Implementation of Bourbaki's Elements of Mathematics in Coq: Part Three Structures

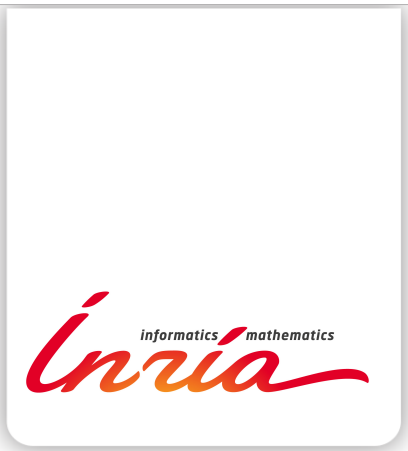
José Grimm

Implementation of Bourbaki's Elements of Mathematics in Coq: Part Three Structures

José Grimm

RESEARCH REPORT
N° 8997
December 2016
Project-Team Marelle

ISSN 0249-6399 ISRN INRIA/RR--8997--FR+ENG



Implementation of Bourbaki's Elements of Mathematics in Coq: Part Three Structures

José Grimm*

Project-Team Marelle

Research Report n° 8997 — December 2016 — 115 pages

Abstract: This document is a follow-up to two research reports explaining the implementation in the Coq proof assistant of the Theory of Sets of Bourbaki. It explains Book I, Chapter III, Section 7 (Inverse limits and direct limits) and the start of Book I, Chapter IV (structures). The code is available on the Web, under http://www-sop.inria.fr/marelle/gaia.

Key-words: Gaia, Coq, Bourbaki, structure, morphism, isomorphisms

* Email: Jose.Grimm@inria.fr

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Implémentation de la théorie des ensembles de Bourbaki dans Coq
partie 3
Structures

Résumé : Ce document est la suite de deux rapports de recherche expliquant l'implémentation dans l'Assistant de Preuve Coq de la Théorie des Ensembles de Bourbaki. Il décrit le livre I, chapitre III, section 7 (limites inductives et projectives) et le débur du Livre I, chapitre IV (structues). Le code est disponible sur le site Web http://www-sop.inria.fr/marelle/gaia.

Mots-clés : Gaia, Coq, Bourbaki, structure, morphisme, isomorphisme

Chapter 1

Introduction

In this document, we explain the implementation in COQ of Chapter IV of the Theory of Sets, entitled "Structures" as described in [2]. This Chapter (minus the appendix) is included in [4], translated in English in [3]. The implementation of Chapter II and Chapter III is described in [7] and [8] respectively. Section 7 of Chapter III ("Inverse limits and direct limits") forms the last chapter of this document, see page 43. Chapter I ("Description of Formal Mathematics") is discussed in [7].

Chapter IV starts with « The purpose of this chapter is to describe once and for all a certain number of formative constructions and proofs (cf. Chapter I, §1, no. 3 and §2, no. 2) which arise very frequently in mathematics. »

In fact, the idea is to study mathematical structures (ordered sets, groups, topologic spaces etc), functions between structures (morphisms, isomorphisms), and deductions of structures (induced structures, products, quotients, tensor products, etc). However, the exposition is abstract, there is no link to other books, and the ideas presented here are not used elsewhere (neither by Bourbaki, nor by anybody else; people prefer to use a "theory of categories", that has the same purpose). There are some small examples, but the notion of "group" is already too complicated to be given in detail; the way groups are defined in the Book of Algebra [5] gives no indication of how a group structure could be defined (see details below).

Chapter IV contains no theorem at all, but some criteria CST1 through CST23. An example of a criterion is: « C14. *Let* $\boldsymbol{A}$ *be a relation in* $\mathscr{T}$ *and let* $\mathscr{T}'$ *be the theory obtained by adjoining* **A** *to the axioms of* $\mathscr{T}$. *If* $\boldsymbol{B}$ *is a theorem in* $\mathscr{T}'$ *then* $\boldsymbol{A} \Longrightarrow \boldsymbol{B}$ *is a theorem in* $\mathscr{T}$. » Some of the criteria of the chapter have a similar form: if, in the theory obtained by adding to $\mathscr{T}$ some axioms, some relation holds, then some other relation holds in $\mathscr{T}$. Chapter I, §1, no. 3 explains (among other things) what is a valid object in the theory; in particular, there are terms (aka sets) and relations; a letter is a term. The object $(\forall \boldsymbol{x})\boldsymbol{R}$ is valid and a relation whenever $\boldsymbol{x}$ is replaced by a letter and $\boldsymbol{R}$ by a relation. In C14 $\boldsymbol{A}$ is not considered as a letter but as a name. Recall that a theorem is a statement with a proof (in the sense of Chapter I, §2, no. 2). For instance $x = x$ is the first theorem proved by Bourbaki. It follows that $(\forall x)(x = x)$ is a theorem.

Bourbaki uses a first order logic: quantification is only over sets. The expression $(\forall \boldsymbol{A})(\boldsymbol{A} \Longrightarrow \boldsymbol{A})$ it not a formative construction, whatever $\boldsymbol{A}$ (after the quantifier, there must be a letter, which is a set, and cannot be on the LHS of the implication sign). Hence, Bourbaki needs some criteria. For instance, by CF5, if $\boldsymbol{A}$ is a relation, so is $\boldsymbol{A} \Longrightarrow \boldsymbol{A}$, and C8 says: « If $\boldsymbol{A}$ is a relation in $\mathscr{T}$, then $\boldsymbol{A} \Longrightarrow \boldsymbol{A}$ is a theorem in $\mathscr{T}$. » In our implementation of Bourbaki, we allow

quantification over of propositions; this make CF5 unnecessary (the COQ type-checker does the job), and C8 becomes a theorem.

An expresion (term or relation) may have free variables that can be replaced by terms; this does not change the type of the expression. Note that $(\forall x)(x = x)$ has no free variables. One can say: let x be an integer, assume x non-zero, etc. In such a case x is considered a constant of the theory (as long as it is in the scope of the let), and not a free variable. One may replace, in a theorem, any free variable by a term: the result is a theorem. For instance, from $x = x$, on deduces $\emptyset = \emptyset$.

For us, a theorem will be a statement without free variables with a proof in COQ; we consider only one theory. The section mechanism is one way to deal with this problem.

```
1  Section a.
2    Variable x:nat.
3    Section b.
4       Hypothesis A: odd x.
5       Lemma B: odd (x.+2).  by rewrite /= A. Qed.
6       Check B.
7    End b.
8  Check B.
9  End a.
10 Check B.
```

On line 5, we are in a theory with a variable x and an axiom A. We then prove B. The first `Check` prints `odd x.+2`. On line 8, we are in a theory with a variable x and no axiom. The second `Check` prints `odd x -> odd x.+2`. Two lines later, we are in the initial theory, and the last `Check` prints `forall x : nat, odd x -> odd x.+2`.

Chapter IV considers a theory $\mathscr{T}$ stronger than the Theory of Sets (the theory itself, or some extension of it). Whenever x and y are two sets, then $x \in y$ is a relation, as well as $x \subset y$. The relation $x = y$ is equivalent to $x \subset y$ and $y \subset x$. One can construct $\{x, y\}$, $x \cup y$, (x, y), $x \times y$, $\mathfrak{P}(x)$, x^y, $\mathscr{F}(x; y)$, etc. Recall that a function f is a triple (a, b, c) where a is the source, b the target and c the graph; it belongs to the set of functions $\mathscr{F}(a; b)$. The function $f \mapsto c$ is the canonical isomorphism $\mathscr{F}(a; b) \to b^a$; the target is the set of all functional graphs with domain a, the range being a subset of b. Bourbaki writes somewhere « we shall often use the word "function" in place of "functional graph". » This may be confusing because a function has a target and a functional graph has none. For this reason, we shall signal these abuses of language. For Bourbaki, a "mapping of A into B" is a function whose source is equal to A and whose target is equal to B. The subtle difference is that "mapping" is never used alone (there are however some exceptions, so that we shall use "mapping" as a synonym of "function"). For Bourbaki, $x \mapsto x + 2 (x \in \mathbf{N}, x + 2 \in \mathbf{N})$ is a function (source and target being $\mathbf{N}$). A more common notation is $f : \mathbf{N} \to \mathbf{N}, x \mapsto x + 2$. The source and target may be indicated in a different way, or perhaps omitted. For us, the COQ object '`fun x => x.+2`' will be a function (it is not a Bourbaki object, as its type is `nat→nat`). If $x + 2$ denotes the double ordinal successor, then $x \mapsto x+2$ will be considered a function; this is not a Bourbaki function in that it has no source and no target (there is no set containing all ordinals); this is in fact not a Bourbaki object at all. Three strategies are used: (a) the object is considered as a term with some free variables, x, y, etc, denoted by T⁞x⁞ or T⁞x, y⁞, etc, instantiation is denoted by T⁞1⁞ or T⁞2,3⁞, etc. In such a notation, there may be other free variables; (b) there a notation, $\mathfrak{P}(x)$, $x + y$, $x \top y$, x^*, etc. The notation may be overloaded (as in $x + y$) or generic (in what follows $x \top y$ is the generic law of a group); (c) the object is named by a letter (or a sequence

of letters), the value being an index. For instance, Bourbaki says: « Put $\aleph_\alpha = \mathrm{Card}(\omega_\alpha)$; $\aleph_\alpha$ is said to be the *aleph of index* α. » Here ω is not the name of the function, so that $\omega = \omega_0$ makes sense.

In Chapter IV there is a footnote that says « We use the notion of integer in the same manner as in Chapter I, that is to say, in the metamathematical sense of marks arranged in a certain order; this use has nothing to do with the mathematical theory of integers which was developed in Chapter III. » There is a section that explains when $\mathrm{R}\{x_1,\dots,x_n,s_1,\dots,s_p\}$ is a transportable relation for the typification $\mathrm{T}\{x_1,\dots,x_n,s_1,\dots,s_p\}$. In the special case $n=2$, $p=1$, this explains when $\mathrm{R}\{x_1,x_2,s_1\}$ is a transportable relation for the typification $\mathrm{T}\{x_1,x_2,s_1\}$. There is a definition for "echelon construction scheme S on n terms", and of $\langle x_1,\dots,x_n\rangle^{\mathrm{S}}$. The first criterion is then

CST1. *If f_i is a mapping of E_i into E'_i, and if f'_i is a mapping of E'_i into E''_i $(1 \le i \le n)$, then for every echelon construction scheme* S *on n terms we have*

$$\langle f'_1 \circ f_1, f'_2 \circ f_2, \dots, f'_n \circ f_n\rangle^{\mathrm{S}} = \langle f'_1 f'_2, \dots, f'_n\rangle^{\mathrm{S}} \circ \langle f_1, f_2, \dots, f_n\rangle^{\mathrm{S}}.$$

The objective is to convert this criterion into a theorem. Take $n=2$. The criterion becomes: If f_1 is a mapping of E_1 into E'_1, if f_2 is a mapping of E_2 into E'_2, if f'_1 is a mapping of E'_1 into E''_1, if f'_2 is a mapping of E'_2 into E''_2, then for every echelon construction scheme S on 2 terms we have $\langle f'_1 \circ f_1, f'_2 \circ f_2\rangle^{\mathrm{S}} = \langle f'_1, f'_2\rangle^{\mathrm{S}} \circ \langle f_1, f_2\rangle^{\mathrm{S}}$. This could be converted into a theorem, if only we could quantify over S (this is impossible, as S is not in the theory $\mathscr{T}$). Moreover, to say that S is a scheme on 2 terms cannot be expressed in $\mathscr{T}$, and the expression $\langle f_1, f_2\rangle^{\mathrm{S}}$ cannot be constructed in $\mathscr{T}$. Take for S the sequence $(0,1),(0,2),(1,0),(3,0),(2,0),(4,5)$. In this example $\langle f_1, f_2\rangle^{\mathrm{S}}$ can be effectively constructed. Moreover, there are sets A_6 and A'_6 such that, if f_1 and f_2 are as above, then $\langle f_1, f_2\rangle^{\mathrm{S}}$ is a mapping of A_6 into A'_6. For this particular S, the criterion becomes a theorem by quantifying over the ten sets.

