**Recitation 10: Convergence and Uniform Substitutions**
**15-424/15-624/15-824 Logical Foundations of Cyber-Physical Systems**

**Notes by Brandon Bohrer/Yong Kiam Tan.**
**Edits by Katherine Cordwell (kcordwel@cs.cmu.edu).**

# 1 Announcements

- Proposal due next Tuesday. We're looking for some concrete advances since the White Paper and some preliminary results.

# 2 Motivation and Learning Objectives

We shall start by rounding off our discussion of hybrid games with the loop convergence rule. The loop convergence rule is useful because it allows us to prove diamond properties of loops, i.e., properties that look like $\langle\alpha^*\rangle P$ (or also $[\alpha^\times]P$ for hybrid games). The version we shall look at is sound for both hybrid programs (proving that there is some run of the loop satisfying $P$) and hybrid games (proving that Angel can win the repetition game $\alpha$ into $P$). The rule could come in handy if you need to prove liveness (or Angel's) properties of loops for your course projects.

Next, we will discuss uniform substitution, which underlies the design of KeYmaera X. In an earlier recitation, we listed a few things that could "go wrong" even though the model has been verified. One of these concerns was maybe our proof was incorrect. Back then, we trusted KeYmaera X to check our proof. But what if KeYmaera X had a bug? How can we trust what it says then?

Uniform substitution is the key answer: it allows the soundness-critical core of KeYmaera X to be implemented with $< 2000$ lines of code. This means that when KeYmaera X says your proof is correct, you are really depending only on these lines of code rather than all of the automation around it. For example, when you click on the `ODE` tactic in KeYmaera X, it is really running many heuristics/computations to try to prove your ODE goal. However, you do not need to trust any of these heuristics because everything it manages to prove successfully has gone through the aforementioned soundness-critical core.

# 3 Convergence

You should already be very familiar with how to prove a box property of a loop: just use loop induction with a well-chosen loop invariant $J$. Intuitively, since $J$ is true initially, and every run of the loop preserves $J$, then $J$ should be true after all runs of the loop.

For diamond properties of loops though, the same reasoning (of course) does not work. Instead, we need the very opposite! While the loop **invariant** *never* changes, we want a loop **variant** that *definitely* changes.

**Exercise 1:**
Is this formula valid? If so, how would you prove it?

$$\langle (y := x - 1; x := 1)^* \rangle y = 0$$

Answer: It is valid because the postcondition will definitely be true after two runs of the loop. It can be proved using the $\langle * \rangle$ axiom.

**Exercise 2:**
Is this formula valid? If so, how would you prove it?

$$\langle (y := y - 1; x := x + 1)^* \rangle x \geq y$$

Answer: It is valid because the gap between $x, y$ decreases by two per loop iteration. However, if we were to do this in a "proof" like before, we would need to unfold the loop roughly $\frac{y-x}{2}$ times. But that is silly because your (syntactic) proof should not change in length for different (semantic) values of $x, y$.

To prove this latter formula, we will need the convergence rule.

$$\text{(con)} \quad \frac{\Gamma \vdash \exists v\, p(v), \Delta \quad \vdash \forall v{>}0\, (p(v) \to \langle \alpha \rangle p(v-1)) \quad \exists v{\leq}0\, p(v) \vdash Q}{\Gamma \vdash \langle \alpha^* \rangle Q, \Delta} \quad (v \notin \alpha)$$

Here, $p(v)$ is the aforementioned loop variant. Intuitively, $p(v)$ is an abstract distance measure to the goal. The first premise says that there is some $v$ so that $p(v)$ is true. The second premise says that whenever $p(v)$ is true initially (for $v > 0$), then there is a run of $\alpha$ such that $p(v-1)$ is true afterwards. The third premise says that if it is ever the case that $p(v)$ is true for $v \leq 0$, then $Q$ is true.

Let us suppose that in our initial state $\omega_0$ we have that $p(v_0)$ is true, where $v_0 > 0$. By repeatedly using the second premise of con, we can construct a sequence of states (related by transitions of $\alpha$) $\overbrace{\omega_0}^{p(v_0)}, \overbrace{\omega_1}^{p(v_0-1)}, \ldots, \overbrace{\omega_k}^{p(v_0-k)}$, where eventually, the state $\omega_k$ satisfies $p(v_0 - k)$ for $v_0 - k = 0$. The final premise of the con rule says that in such a state, the desired diamond postcondition $Q$ is true.

