \vdash G}$$

$$\text{A-}\leftrightarrow\text{:}\ \frac{K\ldots[F_1/F_2], F_1 \leftrightarrow F_2 \vdash G}{K\ldots, F_1 \leftrightarrow F_2 \vdash G} \qquad \text{A-}\leftrightarrow\text{:}\ \frac{K\ldots, F_1 \leftrightarrow F_2 \vdash G[F_1/F_2]}{K\ldots, F_1 \leftrightarrow F_2 \vdash G}$$

In a proof presentation, we would then write something like:

> We know $F_1 \leftrightarrow F_2$ and we know, e.g., $\neg F_2 \wedge F_3$. Therefore we know $\neg F_1 \wedge F_3$.

4.3 Equality Rules and Quantifier Rules

Equality

- Prove an equality as a goal:

$$\text{P-=:}\ \frac{}{K\ldots\ \vdash t = t}$$

In a proof presentation, we would then write something like:

> We have to prove $t = t$ and are therefore done.

- Use an equality as assumption ("substitution")[5]:

$$\text{A-=:}\ \frac{K\ldots[t_2/t_1], t_1 = t_2 \vdash G}{K\ldots, t_1 = t_2 \vdash G} \qquad \text{A-=:}\ \frac{K\ldots, t_1 = t_2 \vdash G[t_2/t_1]}{K\ldots, t_1 = t_2 \vdash G}$$

$$\text{A-=:}\ \frac{K\ldots[t_1/t_2], t_1 = t_2 \vdash G}{K\ldots, t_1 = t_2 \vdash G} \qquad \text{A-=:}\ \frac{K\ldots, t_1 = t_2 \vdash G[t_1/t_2]}{K\ldots, t_1 = t_2 \vdash G}$$

In a proof presentation, we would then write something like:

> We know $t_1 = t_2$ and we know, e.g., $p(a, f(t_1))$. Therefore we know $p(a, f(t_2))$.

Equivalences and equalities in the knowledge base behave very similar, they both allow for substitution. While equivalences *replace subformulas* by formulas, equalities *replace subterms* by terms.

Universal Quantifier

- Prove universally quantified formula as a goal ("skolemization").

[5] We use the notation $\Gamma[t_2/t_1]$ for the result of replacing in some formula(s) in Γ some occurrence of term t_1 by term t_2. Note that the substitution notation works for both replacing formulas by formulas and replacing terms by terms.

$$\text{P-}\forall\text{:}\ \frac{K\ldots \vdash F[\overline{x}/x]}{K\ldots \vdash (\forall x\colon F)} \quad \text{where } \overline{x} \text{ does not occur in } K\ldots, F$$

In a proof presentation, we would then write something like:

> We prove $(\forall x\colon p(x, f(x)))$. We take arbitrary but fixed $\overline{x}$ and prove $p(\overline{x}, f(\overline{x}))$.

"Arbitrary but fixed" is a phrase that you will meet frequently in mathematical texts, in particular in proofs. In this context, "fixed" indicates that we introduce a *constant* $\overline{x}$, a so-called "Skolem constant". "Arbitrary" means that $\overline{x}$ is a *new* constant about which nothing is known. You may think that if you "randomly" choose some constant, say 5, that this is also "arbitrary", but this is a misunderstanding! 5 is not arbitrary, because we know something about 5 (it is prime, it is odd, it is greater than 4, it divides 20, etc.). If you then *use* any of the knowledge about 5 later in the proof, it is a logical error and, thus, a wrong proof. If you do not use any of the knowledge about 5 later in the proof, then you better choose some unknown symbol, like $\overline{x}$, and you will see the proof still works, it is then a correct proof.

- Use universally quantified formula as an assumption ("instantiation"):

$$\text{A-}\forall\text{:}\ \frac{K\ldots, (\forall x\colon F), F[t/x] \vdash G}{K\ldots, (\forall x\colon F) \vdash G} \quad \text{where } t \text{ is some term made up of symbols occurring in } K\ldots, F$$

In a proof presentation, we would then write something like:

> We know $(\forall x\colon p(x, f(x)))$. Thus we know (for $x := a$) $p(a, f(a))$ and (for $x := g(a)$) $p(g(a), f(g(a))$.

Note, that $(\forall x\colon F)$ stays in the assumptions and can be instantiated again and again, it is like a "knowledge generating engine" that can be applied *arbitrarily often*. Often the problem is to *find suitable term* t that help the proof making progress. If an unsuitable t is chosen, it is not a logical error. It is only that the additional knowledge does not help, yet it does not harm.