The amusing point is that, since $\mathscr{T}$ is stronger than the theory of sets, it is possible to use its integers, in the sense indicated above. It is possible to consider S as object in this theory. Now, if $n=2$, the criterion becomes a theorem (proof by induction on the length of S), where $\langle f_1, f_2\rangle^{\mathrm{S}}$ is defined by induction on the length of S. Finally, we consider lists of objects in $\mathscr{T}$, and the criterion becomes: whatever the integer n, whatever $\mathbf{f}$, $\mathbf{f}'$, $\mathbf{E}$, $\mathbf{E}'$, $\mathbf{E}''$ lists of length n, satisfying some property, if $\mathbf{f}' \circ \mathbf{f}$ has the obvious meaning, then for every echelon construction scheme S on n terms we have $\langle \mathbf{f}' \circ \mathbf{f}\rangle^{\mathrm{S}} = \langle \mathbf{f}'\rangle^{\mathrm{S}} \circ \langle \mathbf{f}\rangle^{\mathrm{S}}$. This is a statement in $\mathscr{T}$ with no free variables that happens to be a theorem (bold face letters are sometimes used in this sense in the Appendix of the Chapter).

The first example of a structure given by Bourbaki is: « take $\mathscr{T}$ to be the theory of sets, and consider the species of structures which has no auxiliary base set, one principal base set A, the typical characterization $s \in \mathfrak{P}(\mathrm{A}\times\mathrm{A})$, and the axiom $s\circ s = s$ and $s \cap s^{-1} = \Delta_{\mathrm{A}}$ (Δ_{A} being the diagonal of $\mathrm{A}\times\mathrm{A}$), which is a transportable relation with respect to the typification $s \in \mathfrak{P}(\mathrm{A}\times\mathrm{A})$, as is easily verified. It is clear that this species of structures is just the *theory of ordered sets* (Chapter III, §1, no. 3). »

Let's try to understand the example. In the case of a vector space E over a field K, there is a significative distinction between E and K; the former is called the principal base set, the latter is called the auxiliary base set. Here we have only one set, the principal base set. The expression $s \in \mathfrak{P}(\mathrm{A}\times\mathrm{A})$ says that s is a set of pairs (x,y) with $x \in \mathrm{A}$ and $y \in \mathrm{A}$. The notation $s\circ s$ denotes the composition of graphs (a variant of the composition of functions introduced above), and s^{-1} is the inverse graph. The relation $s \cap s^{-1} = \Delta_{\mathrm{A}}$ is equivalent to: if $x \in \mathrm{A}$ then $(x,x) \in s$ and if $(x,y) \in s$ and $(y,x) \in s$ then $x = y$. So $s\circ s = s$ and $s \cap s^{-1} = \Delta_{\mathrm{A}}$ means that the relation $(x,y) \in s$ is an order relation. We shall see later what it means for the relation to be

transportable and why this is important.

In the appendix of [2] (not translated into English), one can find the example of a group; it will be discussed in details below. The typical characterization has the form $s \in \mathfrak{P}((\mathrm{E}\times\mathrm{E})\times\mathrm{E})$, and the axiom is a bit complicated. Let's admit the Axiom of Unique Choice (AUC) in the form: if $\mathrm{P}(x,y)$ is a relation that depends on x (maybe on y and some other variables), such that (under some assumptions on y and the other variables) there is a unique x satisfying P, then we can consider "the" x that satisfies P. For instance, if P does not depend on y we may denote it by x_0, otherwise, we may denote it by $x_{\mathrm{P}}(y)$, or some other notations. Example: if P denotes "x and y are integers and $y = 2x$," then the x will be denoted by $y/2$. This makes sense when y is an even integer. Bourbaki uses $\tau_x(\mathrm{P})$ for its Axiom of Choice, this is defined whatever P. How should we interpret $y/2$ when y is not an even integer? in a typed theory like COQ, if $y \mapsto y/2$ is of type `nat→nat`, then the expression becomes ill-typed when y is not an integer. Here typification comes into play: y of type `nat` is replaced by the typification $y \in \mathbf{N}$. In COQ, you can use dependent types; this means that $y/2$ can be a function of two arguments, an integer and a proof that it is even. In some cases this is the right choice. However, using such a mechanism for defining the multiplicative inverse of an element of the field (that exists only for non-zero elements) is much too complicated; using an option type (i.e., saying that $y/2$ is in the disjoint union of $\mathbf{N}$ and some singleton) is sometimes possible, sometimes complicated. In SSREFLECT the half of y is defined by $(y-1)/2$ when y is odd. The specification of a unit ring says: if $x\cdot y = y\cdot x = 1$, then x is a unit; if x is a unit then $x\cdot x^{-1} = x^{-1}\cdot x = 1$. The specification of a field says that every non-zero element is a unit, and $0^{-1} = 0$. One could define $y/2$ to be some default value x_0; however, this should meet the typification (if the typification is $x \in \mathrm{E}$, this requires E to be non-empty; in SSREFLECT, a ring has at least two elements since $0 \neq 1$ is assumed. Another solution would be to define $y/2$ to be y (this works whenever y and $y/2$ have the same typification).

The axiom of the group structure implies that s is a functional graph. Hence there exists an operation $a\top b$, defined whenever a and b are in E such that s is the set of all $((a,b),a\top b)$ for a and b in E (note that s is uniquely defined by $\top$ and E). Moreover the law is associative; there is a unit, every element has an inverse. Using AUC, one may denote the unit by e, the inverse by x^*, and prove that the axiom is equivalent to the usual axiom of a group.

What is a group? the short answer is E and s, the long answer is E, $x\top y$, e, x^*. The second item is the function $x,y \mapsto x\top y$; it could be replaced by $\top$, when it is clear that $\top$ is a binary operator; it could be replaced by top, if $x\top y$ is a notation for $\mathrm{top}(x,y)$. The last item is the function $x \mapsto x^*$; in this case omitting the variable is rare. Note that $\top$ is not an object of the theory of sets of Bourbaki, and not uniquely defined by the group (only $a\top b$ for a and b in E is uniquely defined). So a group could be defined by E, $\top_{\mathrm{E}}$, e, $*_{\mathrm{E}}$ (introducing the graphs of the functions; these are objects of the theory). Alternatively, it could be E and a triple $(\top_{\mathrm{E}}, e, *_{\mathrm{E}})$. In the case of a group $\top_{\mathrm{E}}$ is exactly s. The question is now: how to express relations of the form $x\top x^* = e$, given that the group contains $\top_{\mathrm{E}}$ and $*_{\mathrm{E}}$. One solution would be to split the axiom into a conjunction $\mathrm{A}\wedge\mathrm{B}$, where A implies that $\top_{\mathrm{E}}$, $*_{\mathrm{E}}$, etc, are functional graphs, defined on $\mathrm{E}\times\mathrm{E}$, E, etc. It justifies the use of $\top$ and * in the B-part of the axiom. In practice, people say: a group is a set E, with a binary operation $x\top y$ and a unary operation x^* satisfying B.

Consider now the definition of a group ([5, page 30]): *A set with an associative law of composition, possessing an identity element, and under which every element is invertible, is called a group.* The definition is interesting in that nothing is named. This definition refers to another one, page 15: « let E be a unital magma, $\top$ its law of composition, e its identity element and x and x' two elements of E. x' is called an inverse of x if $x'\top x = x\top x' = e$.

An element x of E is called invertible if it has an inverse. A monoid all of whose elements are invertible is called a group. » The definition explains also what is a left inverse, a right inverse, and is followed by a remark explaining that the inverse is unique.

We once tried to implement algebraic structures in COQ, but it was complicated and unusable (see previous discussion on $\top$ and $\top_E$). However, $\top$ is a COQ object, especially if we replace "if a and b belong to E, then $a \top b$ belongs to E" by "if a and b are of type E, then $a \top b$ is of type E". Moreover $\forall x, x \top x^* = e$ is also a COQ object (there is no need to assume $x \in E$, this simplifies the axiom). Finally, one can pack, in the same object, the functions and the axiom. Here is an example.

```
Record mixin_of (V : Type) : Type := Mixin {
  zero : V;
  opp : V -> V;
  add : V -> V -> V;
  _ : associative add;
  _ : commutative add;
  _ : left_id zero add;
  _ : left_inverse zero opp add
}.
```

As you can see, the definition is straightforward. There is a complicated mechanism, described in [6], that explains how to implement concrete commutative groups (by the way, the previous structure is called Zmodule in the file *ssralg* of SSREFLECT). A more complicated object is the following.

```
Record mixin_of (T : Type) : Type := BaseMixin {
  mul : T -> T -> T;
  one : T;
  inv : T -> T;
  _ : associative mul;
  _ : left_id one mul;
  _ : involutive inv;
  _ : {morph inv : x y / mul x y >-> mul y x}
}.

Structure base_type : Type := PackBase {
  sort : Type;
   _ : mixin_of sort;
   _ : Finite.class_of sort
}.

Structure type : Type := Pack {
  base : base_type;
  _ : left_inverse (one (mixin base)) (inv (mixin base)) (mul (mixin base))
}.
```

This is the definition of a finite group in SSREFLECT. It is preceded by the following comment

We split the group axiomatisation in two. We define a class of "base groups", which are basically monoids with an involutive antimorphism, from which we derive the class of groups proper. This allows use to reuse much of the group notation and algebraic axioms for group subsets, by defining a base group class on them. We use class/mixins here rather than telescopes to be able to interoperate with the type coercions. Another potential benefit (not exploited here) would be to define a class for infinite groups, which could share all of the algebraic laws.

An important part of the definition of a group, ring, etc, is the interface: the internal structure is completely hidden, and there are many useful definitions. A typical example is the order of an element x.

```
1 Definition group_set A := (1 \in A) && (A * A \subset A).
2 Notation Local groupT := ...
3 Definition generated A := \bigcap_(G : groupT | A \subset G) G.
4 Definition cycle x := generated [set x].
5 Definition order x := #|cycle x|.
6 Lemma invg_expg x : x^-1 = x ^+ #[x].-1.
```

By lines 4 and 5, the order of x is the cardinal of the set generated by $\{x\}$; this is a natural number as T is a finite set. Line 3 says that the subgroup generated by A is the intersection of all subgroups G of T, such $A \subset G$; that this is a subgroup follows by induction on the number of elements in the intersection (which is finite, because $\mathfrak{P}(T)$ is finite). Line 2 is part of many lines of the code that helps Coq to expand correctly all notations; it defines `groupT` as the type of all subgroups, the type of all A (elements of $\mathfrak{P}(T)$) satisfying the property of line 1. If A and B are in $\mathfrak{P}(T)$ and $A \star B$ is the set of all $x * y$ for $x \in A$ and $b \in B$, then $\star$ makes $\mathfrak{P}(T)$ a group. So, on line 1, Coq sees to groups, T and $\mathfrak{P}(T)$ and correctly interprets the unit and the law. The lemma says that the inverse of x is x^n, where n is one less than the order of x, and $x^n = x * \ldots * x$.

1.1 Additional code

We consider here an implementation of a list $x_1, x_2, \ldots, x_n$. The first idea would be a function graph with domain $[1, n]$. It is sometimes more interesting to start indices with zero. So $f(0) = x_1$, $f(1) = x_2$, etc. This means $x_i = f(i-1)$. The domain of f becomes now I_n, the set of integers $< n$. Our implementation relies on the fact that $I_n = n$. This simplifies the definition from: a functional graph for which there exists an integer n such that the domain is I_n to: a functional graph whose domain is an integer. The length of the list is the number of elements of the domain, the cardinal of I_n, hence n; this is another simplification.

```
Definition slistp f := fgraph f /\ natp (domain f).
Definition slength f := domain f.
Definition slistpl f n := slistp f /\ slength f = n.
Definition slist_E f E := slistp f /\ sub (range f) E.
Definition Vl l x := Vg l (cpred x).
```

We state here some properties. If i is a cardinal, then $f(i) = x_{i+1}$ (the condition holds in particular if $i < n$). If the list has range in E, if $1 \le i \le n$, then $x_i \in E$. If f and g are the functions associated to the lists $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ of the same length n, if $x_i = y_i$ whenever $1 \le i \le n$, then $f = g$.

```
Lemma slist_domain X: slistp X -> domain X = Nint (slength X).
Lemma slength_nat X:  slistp X -> natp (slength X).
Lemma slist_domainP  X: slistp X ->
  forall i, (inc i (domain X) <-> i <c (slength X)).
Lemma Vl_correct l i: cardinalp i ->  Vg l i = Vl l (csucc i).
Lemma slist_extent f g :
  slistp f -> slistp g -> slength f = slength g ->
```

```
  (forall i, \1c <=c i -> i <=c slength f ->
      (Vl f i = Vl g i))
  -> f =g .
Lemma Vl_p1 f E x : slist_E f E -> \1c <=c x -> x <=c (slength f) ->
  inc (Vl f x) E.
```

