Løsninger, I220, åttende oppgavesett

Bjørn Tore Sund

20. oktober 1997

Exercise 1: 7.1-1 (1)-(3)

Problem

Let A, B be Σ-algebras and $\mathcal{C}$ a class of Σ-algebras.

1. Prove that $\equiv_{\{A\}}=\equiv_h$ where $h : T(\Sigma) \to A$ is the evaluation homomorphism of A (cf. definition 3.16 and Theorem 3.58)
2. Prove that $A \simeq B$ implies $\equiv_{\{A\}}=\equiv_{\{B\}}$. Does the reverse also hold?
3. Prove that, if A is initial in $\mathcal{C}$, then $\equiv_{\{A\}}=\equiv_{\mathcal{C}}$. Does the reverse also hold?

Solution

1. By Definition 3.53 (book);

$$\equiv_{\{A\}} \stackrel{\text{def}}{=} \{(t,u) \in T_\Sigma \times T_\Sigma | A(t) = A(u)\}$$

   By Definition 3.16, the congruence relation induced by a a homomorphism is defined by

$$a \equiv_h = a' \text{ iff } h(a) = h(a')$$

   Which, in the case of the evaluation homomorphism, becomes:

$$\equiv_h \stackrel{\text{def}}{=} \{(t,u) \in T_\Sigma \times T_\Sigma | h(t) = h(u)$$

   But by Fact 3.48 (book), the evaluation homomorphism $h : T(\Sigma) \to A$ is defined by

$$h(t) = A(t)$$

   And thus the two definitions are equal.

2. By Definition 3.53 (book);

$$\begin{aligned}\equiv_{\{A\}} &\stackrel{\text{def}}{=} \{(t,u) \in T_\Sigma \times T_\Sigma | A(t) = A(u)\}\\ \equiv_{\{B\}} &\stackrel{\text{def}}{=} \{(t,u) \in T_\Sigma \times T_\Sigma | B(t) = B(u)\}\end{aligned}$$

   Clearly, we have both $\equiv_{\{A\}} \subseteq T_\Sigma \times T_\Sigma$ and $\equiv_{\{B\}} \subseteq T_\Sigma \times T_\Sigma$. Furthermore, by Theorem 2.38 we have that for $\phi : A \to B$ the isomorophism,

$$A(t) = A(u) \Rightarrow \phi(A(t)) = \phi(A(u)) \Rightarrow \phi^-(\phi(A(t))) = \phi^-(\phi(A(u))$$

   with the same relation being valid for B, and by isomorphism we have

$$A(t) = \phi^-(\phi(A(t)))$$

   which implies that

$$\phi^-(B(t)) = A(t) = A(u) = \phi^-(B(t)) \Leftrightarrow \phi(A(t)) = B(t) = B(u) = \phi(A(t))$$

   In other words, $A(t) = A(u)$ iff $B(t) = B(u)$, and thus the definitions of $\equiv_{\{A\}}$ and $\equiv_{\{B\}}$ are equivalent.

   The reverse does not hold. For one, $\equiv_{\{A\}} = \equiv_{\{B\}}$ does not guarantee that $||A|| = ||B||$, since the evaluation homomorphisms $h_A : T(\Sigma) \to A$ and $h_B : T(\Sigma) \to B$ aren't necessarily surjective - both carrier sets might contain "junk".

3. Since A is initial in $\mathcal{C}$, we have that for all $B \in \mathcal{C}$, there is a unique homomorphism $h : A \to B$. But then, by theorem 2.38, we have that if there is a $B \in \mathcal{C}$ such that $B(t) \neq B(u), u, t \in T_\Sigma$, then $A(t) \neq A(u)$, i.e. $A(t) = A(u)$ if and only if $B(t) = B(u)$ for all $B \in \mathcal{C}$. But that means that

$$\begin{aligned}\equiv_{\{A\}} &\stackrel{\text{def}}{=} \{(t,u) \in T_\Sigma \times T_\Sigma | A(t) = A(u)\}\\ \equiv_{\mathcal{C}} &\stackrel{\text{def}}{=} \{(t,u) \in T_\Sigma \times T_\Sigma | B(t) = B(u) \text{ for each } B \in \mathcal{C}\}\end{aligned}$$

   (from Definition 3.53) are equivalent.

   Again, the reverse does not hold. We cannot conclude from $\equiv_{\{A\}} = \equiv_{\mathcal{C}}$ that A is generated, and thus we cannot conclude that it is initial.

