Substitution into Arithmetic Equations

1 Proof of the Substitution Lemma

The Substitution Lemma concisely states the connection between the meanings of an expression before and after substitution.

Lemma (Substitution).

$$[\![e[x := f]]\!]V = [\![e]\!](V[x \leftarrow [\![f]\!]V]), \tag{1}$$

where for any function, F, and elements a, b, we define $F[a \leftarrow b]$ to be the function G such that

$$G(u) = \begin{cases} b & \text{if } u = a. \\ F(u) & \text{otherwise.} \end{cases}$$

Using the informal terminology of Scheme evaluation, the righthand side of (1) describes evaluating the application of $\lambda(x)e$ to f using an *environment model*, while the lefthand side describes evaluating the application using a *substitution model*.

A standard technique for proving things about inductively defined objects like arithmetic expressions is to use *structural induction*. This is an inductive proof in which the base cases are those of the definition – for ae's this means base cases considering expressions that are *constants* and expressions that are *variables*. The induction cases correspond to the clauses of the induction definition where new objects are defined from prior ones—for ae's this means cases for sums, products, and negations[1] In proving the induction cases, the hypothesis to be proved may be *assumed*—by induction – to hold for the constituents of an object. For example, to prove the hypothesis holds in the case of a sum of ae's, we may assume by induction that the hypothesis holds for each of the two summands.

Structural induction could be justified as a special case of induction on the size of ae's, but when a proof is organized in this way, it's clearer to describe it as proof by structural induction.

For this problem there are two ae's, e and f on which structural induction could conceivably be based, but structural induction on e is the one that works nicely:

Proof. Letting V' be the valuation $V[x \leftarrow [\![f]\!]V]$ and e' be the expression $e[x := f]$, we prove that $[\![e']\!]V = [\![e]\!]V'$ by structural induction on the arithmetic expression e.

[1] After handling the sum and product cases, the negation case is invariably routine, and we usually skip it. This can be justified rigorously by regarding $-e$ as an abbreviation for the ae $(-1) \cdot e$.

Base case (e is a constant) Namely, e is the numeral $\hat{n}$ for some integer, n. So by the definition of valuation, the value of e is n in all valuations. Also, by the definition of substitution, e' is the same numeral, $\hat{n}$. So, the value of e' is also n in all valuations, and in particular, $[\![e']\!]V = n = [\![e]\!]V'$.

Base case (e is a variable) Namely, e is a variable, y.

There are two subcases, depending on whether y is distinct from x.

If $y \neq x$, then by the definition of substitution, e' is also the variable y. So, the value of e' in V is $V(y)$ and the value of e in V' is $V'(y)$. But since x is distinct from y, we have $V'(y) = V(y)$ by definition of V'. So $[\![e']\!]V = V(y) = V'(y) = [\![e]\!]V'$.

If y is the variable x, then by the definition of substitution, e' is the expression f. Also, $[\![f]\!]V = V'(x)$ by definition of V'. So,

$$[\![e']\!]V = [\![f]\!]V = V'(x) = [\![x]\!]V' = [\![e]\!]V'.$$

Structural induction case (e is a sum) So e is of the form $e_1 + e_2$ for some ae's e_1, e_2. By induction we may assume $[\![e'_i]\!]V = [\![e_i]\!]V'$ for $i = 1, 2$. Now,

$$\begin{aligned}
e' &= e'_1 + e'_2, \quad \text{so} && \text{def. of substitution} \\
[\![e']\!]V &= [\![e'_1 + e'_2]\!]V \\
&= [\![e'_1]\!]V + [\![e'_2]\!]V && \text{(def. of } [\![+]\!]) \\
&= [\![e_1]\!]V' + [\![e_2]\!]V' && \text{(ind. hypoth.)} \\
&= [\![e_1 + e_2]\!]V' && \text{(def. of } [\![+]\!]) \\
&= [\![e]\!]V',
\end{aligned}$$

as required.

Structural induction case (e is a product) Similar to the sum case.

Structural induction case (e is a negation) Similar to the sum case.

□

2 Soundness of Substitution

"Substituting equals for equals" is a basic rule of algebra: if $f_1 = f_2$, we can replace occurrences of f_1 by f_2. The Substitution Lemma provides a quick and precise justification for this rule.