We consider now a technique called "induction on a stratified collection". The collection is defined by a property W and a rank ρ. It is not necessary that there exists a set containing all objects satisfying W, but it is assumed that if $W(x)$ holds then $\rho(x)$ is an ordinal, and that, for every ordinal α there is a set W_α containing those x such that $W(x)$ holds and $\rho(x) < \alpha$. This implies that there is a set W'_α such containing those x such that $W(x)$ holds, and $\rho(x) = \alpha$. Let $H(x, f)$ be a functional term. One can define a unique functional term f such that $f(x) = H(x, f_x)$, whenever x satisfies W, where f_x is the (unique surjective) function defined on $W_{\rho(x)}$ such that $f_x(a) = f(a)$, whatever a. The idea is to define by transfinite induction a function f_α on W'_α that depends on the f_β for $\beta < \alpha$. We recall here the assumptions and the main result.

```
Hypothesis OS_idx: forall x, W x -> ordinalp (idx x).
Hypothesis Wi_coll: forall i, ordinalp i ->
     exists E, forall x,  inc x E <-> (W x /\ idx x <o i).

Lemma stratified_fct_pr x  (f := stratified_fct):
  W x -> f x =  H x (Lg (stratified_set (idx x)) f).
```

Recall that '`functions X Y`' is the set of all functions $X \to Y$, and '`bijections X Y`' is the set of all bijections $X \to Y$. If X is a set, f a function, then $f\langle X\rangle$ is the set of all $f(x)$ for $x \in X$. This induces a function $\mathfrak{P}(X) \to \mathfrak{P}(Y)$ called "canonical extension of f to sets of subsets", denoted here by `\Pof` (analogous to `\Po`, the powerset), or $\mathfrak{P}(f)$. The properties we shall use here are the following: if f is a bijection so is $\mathfrak{P}(f)$ and $\mathfrak{P}(f^{-1}) = \mathfrak{P}(f)^{-1}$; $\mathfrak{P}(f \circ g) = \mathfrak{P}(f) \circ \mathfrak{P}(g)$, $\mathfrak{P}(I_E) = I_{\mathfrak{P}(E)}$ if I_E is the identity function of E.

Given a family of functions $f_i : E_i \to F_i$ one can consider the product, the function $\prod E_i \to \prod F_i$ that maps a sequence x_i to a sequence y_i with $y_i = f_i(x_i)$. In what follows, we consider two functions $f : E \to F$, $g : E' \to F'$, the product will be denoted by `f \ftimes g`. We have $(f \times g)(x, y) = (f(x), f(y))$. If both functions are bijections, so is the product and $(f \times g)^{-1} = f^{-1} \times g^{-1}$, $(f \times g) \circ (f' \times g') = (f \circ f') \times (g \circ g')$; $I_E \times I_F = I_{E\times F}$.

We need two additional lemmas: if $1 \le i \le k$, and k is an integer, then $i - 1$ is an integer, $i = (i-1)+1$ and $i - 1 < k$. Our induction principle (version 6) says if $p(0)$ holds, if $p(n+1)$ is true whenever p holds for all integers $\le n$, then p holds everywhere. We replace it by: if $p(n)$ is true whenever p holds for all integers $< n$, then p holds everywhere.

```
Lemma Nat_induction6' (P : property):
  (forall n, natp n -> (forall k, k <c n -> P k) -> P n) ->
  forall n, natp n -> P n.

Lemma cpred_pr6' k i:  natp k -> \1c <=c i -> i <=c k ->
    [/\ natp (cpred i), i = csucc (cpred i) & cpred i <c k ].
```

1.2 Example: the structure of a group

The definition of a group structure in the appendix of [2] uses an unusual form of associativity. For this reason, we start with some preliminaries.

1.2.1 Preparation

Recall that a graph s is a set of pairs, so is a subset of a cartesian product A × B. If A and B are the smallest possible, then A is called the domain and B the range (the domain is the set of all a such that for some b, $(a,b) \in s$, the range is defined similarly). The graph is functional if b is unique. This induces a mapping A → B. We say that a law s on E is a functional graph, with domain E × E, and whose range is a subset of E. The induced mapping will be denoted $a \top b$. Recall that '`J a b`' is the pair (a,b), '`P x`' and '`Q x`' are the first and second component of the pair x, denoted $\mathrm{pr}_1 x$ and $\mathrm{pr}_2 x$, that '`Vg g x`' and '`Vf f x`' are the value of x by a functional graph g or a function f (see [7] and [8] for more notations and definitions).

```
Definition Law s E := [/\ sub s ((E \times E) \times E),
                        fgraph s & domain s = (E \times E)].
Definition VL s a b := Vg s (J a b).

Lemma Law_in s E a b: Law s E -> inc a E -> inc b E ->
  inc (J (J a b) (VL s a b)) s.
Lemma Law_range s E a b: Law s E -> inc a E -> inc b E ->
   inc (VL s a b) E.
Lemma Law_val s E a b c: Law s E -> inc (J (J a b) c) s ->
   c = (VL s a b).
```

We define here the identity on E and the canonical mapping (E × E) × E → E × (E × E). These objects are functional graphs, with a domain, target, evaluation function. Note that only the reflections lemmas are used in what follows.

```
Definition GE_I E := Zo (E\times E) (fun z => P z = Q z).
Definition GE_J E :=
  Zo (((E\times E) \times E)\times (E\times (E\times E)))
     (fun x => [/\ P (P (P x)) = P (Q x),
                   Q (P (P x)) = P (Q (Q x)) &
                   Q (P x) = Q (Q (Q x))]).

Lemma GE_I_incP E x: inc x (GE_I E) <-> [/\ pairp x, P x = Q x & inc (P x) E].
Lemma GE_I_fgraph E : fgraph (GE_I E).
Lemma GE_I_domain E : domain (GE_I E) = E.
Lemma GE_I_range E : range (GE_I E) = E.
Lemma GE_I_Ev E x: inc x E -> Vg (GE_I E) x = x.

Lemma GE_J_P E x: inc x  (GE_J E) <-> exists a b c,
  [/\ inc a E, inc b E, inc c E &  x = (J (J (J a b) c) (J a (J b c)))].
Lemma GE_J_fgraph E : fgraph (GE_J E).
Lemma GE_J_domain E : domain (GE_J E) = (E\times E) \times E.
Lemma GE_J_range E : range (GE_J E) = (E\times (E\times E)).
Lemma GE_J_Ev E a b c: inc a E -> inc b E -> inc c E ->
  Vg (GE_J E) (J (J a b) c) =  (J a (J b c)).
```

We then define a complicated operation, denoted A ⊗ B. The interesting case is when A or B is the identity of E.

```
Definition Sprod A B :=
  Zo (((domain A) \times (domain B)) \times (range A \times (range B)))
     (fun z => inc (J (P (P z)) (P (Q z))) A /\ inc (J (Q (P z)) (Q (Q z))) B).
```

```
Lemma Sprod_P A B x: inc x (Sprod A B) <-> exists a1 b1 a2 b2,
  [/\ inc (J a1 a2) A, inc (J b1 b2) B & x = J (J a1 b1) (J a2 b2)].

Lemma Sprod_Il E B x: inc x (Sprod (GE_I E) B) <->
  [/\ pairp x, pairp (P x), pairp (Q x), P (P x) = P (Q x)
      & inc (P (P x)) E /\ inc (J (Q (P x)) (Q (Q x))) B].
Lemma Sprod_Ir A E x: inc x (Sprod A (GE_I E)) <->
  [/\ pairp x, pairp (P x), pairp (Q x), Q (P x) = Q (Q x)
      & inc (Q (P x)) E /\ inc (J (P (P x)) (P (Q x))) A].
Lemma Sprod_fgraph A B: fgraph A -> fgraph B -> fgraph (Sprod A B).
Lemma Sprod_domain A B: sgraph A -> sgraph B ->
   (domain (Sprod A B)) = ((domain A) \times (domain B)).
Lemma Sprod_range A B: sgraph A -> sgraph B ->
   (range (Sprod A B)) = ((range A) \times (range B)).
```

Recall that B∘A is the set of all (a, b) such that there is c such that $(a, c) \in$ A and $(c, b) \in$ B. In the case where A is a functional graph, there is a function ν_A such that $(a, \nu_A(a)) \in$ A whenever a is in the domain of A. If A and B are composable (the domain of B is a subset of the domain of A), and if B is a functional graph, then $\nu_{B\circ A}(a) = \nu_B(\nu_A(a))$ whenever a is in the domain of A. This leads to an alternate definition of composition (note that there is a third definition, for functions). Composition is associative (this is always the case for the definition given here, is true under conditions for the two other definitions).

Let's consider $A = s \circ (s \otimes I_E)$. This makes sense when $s \subset ((E \times E) \times E))$. There is a simple formula for $x \in$ A, considering the complexity of the definition of $\otimes$. If s is a law, this is the set of all $(((a, b), c), (a \top b) \top c)$, for a, b, and c in E. Consider now $B = s \circ (I_E \otimes s)$. The result is similar. The formula $B \circ J_E$ is similar too.

```
Lemma Sprod_case1 s E x (f := s \cg (Sprod s (GE_I E))):
  sub s ((E \times E) \times E) ->
  (inc x f <-> exists a b c d t,
                [/\ x = J (J (J a b) c) d,
                 inc (J (J t c) d) s & inc (J (J a b) t) s]).
Lemma Sprod_case_l1 s E x (f := s \cg (Sprod s (GE_I E))):
  Law s E ->
  (inc x f <-> exists a b c, [/\ inc a E, inc b E, inc c E &
                x = J (J (J a b) c) (VL s (VL s a b) c) ]).
Lemma Sprod_case2 s E : Law s E ->
   s \cg (Sprod s (GE_I E)) = fun_image ((E\times E) \times E)
        (fun z=> J z  (VL s (VL s (P (P z)) (Q (P z))) (Q z))).

Lemma Sprod_case3 s E x (f := s \cg (Sprod (GE_I E) s)):
  sub s ((E \times E) \times E) ->
  (inc x f <-> exists a b c d t,
                [/\ x =  J (J a (J b c)) d,
                 inc (J (J a t) d) s & inc (J (J b c) t) s]).
Lemma Sprod_case4 s E x (f := (s \cg (Sprod (GE_I E) s)) \cg (GE_J E)):
  sub s ((E \times E) \times E) ->
  (inc x f <-> exists a b c d t,
                [/\ x = J (J (J a b) c) d,
                 inc (J (J a t) d) s & inc (J (J b c) t) s]).
Lemma Sprod_case_l2 s E x (f := (s \cg (Sprod (GE_I E) s)) \cg (GE_J E)):
  Law s E ->
  (inc x f <-> exists a b c, [/\ inc a E, inc b E, inc c E &
                x = J (J (J a b) c) (VL s a (VL s b c)) ]).
```

Associativity is now $s \circ (s \otimes \mathrm{I_E}) = s \circ (\mathrm{I_E} \otimes s) \circ \mathrm{J_E}$.

```
Lemma Bourbaki_assoc s E : Law s E ->
  ( (s \cg (Sprod s (GE_I E))) = ((s \cg (Sprod (GE_I E) s)) \cg (GE_J E))
  <->  forall a b c, inc a E -> inc b E -> inc c E ->
        VL s a (VL s b c) =  VL s (VL s a b) c).
```

1.2.2 The group axiom

We shall describe and comment here the group axiom. It has the form $\mathrm{R_1} \wedge \mathrm{R_2} \wedge \mathrm{R_3} \wedge \mathrm{R_4}$, and depends on two parameters E and s.

