Logic and Set Theory: Commonly Made Mistakes

Sanne Wouda and Bas Luttik

October 28, 2014

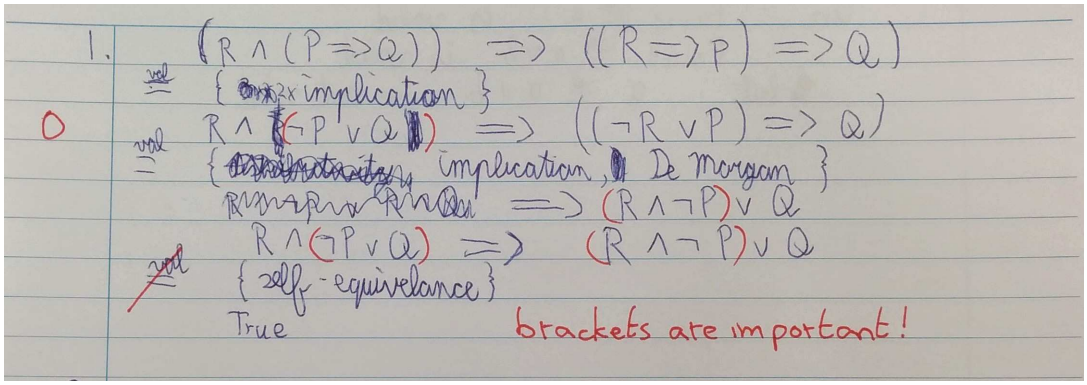
1 Use parentheses when needed.

The inductive definition of the notion of abstract proposition (see Definition 2.3.1 in the book) introduces a pair of parentheses in every step. Not all of these are necessary to reconstruct exactly which abstract proposition is meant. Based on a precedence (see p. 13 in the book) and the standard equivalence *Associativity* it is allowed to omit some parentheses.

For example, consider the following exercise.

Exercise: Show using a calculation that $(R \wedge (P \Rightarrow Q)) \Rightarrow ((R \Rightarrow P) \Rightarrow Q)$ is a tautology.

Take a look at the following erroneous student's solution.



In the first step already, the parentheses around $\neg P \vee Q$ are missing, and in the second step, parentheses are omitted around $R \wedge \neg P$. In fact, looking at the truth table below, you can see that the formulas $R \wedge (\neg P \vee Q)$ and $(R \wedge \neg P) \vee Q$, which are identified above by the omission of the parentheses, are indeed not equivalent:

P	Q	R	$R \wedge (\neg P \vee Q)$	$(R \wedge \neg P) \vee Q$
0	1	0	0	1

Now, if we ignore parentheses, then yes, the left and right hand sides of the implication are identical, and it would be easy to derive True, as is suggested by the student. Note though,

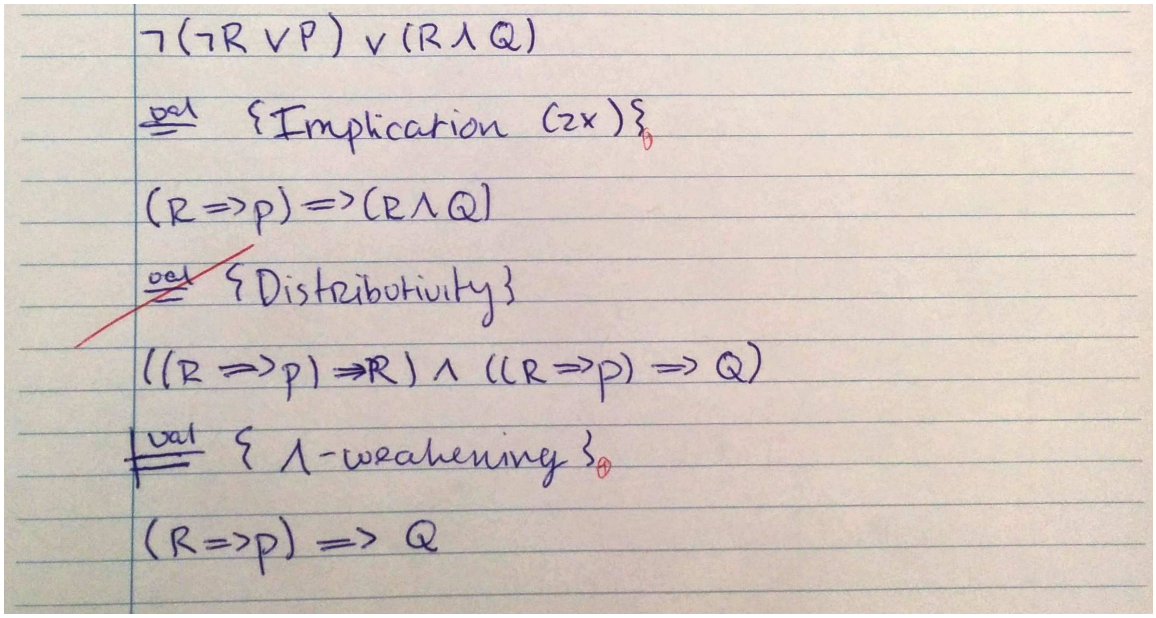
that the standard equivalence named *Self-equivalence* cannot be used for this; instead, using *Implication* and *Excluded Middle* you can calculate as follows: $P \Rightarrow P \stackrel{val}{=\!=} \neg P \vee P \stackrel{val}{=\!=} True$.