Optically, the treatment of a universal quantifier in the goal looks very similar to its treatment in the assumptions. It is of utmost importance to understand that the "arbitrary but fixed"-rule allows us (forces us) to *introduce a new symbol* $\overline{x}$, whereas instantiation means to *construct a term* t using available ingredients.

Existential Quantifier

- Prove an existentially quantified formula as a goal ("instantiation"):

$$\text{P-}\exists\text{:}\ \frac{K\ldots \vdash F[t/x]}{K\ldots \vdash (\exists x\colon F)} \quad \text{where } t \text{ is some term made up of symbols occurring in } K\ldots, F$$

In a proof presentation, we would then write something like:

We have to prove $(\exists x\colon p(x, f(x)))$. We prove (for $x := g(a)$) $p(g(a), f(g(a)))$.

Like when instantiating a universal quantifier in the assumptions, the problem when instantiating an existential goal is to *find* a “witness term” t that lets the proof succeed. If an unsuitable t is chosen, the proof is not wrong, but it will *fail*.

- Use existentially quantified formula as an assumption (“skolemization”):

$$\text{A-}\exists\text{:}\ \frac{K\ldots, F[\overline{x}/x] \vdash G}{K\ldots, (\exists x\colon F) \vdash G} \quad \text{where } \overline{x} \text{ does not occur in } K\ldots, F, G$$

In a proof presentation, we would then write something like:

We know $\exists x\colon p(x, f(x))$. Thus we know $p(\overline{x}, f(\overline{x}))$ for some $\overline{x}$.

Like with skolemization of a universally quantified goal, $\overline{x}$ is an “arbitrary but fixed” Skolem constant. Also this rule can be considered a “knowledge generating engine”, but, since we remove $(\exists x\colon F)$ from the assumptions, it can be applied only *once*.

We have four rules for the two quantifiers, for each of them one rule for the quantifier as outermost symbol in the goal and one rule for the quantifier as outermost symbol in one of the assumptions. Note the “duality” in the rules:

- P-∀ and A-∃ are “essentially the same” and they are easy to apply: just introduce a new Skolem constant and remove the quantifier.
- A-∀ and P-∃ are also “essentially the same”, but they are tricky to apply: we need an “idea” for a “clever instantiation”. However, failing to have the clever idea does not result in a wrong proof, it only produces useless knowledge in the one case (A-∀) and an unprovable goal in the other (P-∃).

Example: A Quantifier Proof

$$
\begin{array}{rcl}
 & \vdash (\exists x\colon \forall y\colon p(x,y)) \to (\forall y\colon \exists x\colon p(x,y)) & \downarrow \\ \hline
\text{P-}\to\text{:} & \exists x\colon \forall y\colon p(x,y) \vdash \forall y\colon \exists x\colon p(x,y) & \\ \hline
\text{P-}\forall\text{:} & \exists x\colon \forall y\colon p(x,y) \vdash \exists x\colon p(x,\overline{y}) & \\ \hline
\text{A-}\exists\text{:} & \forall y\colon p(\overline{x},y) \vdash \exists x\colon p(x,\overline{y}) & \\ \hline
\text{A-}\forall\text{:} & \forall y\colon p(\overline{x},y), p(\overline{x},\overline{y}) \vdash \exists x\colon p(x,\overline{y}) & \\ \hline
\text{P-}\exists\text{:} & \forall y\colon p(\overline{x},y), p(\overline{x},\overline{y}) \vdash p(\overline{x},\overline{y}) & \\ \hline
\text{GoalAssum:} & &
\end{array}
$$

A typical presentation of this proof would be as follows: We prove

$$\exists x : \forall y : p(x, y)) \to (\forall y : \exists x : p(x, y)) \tag{a}$$

We assume

$$\exists x : \forall y : p(x, y) \tag{1}$$

and prove

$$\forall y : \exists x : p(x, y) \tag{b}$$

We take arbitrary but fixed $\overline{y}$ and prove

$$\exists x : p(x, \overline{y}) \tag{c}$$

From (1), we know $\forall y : p(\overline{x}, y)$ for some $\overline{x}$, and from that, we know (for $y := \overline{y}$)

$$p(\overline{x}, \overline{y}). \tag{3}$$

In order to prove (c), let $x := \overline{x}$ and prove $p(\overline{x}, \overline{y})$. QED (3).