We assume moreover

$$s \in \mathfrak{P}(\mathrm{E} \times \mathrm{E} \times \mathrm{E}). \tag{1.1}$$

This is called the typification, it says that an element of s is a triple of elements of E. Axiom $\mathrm{R_1}$ is

$$\begin{aligned} &\forall a \in \mathrm{E}, \forall b \in \mathrm{E}, \exists c, (a,b,c) \in s, \\ &\forall x \in s, \forall y \in s, \mathrm{pr}_1 x = \mathrm{pr}_1 y \implies x = y. \end{aligned} \tag{1.2}$$

We say that $\top$ is a law of composition if $a \top b$ belongs to E, whenever a and b belong to E. We can consider $\top$ as a function $f : \mathrm{E} \times \mathrm{E} \to \mathrm{E}$, or a functional graph g. Note that f is a triple with source $\mathrm{E} \times \mathrm{E}$, target E, and graph g. Bourbaki often identifies f and g.

```
Section GroupExample.

Definition GT E s := inc s (\Po ((E\times E) \times E)).

Definition is_law E f :=  forall x y, inc x E -> inc y E -> inc (f x y) E.
Definition GL E s :=
  (forall a b, inc a E -> inc b E -> exists c, inc (J (J a b) c) s)
  /\ (forall a b, inc a s -> inc b s -> P a = P b -> a = b).

Definition Op E f := Lg (E\times E) (fun z => f (P z) (Q z)).
Definition Opfun E f := Lf (fun z => (f (P z) (Q z)))  (E \times E) E.

Lemma GEl_prop1 E f: is_law E f -> function_prop (Opfun E f) (E\times E) E.
Lemma GEl_prop2 E f: Op E f = graph (Opfun E f).
Lemma GEl_prop3 E f: is_law E f -> GT E (Op E f).
Lemma GEl_prop4 E f: is_law E f -> GL E (Op E f).
```

Bourbaki says: the relation (1.2) is transportable for the typification $s_1 \in \mathfrak{P}(x_1 \times x_1)$. This is obviously a mistake: it should be $s_1 \in \mathfrak{P}(x_1 \times x_1 \times x_1)$, with s_1 replaced by s and x_1 by E. In this case, transportable means: whenever F is a set, g a bijection $\mathrm{E} \to \mathrm{F}$, $\hat{g}$ its extension, $s' = \hat{g}(s)$, and $s \in \mathfrak{P}(\mathrm{E} \times \mathrm{E} \times \mathrm{E})$, then R is transportable if R(E, s) is equivalent to R(F, s').

For simplicity, we do not write down $\hat{g}$, just say that $s' = \hat{g}(s)$, is the set of all $(g(a), g(b), g(c))$ where $(a,b,c) \in s$.

```
Definition transport s g :=
  fun_image s (fun x => J (J (Vf g (P (P x))) (Vf g (Q (P x)))) (Vf g (Q x))).
```

```
Lemma transport_p1 E F s g: GT E s -> bijection_prop g E F ->
   GT F (transport s g).
Lemma transport_p2 E s: GT E s -> (transport s (identity E)) = s.
Lemma transport_p3 E F G s g h: GT E s ->
   bijection_prop g E F -> bijection_prop h F G->
   transport (transport s g) h = transport s (h \co g).
```

The typification is transportable. Relation (1.2) is transportable.

```
efinition transportable R:=
  forall E F s g, bijection_prop g E F -> GT E s ->
   (R E s <-> R F (transport s g)).

Lemma transportable_GT: transportable GT.
Lemma transportable_GL: transportable GL.
```

If s satisfies (1.1) and (1.2) then s is a law in the sense introduced above. If $\top$ is the operation associated to the law, then $\top$ is a law, and its graph is s. Consider a bijection $g : \mathrm{E} \to \mathrm{F}$. We know that s' is a law, hence induces an operation $\perp$ on F. We have $g(a)\perp g(b) = g(a\top b)$.

```
Lemma GE_prop1 E s a: GT E s -> inc a s ->
  [/\ pairp a, pairp (P a), inc (P (P a)) E, inc (Q (P a)) E & inc (Q a) E].
Lemma GE_prop2 E s:  GT E s -> GL E s -> Law s E.
Lemma GE_prop2_stable E s : GT E s -> GL E s ->
  forall a b, inc a E -> inc b E -> inc (VL s a b) E.
Lemma GE_prop3 E s (f := VL s) :  GT E s -> GL E s ->
               is_law E f /\ (Op E f) = s.

Lemma transport_op E F g s: bijection_prop g E F -> GT E s -> GL E s ->
   forall a b, inc a E -> inc b E ->
    Vf g (VL s a b) = VL (transport s g) (Vf g a) (Vf g b).
```

Bourbaki says that the second part of the axiom is $s \circ (s \times \mathrm{I_E}) = s \circ (\mathrm{I_E} \times s) \circ \mathrm{J}$. This makes no sense. The correct relation is

$$s \circ (s \otimes \mathrm{I_E}) = s \circ (\mathrm{I_E} \otimes s) \circ \mathrm{J_E}. \tag{1.3}$$

Its equivalent (modulo the first part of the axiom) to

$$\forall a, b, c \in \mathrm{E}\ \ a\top(b\top c) = (a\top b)\top c. \tag{1.4}$$

The conjunction of the two axioms is transportable. Proof. For any typification T of s, if $\mathrm{R_1}$ is transportable, if g is a bijection $\mathrm{E} \to \mathrm{F}$, and s' the transport of s, then $\mathrm{R_1} \wedge \mathrm{R_2}$ is transportable if, assuming $\mathrm{T}(s)$, $\mathrm{R_1}(\mathrm{E}, s)$, $\mathrm{T}(s')$, $\mathrm{R_1}(\mathrm{F}, s')$, then $\mathrm{R_2}(\mathrm{E}, s)$ is equivalent to $\mathrm{R_2}(\mathrm{F}, s')$. In this context $\mathrm{R_2}$ is equivalent to (1.4) and the result is obvious.

```
Definition GA E s :=
   s \cg (Sprod s (GE_I E)) = (s \cg (Sprod (GE_I E) s)) \cg (GE_J E).
Lemma GE_prop4 E s  (f := VL s): GT E s -> GL E s ->
   (GA E s <-> forall a b c,
   inc a E -> inc b E -> inc c E -> f a (f b c) =  f (f a b) c).

Lemma transportable_GA: transportable (fun E s => GL E s /\ GA E s).
```

Assume now that we have a unit. The Bourbaki definition (first line) uses $s_1(a,b)=c$; we prefer $(a,b,c)\in s$ (second line). The unit is unique; by AUC we may name it. We have $e\in \mathrm{E}$ and $a\top e = e\top a = a$.

$$\begin{gathered}(\exists z)(z\in x_1 \text{ and } (\forall z')((z'\in x_1)\implies (s_1(z,z')=z' \text{and } s_1(z',z)=z'))).\\ \exists e\in \mathrm{E}, \forall x\in \mathrm{E}, (e,z,z)\in s \text{ and } (z,e,z)\in s.\\ e\in \mathrm{E} \text{ and } \forall a\in \mathrm{E}, a\top e = e\top a = a.\end{gathered}\tag{1.5}$$

```
Definition  GU Z s:=
  exists2 z, inc z E &
    forall z', inc z' E -> inc (J (J z z') z') s /\  inc (J (J z' z) z') s.


Definition unit E s e:= forall z, inc z E -> VL s e z = z /\ VL s z e = z.
Definition un E s := select (unit E s) E.

Lemma GE_prop5 E s : GT E s -> GL E s -> GU E s ->
    exists2 z, inc z E & unit E s z.
Lemma GE_prop6 E s z z':
   inc z E -> unit E s z ->  inc z' E -> unit E s z' -> z = z'.
Lemma GE_prop7 E s : GT E s -> GL E s -> GU E s ->
  inc (un E s) E /\ unit E s  (un E s).
```

Bourbaki says: $\mathrm{R}_1\wedge\mathrm{R}_2\wedge\mathrm{R}_3$ is transportable. The reason is that « R_3 is transportable for the typification "T and $z\in x_1$ and $z'\in x_1$".» This statement is a bit strange. Why is it true? Since z and z' are bound variables in R_3, it is transportable (by CT8) for the typification T in the theory obtained from the current theory by adjoining the axiom $x_1\neq\emptyset$. However, adjoining $x_1=\emptyset$ contradicts $\mathrm{R}_1\wedge\mathrm{R}_2\wedge\mathrm{R}_3$. He deduces that $\mathrm{R}_1\wedge\mathrm{R}_2\wedge\mathrm{R}_3$ is transportable.

The trick is that, if E has a unit e, then $g(e')$ is the unit of F, hence R_3 holds for F. Bourbaki says that the unit it is relatively transportable of type x_1 for the typification T_0. Its definition of a unit is

$$\tau_z(z\in x_1 \text{ and } (\forall z')((z'\in x_1)\implies (s_0(z,z')=z' \text{ and } s_0(z',z)=z'))).$$

```
Lemma GE_prop7_rev E s: GT E s -> GL E s ->
  forall x, inc x E ->  unit E s x -> GU E s.
Lemma transport_unit E F g s x:
  bijection_prop g E F -> GT E s -> GL E s ->
  (inc x E /\ unit E s x) ->
  (inc (Vf g x) F /\ unit F (transport s g) (Vf g x)).
Lemma transport_un E F g s:
  bijection_prop g E F -> GT E s -> GL E s -> GU E s ->
  un F (transport s g) = Vf g (un E s).
Lemma transportable_GU:
  transportable (fun E s => (GL E s /\ GA E s) /\ GU E s).
```

We consider now the final axiom.

$$\forall z,z'\in \mathrm{E}, \exists u\in \mathrm{E}, (z,u,z')\in s \text{ and } \exists v\in \mathrm{E}, (v,z,z')\in s.\tag{1.6}$$

It is equivalent to: for every x and y, there is a and b such that $x\top a = y$ and $b\top x = y$. Taking for y the unit, one gets that every element has a left inverse (as well as a right inverse). If a is

a left inverse and b a right inverse then $a = b$. So, the left inverse is unique. By AUC, we may denote it x^*. This is also the right inverse.

```
Definition GI E s: forall z z', inc z E -> inc z' E ->
   (exists2 z'', inc z'' E & inc (J (J z z'') z') s)
   /\ (exists2 z''', inc z''' E & inc (J (J z''' z') z') s).

Definition left_inverse E s (x x': Set) := inc x' E /\ VL s x' x = un E s.
Definition right_inverse E s (x x': Set) := inc x' E /\ VL s x x' = un E s.
Definition inverse E s x := select (fun x'  => VL s x' x = un E s) E.


Lemma GE_prop8l E s: GT E s -> GL E s -> GU E s -> GI E s ->
  forall x, inc x E -> exists a, left_inverse E s x a.
Lemma GE_prop9 E s : GT E s -> GL E s -> GU E s -> GA E s -> GI E s ->
  forall x a b, inc x E ->
  left_inverse E s x a -> right_inverse E s x b -> a = b.
Lemma GE_prop10l E s: GT E s -> GL E s -> GU E s -> GA E s -> GI E s ->
   forall  x a b, inc x E ->
  left_inverse E s x a -> left_inverse E s x b -> a = b.


Lemma GE_prop11l E s: GT E s -> GL E s -> GU E s ->  GA E s -> GI E s ->
  forall x, inc x E ->  left_inverse E s x (inverse E s x).
Lemma GE_prop11r E s: GT E s -> GL E s -> GU E s ->  GA E s -> GI E s ->
  forall x, inc x E -> right_inverse E s x (inverse E s x).
Lemma GE_prop12l E s: GT E s -> GL E s -> GU E s ->  GA E s -> GI E s ->
  forall x y, inc x E ->  inc y E ->  VL s  x (VL s (inverse E s x) y) = y.
```

If every element has a left and a right inverse, then axiom R_4 holds. If y is a left inverse (or the inverse) of x, then $g(y)$ is a left inverse (or the inverse) of $g(x)$. It follows that the conjunction of the four axioms is transportable. Bourbaki says that the inverse is relatively transportable of type x_1 for the typification "T_0 and $u \in x_1$". His definition of the inverse of u is

$$\tau_z(z \in x_1 \text{ and } s_0(z,u) = e \text{ and } s_0(u,z) = e).$$

```
Lemma GE_prop13a E s: GT E s -> GL E s -> GU E s ->  GA E s ->
  (forall x , inc x E ->
    exists y, left_inverse E s x y /\ right_inverse E s x y) ->
  bijection_prop g E F -> GT E s ->  GL E s -> GU E s ->
  inc x E -> left_inverse E s x y ->
  left_inverse F (transport s g) (Vf g x) (Vf g y).


Lemma transport_inv E F g s x y:
  bijection_prop g E F -> GT E s ->  GL E s -> GU E s ->
  inc x E -> left_inverse E s x y ->
  left_inverse F (transport s g) (Vf g x) (Vf g y).
Lemma transport_inverse E F g s x:
  bijection_prop g E F -> GT E s ->  GL E s -> GA E s ->  GU E s ->  GI E s ->
  inc x E -> Vf g (inverse E s x) = inverse F (transport s g) (Vf g x).
Lemma transportable_GI:
  transportable (fun E s => ((GL E s /\ GA E s) /\ GU E s) /\ GI E s).


Section GroupExample.
```

:

1.2.3 Properties

A group is defined by its typification and its axiom, here $R_1 \wedge R_2 \wedge R_3 \wedge R_4$. There are alternative definitions. For instance, in R_4, it suffices to take for z' the unit. Since the left inverse is the right inverse, one can write it as: whenever $a \in E$, there is $b \in E$ such that $a \top b = b \top a = e$.