Exercise 2:7.3-3

Problem

Let (Σ, Φ) and (Σ, Ψ) be initial specifications with $\Psi \models \Phi$. Show that $T(\Sigma, \Psi)$ is the homomorphic image of $T(\Sigma, \Phi)$ for some Σ-homomorphism.

Solution

We have that $\Psi \models \Phi$, which implies that $\forall \varphi : \Phi \models \varphi \Rightarrow \Psi \models \varphi$. By Definition 5.43, we have

$$\begin{aligned} \Psi^* &\stackrel{\text{def}}{=} \{\varphi \in EL(\Sigma) | \Psi \models \varphi\} \\ \Phi^* &\stackrel{\text{def}}{=} \{\varphi \in EL(\Sigma) | \Phi \models \varphi\} \end{aligned}$$

Thus, we have

$$\Phi^* \subseteq \Psi^* \tag{1}$$

and since $Mod(\Phi) = Mod(\Phi^*)$ by Theorem 5.45, we can work with the closures in stead of the original sets of axioms. By 1 above, we can define a set of formulas $\Gamma \in EL(\Sigma)$ by

$$\Gamma = \Psi - \Phi$$

We let $\equiv_\Gamma$ be the congruence relation induced by this set of equations. Then Definition 3.18 gives us a homomorphism $h = hom_{\equiv_\Gamma} : Mod(\Phi) \to Mod(\Psi)$ which is such that $h(T(\Sigma, \Phi)) = T(\Sigma, \Psi)$.

Exercise 3:7.7-4

Problem

Reconsider Example 7.45 but now with $\Phi \subseteq EL(\Sigma)$ consisting of the equation:

$$(\neg a = b \wedge c = d) \vee (a = b \wedge \neg c = d) = True$$

together with the equations fixing the semantics of the operations:

$\_ = \_ \, : \, el \times el \to bool,$

$\neg \, : \, bool \to bool$, etc.

In Example 7.45 it is proved that $Mod(\Phi)$ has no initial algebra. Why does this proof not work in the present example?

What will the carrier of the initial algebra look like?

[Hint: Equations fixing the semantics of the defined equality $\_ = \_$ are only the conditional ones making it into equivalence. (Why isn't it necessary to include any congruence axioms?) In particular, there are no other equations, except for the given one, telling, for instance, $a = b$ or the like.]

Solution

Our signature is now:

$$
\begin{array}{lrrl}
S: & el, bool & & \\
\Omega: & a: & & \to el \\
& b: & & \to el \\
& c: & & \to el \\
& d: & & \to el \\
& False: & & \to bool \\
& True: & & \to bool \\
& \_ = \_: & el \times el & \to bool \\
& \neg: & bool & \to bool \\
& \wedge: & bool \times bool & \to bool \\
& \vee: & bool \times bool & \to bool
\end{array}
$$

However, this time $\Phi \in EL(\Sigma)$, which, by Theorem 7.2, implies that $T(\Sigma, \Phi) \in Mod_{\Sigma}(\Phi)$. This again, by Theorem 3.58, implies that $T(\Sigma, \Phi)$ is initial in $Mod_{\Sigma}(\Phi)$.

This works to convince us that the proof in Example 7.45 is invalid for this case. But we need to find out why. Consider the set of terms T_{Σ}. While the original formula

$$(\neg a = b \wedge c = d) \vee (a = b \wedge \neg c = d)$$

defined a relationship between the elements a, b, c, d, the equation

$$(\neg a = b \wedge c = d) \vee (a = b \wedge \neg c = d) = True$$

merely identifies two terms. All it says is that when applying the congruence induced by Φ,

$$(\neg a = b \wedge c = d) \vee (a = b \wedge \neg c = d) \in [True].$$

Note especially that this formula contains no variables, only constants, and thus only identifies *one* term with $True$, not a class of terms.

Note that given this example, the boolean sort will contain junk. Given the equation above, $\neg a = b \wedge c = d$, for instance, will be neither in $[True]$ or $[False]$, but will be a new element in the boolean sort's carrier set.

Exercise 4:9.1-4

Problem

Complete the proof of Fact 9.6: $\mathcal{M}(sp)$ is an abstract data type for each specification sp of the specification language SL.

Proof: The proof is by induction on the structure of *sp*.

