Some things that seems to be particularly hard to get ones head around

Jonathan Cederberg
jonathan.cederberg@it.uu.se

April 7, 2009

Abstract
This is some comments on things that seems to be particularly hard to get your head around (obviously, it says so in the title). More specifically, it is a collection of oft-asked questions in the course programming theory. Any mistakes in the below text are entirely the author fault (again, obviously), and information of any such discovered errors are highly appreciated.

1 General Hints

• Look in the compendium, use the exact same pattern when writing your own proofs.
• Always list the arithmetic rule you are using. This will decrease the risk that you use it incorrectly.
• Write lemmas, but do it separately and not inside of your proof. You are not allowed to use the assumptions of your proofs when proving the lemmas.
• Make sure you type your proof correctly. In particular, make sure that you are aware that “=” can denote both arithmetic equality and logical equivalence.
• Do not use rules with implication in a proof in the same way as you use rules with equivalence.
• Also the converse: make sure you do not weaken the statement and treat it as if you have an equivalence. For example, $x + 3 = y$ is not equivalent to $x < y$, so you cannot have a proof with equivalences where you go from the first to the second.
• Symmetric problems should not turn asymmetric. This is a very simple sanity-check for proofs.
• Be careful with disjunctions in assumptions. You have to assume the whole disjunction, not the parts individually.
2 Layout of proofs

Although we have seen several proof strategies (see handout on proof theory), the layout should always be the same. It is important to keep a few basic things in mind. The number one rule is that everything should be clearly presented. This means, for example, that the only rules that one is allowed to skip are commutativity and associativity. For the sake of clarity, and also for the sake of not doing unnecessary mistakes, one should always keep parantheses. If one uses some rule of arithmetic, that rule should be explicitly stated.

In general, any solution should be of this form:

Prove:

[Statement you want to prove]

Proof:

Assume: $p_1$

\[ \ldots \]

Assume: $p_k$

\{ k assumptions, where $k$ might be 0 \}

$q_1 = \{ \text{some proof rule} \}$

$q_2 = \{ \text{some proof rule} \}$

$q_3 = \{ \text{some proof rule} \}$

$\vdots$

$q_n = \{ \text{some proof rule} \}$

In particular, this means that one can not assume something in the middle of the proof, and then continue. It is also important to note that the only rules one is allowed to use, are the ones in the compendium, plus any lemmas one might already have proved. If one wants to use for example substitution using a rule saying “$p = q$” to substitute $p$ for $q$, we need to have an expression involving exactly $p$ at our current stage in the proof. We cannot have “almost” $p$.

To sum this up, always write proofs as they are presented in the compendium.

3 How not to use rules with implication

When looking at the set of axioms and theorems, one can easily spot quite a few that is of the form $p \Rightarrow q$. This, one might think, is perfect for proving implications, because one can drop the proofstyle with “$= \{ \text{some proof rule} \}$” everywhere, for the weaker statement “$\Rightarrow$"
[some proof rule involving implication]. However, this highly problematic. As a small example, consider the deduction rule saying that from “p = q”, we can deduce “E(p) = E(q)” (*). However, it is not true in general that from “p \implies q”, one can deduce “E(p) \implies E(q)” (**). As an example, take E(r) = ¬r. Here clearly (*) is true, but not (**).

The bottom line here is of course that one should use the rules involving implication carefully. And, as remarked in the previous Section, be sure to write proofs exactly as in the compendium. There is a reason that the compendium only has proofs with equality.

4 The use of conditional substitution

A very powerful derivation rule, and also a rule which can be hard to use correctly, is the rule of conditional substitution:

\[
\begin{align*}
q_1, \ q_2, \ldots, \ q_n, \ (q_1 \land q_2 \land \ldots \land q_n) \Rightarrow (p_1 = p_2) \\
E(p_1) = E(p_2), \ E(p_2) = E(p_2)
\end{align*}
\]

Let us take a closer look at this derivation rule. It says that if we know that q_1, \ldots, q_n are true, and that \((q_1 \land q_2 \land \ldots \land q_n) \Rightarrow (p_1 = p_2)\) is true, we can deduce \(E(p_1) = E(p_2)\) and \(E(p_2) = E(p_1)\). That is, we can deduce that given the premises, any expression involving p_1 (written as E(p_1) in the derivation rule) is equivalent to the same expression with p_1 substituted for p_2 (written E(p_2)).