You can drop parentheses if, and ONLY if, this is in accordance with the precedence rule, or allowed with an application of *Associativity*. Remember, $\neg$ binds stronger than $\wedge$ and $\vee$, which bind stronger than $\Rightarrow$, which binds stronger than $\Leftrightarrow$. Because there is no precedence defined between $\wedge$ and $\vee$, it is never possible to drop parentheses from a formula that mixes $\wedge$ and $\vee$, for example $P \wedge (Q \vee R)$.

In general, remember the following advice: when in doubt, don't drop parentheses.

2 Distributivity: only for conjunction and disjunction.

The following (part of) a calculation tries to apply distributivity of implication over conjunction. This is not allowed.



Remember above all that a calculation is a proof that is strictly according to the standard equivalences in the Tables for Part 1 (p. 372-373 of the book). Distribution of $\Rightarrow$ over $\wedge$ is not a standard equivalence. There are two valid distributivity rules: distributivity of $\vee$ over $\wedge$ and distributivity of $\wedge$ over $\vee$.

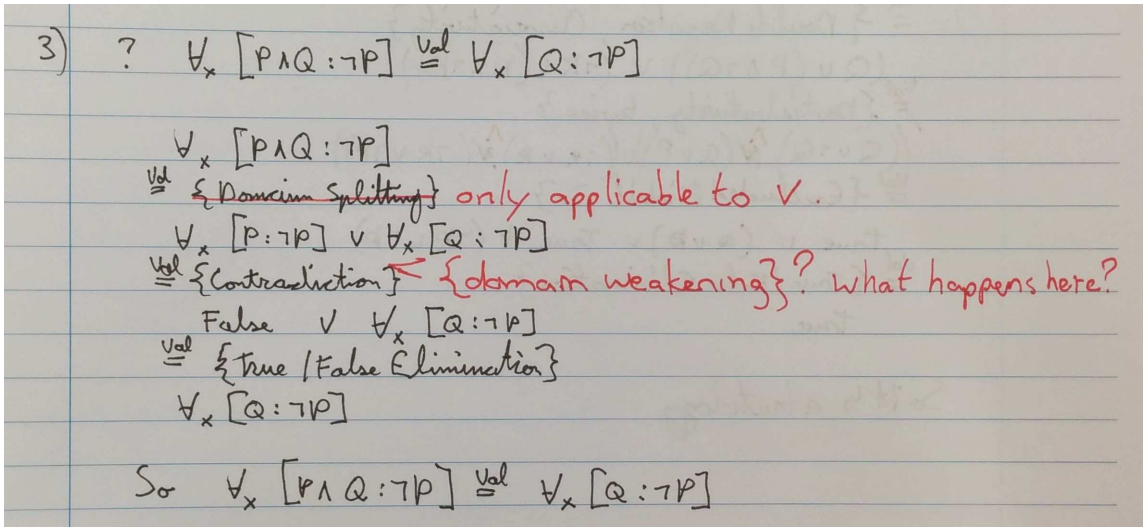
The calculation should have been as follows:

$$
\begin{array}{ll}
 & (R \Rightarrow P) \Rightarrow (R \wedge Q) \\
\overset{val}{=\!=} & \quad \{\ \textit{Implication}\ \} \\
 & \neg(R \Rightarrow P) \vee (R \wedge Q) \\
\overset{val}{=\!=} & \quad \{\ \textit{Distributivity}\ \} \\
 & (\neg(R \Rightarrow P) \vee R) \wedge (\neg(R \Rightarrow P) \vee Q) \\
\overset{val}{=\!=} & \quad \{\ \textit{Implication}\ (2\text{x})\ \} \\
 & ((R \Rightarrow P) \Rightarrow R) \wedge ((R \Rightarrow P) \Rightarrow Q)
\end{array}
$$

As you can see, the conclusion of the earlier calculation was correct. However, this is a lucky coincidence. If we considered "distributivity of $\Rightarrow$ over $\wedge$", it does not hold in general: $(P \wedge Q) \Rightarrow R$ is not equivalent to $(P \Rightarrow R) \wedge (Q \Rightarrow R)$ as you can check yourself by making a truth table.