Induction basis If *sp* s an atomic specification, the property directly results from the definition of $\mathcal{M}(sp)$ in Chapters 6 to 8.

Induction step One distinguishes between cases (ii) to (vii) of Definition 9.1.

...

The proof for the cases (iii) to (vii) is left as Exercise 9.1-4.

Solution

3. $sp =$(**rename** sp' **by** μ):
Assume $A \in \mathcal{M}(sp)$ and $B \simeq A$. We need to prove that $B \in \mathcal{M}(sp)$. Let $i : B \to A$ be the isomorphism. By Definition 4.9 and Fact 4.10,

$$(i \,|\, \mathcal{S}(sp')) \;:\; (B \,|\, \mathcal{S}(sp')) \to (A \,|\, \mathcal{S}(sp'))$$

is a homomorphism too. Since i is bijective, $(i \,|\, \mathcal{S}(sp'))$ is an isomorphism, and $(B \,|\, \mathcal{S}(sp')) \simeq (A \,|\, \mathcal{S}(sp'))$. By definition of the semantics of (**rename** *sp* **by** μ), $(A \,|\, \mathcal{S}(sp')) \in \mathcal{M}(sp')$. Since $\mathcal{M}(sp')$ is an abstract data type by induction hypothesis, $(B \,|\, \mathcal{S}(sp')) \in \mathcal{M}(sp')$, and thus $B \in \mathcal{M}(sp)$.

4. $sp =$(sp' **forget** (S, Ω)):
Assume $A \in \mathcal{M}(sp)$ and $B \simeq A$. We need to prove that $B \in \mathcal{M}(sp)$. Let $i : B \to A$ be the isomorphism. Clearly,

$$\mu(i) \;:\; \mu(B) \to \mu(A),$$

where μ is the inclusion morphism adding (S, Ω) to $\mathcal{S}(sp')$, is also an isomorphism. Since $\mu(A) \in \mathcal{M}(sp')$ by definition of semantics of (sp' **forget** (S, Ω)), and $\mathcal{M}(sp')$ is an abstract data type by induction hypothesis, $\mu(B) \in \mathcal{M}(sp')$, and thus $B \in \mathcal{M}(sp)$.

5. *sp*=(**extend** *sp* **by** (S, Ω)):
This proof is identical to the one for **rename**.

Exercise 5:9.1-5

Problem

1. Simplify the definition of $\mathcal{M}(sp_1 + sp_2)$ for the following cases:
    - $\mathcal{M}(sp_1) \subseteq \mathcal{M}(sp_2)$;
    - $\mathcal{M}(sp_1) \subseteq (\mathcal{M}(sp_2) \,|\, Sig(sp_1))$;
    - $\mathcal{M}(sp_1) = \mathsf{Alg}(Sig(sp_1))$.

2. Is the construct "+" associative?
3. Is $\mathcal{M}(sp_1 + sp_2)$ a monomorphic abstract data type, when $\mathcal{M}(sp_1)$ and $\mathcal{M}(sp_2)$ are?
4. Does $\mathcal{M}(sp_1 + sp_2)$ have an initial algebra, when $\mathcal{M}(sp_1)$ and $\mathcal{M}(sp_2)$ do?

Solution

1. We are to simplify $\mathcal{M}(sp_1 + sp_2)$ given some constraints on the element specifications. First of all, it is important to note that for any specification sp expressed in SL, $\mathcal{M}(sp)$ is a set of algebras, by Definition 9.4. For the sets $\mathcal{M}(sp_1)$ and $\mathcal{M}(sp_2)$ to have any other relation than $\mathcal{M}(sp_1) \cap \mathcal{M}(sp_2) = \emptyset$, it is necessary that the two specifications are over the same signature.

   We have

$$\begin{aligned}
&\mathcal{M}(sp_1 + sp_2) \\
&\quad\stackrel{\text{def}}{=} \quad \{A \in \mathsf{Alg}(Sig(sp_1 + sp_2))| \\
&\qquad\quad (A \,|\, Sig(sp_1)) \in \mathcal{M}(sp_1) \wedge (A \,|\, Sig(sp_2)) \in \mathcal{M}(sp_2)\}
\end{aligned}$$