Proposition 2.1. *If* $\models e_1 \leq e_2$ *and* $\models f_1 = f_2$, *then* $\models e_1[x := f_1] \leq e_2[x := f_2]$.

Proof. Suppose $\models e_1 \leq e_2$ and $\models f_1 = f_2$. So by definition of $\models$, we have

$$[\![e_1]\!] \leq [\![e_2]\!] \tag{2}$$

and

$$[\![f_1]\!] = [\![f_2]\!]. \tag{3}$$

Now for any valuation, V,

$$\begin{aligned}
[\![e_1[x := f_1]]\!]V =& [\![e_1]\!](V[x \leftarrow [\![f_1]\!]V]) && \text{(by Subst. Lemma)}\\
\leq& [\![e_2]\!](V[x \leftarrow [\![f_1]\!]V]) && (by\ (2))\\
=& [\![e_2]\!](V[x \leftarrow [\![f_2]\!]V]) && (by\ (3))\\
=& [\![e_2[x := f_2]]\!]V && \text{(by Subst. Lemma))}
\end{aligned}$$

Since V was arbitrary, we conclude that $[\![e_1[x := f_1]]\!]V \leq [\![e_2[x := f_2]]\!]V$ for *all* valuations V, that is, $\models e_1[x := f_1] \leq e_2[x := f_2]$. □

Since, $\models e_1 = e_2$ iff [$\models e_1 \leq e_2$ and $\models e_2 \leq e_1$], we can immediately conclude that Proposition 2.1 holds for substituting into equalities as well as inequalities:

Corollary 2.2. *If* $\models e_1 = e_2$ *and* $\models f_1 = f_2$, *then* $\models e_1[x := f_1] = e_2[x := f_2]$.

On the other hand, substituting *un*equals is another story, because

$$\models f_1 \leq f_2 \text{ does } \textit{not} \text{ imply } \models e[x := f_1] \leq e[x := f_2].$$

Just consider the case when e is $-x$, f_1 is 0, and f_2 is 1.

3 Substitution as a Derived Rule

Proposition 2.1 shows that substituting equals *into* inequalities is a *sound* rule for proving new inequalities. So this is an inference rule we might add to strengthen our formal system, $\vdash_{\leq}$, for proving inequalities. But this is not necessary, because it turns out that any inequality that could be proved using the proposed new rule could actually have been proved without adding the rule. So we can use substitution into inequalities as a *derived rule*: even though substitution is not one of the given inference rules of the proof system, we can use it in showing that an inequality is provable:

Proposition 3.1. *If* $\vdash e_1 \leq e_2$ *and* $\vdash f_1 = f_2$, *then* $\vdash (e_1[x := f_1] \leq e_2[x := f_2])$.

The proof of Proposition 3.1 will follow from two special cases.

Lemma 3.2. *If* $\vdash f = g$, *then* $\vdash (e[x := f] = e[x := g])$.

By completeness we have that $\models (f = g)$ iff $\vdash_= (f = g)$, so Lemma 3.2 about $\vdash$ follows immediately from the corresponding property of $\models$ given in Corollary 2.2.

Nevertheless we give a different proof of Lemma 3.2. This proof in instructive because it depends only on the proof rules for equality, unlike completeness, which depends on the Ring properties of real numbers and integers.

Proof. By structural induction on e.

Base case (e is a constant) So e is c, where c is a numeral. Then by the definition of substitution, $e[x := f]$ is actually c. Likewise, $e[x := g]$ is actually c. So $(e[x := f] = e[x := g])$ is actually the equation $c = c$, which is immediately provable by reflexivity.

We were careful to say above that "$e[x := f]$ *is actually* the arithmetic expression c" instead of "$c = e[x := f]$." We don't want to confuse an argument that two different *arithmetic* expressions are provably equal, with an argument showing that two different *mathematical* expressions describe the same arithmetic expression. This confusion sometimes traps students: they assert that since

- $c = e[x := f]$ and
- $c = e[x := g]$, it follows that
- $e[x := f] = e[x := g])$ is provable *by transitivity and symmetry*. (NO!)