Example: Another Quantifier Proof

$$\begin{array}{rcc}
 & \vdash ((\exists x\colon p(x)) \wedge (\forall x\colon p(x) \longrightarrow \exists y\colon q(x, y))) \longrightarrow \exists x, y\colon q(x, y) & \downarrow \\ \cline{2-2}
\text{P-}\longrightarrow\text{:} & (\exists x\colon p(x)) \wedge (\forall x\colon p(x) \longrightarrow \exists y\colon q(x, y)) \vdash \exists x, y\colon q(x, y) & \\ \cline{2-2}
\text{A-}\wedge\text{:} & \exists x\colon p(x), \forall x\colon p(x) \longrightarrow \exists y\colon q(x, y) \vdash \exists x, y\colon q(x, y) & \\ \cline{2-2}
\text{A-}\exists\text{:} & p(\overline{x}), \forall x\colon p(x) \longrightarrow \exists y\colon q(x, y) \vdash \exists x, y\colon q(x, y) & \\ \cline{2-2}
\text{A-}\forall\text{, Drop:} & p(\overline{x}), p(\overline{x}) \longrightarrow \exists y\colon q(\overline{x}, y) \vdash \exists x, y\colon q(x, y) & \\ \cline{2-2}
\text{MP,Drop:} & \exists y\colon q(\overline{x}, y) \vdash \exists x, y\colon q(x, y) & \\ \cline{2-2}
\text{A-}\exists\text{:} & q(\overline{x}, \overline{y}) \vdash \exists x, y\colon q(x, y) & \\ \cline{2-2}
\text{P-}\exists\text{:} & q(\overline{x}, \overline{y}) \vdash q(\overline{x}, \overline{y}) & \\ \cline{2-2}
\text{GoalAssum:} & &
\end{array}$$

A typical presentation of this proof would be as follows: In order to prove the goal we assume

$$(\exists x\colon p(x)) \wedge (\forall x\colon p(x) \longrightarrow \exists y\colon q(x, y)) \tag{1}$$

and show

$$\exists x, y\colon q(x, y) \tag{b}$$

From (1), we know

$$(\exists x\colon p(x)) \tag{2}$$

$$(\forall x\colon p(x) \longrightarrow \exists y\colon q(x, y))\,. \tag{3}$$

From (2), we know

$$p(\overline{x}) \qquad \text{for some } \overline{x}. \tag{4}$$

From (3) we know (for $x := \overline{x}$) $(p(\overline{x}) \longrightarrow \exists y\colon q(\overline{x}, y))$ and together with (4), this gives $\exists y\colon q(\overline{x}, y)$. From this, we know

$$q(\overline{x}, \overline{y}) \qquad \text{for some } \overline{y}. \tag{7}$$

In order to prove (b), let $x := \overline{x}$ and $y := \overline{y}$ and prove $q(\overline{x}, \overline{y})$. QED (7).

Example: A Quantifier Proof with Branches

$$
\begin{array}{rc}
\text{P-}\to\text{:} & \vdash \big((p(a)\lor q(b))\land(\forall x\colon p(x)\to r(x))\land(\forall x\colon q(x)\to r(f(x)))\big)\to\exists x\colon r(x) \quad \downarrow \\ \hline
\text{A-}\land\text{:} & (p(a)\lor q(b))\land(\forall x\colon p(x)\to r(x))\land(\forall x\colon q(x)\to r(f(x)))\vdash\exists x\colon r(x) \\ \hline
\text{A-}\lor\text{, Drop:} & p(a)\lor q(b), (\forall x\colon p(x)\to r(x)), (\forall x\colon q(x)\to r(f(x)))\vdash\exists x\colon r(x) \\ \hline
 & \begin{array}{cc}
\begin{array}{rc}
\text{A-}\forall\text{, Drop:} & p(a), (\forall x\colon p(x)\to r(x))\vdash\exists x\colon r(x) \\ \hline
\text{MP:} & p(a), p(a)\to r(a)\vdash\exists x\colon r(x) \\ \hline
\text{P-}\exists\text{:} & p(a), r(a)\vdash\exists x\colon r(x) \\ \hline
\text{GoalAssum:} & p(a), r(a)\vdash r(a) \\ \hline
\end{array}
&
\begin{array}{rc}
\text{A-}\forall\text{, Drop:} & q(b), (\forall x\colon q(x)\to r(f(x)))\vdash\exists x\colon r(x) \\ \hline
\text{MP:} & q(b), q(b)\to r(f(b))\vdash\exists x\colon r(x) \\ \hline
\text{P-}\exists\text{:} & q(b), r(f(b))\vdash\exists x\colon r(x) \\ \hline
\text{GoalAssum:} & q(b), r(f(b))\vdash r(f(b)) \\ \hline
\end{array}
\end{array}
\end{array}
$$

A typical presentation of this proof would be as follows: In order to prove the goal we assume

$$p(a)\lor q(b) \tag{1}$$

$$(\forall x\colon p(x)\to r(x)) \tag{2}$$

$$(\forall x\colon q(x)\to r(f(x))) \tag{3}$$

and prove

$$\exists x\colon r(x) \tag{b}$$

From (1), we have two cases:

1. We assume $p(a)$: From (2), we know $p(a)\to r(a)$ (for $x := a$), and together with the case assumption, this gives (4) $r(a)$. In order to prove (b), let $x := a$ and prove $r(a)$. QED (4).