   - $\mathcal{M}(sp_1) \subseteq \mathcal{M}(sp_2)$:
     Since $\mathcal{M}(sp_1) \subseteq \mathcal{M}(sp_2)$, we have $Sig(sp_1) = Sig(sp_2) = Sig(sp_1 + sp_2)$. We term this signature Σ. Since all the signature are the same, $(A \,|\, Sig(sp_1)) = A$ and $(A \,|\, Sig(sp_2)) = A$, and thus

$$\begin{aligned}
\mathcal{M}(sp_1 + sp_2) &= \{A \in \mathsf{Alg}(\Sigma) | A \in \mathcal{M}(sp_1) \wedge A \in \mathcal{M}(sp_2)\} \\
&= \mathcal{M}(sp_1) \cap \mathcal{M}(sp_2) \\
&= \mathcal{M}(sp_1)
\end{aligned}$$

   - $\mathcal{M}(sp_1) \subseteq (\mathcal{M}(sp_2) \,|\, Sig(sp_1))$:
     We know that there is a $\mu : Sig(sp_1) \to Sig(sp_2)$, where μ is an inclusion. In other words, we have $Sig(sp_1) \subseteq Sig(sp_2)$, which implies that
     $$Sig(sp_1 + sp_2) = Sig(sp_1) \cup Sig(sp_2) = Sig(sp_2)$$
     That $\mathcal{M}(sp_1) \subseteq (\mathcal{M}(sp_2) \,|\, Sig(sp_1))$ gives us that
     $$\forall A \in \mathcal{M}(sp_1) \,:\, \exists A' \in \mathcal{M}(sp_2).(A' \,|\, Sig(sp_1)) = A$$
     But then we have
     $$\mathcal{M}(sp_1 + sp_2) = \{A' \in \mathsf{Alg}(Sig(sp_2)) | (A' \,|\, Sig(sp_1)) \in \mathcal{M}(sp_1)\}$$

   - $\mathcal{M}(sp_1) = \mathsf{Alg}(Sig(sp_1))$:
     This clearly put no restraints on the signature, thus $Sig(sp_1 + sp_2) = Sig(sp_1) \cup Sig(sp_2)$, in accordance with Definition 9.1. Plugging in the semantic definition, we get
     $$\mathcal{M}(sp_1 + sp_2)\{A \in \mathsf{Alg}(Sig(sp_1 + sp_2)) | (A \,|\, Sig(sp_2)) \in \mathcal{M}(sp_2)\}$$
     since there is no constraint on sp_1 at all.

2. The "+" operation is clearly associative. The signature is defined as a simple union, which is an associative operation, and the definition of the model class is based on $\wedge$, which is also associative.

3. By Fact 9.6, $\mathcal{M}(sp_1 + sp_2)$ is an abstract data type. We need to verify that it is monomorphic. Assume that it isn't, i.e. that there exists two algebras $A, B \in \mathcal{M}(sp_1 + sp_2)$ such that $A \not\simeq B$. But since A and B are defined exclusively based on their reducts vs. the two specifications, this would imply either that $(A \mid Sig(sp_1)) \not\simeq (B \mid Sig(sp_1))$ or $(A \mid Sig(sp_2)) \not\simeq (B \mid Sig(sp_2))$. But this clearly contradicts the monomorphicness of $\mathcal{M}(sp_1)$ and $\mathcal{M}(sp_2)$, and thus $\mathcal{M}(sp_1 + sp_2)$ is monomorphic.

4. It is possible for $\mathcal{M}(sp_1 + sp_2)$ to be empty, in which case it has no initial algebras.

   However, assuming that $\mathcal{M}(sp_1 + sp_2)$ is non-empty *and* that the initial algebras are quotient algebras, i.e. $I_1 = T(\Sigma)_1/_{\equiv_1}$, $I_2 = T(\Sigma)_2/_{\equiv_2}$. Our claim is now that for $\equiv$ being the smallest congruence on the term algebra $T(\Sigma)$ for $sp_1 + sp_2$ which contains $\equiv_1 \cup \equiv_2$, $T(\Sigma)/_{\equiv}$ is initial in $\mathcal{M}(sp_1 + sp_2)$.

   By Fact 5.28, if $T(\Sigma)/_{\equiv}$ is a model of $\mathcal{M}(sp_1 + sp_2)$, it is initial in $\mathcal{M}(sp_1 + sp_2)$. Assume that $T(\Sigma)/_{\equiv}$ isn't a model of $\mathcal{M}(sp_1 + sp_2)$. This implies that there is some $\varphi \in \equiv_1 \cup \equiv_2$ such that $\varphi \not\in \equiv$, but this contradicts the definition of $\equiv$, and thus $T(\Sigma)/_{\equiv}$ is a model of $\mathcal{M}(sp_1 + sp_2)$.