3 Domain splitting and contradiction.

Take a look at the following calculation.



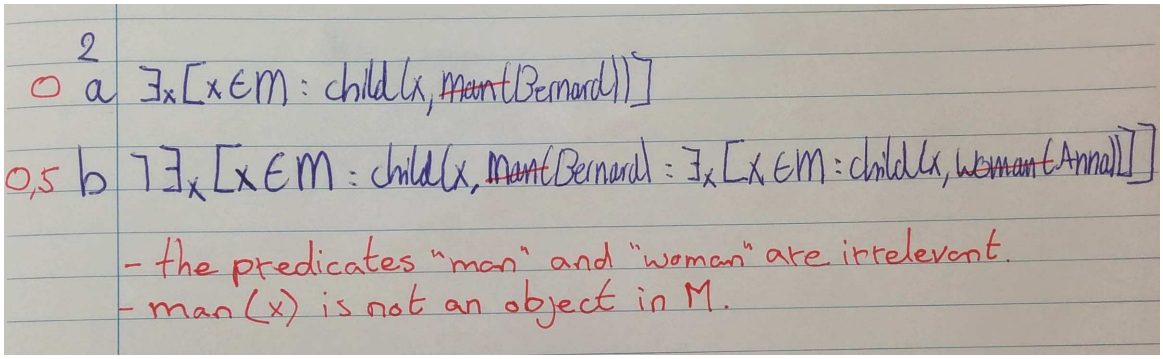
The "domain splitting" rule cannot be applied here. You can only split the domain if the domain is specified as a disjunction (with $\vee$).

Note that the formula $\forall_x[P : \neg P]$ is not a contradiction. If we rewrite the formula using domain weakening, we get $\forall_x[P \Rightarrow \neg P]$, which is by *Implication* and Idempotence equivalent to $\forall_x[\neg P]$.

4 Types

When talking about predicates, it is helpful to consider the types of expressions. It is important to realise that every expression has a type, and that the types of the expressions on which a particular predicate can be applied is fixed.

Consider the next example.



Here, `man(Bernard)` is a proposition expressing that Bernard is a man; "evaluating" `man(Bernard)` yields a truth value. `child` is a predicate that should be applied to elements of M (in the exercise `child` is declared as a 'predicate on M^2'), and not to truth values.

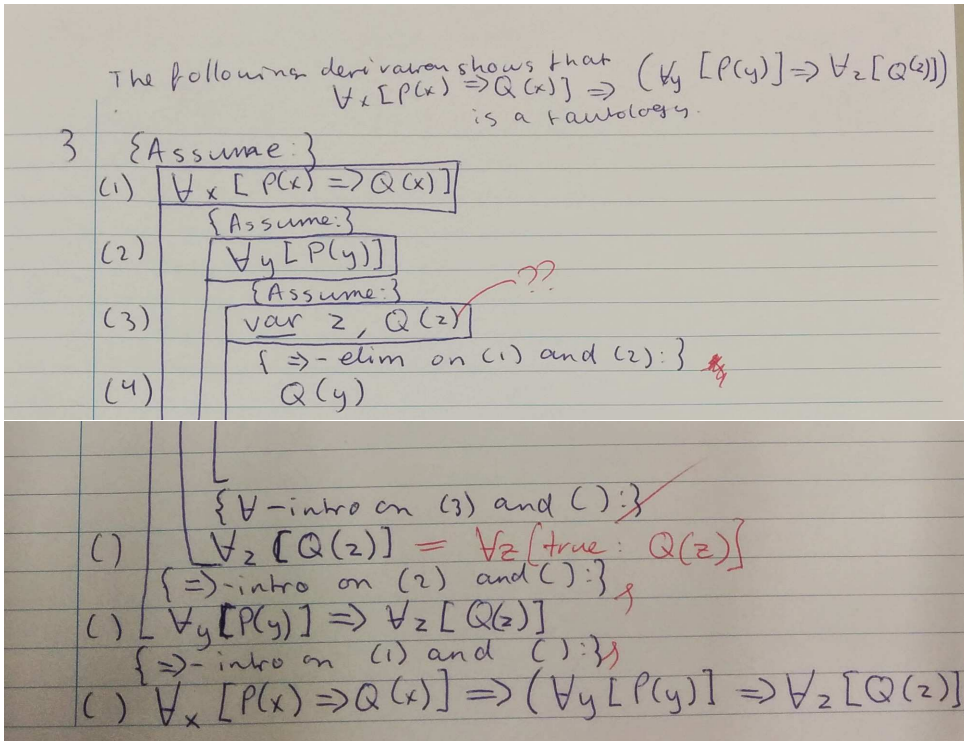
If you wanted to express that Bernard is a man and x is a child of Bernard, you write:

`child`(x`, Bernard`) $\wedge$ `man(Bernard)`

5 Predicates of a quantifier.

The following derivation is incorrect. To apply the ∀-intro rule, one should declare an element (variable) satisfying the domain predicate (the predicate on the left-hand side of the colon), and then derive that it also satisfies the predicate on the right-hand side of the colon. When no domain predicate is given (as is the case with the formula $\forall_z[Q(z)]$ below) it is True by default. So $\forall_z[Q(z)] \overset{val}{=\!=} \forall_z[True : Q(z)]$.

Note also that the ⇒-elimination for line 4 needs two ∀-eliminations, with z on (1) and with z on (2), first.



6 Flag discharge

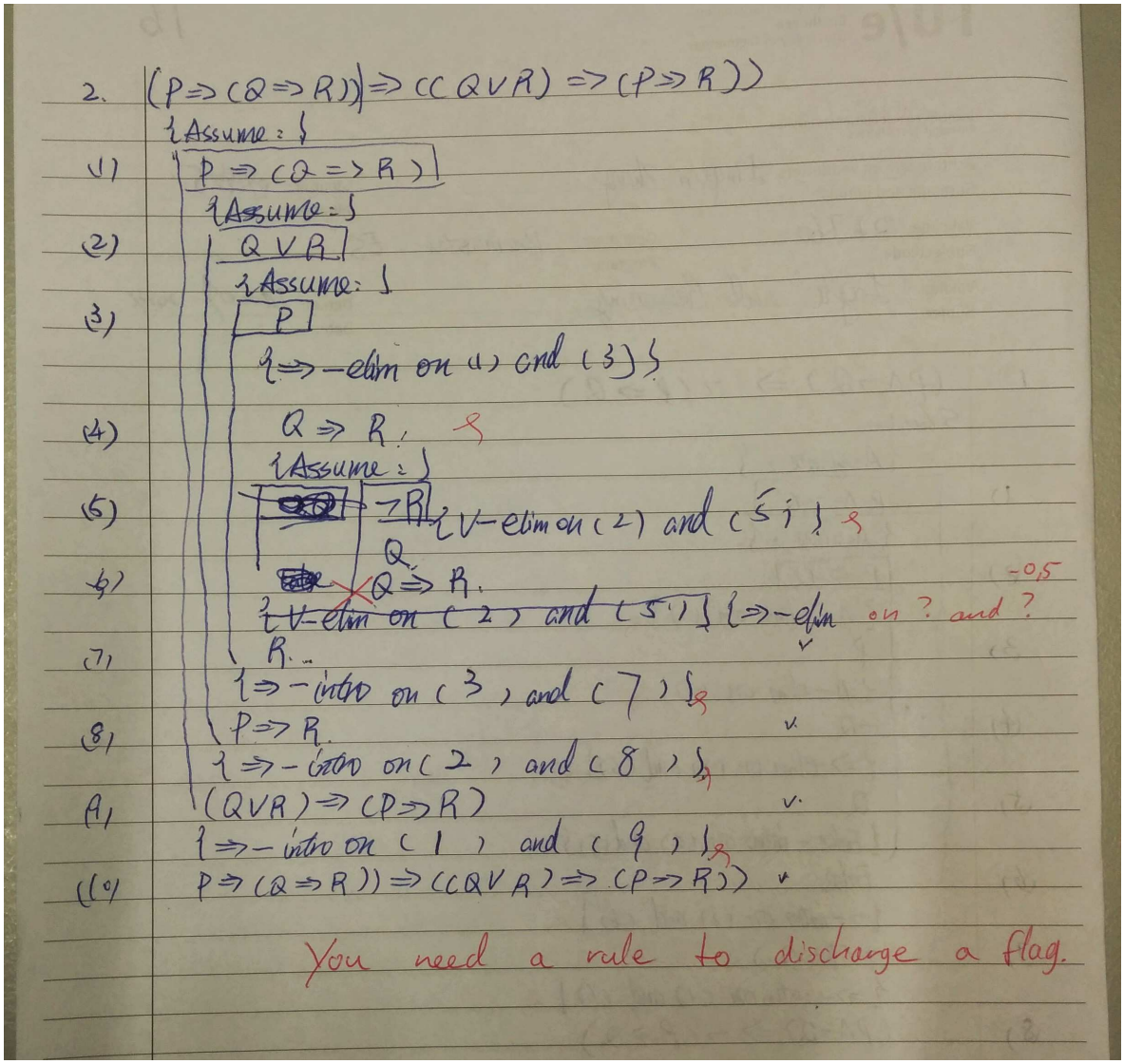
When proving using derivation, it is key to realise when you are allowed to raise flags, and when you can discharge them.