2. We assume $q(b)$: From (3), we know $q(b)\to r(f(b))$ (for $x := b$), and together with the case assumption, this gives (5) $r(f(b))$. In order to prove (b), let $x := f(b)$ and prove $r(f(b))$. QED (5).

The last two examples showed a powerful strategy for instantiating universally quantified assumptions. If we have an assumption of the form

$$\forall x\colon (P\to Q),$$

where P and Q are arbitrary formulas containing the variable x, and there is another assumption R, such that

$$R = P[t/x] \qquad \text{for some term } t,$$

then it is usually a good idea to instantiate the universally quantified variable x with this t in order to obtain

$$R\to Q[t/x].$$

Together with the assumption R, we can then *always* apply "modus ponens" and derive new knowledge $Q[t/x]$.

5 The Pragmatics of Proving

5.1 Proving Strategies

We consider proving as the “syntactic process” of constructing a proof tree. It is like a game, where in each step we need to find an inference rule that fits to one of the open proof situations. We win the game if we are able to close all open proof situations, i.e., all branches of the proof tree. In many cases it turns out that already a major part of a proof can be elaborated (possibly even completed) by simple syntactic proof tree construction. All this is mostly “craft”, and there are usually only few steps that can be considered “art”, e.g., clever instantiation or the application of the cut rule.

The following strategies often help:

1. First work on the goal, i.e., apply only the “goal-oriented” rules. The rules are built-up in such a way that they always *decompose* the goal into one or more simpler goals. When an existential quantifier is reached as the outermost symbol, or when the goal is an atomic formula, then stop and continue with other rules.

2. Then work on the assumptions, i.e., apply the “assumption-oriented” rules. By these rules complex assumptions are decomposed into simpler ones. Immediately skolemize existentially quantified assumptions, instantiate universally quantified assumptions if you have reasonable indications how to instantiate.

3. Finally “close the gap” between assumptions and goal. This can be done on the one hand by deriving assumptions that coincide with atomic goal, and, on the other hand, by instantiating existentially quantified goal in such a way that the body of the instantiated formula appears among the assumptions.

5.2 Mathematical Proofs

Mathematical proofs are typically written in a much more informal style. In mathematics, when people talk about a proof, they do not talk about a proof tree. Rather, what they mean is an easily readable “sketch of the logical argumentation” that just gives the essential information, which would allow to reconstruct a corresponding formal proof, i.e., a proof tree. Usually, they

- do not mention all steps,
- combine several steps into one,
- reuse the name of a variable for the Skolem constant,
- use hidden assumptions,
- etc.

Theorem: Suppose a divides b if and only if, for some $t \in \mathbb{N}$, $b = t \cdot a$. Then, if a divides b it also divides every multiple of b.

Proof: Assume $a, b, s \in \mathbb{N}$ arbitrary but fixed such that a divides b. We show that a divides $s \cdot b$, i.e. $\exists t \in \mathbb{N}\colon\ s \cdot b = t \cdot a$. Since a divides b, we know $b = \bar{t} \cdot a$ for some $\bar{t} \in \mathbb{N}$, thus, we have to find $t \in \mathbb{N}$ with $s \cdot \bar{t} \cdot a = t \cdot a$. Let now $t := s \cdot \bar{t} \in \mathbb{N}$, we have to show $s \cdot \bar{t} \cdot a = s \cdot \bar{t} \cdot a$. QED.

Every sentence in the proof is justified by one or more proof rules. Trivial steps (e.g. split conjunction in assumptions) are not mentioned explicitly. The corresponding proof tree with all logical steps spelled out is shown in Figure 1. Note, that in both branches we use a hidden assumption in the final step. In the first branch, it is the assumption that multiplication is closed over the natural numbers, and in the second branch it is the associativity of multiplication. It is important to understand that behind every mathematical proof there must be a complete and formal proof. Even if the mathematical proof is only a few lines, it must be possible to expand it into a fully formal proof tree. It should be mentioned also, that in mathematics the forward presentation style, see page 6, is very popular, which has the side-effect, that it is not always that easy to see, how the proof was actually constructed.