Exercise 5:9.1-6

Problem

Let sp be the (atomic) loose specification of Example 6.10. Determine the abstract data type defined by the following specifications:

- **extend** sp **by opns** $Isempty : list \to bool$;
- (**extend** sp **by opns** $Isempty : list \to bool$) **model** (**vars**$l : list$, $e : el$ **axioms** $Isempty(Add(e, l)) = False$));
- sp **model** (**vars** $l, m : list$ **axioms** $l.m = m.l$).

Solution

loose spec sorts el, **freely generated** $bool$, $list$
opns **constr** $True : \to bool$
constr $False : \to bool$
constr $[\,] : \to list$

$$
\begin{array}{ll}
 & \textbf{constr } Add : el \times list \to list \\
 & \_.\_ : list \times list \to list \\
 & Isprefix : list \times list \to bool \\
\textbf{vars} & l, m, n : list,\ e : el \\
\textbf{axioms} & [\,].l = l \\
 & Add(e,l).m = Add(e, l.m) \\
 & (Isprefix(l,m) = True) \equiv (\exists n.(m = l.n))
\end{array}
$$

endspec

This is a strictly adequate specification of the "classical" algebra of lists of elements.

- **extend** *sp* **by opns** $Isempty : list \to bool$:
  By the definition of the semantics of **extend**, the abstract data type is expanded to contain algebras with all possible interpretations of $Isempty$.
- (**extend** *sp* **by opns** $Isempty : list \to bool$) **model** (**vars**$l : list$, $e : el$ **axioms** $Isempty(Add(e,l)) = False$)):
  This is the abstract data type where the operation $Isempty$ is always interpreted correctly for non-empty list. But note that the given axiom does *not* specify that $Isempty([\,]) = True$, thus the abstract data type also contains, for instance, the algebra where $\forall l \in list.Isempty(l) = False$.
- *sp* **model** (**vars** $l, m : list$ **axioms** $l.m = m.l$):
  This is the abstract data type of unordered lists. Note that there is no restraint against the same element appearing several times, so this is *not* the abstract data type of sets.

Exercise 6:9.1-7

Problem

Let *sp*, μ, $(\mathcal{S}, \Omega)$ and Φ be as in Definition 9.1. Show the following facts:

1. $\mathcal{M}$(**rename** *sp* **by** μ) is a persistent μ-extension of $\mathcal{M}(sp)$;
2. $\mathcal{M}$(**extend** *sp* **by** $(\mathcal{S}, \Omega)$) is a persistent extension of $\mathcal{M}(sp)$;
3. $\mathcal{M}$(*sp* **model** Φ) is a persistent extension of $\mathcal{M}(sp)$ if and only if $\mathcal{M}(sp) \models \Phi$.

(For the notions of "μ-extension" and "extension" see Definition 4.13.)

Solution

1. By definition 9.4, for $\mu : \Sigma \to \Sigma'$

$$\begin{aligned}&\mathcal{M}(\textbf{rename } sp \textbf{ by } \mu)\\ &\quad = \{A \in Alg(\Sigma') | (A \,|\, \mu) \in \mathcal{M}(sp)\}\end{aligned}$$

   By definition 4.13, the persistent μ-extension of $\mathcal{M}(sp)$ is a class of algebras

$$\{A \in \mathsf{Alg}(\Sigma') | (A \,|\, \mu) \in \mathcal{M}(sp)\}$$

   The definitions are clearly identical.

2. By definition 9.4, for $\mu : \Sigma \to \Sigma' = \Sigma \cup (S, \Omega)$,

$$\begin{aligned}&\mathcal{M}(\textbf{extend } sp \textbf{ by } (\mathcal{S}, \Omega))\\ &\quad = \{A \in \mathsf{Alg}(\Sigma') | (A \,|\, \Sigma) \in \mathcal{M}(sp)\}\end{aligned}$$

   By definition 4.13, the persistent extension of $\mathcal{M}(sp)$ is a class of algebras

$$\{A \in \mathsf{Alg}(\Sigma') | (A \,|\, \Sigma) \in \mathcal{M}(sp)\}$$

   The definitions are clearly identical.