A flag is always raised as the result of applying backward reasoning. One raises flags to simplify the goal of a proof. The flags represent assumptions generated by the (backwards) applications of introduction rules.

The following derivation attempts a proof by contradiction starting on line 5. The intuition is correct here; it is indeed the right approach. However, between line 6 and 7, a number of steps are missing.

For a correct proof by contradiction, you assume the negation of your proof obligation or goal. To discharge this new flag, the rule for $\neg$-introduction tells you that you now need to derive False.

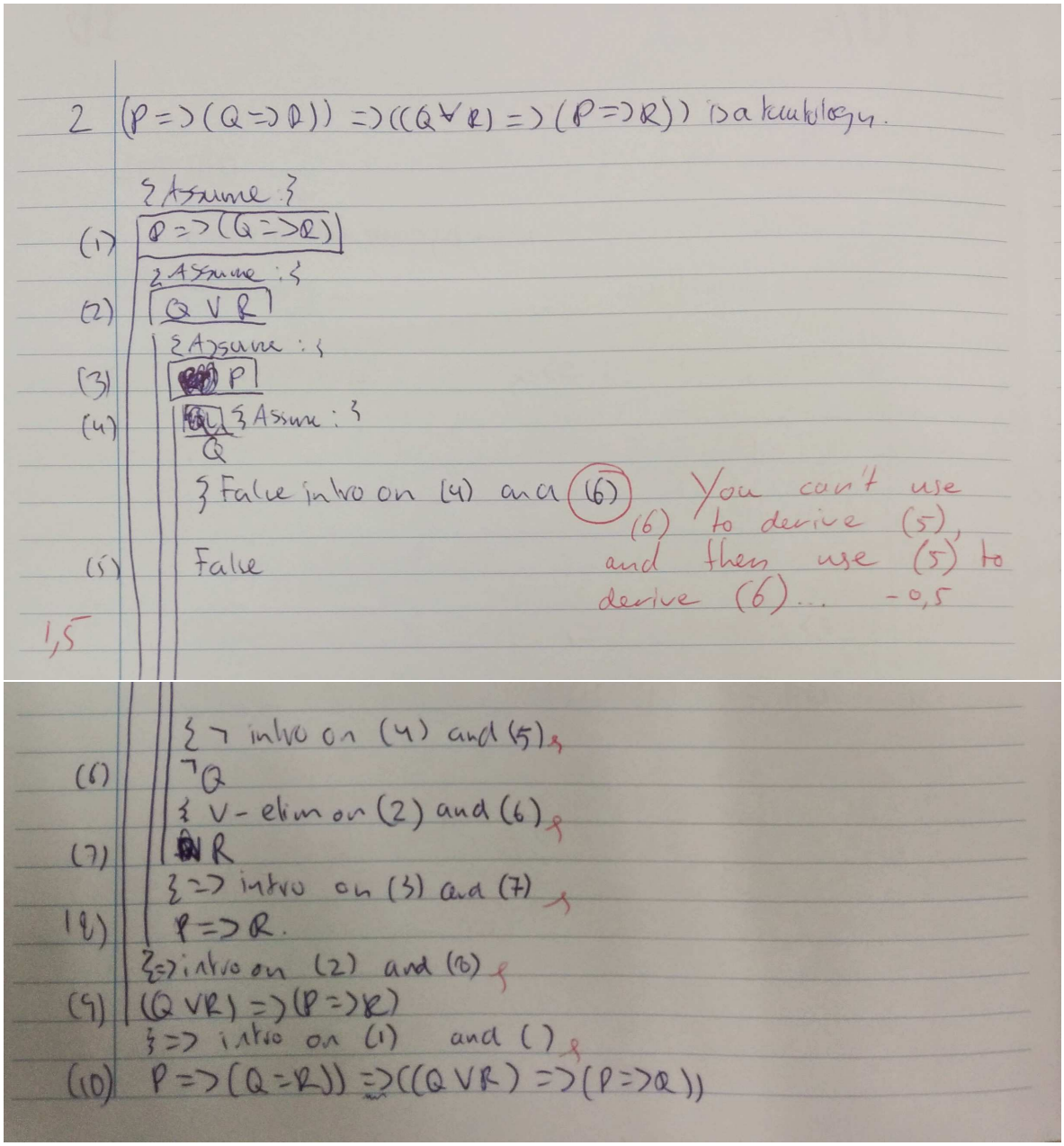
In the missing derivation steps are as follows. First, $\Rightarrow$-elimination on Q and $Q \Rightarrow R$ results in R, but it does not discharge the flag. Next, $\neg$-elimination on the assumption of $\neg R$ and R gives False. Then, $\neg$-introduction gives $\neg\neg R$ and discharges the assumption $\neg R$. Finally, the application of $\neg\neg$-elimination results in R and the proof is complete.