5.3 Mathematical Induction

When proving statements about the natural numbers, we have an additional proving technique available. More precisely, when proving universally quantified formulas over the natural numbers, i.e., statements of the form[6]

$$\forall n \in \mathbb{N}_{\geq m}\colon F$$

(for some given m), we have an alternative to the standard rule "take $n \in \mathbb{N}_{\geq m}$ arbitrary but fixed". Induction is *not a new logical inference rule*, but it is *technique* based on the special structure of the domain of natural numbers. These are characterized by the *Peano axioms*, and it is the *axiom of induction*, which is of special interest now. This axiom tells us for every formula F

$$(F[m/n] \wedge \forall y \in \mathbb{N}_{\geq m}\colon (F[y/n] \rightarrow F[y+1/n])) \rightarrow (\forall n \in \mathbb{N}_{\geq m}\colon F).$$

Note that we cannot express the induction axiom using *one formula* $\forall F\colon \ldots$ in first-order predicate logic, because this would require quantification over formulas, which is not allowed in first-order predicate logic, see Section 6.1 for more details. Thus, we have an induction axiom for every formula F, so there are, in fact, *infinitely many* induction axioms. Nevertheless, when, *for a concrete F*,

$$\text{we want to prove} \quad \forall n \in \mathbb{N}_{\geq m}\colon F$$

then it is "a good idea", using the cut-rule, to

[6] Recall $\mathbb{N}_{\geq m} = \{m, m+1, m+2, \ldots\}$ stands for the natural numbers greater equal m.

$$
\begin{array}{rl}
 & \forall a, b\colon (a \mid b \leftrightarrow \exists t \in \mathbb{N}\colon b = t \cdot a) \vdash \forall a, b, s \in \mathbb{N}\colon (a \mid b \rightarrow a \mid s \cdot b) \qquad \downarrow \\ \hline
\text{P-}\forall\text{:} & \forall a, b\colon (a \mid b \leftrightarrow \exists t \in \mathbb{N}\colon b = t \cdot a), \bar{a}, \bar{b}, \bar{s} \in \mathbb{N} \vdash \bar{a} \mid \bar{b} \rightarrow \bar{a} \mid \bar{s} \cdot \bar{b} \\ \hline
\text{P-}\rightarrow\text{:} & \forall a, b\colon (a \mid b \leftrightarrow \exists t \in \mathbb{N}\colon b = t \cdot a), \bar{a}, \bar{b}, \bar{s} \in \mathbb{N}, \bar{a} \mid \bar{b} \vdash \bar{a} \mid \bar{s} \cdot \bar{b} \\ \hline
\text{A-}\forall\text{:} & \forall a, b\colon (a \mid b \leftrightarrow \exists t \in \mathbb{N}\colon b = t \cdot a), \bar{a}, \bar{b}, \bar{s} \in \mathbb{N}, \bar{a} \mid \bar{b}, \bar{a} \mid \bar{s} \cdot \bar{b} \leftrightarrow \exists t \in \mathbb{N}\colon \bar{s} \cdot \bar{b} = t \cdot \bar{a} \vdash \bar{a} \mid \bar{s} \cdot \bar{b} \\ \hline
\text{A-}\leftrightarrow\text{, Drop:} & \forall a, b\colon (a \mid b \leftrightarrow \exists t \in \mathbb{N}\colon b = t \cdot a), \bar{a}, \bar{b}, \bar{s} \in \mathbb{N}, \bar{a} \mid \bar{b} \vdash \exists t \in \mathbb{N}\colon \bar{s} \cdot \bar{b} = t \cdot \bar{a} \\ \hline
\text{A-}\forall\text{:} & \forall a, b\colon (a \mid b \leftrightarrow \exists t \in \mathbb{N}\colon b = t \cdot a), \bar{a}, \bar{b}, \bar{s} \in \mathbb{N}, \bar{a} \mid \bar{b}, \bar{a} \mid \bar{b} \leftrightarrow \exists t \in \mathbb{N}\colon \bar{b} = t \cdot \bar{a} \vdash \exists t \in \mathbb{N}\colon \bar{s} \cdot \bar{b} = t \cdot \bar{a} \\ \hline
\text{A-}\leftrightarrow\text{, Drop:} & \bar{a}, \bar{b}, \bar{s} \in \mathbb{N}, \exists t \in \mathbb{N}\colon \bar{b} = t \cdot \bar{a} \vdash \exists t \in \mathbb{N}\colon \bar{s} \cdot \bar{b} = t \cdot \bar{a} \\ \hline
\text{A-}\exists\text{:} & \bar{a}, \bar{b}, \bar{s} \in \mathbb{N}, \bar{t} \in \mathbb{N} \wedge \bar{b} = \bar{t} \cdot \bar{a} \vdash \exists t \in \mathbb{N}\colon \bar{s} \cdot \bar{b} = t \cdot \bar{a} \\ \hline
\text{A-}\wedge\text{:} & \bar{a}, \bar{b}, \bar{s} \in \mathbb{N}, \bar{t} \in \mathbb{N}, \bar{b} = \bar{t} \cdot \bar{a} \vdash \exists t \in \mathbb{N}\colon \bar{s} \cdot \bar{b} = t \cdot \bar{a} \\ \hline
\text{A-=, Drop:} & \bar{a}, \bar{b}, \bar{s} \in \mathbb{N}, \bar{t} \in \mathbb{N} \vdash \exists t \in \mathbb{N}\colon \bar{s} \cdot (\bar{t} \cdot \bar{a}) = t \cdot \bar{a} \\ \hline
\text{P-}\exists\text{:} & \bar{a}, \bar{b}, \bar{s} \in \mathbb{N}, \bar{t} \in \mathbb{N} \vdash \bar{s} \cdot \bar{t} \in \mathbb{N} \wedge \bar{s} \cdot (\bar{t} \cdot \bar{a}) = (\bar{s} \cdot \bar{t}) \cdot \bar{a} \\ \hline
\text{P-}\wedge\text{, Drop:} & \bar{s} \in \mathbb{N}, \bar{t} \in \mathbb{N} \vdash \bar{s} \cdot \bar{t} \in \mathbb{N} \qquad \bar{a}, \bar{s} \in \mathbb{N}, \bar{t} \in \mathbb{N} \vdash \bar{s} \cdot (\bar{t} \cdot \bar{a}) = (\bar{s} \cdot \bar{t}) \cdot \bar{a} \\ \hline
\text{Hidden:} & \qquad\qquad\qquad\qquad \text{P-=, Hidden:}
\end{array}
$$