If instead $v_0 \leq 0$, then we immediately have that $Q$ is true by the third premise.

**Exercise 3:**
What $p(v)$ should we choose for the formula above?
Answer: Probably the most straightforward choice would be $y - x \leq v$. Notice that when $v \leq 0$, this implies the postcondition $x \geq y$. Let us check the other two branches. For the initial branch, we need to check that $\exists v\, y - x \leq v$. This is just true by real arithmetic. More importantly, the middle branch:

$$\dfrac{\dfrac{\dfrac{*}{v>0, y-x\leq v \vdash (y-x-2\leq v-1)}\,\mathbb{R}}{v>0, y-x\leq v \vdash (y-1)-(x+1)\leq v-1}\,\langle;\rangle,\langle:=\rangle}{v>0, y-x\leq v \vdash \langle y:=y-1; x:=x+1\rangle y-x\leq v-1}$$

**Note: The accompanying archive has these two liveness questions as rec10conv1 and rec10conv2 respectively.**
**Note: The convergence rule works well for proving diamond properties of loops but we have not discussed diamond properties of ODEs in very much detail. ODEs with polynomial solutions can be handled in KeYmaera X using its solving capabilities. However, diamond properties for non-solvable (or even just non-linear ODEs) are much more difficult. Please come chat with us if you encounter these types of questions for your course projects.**

# 4 Uniform Substitution

To motivate uniform substitutions, let us examine the assignment axiom that we used several times in our convergence proofs earlier.

$$([:=]) \quad [x:=e]p(x) \leftrightarrow p(e)$$

In the course so far, we were quite happy to use this axiom **schema** but remember that we had to be extra careful when replacing $x$.

**Exercise 4:**
Which of these instances of [:=] are valid?

1. $[x:=x+y]x=1 \leftrightarrow (x+y)=1$
2. $[x:=x+y]x+x=1 \leftrightarrow (x+y)+x=1$
3. $[x:=x+y][x:=x+y]x=1 \leftrightarrow [x:=(x+y)+y]x=1$
4. $[x:=x+y][x:=x+y]x=1 \leftrightarrow [x:=(x+y)+y](x+y)=1$
5. $[x:=x+y][x:=x+y]x=1 \leftrightarrow [x:=x+y](x+y)=1$
6. $[x:=x+y][\{x'=x+y\}]x=1 \leftrightarrow [\{x'=(x+y)+y\}]x=1$
7. $[x:=x+y][(y:=y+1)^*]x=1 \leftrightarrow [(y:=y+1)^*]x+y=1$
8. $[x:=x+y][(y:=y)^*]x=1 \leftrightarrow [(y:=y)^*]x+y=1$

**Note: You can see what KeYmaera X does for each of these assignments. The examples are in rec10clashes.**
Answer:

1. $[x := x + y]x = 1 \leftrightarrow (x + y) = 1$. Valid.
2. $[x := x + y]x + x = 1 \leftrightarrow (x + y) + x = 1$. Not valid. Forgot to substitute an $x$.
3. $[x := x + y][x := x + y]x = 1 \leftrightarrow [x := (x + y) + y]x = 1$. Valid.
4. $[x := x + y][x := x + y]x = 1 \leftrightarrow [x := (x + y) + y](x + y) = 1$. Not valid. Substituted too many $x$.
5. $[x := x + y][x := x + y]x = 1 \leftrightarrow [x := x + y](x + y) = 1$. Valid. In context.
6. $[x := x + y][\{x' = x + y\}]x = 1 \leftrightarrow [\{x' = (x + y) + y\}]x = 1$. Not valid. Substitution into context where $x$ gets bound.
7. $[x := x + y][(y := y + 1)^*]x = 1 \leftrightarrow [(y := y + 1)^*]x + y = 1$. Not valid. Substitution into context where $y$ gets bound.
8. $[x := x + y][(y := y)^*]x = 1 \leftrightarrow [(y := y)^*]x + y = 1$. Valid, but not a substitution instance!

If we were to implement the [:=] axiom schema, we would also need to implement all of these checks that we just did for the questions above. Even worse: the implementation of every new axiom that comes with soundness side conditions would need these checks as well.