7 Validity in a derivation

Some statements inside a derivation do not stay valid for the entire duration of the proof. When we discharge a flag, both the assumption and the propositions derived from it lose their validity. The statements that are valid at a certain point in the proof, are together called the context.

Additionally, only earlier propositions (as long as they do not occur in a closed flag) can be used to derive new propositions. In the next example, this latter rule is violated.



On line 5, False is derived from 4 and 6. This creates a circular reasoning, resulting in an incorrect proof.

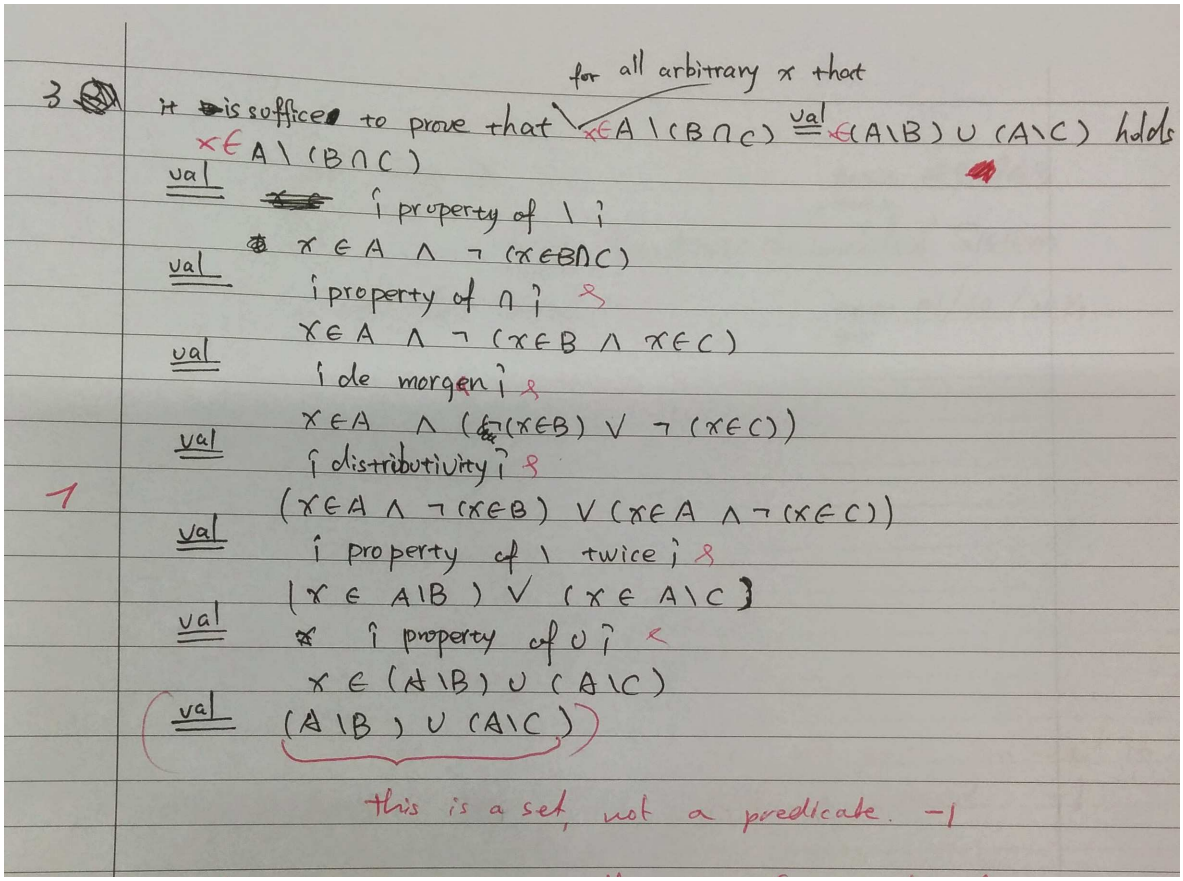
Refer to Chapter 13 of the book for more details on validity in derivation proofs.

8 Set types

Related to common mistake 4, when dealing with proofs related to sets, always check your derivation or calculation with respect to types. Most importantly, a set can never be equivalent to a proposition.