Figure 1: Proof tree for divisibility proof

$$\text{prove} \quad \underbrace{F[m/n]}_{(1)} \wedge \underbrace{\forall y \in \mathbb{N}_{\geq m} \colon (F[y/n] \rightarrow F[y+1/n])}_{(2)},$$

because then, by modus ponens and the appropriate induction axiom for F, we can *derive*

$$\forall n \in \mathbb{N}_{\geq m} \colon F,$$

and the proof is *finished*, since this assumption is identical to the goal. We call (1) the *induction base*, and the proof of (2) then proceeds by the usual "take $\bar{n} \in \mathbb{N}_{\geq m}$ arbitrary but fixed, assume left-hand side, prove right-hand side"-rule. We can write this technique as one compact inference rule

$$\text{Induction: } \frac{K \ldots \vdash F[m/n] \qquad K \ldots, \bar{n} \in \mathbb{N}_{\geq m}, F[\bar{n}/n] \vdash F[(\bar{n}+1)/n]}{K \ldots \vdash \forall n \in \mathbb{N}_{\geq m} \colon F}$$

Hence, a proof of $\forall n \in \mathbb{N}_{\geq m} \colon F$ by induction over n proceeds as follows:

- *Induction base:* prove that F holds for m.
- *Induction hypothesis:* assume that F holds for new constant $\overline{n} \geq m$.
- *Induction step:* prove that then F also holds for $\overline{n} + 1$.

As an example, we prove the "sum of squares" formula

$$\forall n \in \mathbb{N} \colon \sum_{i=1}^{n} i^2 = \frac{n \cdot (n+1) \cdot (2n+1)}{6}$$

by induction on n.

- Induction Base: in this case $m = 0$.

$$\sum_{i=1}^{0} i^2 = 0 = \frac{0 \cdot (0+1)(2 \cdot 0 + 1)}{6}$$

- Induction Hypothesis: assume

$$\sum_{i=1}^{\overline{n}} i^2 = \frac{\overline{n} \cdot (\overline{n}+1) \cdot (2\overline{n}+1)}{6} \qquad (*)$$

- Induction Step: prove

$$\sum_{i=1}^{\overline{n}+1} i^2 = (\overline{n}+1)^2 + \sum_{i=1}^{\overline{n}} i^2 \overset{(*)}{=} (\overline{n}+1)^2 + \frac{\overline{n} \cdot (\overline{n}+1) \cdot (2\overline{n}+1)}{6} =$$
$$= \frac{(\overline{n}+1) \cdot (6 \cdot (\overline{n}+1) + \overline{n} \cdot (2\overline{n}+1))}{6} = \frac{(\overline{n}+1) \cdot (2\overline{n}^2 + 7\overline{n} + 6)}{6} =$$
$$= \frac{(\overline{n}+1) \cdot (\overline{n}+2) \cdot (2\overline{n}+3)}{6} = \frac{(\overline{n}+1) \cdot ((\overline{n}+1)+1) \cdot (2(\overline{n}+1)+1)}{6} \qquad \square$$