But we proved mathematically that "$e[x := f]$" and "$e[x := g]$" describe the same constant expression, c. Our *mathematical argument* implicitly used the transitivity and symmetry properties of "sameness." On the other hand, the *formal proof* of the equation $c = c$ follows from reflexivity, not symmetry or transitivity.

Note also that in this base case, we did not need to use the hypothesis $\vdash f = g$.

Base case (e is a variable) Namely, e is a variable, y.

There are two subcases, depending on whether y is distinct from x. If y is distinct from x, then both $e[x := f]$ and $e[x := g]$ are the same as y, and $\vdash y = y$ follows from reflexivity as in the constant case.

If e is x, then by the definition of substitution, $e[x := f]$ is the expression f. Likewise $e[x := g]$ is g. That is, this base case requires that we show that $\vdash f = g$. But this is exactly the given hypothesis, so we are done.

Structural induction case (e is a sum) So e is of the form $e_1 + e_2$. Assume $\vdash f = g$. Now by structural induction, we may assume that

$$\vdash (e_1[x := f] = e_1[x := g]),$$
$$\vdash (e_2[x := f] = e_2[x := g]).$$

It follows by (+-congruence) that

$$\vdash (e_1[x := f] + e_2[x := f] = e_1[x := g] + e_2[x := g]) \tag{4}$$

But by the definition of substitution, the lefthand side of equation (4) is actually $e[x := f]$, and the righthand side is actually $e[x := g]$, so (4) actually says that $\vdash (e[x := f] = e[x := g])$, as required.

Structural induction cases (product and negation) Essentially the same as the sum case.

□

Lemma 3.3.

$$\vdash f = g \textit{ implies } \vdash (f[x := e] = g[x := e]),$$
$$\vdash f \leq g \textit{ implies } \vdash (f[x := e] \leq g[x := e]).$$

Proof. Structural induction on formulas is not always the way to go. This time to prove Lemma 3.3 we use induction on *the length of the formal proof* in $\vdash_{\leq}$ of the given provable equation or inequality.

Base case The proof of $f = g$ (or $f \leq g$) is of length 1. This means that it is an axiom. Now we claim that $f[x := e] = g[x := e]$ is another instance of the *same* axiom, and so also has a proof of length 1.

We illustrate why the claim holds for the +-commutativity axiom, $f + g = g + f$. Namely, we claim that the equation

$$(f + g)[x := e] = (g + f)[x := e] \tag{5}$$

is also an instance of +-commutativity. To see this, let h_1 be the expression $f[x := e]$ and h_2 be $g[x := e]$. But by the definition of substitution, the expression on the lefthand side of equation (5) describes the term $h_1 + h_2$ and the righthand side describes $h_2 + h_1$, confirming the claim for this axiom.

The proofs of the claim for the other axioms follow similarly.

Induction The proof of $f = g$ (or of $f \leq g$) is of length $n + 1$ for some $n \geq 1$.

In this case we use the fact that all the formal inference rules of Notes 3, Tables 1 & 3, have a property similar to that claimed for the axioms. Namely, if we perform the same substitution on all the antecedents and the consequent of a rule, we obtain another instance of the same rule. The argument proving this is similar to the argument above for the commutativity axiom. We leave it to the reader to check this fact.

Now, if $f = g$ is an axiom, the proof for the base case shows that $f[x := e] = g[x := e]$.

Otherwise, $f = g$ must follow from an instance of an inference rule whose antecedents appear earlier in the formal proof. The antecedents must have proofs of length at most n, so by induction, we may assume that the result of substituting e for x in any antecedent equation or inequality is also provable.

Now we just observed that substituting e for x into the antecedents and consequent of the final inference rule used to prove $f = g$ yields another instance of the rule. But since the substituted antecedents are provable, we can conclude that the substituted consequent, namely $f[x := e] = g[x := e]$, is also provable, as required.

□

Now suppose $e_1 \leq e_2$ and $f_1 = f_2$ are both provable. Then

$$e_1[x := f_1] \leq e_2[x := f_1]$$

is provable by Lemma 3.3, and

$$e_2[x := f_1] = e_2[x := f_2]$$

is provable by Lemma 3.2. So by (transitivity),

$$\vdash e_1[x := f_1] \leq e_2[x := f_2]$$

which completes the proof Proposition 3.1.