When proving that two sets, say A and B, are equal, one needs to prove that, for every

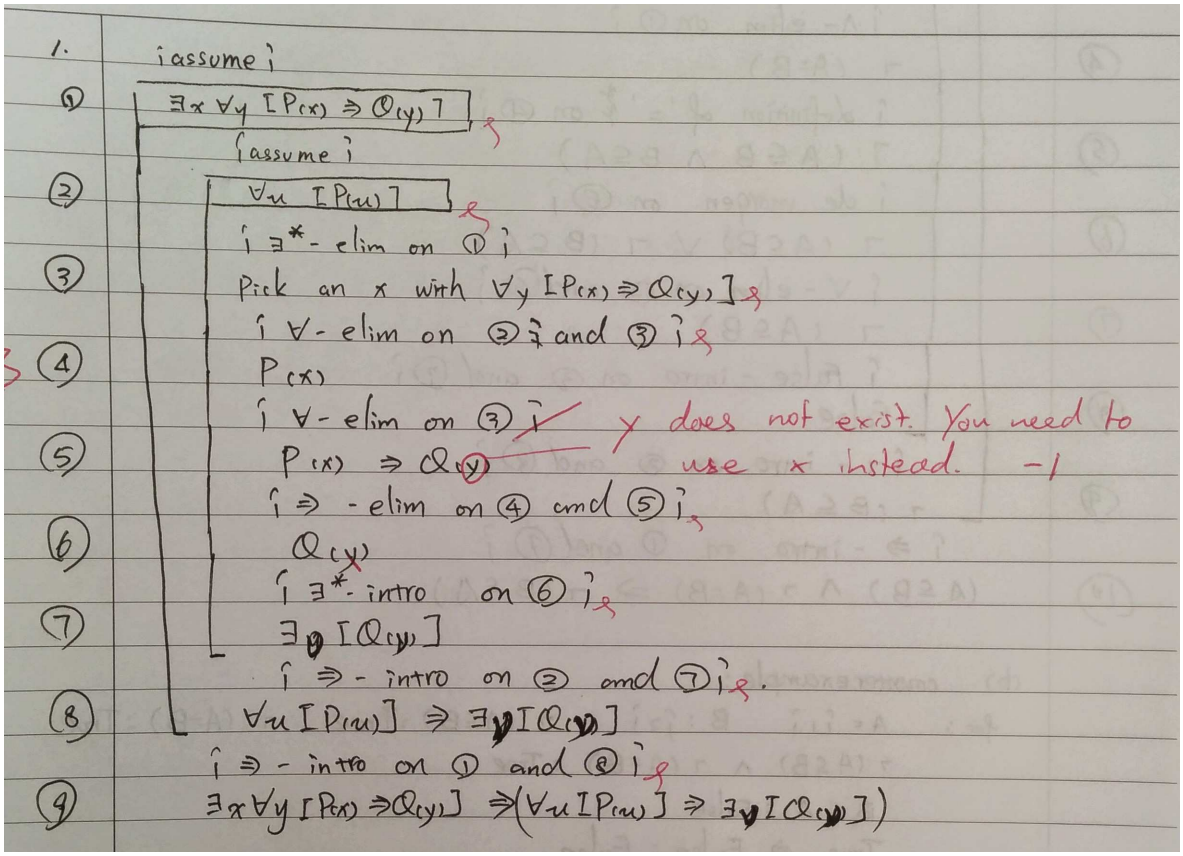
arbitrary x it holds that $x \in A \overset{val}{=\!=} x \in B$. The following calculation does prove that this equivalence holds, however, it is incorrect to write a set where a proposition is required.



9 No ∀-elimination on non-existing elements.

When you apply ∀-elimination, you need an existing element which satisfies the domain predicate. So, if you have an element x for which the predicate $P(x)$ holds and a formula $\forall_y[P(y) : Q(y)]$, then you can conclude $Q(x)$. Note that it is irrelevant that the quantified variable is y, since it is only a name used to establish a binding between the universal quantifier and arguments of the predicates P and Q. The element x, however, refers to a specific element.