The relation R_s

$$(\forall z), ((z \in v) \implies (z^{-1} \in v \text{ and } (\forall z')((z' \in v \implies (s_0(z, z') \in v))))$$

is relatively transportable for the typification T_0 and $v \in \mathfrak{P}(x_1)$. Proof. Same notations as above. Assume $v \subset E$; then v' is the set of all $g(x)$ for $x \in v$. Denote by x^* the inverse in E, by x^{-1} the inverse in F. Take $x' \in v'$, so that $x' = g(x)$ for some $x \in v$. Note that $x \in E$, and $g(x)$ is well defined. The relation R_s says $x^* \in v$, so that $g(x^*) \in v'$. But $g(x^*) = x'^{-1}$. Assume $y' \in v'$, so that $y' = g(y)$ with $y \in v$. Now, $x \top y \in v$ so $x' \perp y' = g(x \top y) \in v'$. The conclusion follows. The relation R_s is interpreted as v is a subgroup. The subgroup generated by w is relatively transportable of type $\mathfrak{P}(x_1)$ for the typification T_0 and $w \in \mathfrak{P}(x_1)$. The relation

$$(\exists z')(\exists z'')(z' \in x_1 \text{ and } z'' \in x_1 \text{ and } z = s_0(z', s_0(s'', s_0(z'^{-1}, z''^{-1}))))$$

is read: z is a commutator. It is transportable for the typification T_0 and $z \in x_1$. So: the commutator subgroup is relatively transportable of type $\mathfrak{P}(x_1)$.

Note: it should be easy to formalize the notion of: $R(z)$ is relatively transportable of type A for a typification T and $T'(z)$; then prove the previous statements in Coq.

1.3 Trees

Recall that (1.1) says $x \in \mathfrak{P}(E \times E \times E)$. Write it in the form $s \in S(E)$. If g is a bijection $E \to F$, we consider its extension $g^S : S(E) \to S(F)$. In what follows, S will be implemented by an echelon. However, there is more than one echelon such that $S(E) = \mathfrak{P}(E \times E \times E)$, and it is not clear whether these echelons give the same g^S. For this reason we shall consider another formalism: that of a tree. In the case of a vector set E over a field K, there are two sets, E and K. These are called the base sets, and numbered x_1, x_2.

There are three possibilities for a tree: There is a base case T_b (for instance if $x = T_b(0)$ then $S(x) = E$); there is the case of a product, for instance, if $y = T_x(T_x(x, x), x)$ then $S(y) = (E \times E) \times E$; there is the case of the power set, for instance $S(T_p(y)) = \mathfrak{P}((E \times E) \times E)$.

We show here an example of tree, in Coq, and show two functions on trees, depth and size.

```
Inductive Tree :=
  | Tbase: nat -> Tree
  | Tpowerset : Tree -> Tree
  | Tproduct : Tree -> Tree -> Tree.

Fixpoint Tdepth e:=
  match e with
    | Tbase _ => 0
```

```
    | Tpowerset e' => (Tdepth e').+1
    | Tproduct e' e'' => (maxn (Tdepth e') (Tdepth e'')).+1
end.

Fixpoint Tsize e:=
  match e with
    | Tbase n => n
    | Tpowerset e' => Tsize e'
    | Tproduct e' e'' => maxn (Tsize e') (Tsize e'')
  end.
```

Let's try to implement a tree as a Bourbaki set. The result will be called a `tree`: a tree is either a pair $(0,n)$, a pair $(1,\mathrm{T})$ or a triple $(2,\mathrm{T},\mathrm{T}')$ where T and T′ are trees. This is not a structure, in the Bourbaki sense, although the notion of morphisms and isomorphisms are defined and share the same properties as morphisms and isomorphisms of other structures.

```
Definition Tb x := J \0c x.
Definition Tp x := J \1c x.
Definition Tx x y := J \2c (J x y).
```

Can we have $\mathrm{T} = (1,\mathrm{T})$? In an earlier version of Bourbaki, pairs were defined by an axiom, so the answer is: maybe. If we use the Kuratowski definition, then the answer is still maybe, but, if the foundation axiom holds then answer is no, if the anti-foundation axiom holds, then the answer is yes, and T is unique. If the answer is yes, we cannot define the depth of a tree, since the depth d of this tree would satisfy $d = d+1$. Instead of 0, 1, 2, we could use other markers, in order to solve this difficulty; however, there is no guarantee that the tree will be finite. On the other hand, if we assume the existence of a depth function, one can proceed by induction (more technically by stratified induction, see above). We have studied in [8] the set of formulas, and we use here the same techniques.

The idea is to consider the set of trees of depth $\leq n$, defined by induction.

```
Definition tset_base := fun_image Nat Tb.

Definition Tset_next E :=
  fun_image E Tp
  \cup fun_image (E \times E) (fun p => J \2c p)
  \cup E.


Lemma tset_baseP x: inc x tset_base <-> exists2 n, natp n & x = Tb n.
Lemma tset_basei n: natp n -> inc (Tb n) tset_base.
Lemma tset_nextP E x:  inc x (tset_next E) <->
  [\/ exists2 y, inc y E  & x = Tp y,
      exists y z, [/\ inc y E, inc z E & x = Tx y z]
    | inc x E].
```

We define by induction $\mathscr{T}_{n+1} = f(\mathscr{T}_n)$, then $\mathscr{T} = \bigcup_{n\in\mathbf{N}} \mathscr{T}_n$. We say that an element x of $\mathscr{T}$ is a *tree*. We have either $x \in \mathscr{T}_0$, or there is an integer n such that $x \in \mathscr{T}_{n+1}$ and $x \notin \mathscr{T}_n$.

```
Definition tset_index := induction_term (fun _ x => tset_next x) tset_base.
Definition tset := unionf Nat tset_index.
Definition treep x := inc x tset.
```

```
Lemma tset_index0: tset_index \0c = tset_base.
Lemma tset_indexS n:  natp n ->
    tset_index (csucc n) = tset_next (tset_index n).
Lemma tsetP x: treep x <-> exists2 n, natp n & inc x (tset_index n).
Lemma tset_base_hi x: inc x tset_base -> treep x.
Lemma tset_min x: treep x ->
  inc x tset_base \/
  exists n, [/\ natp n, inc x (tset_index (csucc n)) & ~inc x (tset_index n)].
```

Let's define the *depth* $d(x)$ of x as the least n such that $x \in \mathscr{T}_n$. Since $\mathscr{T}_n \subset \mathscr{T}_{n+1}$, we have $x \in \mathscr{T}_n$ whenever $n \geq d(x)$ is at least the depth, and conversely. This implies that, if $d(x) = 0$, then $x \in \mathscr{T}_0$, if $d(x) = n+1$, then $x = (1, x')$ or $x = (2, (x', x''))$, where x' and x'' are trees of depth $\leq n$. Conversely, if n is an integer, $(0, n)$ is a tree of depth 0, if x is a tree then $(1, x)$ is tree of depth $d(x)+1$, if x' is a tree, then $(2, (x, x'))$ is a tree of depth $1 + \max(d(x), d(x'))$. We deduce a principle of induction similar to `Tree_ind`.

```
Definition tdepth x := intersection (Zo Nat (fun n => inc x (tset_index n))).

Lemma tdepth1 x (n:= tdepth x): treep x ->
   [/\ natp n, inc x (tset_index n) &
    forall m, natp m -> inc x (tset_index m) -> n <=c m].
Lemma NS_tdepth x: treep x -> natp (tdepth x).
Lemma tdepth2 x m: treep x -> natp m -> (tdepth x) <=c m ->
  inc x (tset_index m).
Lemma tdepth3 x m: natp m -> inc x (tset_index m) -> (tdepth x) <=c m.
Lemma tdepth4 x: treep x -> tdepth x = \0c -> inc x tset_base.


Lemma tdepth_prop x n: treep x -> natp n -> tdepth x = (csucc n) ->
  (exists y, [/\ treep y, tdepth y <=c n & x = Tp y]) \/
  (exists y z, [/\ treep y, treep z, tdepth y <=c n, tdepth z <=c n &
     x = Tx y z]).

Lemma tdepth_prop_inv:
  [/\ forall n, natp n -> treep (Tb n)  /\ tdepth (Tb n) = \0c,
  forall t, treep t -> treep (Tp t) /\ tdepth (Tp t) = csucc (tdepth t) &
  forall t t', treep t -> treep t' -> treep (Tx t t') /\
           tdepth (Tx t t') = csucc (cmax (tdepth t) (tdepth t'))  ].

Lemma TS_base n: natp n ->treep (Tb n).
Lemma TS_powerset t: treep t ->  treep (Tp t).
Lemma TS_product t t': treep t -> treep t' ->  treep (Tx t t').

Lemma tree_ind (p: property):
  (forall n, natp n -> p (Tb n)) ->
  (forall x, treep x -> p x -> p (Tp x)) ->
  (forall x x', treep x -> treep x' -> p x -> p x' -> p(Tx x x')) ->
  (forall x, treep x -> p x).
```

Recall the definition by stratified induction. It depends on a property W (being a tree) a function ρ (the depth), an operator H, specified below. We first must show that $\rho(x)$ is an ordinal, whenever $W(x)$ holds; this is trivial. We then must show that there is a set W_α (for every ordinal α), such that $x \in W_\alpha$ if and only if x is a tree of depth $< \alpha$. If $\alpha = 0$, then W_α must be empty; if α is infinite then $W_\alpha = \mathscr{T}$ since all trees have a finite depth. If n is finite, then $W_{n+1} = \mathscr{T}_n$.

```
Definition tree_stratified i E :=
  forall x, inc x E <-> treep x /\ tdepth x <o i.
Definition tstratified i :=
  Yo (i = \0c) emptyset
     (Yo (omega0 <=o i) tset (tset_index (cpred i))).

Lemma tree_rec_def_aux1: forall x, treep x -> ordinalp(tdepth x).
Lemma tree_rec_def_aux2a: tree_stratified \0c emptyset.
Lemma tree_rec_def_aux2b i: omega0 <=o i -> tree_stratified i tset.
Lemma tree_rec_def_aux2c i: i <o omega0 -> i <> \0c ->
   tree_stratified i (tset_index (cpred i)).
Lemma tree_rec_def_aux2: forall i, ordinalp i -> exists E, tree_stratified i E.
Lemma tstratified_val i: ordinalp i ->
  stratified_set treep tdepth i = tstratified i.
```

The previous lemmas allow us to define a function f such that $f(x) = \mathrm{H}(x, f_x)$, where f_x is the restriction of f to $\mathrm{W}_{\rho(x)}$. The following definitions and lemmas are stated in a context where h_1, h_2 and h_3 are three functional terms. We can then construct a function f such that $f(\mathrm{T}_b(n)) = h_1(n)$, $f(\mathrm{T}_p(x)) = h_2(f(x))$ and $f(\mathrm{T}_x(xy)) = h_3(f(x), f(y))$. If the function h_i take their values in a set F, then $f(x) \in \mathrm{F}$.

```
Definition tree_rec_prop x f :=
  Yo (P x = \0c) (h1 (Q x))
     (Yo (P x = \1c) (h2 (Vg f (Q x))) (h3 (Vg f (P (Q x))) (Vg f (Q (Q x))))).

Definition tree_rec := stratified_fct treep tree_rec_prop tdepth.

Lemma tree_recdef_p x: treep x ->  tree_rec x =
   tree_rec_prop x (Lg (tstratified (tdepth x))  tree_rec).
Lemma tree_recdef_pb' n: natp n -> tree_rec (Tb n) = h1 n.
Lemma tree_recdef_pb x : inc x ttset_base ->  tree_rec x = h1 (Q x).
Lemma tree_recdef_pp x: treep x -> tree_rec (Tp x)  = h2 (tree_rec x).
Lemma tree_recdef_px x y: treep x -> treep y ->
   tree_rec (Tx x y)  = h3 (tree_rec x) (tree_rec y).

Lemma tree_rectdef_stable E:
  (forall n, natp n -> inc (h1 n) E) ->
  (forall x, inc x E -> inc (h2 x) E) ->
  (forall x x', inc x E -> inc x' E -> inc (h3 x x') E) ->
  (forall x, treep x -> inc (tree_rec x) E).
```

Let's consider an example: definition of the depth by induction. We show that this is the same as the previous definition.

```
Definition Tree_depth_alt :=
  tree_rec (fun _ => \0c) csucc (fun a b => csucc (cmax a b)).
Lemma tree_depth_altE  x: treep x -> (tree_depth_alt x) = tdepth x.
```