Suppose now we want to prove

$$\forall n \in \mathbb{N}_{n\geq 4}\colon n^2 \leq 2^n$$

- Induction base: in this case $m = 4$, i.e., we show

$$4^2 = 16 = 2^4.$$

- Induction hypothesis: we assume for $n \geq 4$

$$n^2 \leq 2^n. \tag{$*$}$$

- Induction step: we show

$$\begin{aligned}(n+1)^2 = n^2 + 2n + 1 &\overset{1\leq n}{\leq} n^2 + 2n + n = n^2 + 3n \overset{0\leq n}{\leq} n^2 + 4n \\ &\overset{4\leq n}{\leq} n^2 + n \cdot n = n^2 + n^2 = 2n^2 \overset{(*)}{\leq} 2 \cdot 2^n = 2^{n+1} \qquad \square\end{aligned}$$

In our final example, we prove a formula with more than one quantifier. In this case, one needs to decide, for which variable we apply induction. We define addition on $\mathbb{N}$ by primitive recursion:

$$\begin{aligned} x + 0 &:= x && (1)\\ x + (y+1) &:= (x+y) + 1 && (2)\end{aligned}$$

Our goal is to prove the associativity law

$$\forall x \in \mathbb{N}, y \in \mathbb{N}, z \in \mathbb{N}\colon x + (y+z) = (x+y) + z$$

For this purpose, we fix arbitrary $x_0, y_0 \in \mathbb{N}$ and then prove

$$\forall z \in \mathbb{N}\colon x_0 + (y_0 + z) = (x_0 + y_0) + z$$

by induction[7] on z.

- Induction base: we prove

$$x_0 + (y_0 + 0) \overset{(1)}{=} x_0 + y_0 \overset{(1)}{=} (x_0 + y_0) + 0.$$

- Induction hypothesis: we assume for $z_0 \in \mathbb{N}$

$$x_0 + (y_0 + z_0) = (x_0 + y_0) + z_0. \tag{$*$}$$

- Induction step: we show

$$\begin{aligned} x_0 + (y_0 + (z_0 + 1)) &\overset{(2)}{=} x_0 + ((y_0 + z_0) + 1) \overset{(2)}{=} (x_0 + (y_0 + z_0)) + 1 = \\ &\overset{(*)}{=} ((x_0 + y_0) + z_0) + 1 \overset{(2)}{=} (x_0 + y_0) + (z_0 + 1). \qquad \square\end{aligned}$$

[7] Sometimes the appropriate choice of the induction variable is critical.

6 Expressiveness and Limitations of First-Order Predicate Logic

6.1 Expressiveness of First-Order Predicate Logic

In this section we want to briefly discuss what can and what cannot be expressed in first order logic. One of the characteristics of *first-order* predicate logic is that variables range over *elements of the domain*, see variable assignment in Part 1. The elements of the domain are the objects of our discourse, therefore variables are sometimes also called *object variables*. This terminology is used then in particular, when we have other types of variables also, e.g.,

- *function variables*, i.e., variables ranging over functions, or
- *predicate variables*, i.e., variables ranging over predicates.

As a consequence, quantifiers always range over objects, because the variable in a quantified expression can only range over objects. This means that there is no quantification possible over functions and predicates of the domain. This would require second-order predicate logic.

Nevertheless, we want to express statements such as "every bijective function has an inverse" in first-order logic. More formally, we want to have a statement of the form

$$\forall A, B, f\colon \mathrm{isFun}(f, A, B) \wedge \mathrm{bijective}(f) \longrightarrow \exists g\colon \mathrm{isFun}(g, B, A) \wedge \forall x \in B\colon f(g(x)) = x$$

where $\mathrm{isFun}(f, A, B)$ and $\mathrm{isFun}(g, B, A)$ express that f and g are *functions* from A to B and from B to A, respectively. In this form, f and g would be function variables, and we would have quantification over functions.

This is where *set theory* comes into play. Using the language of set theory allows us to express functions and predicates as *objects of the domain*, which turns function and predicate variables into usual object variables and permits quantification over these. For instance, $\mathrm{isFun}(f, A, B)$ means that $f \subseteq A \times B$ s.t.

$$\forall a \in A\colon \exists b \in B\colon (a, b) \in f$$
$$\forall a, b, b'\colon (a, b) \in f \wedge (a, b') \in f \longrightarrow b = b'.$$

Hence, *functions are objects*, namely certain sets of pairs, and quantifiers range over all sets. As an example, the function f that maps every $x \in \{1, 2, 3, 4\}$ to its square plus one would then be represented by the set

$$f = \{(1, 2), (2, 5), (3, 10), (4, 17)\} \subseteq \{1, 2, 3, 4\} \times \mathbb{N}.$$