Suppose we are in the middle of a proof and we want to use conditional substitution with some rule saying “\((q_1 \land q_2) \Rightarrow (p_1 = p_2)\)” to replace “E(p_1)” with “E(p_2)”. According to the rule above, we need establish that q_1 and q_2 are true, in order for the substitution to be valid. The question is how this can be established. The answer here is very simple. The premises q_1 and q_2 can be in two places: either they are assumptions, or they are axioms or theorems. In other words, the following proof outline would be the only valid way to use conditional substitution and the rule above (the [...] means a skipped part of the proof that is irrelevant to illustrate the point here, but one should remember that the proof always should conform to the rules of the last Section):

Proof outline

[...]

Proof:

Assume: q_1

Assume: q_2

[...]

E(p_1) = \{\text{some proof rule}\}

E(p_2) = \{\text{Conditional Substitution using (q_1 \land q_2) \Rightarrow (p_1 = p_2) and assumptions “q_1” and “q_2”}\}

[...]
There is actually three more cases, namely those where \( q_1, q_2 \) or both \( q_1 \) and \( q_2 \) are theorems and not assumptions.

There is also the question of how to handle the case when there is an assumption that “kind of” fits some rule for conditional substitution. One example might be that there was an assumption of the form “\( m < n + 1 \)”, but the rule one wants to use for conditional substitution is of the form “\( (m \leq n) \Rightarrow (p = q) \)”. The answer is that there is a very simple way to handle this situation: prove a lemma stating “\( (m < n + 1) \Rightarrow (p = q) \)” and then use conditional substitution using this lemma and the assumption.

To sum this up, conditional substitution is powerful, but needs to be treated with respect. One should make sure to always state the rules and the assumptions that the conditional substitution is based on, if not only for the fact that this is an extra sanity test to see that is is really applicable.

5 The definition of the alternative command and the alternative command theorem

The alternative command has the form

\[
IF: \quad \text{if } B_1 \rightarrow S_1 \\
\square \quad : \rightarrow : \\
\square \quad B_n \rightarrow S_n \\
\text{fi}
\]

Intuitively, the behaviour of this command is that non-deterministically, a guard \( B_i \) that evaluates to true in the current state is chosen, and the statement \( S_i \) is executed.

Definition of the weakest precondition for the alternative command The weakest precondition of the alternative command theorem is defined as follows:

\[
wp(IF, R) = (B_1 \lor \ldots \lor B_n) \land (B_1 \Rightarrow wp(S_1, R)) \land \ldots \land (B_n \Rightarrow wp(S_n, R))
\]

Let us examine conjunction on the left-hand side of this definition closer. Recall that \( wp(IF, R) \) defines the set of states such that \( IF \) can be executed and we end up in in a state satisfying \( R \). The first conjunct \( (B_1 \lor \ldots \lor B_n) \) says that for all the states, there should be at least one true guard. Each of the following conjuncts are of the form \( B_i \Rightarrow wp(S_i, R) \).

What does this mean? Suppose that the guard \( B_i \) is true. Then you must be sure to end up in a state satisfying \( R \) after executing \( S_i \).
**Alternative Command Theorem**  If a predicate $Q$ satisfies

(1) $Q \Rightarrow (B_1 \lor \ldots \lor B_n)$

(2) $(Q \land B_i) \Rightarrow \text{wp}(S_i, R)$ for all $i$ such that $1 \leq i \leq n$

Then, and only then, $Q \Rightarrow \text{wp}(IF, R)$.

The question is now why this theorem is useful. The answer here, is that the alternative command provides a shortcut for proving the correctness of programs of the form

\[
\begin{align*}
\{ Q \} \\
\text{IF} \\
\{ R \}
\end{align*}
\]

The theorem can not be used (directly) to solve problems of the form “simplify $\text{wp}(IF, R)$”, since the theorem is stated including $Q$. This might be more clear if we consider the following, alternative, formulation of the theorem:

\[
((1) \land (2)) \Rightarrow (Q \Rightarrow \text{wp}(IF, R))
\]

Here, we have of course (1) and (2) as above. With this formulation, we can see that the alternative command theorem does not speak about $\text{wp}(IF, R)$ directly, but rather its relation with $Q$.

To sum up, if you want to simplify an expression that includes the alternative command, use its definition of the weakest precondition. If you want to prove a program correct, and the program has exactly the form of $IF$ above, use the alternative command theorem.