Let's say that a tree is positive if the arguments of T_b are all non-zero. In Coq, this would be

```
Fixpoint Tpositive e:=
  match e with
    | Tbase n =>  n != 0
```

```
    | Tpowerset e' =>  Tpositive e'
    | Tproduct e' e'' => (Tpositive e') && (Tpositive e'')
  end.
```

The principle of induction of COQ allows us to define a function of any type; but in Bourbaki, we are limited to sets (in particular, we cannot define a function that associates a Tree to a tree). What we can do is define a boolean, and convert it to a proposition. If zero means false and one means true, then the min function corresponds to boolean or. We first state a lemma that says that our function f takes only 0 and 1 as values. We then get: the tree $(0,n)$ is positive if and only if n is non-zero, the tree $(1,x)$ is positive if and only if x is positive, and $(2,(x,x'))$ is positive if and only if x and x' are positive.

```
Definition tree_to_pos :=
  tree_rec (fun  n => Yo (n = \0c) \0c \1c) id (fun a b => (cmin a b)).
Definition tree_is_pos x := tree_to_pos x = \1c.

Lemma tree_rec_bool h1 (f := tree_rec_ h1 id (fun a b => (cmin a b))):
   (forall x, natp x -> h1 x <=c \1c) -> (forall x, treep x -> f x <=c \1c).

Lemma tree_to_pos_p1:
  [/\ (forall x, natp x -> tree_to_pos (Tb (csucc x)) = \1c),
   (tree_to_pos (Tb \0c) = \0c),
   (forall x, treep x -> tree_to_pos (Tp x) = tree_to_pos x) &
   (forall x x', treep x -> treep x' ->
      tree_to_pos (Tx x x') = cmin (tree_to_pos x) (tree_to_pos x'))].
Lemma tree_to_pos_p2:
  [/\ (forall x, natp x -> (tree_is_pos (Tb x) <-> x <> \0c)),
   (forall x, treep x -> (tree_is_pos (Tp x) <-> tree_is_pos x)) &
   (forall x x', treep x -> treep x' ->
      (tree_is_pos (Tx x x') <-> ((tree_is_pos x) /\ (tree_is_pos x'))))].
```

We define here by induction the size of a tree.

```
Definition tree_size := tree_rec id id cmax.

Lemma tree_size_p:
  [/\ (forall x, natp x -> tree_size (Tb x) = x),
   (forall x, treep x -> tree_size (Tp x) = tree_size x) &
   (forall x y, treep x ->treep y  ->
   tree_size (Tx x y) = cmax (tree_size x) (tree_size y)) ].
Lemma NS_rree_size x: treep x -> natp (tree_size x).
```

We define now a function that maps a Tree to a tree. This is a bijection. It respects all the properties defined above.

```
Fixpoint Tree_to_tree e:=
  match e with
    | Tbase n =>  J \0c (nat_to_B n)
    | Tpowerset e' => J \1c (Tree_to_tree e')
    | Yproduct e' e'' => J \2c (J (Tree_to_tree e') (Tree_to_tree e''))
end.

Lemma Tree_to_tree_prop e (t := Tree_to_tree e):
  [/\ treep t,
```

```
      tdepth t = nat_to_B (Tdepth e),
      tree_size t = nat_to_B (Tsize e)&
      tree_is_pos t <-> Tpositive e].
Lemma Tree_to_tree_injective: injective Tree_to_tree.
Lemma Tree_to_tree_surjective x: treep x -> exists e, x = Tree_to_tree e.
```

Chapter 2

Structures and isomorphisms

In the example of the introduction we had one set E, one structure s, and an axiom. We considered a second set F, a bijection g, then deduced s'. We showed that the axiom was transportable: this means that if the axiom holds for E and s, if holds for F and s'. We have also seen that if v is a subgroup, so is v'. The way s', v', etc, are constructed depends only on the typification of s, v, etc. For instance, v' is the set of all $g(x)$ for $x \in v$. Since v is a subset of the source of g, this is well defined and is a subset of the target of g, namely F. The idea behind the typification is to provide a systematic way to transport objects. In the case of vector space E over K, there are two laws, with typification $s_1 \in \mathfrak{P}((\mathrm{E} \times \mathrm{E}) \times \mathrm{E})$ and $s_2 \in \mathfrak{P}((\mathrm{K} \times \mathrm{E}) \times \mathrm{E})$; so that $(s_1, s_2) \in \mathfrak{P}((\mathrm{E} \times \mathrm{E}) \times \mathrm{E}) \times \mathfrak{P}((\mathrm{K} \times \mathrm{E}) \times \mathrm{E})$. This has the form $(s_1, s_2) \in \mathrm{S}(\mathrm{E}, \mathrm{K})$.

What we want to do is to formalise the quantity $\mathrm{S}(\mathrm{E}, \mathrm{K})$. This is called "echelon construction of scheme S on E and K", where S is called an "echelon construction scheme"; for simplicity, we just say echelon for S. The relation is equivalent to "$s_1 \in \mathrm{S}_1(\mathrm{E}, \mathrm{K})$ and $s_2 \in \mathrm{S}_2(\mathrm{E}, \mathrm{K})$". Such a relation is called a typification of the letters s_1 and s_2. There is a small problem here: Bourbaki assumes that S_1 and S_2 are echelons on two terms (this makes sense, as they are applied to two terms). However $\mathrm{S}_1(\mathrm{E}, \mathrm{K})$ is independent of K, so is morally an echelon on one term. It is however possible to modify S_1 such that it has the correct size, without changing the value. The problem is now: if we transport our terms with the modified echelon, do we get the same value or not?

For this reason, we shall give two implementations; in the first variant an echelon will be a tree and there will be uniqueness. The second implementation a linearized version of the tree, i.e., a list with some properties.

We proceed as follows: take $\mathrm{A}_1 = \mathrm{E}$, $\mathrm{A}_2 = \mathrm{A}_1 \times \mathrm{A}_1$, $\mathrm{A}_3 = \mathrm{A}_2 \times \mathrm{A}_1$, $\mathrm{A}_4 = \mathfrak{P}(\mathrm{A}_3)$, $\mathrm{A}_5 = \mathrm{K}$, $\mathrm{A}_6 = \mathrm{A}_5 \times \mathrm{A}_1$, $\mathrm{A}_7 = \mathrm{A}_6 \times \mathrm{A}_1$, $\mathrm{A}_8 = \mathfrak{P}(\mathrm{A}_7)$, $\mathrm{A}_9 = \mathrm{A}_4 \times \mathrm{A}_8$. Now $\mathrm{S}(\mathrm{E}, \mathrm{K})$ is the last set in the list, namely A_9. In order to define S as a function of two arguments, we replace E by x_1, and K by x_2. We can reduce the length of this character string: first remove every equal sign and what is on the left (this is redundant information); second, in the case of a product, just keep the indices; in the case of a powerset, keep the index, followed by 0, otherwise keep the index, preceded by zero (this works as no index is zero). We get 0,1,1,1,2,1,...,4,8, a list of 18 integers. Add some parentheses; we get

$$(0,1),(1,1),(2,1),(3,0),(0,2),(5,1),(6,1),(7,0),(4,8).$$

We get a list of 9 pairs of integers. This is an example of an *echelon*, in the Bourbaki sense.

We shall consider below the case of (0,1), (0,2), (1,0), (3,0), (2,0), (4, 5), denoted S_3, and show that this gives $\mathrm{S}_3(\mathrm{E}, \mathrm{F}) = \mathfrak{P}(\mathfrak{P}(\mathrm{E})) \times \mathfrak{P}(\mathrm{F})$. Bourbaki considers also (0,2), (0,1), (1,0), (2,0),

(4,0), (5, 3), denoted S_4, says $S_4(E,F) = S_3(E,F)$ and deduces « Distinct schemes may therefore give rise to the same echelon on the same terms. » There is a simpler example: if we add (4,5) (the last term of S_3) to the right of S_4 we get a longer scheme with the same behavior. This new scheme is not minimal (some sets are useless). The scheme (0,1), (1,1), (1,1), (2,3) is not minimal, as it could be replaced by (0,1), (1,1), (2,2). In the example above both schemes are minimal, yet are different and behave the same.

Add now before each pair its index: we get

$$(0,0,1),(1,1,1),(2,2,1),(3,3,0),(4,0,2),(5,5,1),(6,6,1),(7,7,0),(8,4,8).$$

What we get now is a list of triples. If the triple is (i,a,b), we must have $a \leq i$ and $b \leq i$ (except when $a = 0$, case where $1 \leq b \leq 2$ is required in order to apply it to n terms). In the first chapter, we manipulated lots of triples, of the form $((a,b),c)$. Here they are of the form $(i,(a,b))$. With this interpretation, the set of these elements is a functional graph, whose domain is a subset of **N**, and whose range is a subset of $\mathbf{N} \times \mathbf{N}$.

2.1 Echelons

« An *echelon construction scheme* is a sequence $c_1, c_2, \ldots, c_m$ of ordered pairs of natural integers $c_i = (a_i, b_i)$ satisfying the following conditions: (a) if $b_i = 0$, then $1 \leq a_i \leq i-1$, (b) if $a_i \neq 0$ and $b_i \neq 0$, then $1 \leq a_i \leq i-1$ and $1 \leq b_i \leq i-1$. If n is the largest of the integers b_i which appear in the pairs $(0, b_i)$ then $c_1, c_2, \ldots, c_m$ is said to be an echelon construction scheme *on n terms.* »

As mentioned above, an echelon will be a functional graph, with some properties. The integer m is called the *length* of the list, and n the *size* of the list. Because our indices start at zero, the condition $1 \leq a_i \leq i-1$ becomes $1 \leq a_i \leq i$; we say that a_i is good (with respect to i); this relation implies $0 \leq a_i - 1 < i$. It particular, it is false when $i = 0$. Thus $a_1 = 0$ and $b_1 > 0$, and the size is well defined (provided that m non-zero). Our definition of the size works in any case (the size of the empty list being zero; it is non-zero otherwise).

```
Definition ech_good x i :=  \1c <=c x /\ x <=c i.
Definition echelon c :=
  slist_E c (Nat \times Nat) /\
  forall i, i <c (slength c) ->
            let a:= P (Vg c i) in
            let b:= Q (Vg c i) in
            (b = \0c -bo > ech_good a i) /\
            (b <> \0c -> a <> \0c -> ech_good a i /\ ech_good b i).

Definition esize c :=
  \csup(range (Lg (domain c) (fun i=> Yo (P (Vg c i) = \0c) (Q (Vg c i)) \0c))).


Lemma echelon_p1 c:  echelon c ->
  \0c <c slength c ->
  exists b, [/\ natp b, \0c <c b, Vl c \1c = J \0c b].
Lemma echelon_p1' c: echelon c ->
  \0c <c slength c ->
  exists b, [/\ natp b, \0c <c b & Vg c \0c = J \0c b].
Lemma esize_empty c : echelon c ->
  slength c = \0c -> esize c = \0c.
```

```
Lemma esize_prop1 c (n:= esize c) (m:=slength c):
  echelon c -> \0c  <c m ->
  [/\ natp n, \0c <c n, exists2 j, j <c m & Vg c j = J \0c n &
   forall j, j <c m -> P (Vg c j) = \0c -> Q (Vg c j) <=c n].
Lemma esize_prop2 c n (m:=slength c):
  echelon c ->
  (exists2 j, j <c m & Vg c j = J \0c n) ->
  (forall j, j <c m -> P (Vg c j) = \0c -> Q (Vg c j) <=c n) ->
  esize c = n.
Lemma NS_esize c: echelon c ->
  natp (esize c).
```

In the Appendix to Chapter IV, in a footnote, Bourbaki explains how to build a scheme from other schemes, so that $s'(\mathrm{E}) = \mathfrak{P}(s(\mathrm{E}))$ or $s'(\mathrm{E}) = s_1(\mathrm{E}) \times s_2(\mathrm{E})$. We consider a third case, it produces an echelon of length one from an integer.

```
Definition Ech_base n := Lg \1c (fun z => (J \0c n)).

Lemma Ech_base_prop n (c:= Ech_base n):
  natp n -> \0c <c n ->
   [/\ echelon c, Vg c \0c = J \0c n, \0c <c slength c & esize c = n].
```

The second operation is easy; if s is of length m, it suffices to add $(m,0)$ at the end.

```
Definition Ech_powerset c:=
  c +s1 J (slength c) (J (slength c) \0c).

Lemma Ech_powerset_prop c (m := slength c)(c':= Ech_powerset c):
  echelon c -> \0c <c m ->
  [/\ echelon c',  slength c' =  csucc m, esize c' =  esize c,
     Vg c' m = J m \0c & forall k, k <c m -> Vg c' k = Vg c k].
```