When we "apply that function" to 2, we get the function value of f at 2, which is 5, because $(2, 5) \in f$. Of course, we cannot write $f(2)$ now, because we cannot *apply* an object f to another object 2. When viewing functions in their set theory interpretation, i.e., as sets of pairs, then $f(x)$ is just a *notation* for "function application" $\mathrm{apply}(f, x)$ with

$$\mathrm{apply}(f, x) := \textbf{the } y\colon (x, y) \in f.$$

The term $f(g(x))$ then stands for

$$\text{apply}(f, \text{apply}(g, x)).$$

A similar approach is taken for *predicates*, namely n-ary predicates are just sets of n-tuples. In the simplest case $n = 1$ a unary predicate p is just represented by the set containing those objects, for which p holds, and $p(x)$ is just a *notation* for $x \in p$. For example, the predicate "prime" that expresses that $x \in \{1, 2, 3, 4\}$ is a prime number would be represented by the set

$$\text{prime} = \{2, 3\}.$$

Expressing that 2 is prime would be written in standard predicate logic as prime(2), which will be read in the set theory interpretation as $2 \in \text{prime}$. For the case $n = 2$, a binary predicate p is represented by the set of pairs consisting of those objects x and y, for which p holds, and $p(x, y)$ is a *notation* for $(x, y) \in p$. For example, the predicate "divides" that expresses that $x \in \{1, 2, 3, 4\}$ divides $y \in \{1, 2, 3, 4\}$ would then be represented by the set

$$\text{divides} = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4)\} \subseteq \{1, 2, 3, 4\} \times \{1, 2, 3, 4\}.$$

Expressing that 2 divides 4 would be written in predicate logic as divides(2, 4), which will be read in the set theory interpretation as $(2, 4) \in \text{divides}$. For $n > 2$ we proceed analogously.

First-order predicate logic over the domain of sets is the "working horse" of mathematics. Virtually all of mathematics is formulated in this framework.

6.2 Limitations of First-Order Predicate Logic

We conclude with some results concerning the power of first-order predicate logic. As a first result, we want to mention the *Completeness Theorem* as formulated by Kurt Gödel in 1929, which gives a connection between the semantic notion of "logical consequence" ($\models$) and the notion of "derivability in a proof calculus" ($\vdash$).

Completeness Theorem (Kurt Gödel, 1929):
First-order predicate logic has a proof calculus for which the following holds:

- Soundness: if a conclusion F can be derived from a set of assumptions Γ by the rules of the calculus, then F is a logical consequence of Γ, i.e.,

$$\text{if } \Gamma \vdash F \text{ then } \Gamma \models F.$$

- Completeness: if F is a logical consequence of Γ, then F can be derived from Γ by the rules of the calculus, i.e.,

$$\text{if } \Gamma \models F \text{ then } \Gamma \vdash F.$$

No logic that is stronger (more expressive) than first-order predicate logic has a proof calculus that also enjoys both soundness and completeness. Still, the existence of a complete proof calculus does not mean that the truth of every formula is *algorithmically decidable*. Completeness only covers the case $\Gamma \models F$, but it does not say anything about the case $\Gamma \not\models F$. Well, soundness tells us that there cannot be a derivation, but how should we decide, *whether* there is one or not? Church and Turing tell us that we cannot.

Undecidability (Church/Turing, 1936/1937):
There does not exist any algorithm that for given formula set Γ and formula F always terminates and says whether $\Gamma \models F$ holds or not.

But, due to completeness, we can state a slightly weaker result.

Semidecidability:
There exists an algorithm, that for given Γ and F, if $\Gamma \models F$, detects this fact in a finite amount of time.

This algorithm searches for a proof of $\Gamma \vdash F$ in a complete proof calculus. If such a proof exists, it will eventually detect it and the answer is $\Gamma \models F$. However, if no such proof exists, the search runs forever. Automatic proof search is not able to detect that a formula is not true.

Gödel's completeness theorem in some sense is a "positive result", but Gödel also proved a "negative result", namely that not every structure can be completely described by a finite set of formulas.

Incompleteness Theorem (Kurt Gödel, 1931):
It is in no sound logic possible to prove all true arithmetic statements, i.e., all statements about natural numbers with addition and multiplication.

In other words, in every sound formal system that is sufficiently rich there are statements that can neither be proved nor disproved. For instance, to adequately characterize $\mathbb{N}$, the (infinite) axiom scheme of mathematical induction has to be added. In practice, complete reasoners for first-order logic are often supported by (complete or incomplete) reasoners for special theories.