The realization behind uniform substitution is that most (if not all) of these side conditions can be dealt with cleanly by a substitution admissibility conditions on the uniform substitution proof rule. Before we get there, however, we need to (as usual) build up our syntax and semantics.

## 4.1 Syntax and Semantics

The syntax of terms, formulas, and programs are respectively extended with $k$-ary function symbols ($f$), $k$-ary predicate symbols ($p$), and program constant symbols $a$:

$$\begin{aligned} e &::= \ldots \mid f(e_1, \ldots, e_k) \\ P, Q &::= \ldots \mid p(e_1, \ldots, e_k) \\ \alpha &::= \ldots \mid a \end{aligned}$$

At first glance, it does not seem like we have added much at all. For example, why write down $a$ when we have been perfectly fine writing $\alpha$ all the time? The important thing to note is that we can now write down a *concrete* formula such as:

$$[a](p(x) \land f(x, y) \geq 0)$$

In contrast, for most of this course, we just wrote something like:

$$[\alpha](P(x) \land e(x, y) \geq 0)$$

This really means that $\alpha, P, e$ are placeholders for concrete programs that you would fill in later. Thus, it is not a formula but rather a schematic that stands for an infinite number of possible formulas. In the same way, [:=] that we have been calling an "axiom" is really an axiom schema that stands for an infinite number of possible concrete instances (minus those that fail the side condition).

Once we admit these new terms, formulas and programs, we also need to say what they mean.

**Exercise 5:**
So what should a function symbol like $f(e_1, \ldots, e_k)$ mean?
Answer: We might first try something like:

$$\omega[\![f(e_1, \ldots, e_n)]\!] = f(\omega[\![e_1]\!], \ldots, \omega[\![e_n]\!])$$

But that does not make sense! Remember that $f$ is just a piece of syntax, but on the right-hand side we are treating it as a $k$-ary function over the reals. The solution is to recall how we dealt with the semantics of variables $x$. We added the additional state $\omega$ that tracks the value of $x$ and defined:

$$\omega[\![x]\!] = \omega(x)$$

Just like $\omega$ tells us the meaning of the variables, we shall fix an interpretation $I$ which we use to tell us the meaning of the new symbols like $f$.[1]

Given this interpretation $I$, the semantics of the newly added symbols become straightforward lookups in the interpretation:

$$\begin{aligned} \omega[\![f(e_1, \ldots, e_n)]\!] &= I(f)(\omega[\![e_1]\!], \ldots, \omega[\![e_n]\!]) \\ [\![p(e_1, \ldots, e_n)]\!] &= \{\omega \mid (\omega[\![e_1]\!], \ldots, \omega[\![e_n]\!]) \in I(p)\} \\ [\![a]\!] &= I(a) \end{aligned}$$

Now that we know the syntax and semantics, we can revisit the assignment axiom schema and rewrite it as a single concrete axiom ($p$ is a 1-ary predicate symbol while $f$ is a 0-ary function symbol):

$$([:=]) \quad [x := f]p(x) \leftrightarrow p(f)$$

A note on notation: Sometimes we'll write $c()$ for a 0-ary function symbol.

**Exercise 6:**
Make sure you understand the difference between this assignment **axiom** versus the earlier assignment **axiom schema**. In particular, this axiom has no side conditions.

[1] This follows the style of definition given in the textbook. More formally, one would redefine all of the semantics to carry the interpretation around explicitly.

**Exercise 7:**
Prove the soundness of the axiom [:=].
Answer: If you get stuck, see Lemma 18.2 in the textbook.

We have got rid of the problem of having a schematic assignment axiom, but we have not really made the proof calculus much better (yet). The new [:=] axiom does not allow us to do much by itself!

## 4.2 Uniform Substitution

The key proof rule that enables effective use of [:=] is uniform substitution:

$$\text{(US)} \quad \frac{\phi}{\sigma(\phi)}$$

which is sound provided $\mathrm{FV}(\sigma|_{\Sigma(\theta)}) \cap \mathrm{BV}(\otimes(\cdot)) = \emptyset$ for each operation $\otimes(\theta)$ in $\phi$.

This proof rule says that if we can prove formula $\phi$, then we also have a proof of the conclusion $\sigma(\phi)$ which is obtained by applying the substitution $\sigma$ on formula $\phi$. We define $\sigma(\phi)$ by structural induction on $\phi$ (see Figure 18.1 in the textbook).