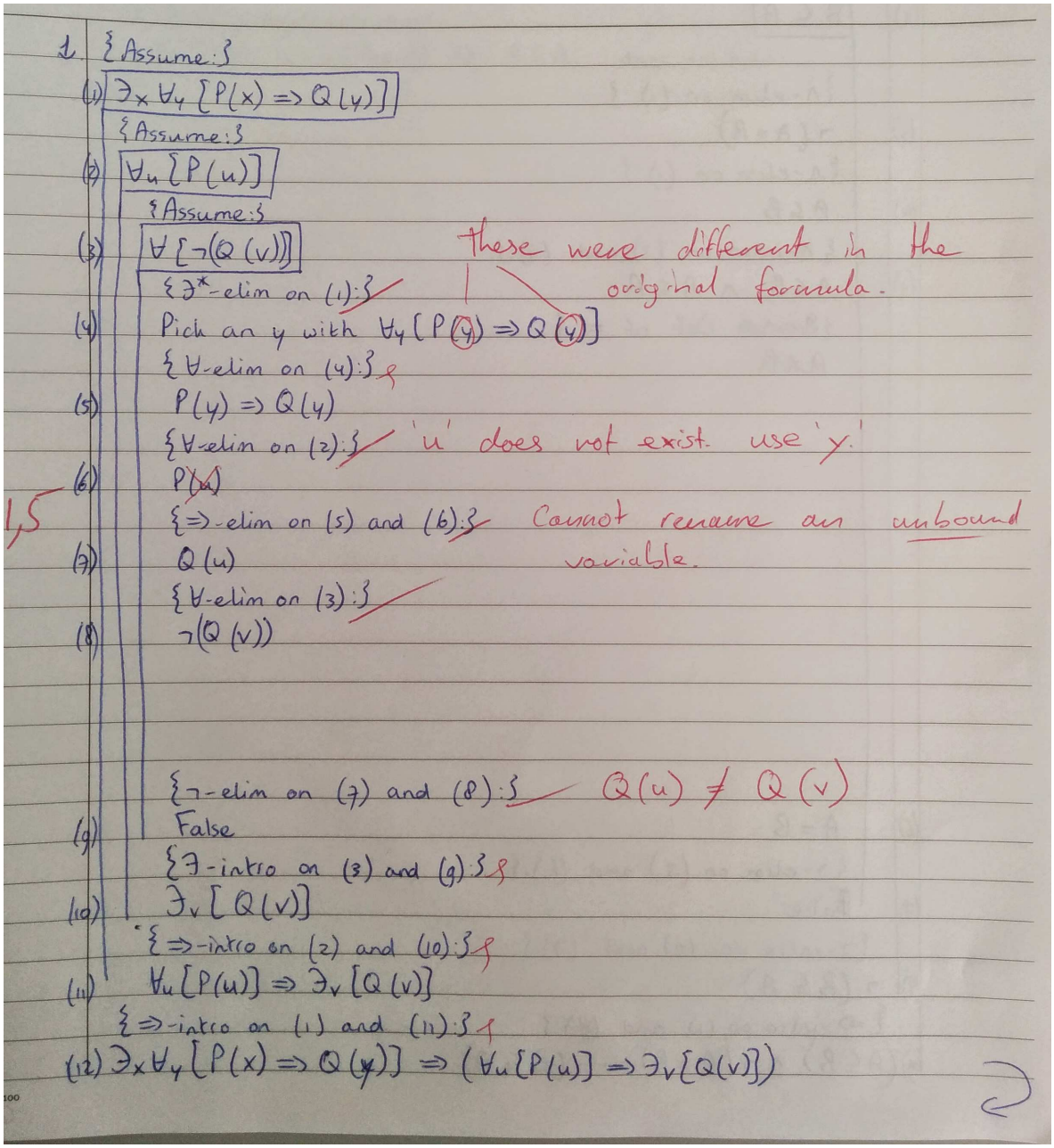
Consider the following example. Here, an unknown object y is used to perform ∀-elimination on line 5, instead of some known object. This is logically incorrect. Note that if we allowed this type of reasoning, it would be possible to derive $\exists_x[P(x) : Q(x)]$ from $\forall_x[P(x) : Q(x)]$. However, this is not true, since, if $P(x)$ is False for all x, then $\forall_x[P(x) : Q(x)]$ is valid, but there is no x that satisfies both P and Q.



10 Renaming of bound variables.

As hinted at in Commonly Made Mistake 9, the name of a quantified variable is not relevant. You can therefore rename bound variables in quantified formulas at will.

On the other hand, unbound variables refer to specific elements and cannot be renamed. The following example takes some liberties with respect to unbound variables.



First of all, something interesting happens on line 4. Here, y is declared as a free variable, but it also occurs as a bound variable inside the universal quantification. This application of $\exists^*$-elimination is invalid, because the meaning of the universal quantification changes in the process

Next, on line 6, an unknown object u is used for $\forall$-elimination. As explained in the previous item, this is not allowed.

In line 7 two things go wrong. First of all, $P(u)$ cannot be used for the $\Rightarrow$-elimination, since u is not necessarily equal to y. Furthermore, the conclusion cannot be $Q(u)$, again, since u and y are not identified.

Finally, the application of $\neg$-elimination on line 9 is incorrect, for the same reason: you cannot simply identify two unbound variables without first deriving $u = v$.