The third operation is more complex. If m_1 and m_2 are the sizes of s_1 and s_2, we construct a list of size $s_1 + s_2 + 1$ formed of s_1, a modified version of s_2 and a final term. If m_1 and m_2 are zero, this final term is (0,0), and the construction is invalid. If m_1 is non-zero, m_2 is zero, we get an object that evaluates as $s_1(\mathrm{E}) \times s_2(\mathrm{E})$.

```
Definition ech_shift n v:=
  Yo (P v = \0c) v (Yo (Q v = \0c) (J (P v +c n) \0c)
                       (J (P v +c n) (Q v +c n))).
Definition ech_product1 f g n m i:=
   Yo (i <c n) (Vg f i)
    (Yo (i = n +c m) (J n (n +c m)) (ech_shift n (Vg g (i -c n)))).
Definition Ech_product f g :=
  let n := (slength f)  in let m := (slength g) in
  Lg (csucc (n +c m))(ech_product1 f g  n m).


Lemma ech_product_prop1 f g n m i (v:= ech_product1 f g n m):
  natp n -> natp m ->
  [/\ i <c n -> v i = (Vg f i),  v(n +c m) =  (J n (n +c m)) &
      i <c m -> v (n +c i) =  ech_shift n (Vg g i)].

Lemma Ech_product_prop f g (n := slength f) (m:= slength g)
```

```
  (h := Ech_product f g):
  echelon f ->
  echelon g ->
  \0c <c n ->
  [/\ echelon h,
      slength h = csucc (n +c m),
      esize h = cmax  (esize f) (esize g) &
      [/\ forall i, i <c n -> Vg h i = (Vg f i),
          Vg h (n +c m) =  (J n (n +c m)) ,
          forall i, i <c m -> Vg h (n +c i) =  ech_shift n (Vg g i) &
          forall i, n <=c i -> i <c n +c m ->
              Vg h i = ech_shift n (Vg g (i -c n))]].
```

What we would like to do now is: define a function that maps an echelon to a tree, and conversely. This second operation is easy to do.

```
Fixpoint Tree_to_echelon t :=
  match t with
    | Tbase n  => Ech_base (nat_to_B n.+1)
    | Tpowerset t' =>  Ech_powerset (Tree_to_echelon t')
    | Tproduct t' t'' =>
      Ech_product (Tree_to_echelon t') (Tree_to_echelon t'')
end.

Definition tree_to_echelon x := tree_rec
  (fun n => Ech_base (csucc n))
  (fun t => Ech_powerset t)
  (fun t t'  => Ech_product  t t') x.

Lemma tree_to_echelon_E (f:=tree_to_echelon) :
  [/\ forall n, natp n -> f (Tb n) =  Ech_base (csucc n),
    forall t, treep t -> f (Tp t) =  Ech_powerset (f t) &
    forall t t', treep t -> treep t' ->
                f (Tx t t') =   Ech_product (f t) (f t')].


Lemma tree_to_echelon_E (f:=tree_to_echelon) :
  [/\ forall n, natp n -> f (Tb n) =  Ech_base (csucc n),
    forall x, treep x -> f (Tp x) =  Ech_powerset (f x) &
   forall x y, treep x -> treep y ->
                f (Tx x y) =   Ech_product (f x) (f y)].

Lemma tree_to_echelon_prop2 t:
  tree_to_echelon (Tree_to_tree t) = Tree_to_echelon t.

Lemma tree_to_echelon_ok t (c := tree_to_echelon t): treep  t ->
  [/\ echelon c, \0c <c slength c &  esize c = csucc (tree_size t)].
```

The converse operation is a bit trickier. It requires a new induction principle. As previously, we consider three functions h_1, h_2 and h_3. We combine them into a single function p, that takes 3 arguments; f, a, b, and is defined by: if $a = 0$, then $h_1(b)$, if $b = 0$ then $h_2(f(a))$, otherwise $h_3(f(a), f(b))$. It satisfies the following property: if f_1 and f_2 agree for values $< i$, and c_i is equal to (a, b), then $p(f_1, a, b) = p(f_2, a, b)$.

```
Definition Erecdef_combine h1 h2 h3 :=
```

```
  fun f a b => Yo (a = \0c) (h1 b)
                  (Yo (b = \0c) (h2  (Vl f a)) (h3 (Vl f a) (Vl f b))).
Definition echelon_recdef_prop c  (p: Set -> Set -> Set -> Set):=
   forall g1 g2 i,
   i <c  slength c ->
   (forall j, j <c i -> Vg g1 j = Vg g2 j) ->
   p  g1 (P (Vg c i)) (Q (Vg c i)) = p  g2 (P (Vg c i)) (Q (Vg c i)).

Lemma erecdef_prop1 c:
  echelon c -> echelon_recdef_prop c (erecdef_combine h1 h2 h3).
```

The idea is to define by transfinite induction a function f on the set of all integers, then restrict its graph to I_m, where m is the domain of the echelon. The resulting list is the unique one that satisfies the relation. We then state: if $c_i = (a, b)$ then in case $a = 0$, we have $1 \leq b \leq n$ and $f_i = h_1(b)$; in case $b = 0$, we have $1 \leq a \leq i$ and $f_i = h_2(f(a))$, otherwise both a and b are between 1 and i and $f_i = h_3(f(a), f(b))$.

```
Definition echelon_recdef c (p :=  erecdef_combine h1 h2 h3) :=
  restr (graph (transfinite_defined Nat_order
     (fun u => (p  (graph u) (P (Vg c (source u))) (Q (Vg c (source u)))))))
        (slength c).
Lemma erecdef_prop c (m := slength c)(f := echelon_recdef c)
      (p :=  erecdef_combine h1 h2 h3):
  echelon c ->
  [/\ fgraph f, domain f = m &
   forall i, i <c m -> Vg f i = p f (P (Vg c i)) (Q (Vg c i))].
Lemma erecdef_unique c f (m := slength c) (p :=  erecdef_combine h1 h2 h3):
  echelon c ->
  slistpl f m ->
  (forall i, i <c m -> Vg f i = p f (P (Vg c i)) (Q (Vg c i))) ->
  f =  echelon_recdef c.
Lemma ecrecdef_unique1 c f (m := slength c):
  echelon c ->
  slistpl f m ->
  (forall i, i <c m ->
    let a:= P (Vg c i) in let b := Q (Vg c i) in
    [/\ a = \0c -> Vg f i = h1 b,
        b = \0c -> Vg f i = h2 (Vl f a)
      & a <> \0c -> b <> \0c -> Vg f i = h3 (Vl f a) (Vl f b)]) ->
  f = echelon_recdef c.

Lemma erecdef_prop2 c (m := slength c)(f :=  echelon_recdef c)
   (n:=  esize c):
  echelon c -> forall i, i <c m ->
  let a:= P (Vg c i) in let b := Q (Vg c i) in
  [/\ a = \0c -> [/\ \1c <=c b, b <=c n & Vg f i = (h1 b)],
      b = \0c -> [/\ \1c <=c a, a <=c i & Vg f i = h2 (Vl f a) ]
    & a <> \0c -> b <> \0c -> [/\ \1c <=c a, a <=c i, \1c <=c b, b <=c i &
   Vg f i = h3 (Vl f a) (Vl f b)]].
```

Define $g(c)$ as $f(i)$ where i is the length of c. If $c' = \mathfrak{P}(c)$ then $g(c') = h_2(g(c))$, if $c'' = c \times c'$, then $g(c'') = h_3(g(c), g(c'))$. The first property is easy; the second is a bit more complicated because the product of two echelon is non-trivial.

```
Lemma erecdef_restr c n:
```

```
   echelon c -> n <=c slength c ->
   echelon_recdef (restr c n) = restr (echelon_recdef c) n.

Lemma echelon_recdef_extent2 c c' i:
  echelon c -> echelon c' -> i <=c slength c -> i <=c slength c' ->
  i <> \0c ->
  (forall k, k<c i -> Vg c k = Vg c' k) ->
  Vl (echelon_recdef c) i = Vl (echelon_recdef c') i.

Definition echelon_recdef_last c := Vl (echelon_recdef c) (slength c).

Lemma erecdef_base n (c := Ech_base n):
  natp n -> \0c <c n -> echelon_recdef_last c = h1 n.
Lemma erecdef_powerset c (c' := Ech_powerset c):
  echelon c -> \0c <c slength c ->
  echelon_recdef_last c' = h2 ( echelon_recdef_last c).
Lemma erecdef_product c c' (c'' := Ech_product c c'):
  echelon c -> echelon c' -> \0c <c slength c -> \0c <c slength c' ->
  echelon_recdef_last c'' = h3 (echelon_recdef_last c) (echelon_recdef_last c').
```

Example. Take $h_1(x) = \mathrm{T}_b(x-1)$, $h_2 = \mathrm{T}_p$ and $h_3 = \mathrm{T}_x$. It is easy to show, by induction, that the result is a list of trees.

```
Definition echelon_to_trees :=  echelon_recdef (fun b => Tb (cpred b)) Tp Tx.

Lemma echelon_to_trees_prop c (m := slength c)(f := echelon_to_trees c)
      (n := esize c):
  echelon c ->
  [/\ fgraph f, domain f = m,
      forall i, i <c m -> treep (Vg f i) &
      forall i, i <c m ->
        let a:= P (Vg c i) in let b := Q (Vg c i) in
         [/\ a = \0c -> [/\ \1c <=c b, b <=c n & Vg f i =  Tb (cpred b)],
           b = \0c -> [/\ \1c <=c a, a <=c i &  Vg f i = Tp (Vl f a)]
          & a <> \0c -> b <> \0c -> [/\ \1c <=c a, a <=c i, \1c <=c b, b <=c i
               & Vg f i = Tx (Vl f a) (Vl f b)]]].
```

We now rewrite this result using Trees.

```
Lemma ET_val1 c i (f := echelon_to_trees c):
  echelon c -> i <c (slength c) -> P (Vg c i) = \0c ->
  exists n,  Q (Vg c i) = csucc (nat_to_B n) /\
  Vg f i = Tree_to_tree (Tbase n).
Lemma ET_val2 c i (f := echelon_to_trees c):
  echelon c -> i <c (slength c) -> Q (Vg c i) = \0c ->
  exists2 E, Tree_to_tree E = (Vl f (P (Vg c i))) &
              Tree_to_tree (Tpowerset E) = Vg f i.
Lemma ET_val3 c i (f := echelon_to_trees c)
      (a := (P (Vg c i))) (b := Q (Vg c i)):
  echelon c ->  i <c (slength c) -> a <> \0c -> b <> \0c ->
  exists E F, [ /\  Tree_to_Tree E = Vl f a,
                    Tree_to_Tree F = Vl f b&
                    Tree_to_Tree (Tproduct E F) = Vg f i ].
```

We are now ready to continue with the Bourbaki text.

« Given an echelon construction scheme $S = (c_1, c_2, \ldots, c_m)$ on n terms, and given n terms $E_1, E_2, \ldots, E_n$ in a theory $\mathcal{T}$ which is stronger than the theory of sets, an *echelon construction of scheme* S *on* $E_1, \ldots, E_n$ is defined to be a sequence $A_1, A_2, \ldots, A_m$ of m terms in the theory $\mathcal{T}$, defined step by step by the following conditions:

(a) if $c_i = (0, b_i)$, then A_i is the term E_{b_i},

(b) if $c_i = (a_i, 0)$, then A_i is the term $\mathfrak{P}(A_{a_i})$,

(c) if $c_i = (a_i, b_i)$ where $a_i \neq 0$ and $b_i \neq 0$, then A_i is the term $A_{a_i} \times A_{b_i}$.