The substitution $\sigma$ maps the various symbols that we have introduced earlier to concrete terms, formulas and programs (which may themselves mention free variables).

The condition that "$\mathrm{FV}(\sigma|_{\Sigma(\theta)}) \cap \mathrm{BV}(\otimes(\cdot)) = \emptyset$ for each operation $\otimes(\theta)$ in $\phi$" cleanly encapsulates/handles the side conditions that we saw before. This mechanical check prevents us from making unsound conclusions. The intuitive way to remember the condition is that the substitution should never introduce a free variable into an environment where it is bound.[2]

As an example, let us consider the substitution:

$$\sigma_1 = \{f \mapsto x + y, p(\cdot) \mapsto \cdot = 1\}$$

Applying $\sigma_1$ on the [:=] axiom yields the following proof of the very first example in Exercise 4.

$$\dfrac{\dfrac{*}{\vdash [x := f]p(x) \leftrightarrow p(f)}\,[:=]}{\vdash [x := x + y]x = 1 \leftrightarrow x + y = 1}\,\text{US}$$

In order to conclude that this use of US is sound/works, we perform the following calculation, following the recursive definition in Figure 18.1:

$$\begin{aligned}\sigma_1([x := f]p(x) \leftrightarrow p(f)) &= \sigma_1([x := f]p(x)) \leftrightarrow \sigma_1(p(f)) \\ &= [\sigma_1(x := f)]\sigma_1(p(x)) \leftrightarrow \sigma_1(p(f))\end{aligned}$$

Here already we need to check a side condition: in order to be able to get to the third line, $\sigma_1$ must be $BV(\sigma_1(x := f))$-admissible for $p(x)$, i.e. $\{x\}$-admissible for $p(x)$. That is, we

[2] Or else you will be sent to logic jail!

need $\mathrm{FV}(\sigma_1|_{\Sigma(p(x))}) \cap \{x\} = \emptyset$, where $\mathrm{FV}(\sigma_1|_{\Sigma(p(x))})$ restricts attention to all the symbols in $p(x)$ that are mapped by $\sigma_1$, i.e. $p$. $\mathrm{FV}(\sigma_1|_{\{p\}})$ is empty, so this condition is satisfied. This check follows Definition 18.7 in the textbook.

Now we can continue our calculation:

$$\begin{aligned}[\sigma_1(x := f)]\sigma_1(p(x)) \leftrightarrow \sigma_1(p(f)) &= [x := \sigma_1(f)](\sigma_1(p))(\sigma_1(x)) \leftrightarrow \sigma_1(p)\sigma_1(x) \\ &= [x := x + y](\sigma_1(p))(\sigma_1(x)) \leftrightarrow (\sigma_1(p))(\sigma_1(x))\end{aligned}$$

By definition, $(\sigma_1(p))(\sigma_1(x))$ is $\{\cdot \mapsto x\}(\cdot = 1)$, which simplifies to $x = 1$. Similarly, $(\sigma_1(p))(\sigma_1(f))$ is $\{\cdot \mapsto x + y\}(\cdot = 1)$, which simplifies to $x + y = 1$. Substituting yields the desired goal:

$$\sigma_1([x := f]p(x) \leftrightarrow p(f)) \leftrightarrow [x := x + y]x = 1 \leftrightarrow x + y = 1.$$

So US was indeed safe to use for $\sigma_1$. We can also come up with a substitution that would work for the second example in Exercise 4, but using US should *not* be sound with this substitution.

**Exercise 8:**
Give a substitution $\sigma_2$ that would yield the second example in Exercise 4.
Answer: Recall that the bug in this example was that we left out one of the $x$. Concretely, this features in the substitution as follows:

$$\sigma_2 = \{f \mapsto x + y, p(\cdot) \mapsto \cdot + x = 1\}$$

In other words, the substitution $p(\cdot)$ introduces $x$ free. It is essential that the US proof rule *does not* allow us to conclude this from the assignment axiom as that would be unsound.

The US proof rule prevents this with an admissibility condition on $\sigma$.