3. Since the two classes of algebras are over the same signature, we need to prove that $\mathcal{M}(sp) = \mathcal{M}(sp \textbf{ model } \Phi)$. This, by Definition 9.4, is equivalent to proving that

$$\mathcal{M}(sp) \cap Mod(\Phi) = \mathcal{M}(sp)$$

   This equation is clearly true if $Mod(\Phi) \supseteq \mathcal{M}(sp)$. By Fact 5.32, $\mathcal{M}(sp) \models \Phi$ iff $\forall A \in \mathcal{M}(sp).A \models \Phi$. Now, $A \models \Phi$ implies that $A \in Mod(\Phi)$. Consequently, $\mathcal{M}(sp) \models \Phi$ implies that any $A \in \mathcal{M}(sp)$ is also in $Mod(\Phi$, and so $\mathcal{M}(sp) \subseteq Mod(\Phi$.

   Conversely, we need to prove that $\mathcal{M}(sp) = \mathcal{M}(sp) \cap Mod(\Phi) \models \Phi$. Assume that this is not the case, that $\exists \varphi \in \Phi$ such that $\mathcal{M}(sp) \not\models \varphi$. But we know that $\mathcal{M}(sp) \subseteq Mod(\Phi)$, which, by Theorem 5.40 implies that $\Phi^* \subseteq Th(\mathcal{M}(sp))$. But then $\forall \varphi \in \Phi.\varphi \in Th(\mathcal{M}(sp))$, and hence, by definition 5.36, $\mathcal{M}(sp) \models \varphi$.

Exercise 8

Problem

Show that using **derive** (notes) one can define **rename** and **forget**.

Solution

- **forget** as **derive**:
  We define $\mu : \Sigma \to \mathcal{S}(sp_1)$ by $\mu(\Sigma) = \Sigma \cup (S, \Omega)$. We define $sp =$ **derive from** sp_1 **by** μ. The signature is Σ. The model class is defined as

  $$\mathcal{M}(sp) = \{(A \,|\, \Sigma) : A \in \mathcal{M}(sp_1)\}$$

  If we define $sp = sp_1$ **forget** (S, Ω), we also have the signature Σ, and a model class that is defined by

  $$\mathcal{M}(sp) = \{(A \,|\, \Sigma) | A \in \mathcal{M}(sp_1)\}$$

- **rename** as **derive**:
  We define a signature morphism μ' by $\mu'(\Sigma') = \mathcal{S}(sp_1))$, i.e. the inverse of $\mu : \mathcal{S}(sp_1) \to \Sigma'$. Since μ is a simple renaming, and thus is bijective, this definition causes no problems.

  We define $sp =$ **derive from** sp_1 **by** μ'. The signature is Σ'. The model class is defined as

  $$\mathcal{M}(sp) = \{(A \,|\, \Sigma') : A \in \mathcal{M}(sp_1)\}$$

  Then we take $sp =$ **rename** sp_1 **by** μ. The signature is Σ'. The model class is defined as

  $$A \in \mathsf{Alg}(\Sigma') : (A \,|\, \mathcal{S}(sp_1)) \in \mathcal{M}(sp_1)\}$$

  The two definitions are clearly identical.

Exercise 9

Problem

Show the claim after equation (10.1) from the notes, i.e. that any ADT specified using **union** can be expressed without it (using other specification building operations).

Be careful with possible name clashes when applying the induction! The first step, for **rename** will look as follows:

$sp = sp_1 \oplus \mu(sp_2)$: by IH, $\forall sp_1 : sp_1 \oplus sp_2 = sp'$ where sp' does not use any **union**. The symbols in the signature Σ of sp can be of three kinds: either from

1. $Sig(sp_1) \cap \mu(Sig(sp_2))$ or from
2. $Sig(sp_1) \backslash \mu(Sig(sp_2))$, or from
3. $\mu(Sig(sp_2)) \backslash Sig(sp_1)$.

Rename sp_1 by ν so that names in 1) coincide with the respective names in sp_2 (i.e., $\mu^-(n)$), while names in 2) are different from all sp_2 names. By IH, $sp_n = \nu(sp_1) \oplus sp_2$ can be expressed without **union**. Apply now a new renaming π to this resulting sp_n, so that for all names from 1) and 3) $\pi(n) = \mu(n)$, while for those from 2) $\pi(n) = \nu^-(n)$. Show that $\pi(sp_n) = sp$.

Solution