Let us concretely see what goes wrong in $\sigma_2$ by working through applying the substitution. At the top-level we simply recursively apply the substitution on both sides of the equivalence:

$$\sigma_2([x := f]p(x) \leftrightarrow p(f)) = \sigma_2([x := f]p(x)) \leftrightarrow \sigma_2(p(f))$$

For the RHS, the substitution works fine:

$$\begin{aligned}\sigma_2(p(f)) &= \sigma_2(p)(\sigma_2(f)) \\ &= \sigma_2(p)(x + y) \\ &= ((x + y) + x = 1)\end{aligned}$$

On the LHS, we can also recursively apply the substitution along each of the operators:

$$\begin{aligned}\sigma_2([x := f]p(x)) &= [\sigma_2(x := f)]\sigma_2(p(x)) \\ &= [x := \sigma_2(f)]\sigma_2(p)(\sigma_2(x)) \\ &= ([x := x + y]x + x = 1)\end{aligned}$$

**Exercise 9:**
Which step did we make a mistake?
Answer: The very first step actually fails. The expansion for substituting a box modality $\sigma([\alpha]\phi)$ requires that the free variables introduced by the substitution in $\phi$ are not contained in $BV(\sigma(\alpha))$. However, we have done exactly that because the variable $x$ is bound by $x := x+y$, but is introduced free by the substitution $p(\cdot) \mapsto \cdot + x = 1$. Notice that substitution (and checking) is completely mechanical once we have got the definitions correct.

Contrast this with $\sigma_1$ which maps $p(\cdot) \mapsto \cdot = 1$. Here, we are safe because $x$ is not introduced free into a context where it is bound.

Let us go through each of the remaining cases in Exercise 4 and determine why (or if) a clash would occur. This should give you lots of practice with writing down substitutions and determining when clashes occur. Assignments, of course, are not the only case where subtle binding issues can occur.

**Exercise 10:**
Give a substitution $\sigma_3$ for the 3rd example from Exercise 4.
Answer:

$$\sigma_3 = \{f \mapsto x + y, p(\cdot) \mapsto [x := \cdot + y]x = 1\}$$

This substitution $\sigma_3$ is fine, unlike $\sigma_2$ because $p(\cdot) \mapsto [x := \cdot]x = 1$ does not introduce $x$ free. Here is the calculation:

$$\begin{aligned}
\sigma_3([x := f]p(x) \leftrightarrow p(f)) &= \sigma_3([x := f]p(x)) \leftrightarrow \sigma_3(p(f)) \\
&= [x := \sigma_3(f)]\sigma_3(p(x)) \leftrightarrow \sigma_3(p(f)) \\
&= [x := x + y]\sigma_3(p)\sigma_3(x) \leftrightarrow \sigma_3(p)\sigma_3(f) \\
&= [x := x + y]\{\cdot \mapsto x\}([x := \cdot + y]x = 1) \leftrightarrow \{\cdot \mapsto x + y\}[x := \cdot + y]x = 1 \\
&= [x := x + y][x := x + y]x = 1 \leftrightarrow [x := (x + y) + y]x = 1,
\end{aligned}$$

which is the desired conclusion. The side condition in the second line is okay because $\sigma_3$ is $BV(\sigma_3(x := f))$-admissible for $p(x)$, i.e. we need $FV(\sigma_3|_{\{p\}}) \cap \{x\} = \emptyset$, which holds because $x$ is not free in $p(\cdot) \to [x := \cdot + y]x = 1$.

**Exercise 11:**
Give a substitution $\sigma_4$ for the 4th example from Exercise 4 and explain why it clashes.
Answer:

$$\sigma_4 = \{f \mapsto x + y, p(\cdot) \mapsto [x := \cdot]\cdot = 1\}$$

The reason for this clash is different from before. This time, the LHS actually substitutes just fine because $p(\cdot) \mapsto [x := \cdot]\cdot = 1$ does not introduce $x$ free (like we argued for $\sigma_3$).

Instead, it is the RHS which causes problems because we tried to apply the substitution $\cdot \mapsto x + y$ to $[x := \cdot]\cdot = 1$, which causes $x$ to become bound.

**Exercise 12:**
Give a substitution $\sigma_5$ for the 5th example from Exercise 4.
Answer: This is not a well-formed question because this example is not obtained directly by substitution from [:=]. Rather, it follows from the first example of Exercise 4 together with the contextual equivalence proof rule:

$$\text{(CE)} \quad \frac{P \leftrightarrow Q}{C(P) \leftrightarrow C(Q)}$$

Since $P \leftrightarrow Q$ is true in all states, the rule CE allows us to replace them equivalently in any context $C$.

**Exercise 13:**
Give a substitution $\sigma_6$ for the 6th example from Exercise 4 and explain why it clashes.
Answer:

$$\sigma_6 = \{f \mapsto x + y, p(\cdot) \mapsto [\{x' = \cdot + y\}]x = 1\}$$

Actually, there are two clashes here. When we consider $\sigma_6([x := f]p(x)) \leftrightarrow \sigma_6(p(f))$, on the LHS we have $\sigma_6([x := f]p(x))$ and on the RHS we have $\sigma_6(p(f))$. Each of these causes a clash.

On the LHS of the $\leftrightarrow$, we can't change $\sigma_6([x := f]p(x))$ to $[\sigma_6(x := f)]p(x))$, because $x$ is free in $\{x' = \cdot + y\}x = 1$.

On the RHS of the $\leftrightarrow$, we'd end up with $\{\cdot \mapsto x + y\}[\{x' = \cdot + y\}]x = 1$ and we have another clash. Intuitively, we're trying to substitute a free occurrence of $x$ into the RHS of an ODE when it is bound by the LHS of the ODE.

**Exercise 14:**
Give a substitution $\sigma_7$ for the 7th example from Exercise 4 and explain why it clashes.
Answer:

$$\sigma_7 = \{f \mapsto x + y, p(\cdot) \mapsto [(y := y + 1)^*]\cdot = 1\}$$

This clashes on the RHS because it tries to substitute $\cdot \mapsto x + y$ in $[(y := y + 1)^*]\cdot = 1$ which causes a free instance of $y$ to become bound.

**Exercise 15:**
Give a substitution $\sigma_8$ for the 8th example from Exercise 4 and explain why it clashes.
Answer:

$$\sigma_8 = \{f \mapsto x + y, p(\cdot) \mapsto [(y := y)^*]\cdot = 1\}$$

Similarly to $\sigma_7$, this clashes only on the RHS because it tries to substitute $\cdot \mapsto x + y$ in $[(y := y)^*]\cdot = 1$ which causes a free instance of $y$ to become bound.

However, as we noted earlier, the conclusion is actually valid, just not directly provable with uniform substitution.

Now, let us revisit the V axiom. Crucially, $p$ is a 0-ary predicate symbol without any arguments:

$$\text{(V)} \quad p \to [a]p$$

**Exercise 16:**
Is the following formula valid? Does it follow from V?

$$x = 1 \to [x := 1]x = 1$$

Answer: This formula is valid and it follows from [:=], but NOT from the vacuity axiom.

The substitution clashes because $x$ is bound in the replacement for $a$ in the RHS of the implication. Notice that it is crucial for $p$ to be 0-ary rather than $p(x)$, for example, in the axiom:

$$\sigma = \{p \mapsto x = 1, a \mapsto x := 1\}$$

**Exercise 17:**
Give an example where the sequential composition axiom would clash. Here, $\bar{x}$ stands for the vector of all variables that we are concerned with.

$$([;]) \quad [a; b]p(\bar{x}) \leftrightarrow [a][b]p(\bar{x})$$

Answer: It is actually not possible to produce a clash when you have correctly chosen the substitution. Think about where such a clash would occur.

On the LHS of the equivalence, a clash would occur if we introduced in the substitution for $p(\bar{x})$ a free variable that is bound in the replacement of $a; b$ but $p(\bar{x})$ has license to mention all of the variables $\bar{x}$, and so it does not mention any variables free.

Similarly on the RHS, $p(\bar{x})$ already has license to mention all of the variables, so the resulting substitution does not need to mention any variables free.

**Exercise 18:**
Concretely, consider the formula $[x := 1; y := 1]x + y = 2 \leftrightarrow [x := 1][y := 1]x + y = 2$. Give a substitution that clashes and one that does not clash.

This following substitution clashes because, even though $p$ had license to mention all of its arguments, it chooses not to but instead mentions $x, y$ free (which are bound).

$$\sigma_\dagger = \{a \mapsto x := 1, b \mapsto y := 1, p(\cdot, \star) \mapsto x + y = 2\ \}$$

However, since $p$ has license to mention all of its arguments, the following substitution works (since $p$ mentions no variables free):

$$\sigma = \{a \mapsto x := 1, b \mapsto y := 1, p(\cdot, \star) \mapsto \cdot + \star = 2\ \}$$