The last term A_m of the echelon construction of scheme S on $E_1, \ldots, E_n$ is called the *echelon of scheme* S *on the base sets* $E_1, \ldots, E_n$; in the general arguments that follow, it will be denoted by the notation $S(E_1, \ldots, E_n)$. »

We shall denote by $\bar{S}(E)$ the list of the A_i, so that $S(E) = \bar{S}(E)_m$. The lemmas that follow are trivial.

```
Definition echelon_value c E :=
  echelon_recdef  (fun b => (Vl E b)) powerset product c.

Definition echelon_of_base c E :=
  Vl (echelon_value c E) (slength c).


Lemma echelon_of_baseE c E:
  echelon_of_base c E =
  echelon_recdef_last (fun b => (Vl E b)) powerset product c.
Lemma echelon_value_prop c E i (m := slength c)(f :=  echelon_value c E)
   (n:=  esize c):
  echelon c -> i <c m ->
  let a:= P (Vg c i) in let b := Q (Vg c i) in
  [/\ a = \0c -> [/\ \1c <=c b, b <=c n & Vg f i = (Vl E b)],
      b = \0c -> [/\ \1c <=c a, a <=c i & Vg f i = \Po (Vl f a) ]
    & a <> \0c -> b <> \0c -> [/\ \1c <=c a, a <=c i, \1c <=c b, b <=c i &
   Vg f i = (Vl f a) \times (Vl f b)]].
```

We can evaluate a tree in the same way as an echelon. If T is a tree, converted to an echelon S, then, whatever E, $T(E) = S(E)$.

```
Definition tree_value E x := tree_rec
  (fun n => Vg E n)
  (fun t => \Po t)
  (fun t t'  => t \times t') x.

Fixpoint Tree_value E e:=
  match e with
    | Tbase n  => Vg E (nat_to_B n)
    | Tpowerset e' => \Po (Tree_value E e')
    | Tproduct e' e'' =>
      (Tree_value E e') \times (Tree_value E e'')
end.

Lemma tree_value_prop E:
  [/\ (forall n, natp n -> tree_value E (Tb n) = Vg E n),
```

```
     (forall x, treep x -> tree_value E (Tp x) = \Po (tree_value E x))&
     (forall x y, treep x -> treep y ->
        tree_value E (Tx x y) = (tree_value E x) \times (tree_value E y))].
Lemma Tree_value_compat E e:
  tree_value E (Tree_to_tree e) = Tree_value E e.
Lemma tree_value_extent T E E': treep T ->
  (forall i, i<=c (tree_size T) -> Vg E i = Vg E' i) ->
  tree_value E T = tree_value E' T.

Lemma echelon_of_base_of_tree t E: treep t ->
   echelon_of_base (tree_to_echelon t) E = tree_value E t.
```

Tree evaluation is injective. This means that if we have two trees, T and T′, such that, for every E, T(E) = T′(E) holds, then T = T′. In what follows, we consider a property P, and the assumption H_P that, whenever E satisfies P, then T(E) = T′(E). We can make strong assumptions on P, for, if Q is such that Q $\implies$ P, then H_Q implies H_P. We may for instance assume that E is a list of length m with $n < m$ and $n' < m$, where n and n' are the sizes of T and T′. If the condition fails, at least one of T(E) and T′(E) is not correctly defined (note that E_3 is defined even if E = ∅, but whether or not this is equal to E_4 is left unspecified). We may assume that E has length $1 + \max(n, n')$, for we can always restrict the list to a smaller one. We may assume that the elements of E are 1 and 3. This means that only a finite number of lists have to be checked.

The proof is by induction. There are six cases to consider, since T can be $T_b(n)$, $T_p(x)$, $T_x(x, x')$, likewise for T′ which could be $T_b(m)$, $T_p(y)$, $T_x(y, y')$. The evaluation has the form E_n, $\mathfrak{P}(X)$, $X \times X'$, E_m, $\mathfrak{P}(Y)$, $Y \times Y'$. Assume $E_n = E_m$; in order to get $n = m$, it suffices to allow two distinct values for E. Assume $\mathfrak{P}(X) = \mathfrak{P}(Y)$; then X = Y, since $X \in \mathfrak{P}(X)$, thus $X \in \mathfrak{P}(Y)$, so $X \subset Y$, we conclude by extensionality. Assume $X \times X' = Y \times Y'$. It could be that X′ and Y′ are empty. We exclude this case by assuming E_i non-empty, so that T(E) is non-empty, whatever T. In this case X = Y and X′ = Y′. We now must show that $E_n = \mathfrak{P}(X)$ is absurd. Here we take $E_n = 3$ (our proof relies in the fact that $0 = \emptyset$, $1 = \{0\}$, $2 = \{0,1\}$ and $3 = \{0,1,2\}$; but obviously a set with three elements cannot be a power set, since the cardinal of a power set is a power of two). Note that $\mathfrak{P}(X) \neq Y \times Y'$. If pairs are defined via an axiom (as was the case in earlier versions of Bourbaki), this statement is hard to prove (maybe false with our limited choice of sets for E). However, defining pairs as doubletons ensures that the empty set is not a pair; it belongs to the power set, but not to the product. Finally, we have to exclude the case $E_n = Y \times Y'$. It suffices to take $E_n = 1$ (recall that 1 is the powers et of 0).

```
Lemma tree_val_ne n  E : (forall i, i <c n -> nonempty (Vg E i)) ->
  forall t, treep t -> tree_size t <c n -> nonempty(tree_value E t).

Lemma powerset_injective: injective powerset.
Lemma product_injective A B C D:
  nonempty (C \times D) -> A\times B = C\times D -> A = C /\ B = D.
Lemma not_a_powerset3 x: \3c <> powerset x.
Lemma powerset_not_product x y z:  powerset x <> y \times z.
Lemma not_a_product1 x y: \1c <> x \times y.
```

The proof is a bit long (160 lines) but is straight forward.

```
Definition slist_good n m E :=
  [/\ slistp E, slength E = csucc(cmax n m) &
  forall i, i <c slength E ->  (Vg E i) = \1c \/ (Vg E i) = \3c ].
```

```
Lemma tree_value_injective t1 t2:
  treep t1 -> treep t2 ->
  (forall E, slist_good (tree_size t1) (tree_size t2) E ->
             tree_value E t1 = tree_value E t2) ->
  t1 =  t2.
```

Let S be an echelon converted to a tree list $T_1 \dots T_m$. Then $A_i = T_i(E)$ for every index i. In particular S(E), the last element of the list is $T_m(E)$. If S′ is another echelon such that the last tree T'_k is equal to T_m, then S(E) = S′(E).

```
Lemma tree_value_commmutes E c (f := echelon_value c E)
      (t :=echelon_to_trees c)
      (g := Lg (domain c) (fun i => (tree_value E (Vg t i)))):
   echelon c -> f = g.

Definition echelon_to_tree c := Vl (echelon_to_trees c) (slength c).

Lemma tree_value_commmute_bis E c1 c2:
  echelon c1 -> echelon c2 -> \0c <c slength c1 -> \0c <c slength c2 ->
  echelon_to_tree c1 = echelon_to_tree c2 ->
  echelon_of_base c1 E = echelon_of_base c2 E.
```

Example

Bourbaki considers the scheme (0,1),(0,2),(1,0),(3,0),(2,0),(4,5), and a similar one. This requires to introduce the integer 6 (the length of the list) and some properties (omitted here).

```
Definition card_six := csucc card_five.
Notation "\5c" := card_five.
Notation "\6c" := card_six.
```

We define now the lists a,b, and a,b,c,d,e,f of length 2 and 6, then show that we have two echelons, of size 2.

```
Definition  slist1 a:= Lg \1c (fun z => a).
Definition slist2 a b := Lg \2c (fun z => Yo (z = \0c) a b).
Definition slist6 a b c d e f:=
  Lg \6c (fun z => Yo (z = \0c) a (Yo (z = \1c) b
       (Yo (z = \2c) c (Yo (z = \3c) d (Yo (z = \4c) e f))))).

Lemma slist1_prop a (s := slist1 a):
  slistpl s \1c /\ Vg s \0c = a.
Lemma slist2_prop a b (c:= slist2 a b):
  [/\ slistpl c \2c, Vg c \0c = a & Vg c \1c = b].

Lemma slist6_prop a b c d e f (E:= slist6 a b c d e f):
  [/\ slistpl E \6c, Vg E \0c = a, Vg E \1c = b &
   [/\ Vg E \2c = c, Vg E \3c = d , Vg E \4c = e & Vg E \5c = f ]].


Definition scheme_ex1 := slist6 (J \0c \1c) (J \0c \2c) (J \1c \0c)
                          (J \3c \0c) (J \2c \0c) (J \4c \5c).
```

```
Definition scheme_ex2 := slist6 (J \0c \2c) (J \0c \1c) (J \1c \0c)
                          (J \2c \0c) (J \4c \0c) (J \5c \3c).

Lemma scheme_ex1_ok1 (E := scheme_ex1):
 [/\ echelon E, slength E = \6c, esize E = \2c
  & [/\ Vg E \0c = J \0c \1c, Vg E \1c = J \0c \2c, Vg E \2c = J \1c \0c,
      Vg E \3c = J \3c \0c&  (Vg E \4c =J \2c \0c /\  Vg E \5c =J \4c \5c) ]].

Lemma scheme_ex2_ok1 (E := scheme_ex2):
 [/\ echelon E, slength E = \6c, esize E = \2c
  & [/\ Vg E \0c = J \0c \2c, Vg E \1c = J \0c \1c, Vg E \2c = J \1c \0c,
      Vg E \3c = J \2c \0c&  (Vg E \4c =J \4c \0c /\  Vg E \5c =J \5c \3c) ]].
```

We can convert the echelon into a tree, and compute the value, step by step. We show the full result in the first case. The same can be done for the second example. The lists are different, but the last tree is the same. This means that, whatever the list l, $S_1(l) = S_2(l)$.

```
Definition Tree6 := echelon_to_trees scheme_ex1.

Lemma tree6_1: [/\
     Vg Tree6 \0c = Tree_to_tree (Tbase 0),
     Vg Tree6 \1c = Tree_to_tree (Tbase 1),
     Vg Tree6 \2c = Tree_to_tree (Tpowerset (Tbase 0)),
     Vg Tree6 \3c = Tree_to_tree (Tpowerset (Tpowerset (Tbase 0))) &
    Vg Tree6 \4c = Tree_to_tree (Tpowerset (Tbase 1)) /\
    Vg Tree6 \5c =
       Tree_to_tree
         (Tproduct (Tpowerset (Tpowerset (Tbase 0))) (Tpowerset (Tbase 1)))].

Lemma tree6_2:  echelon_to_tree scheme_ex1 = echelon_to_tree scheme_ex2.
```

Evaluating S_1 on the list U, V is similar. We get $(\mathfrak{P}(\mathfrak{P}(E)) \times \mathfrak{P}(F)$

```
Definition scheme_val1 U V:=
  slist6 U V  (\Po U) (\Po(\Po U)) (\Po V)
   ((\Po(\Po U)) \times (\Po V)).

Lemma echelon_ex1_value U V:
   echelon_value scheme_ex1 (slist2 U V) = scheme_val1 U V.
Lemma echelon_of_base_ex1 U V:
  echelon_of_base scheme_ex1 (slist2 U V) =
  ((\Po(\Po U)) \times (\Po V)).
```

2.2 Canonical Extensions of Mappings

« Let $S = (c_1, c_2, \ldots, c_m)$ be an echelon construction scheme on n term. Let $E_1, \ldots, E_n, E'_1, \ldots E'_n$ be sets (terms in $\mathcal{T}$) and let $f_1, \ldots f_n$ be terms in $\mathcal{T}$ such that the relations "f_i is a mapping of E_i onto E'_i" are theorems in $\mathcal{T}$ for $1 \leq i \leq n$. Let $A_1, \ldots, A_m$ (resp. $A'_1, \ldots, A'_m$) be the echelon construction of scheme S on $E_1, \ldots, E_n$ (resp. $E'_1, \ldots, E'_n$). We define step by step a sequence of m terms $g_1, \ldots, g_m$ such that g_i is a *mapping of* A_i *into* A'_i (for $1 \leq i \leq m$) by the following conditions:

(a) if $c_i = (0, b_i)$, so that $A_i = E_{b_i}$ and $A'_i = E'_{b_i}$, then g_i is the mapping f_{b_i},

(b) if $c_i = (a_i, 0)$, so that $\mathrm{A}_i = \mathfrak{P}(\mathrm{A}_{a_i})$ and $\mathrm{A}'_i = \mathfrak{P}(\mathrm{A}'_{a_i})$ then A_i is the *canonical extension* $\hat{g}_{a_i}$ of g_{a_i} to the set of subsets (Chapter II, §5, no. 1),

(c) if $c_i = (a_i, b_i)$ where $a_i \neq 0$ and $b_i \neq 0$, so that $\mathrm{A}_i = \mathrm{A}_{a_i} \times \mathrm{A}_{b_i}$ and $\mathrm{A}'_i = \mathrm{A}'_{a_i} \times \mathrm{A}'_{b_i}$, then A_i is the *canonical extension* $g_{a_i} \times g_{b_i}$ of g_{a_i} and g_{b_i} to $\mathrm{A}_{a_i} \times \mathrm{A}_{b_i}$ (Chapter II, §3, no. 9).

The last term g_m of this sequence is called the *canonical extension, with scheme* S*, of the mappings* $f_1, \ldots, f_n$, and will be denoted by $\langle f_1, \ldots, f_n \rangle^{\mathrm{S}}$. »

The definition is as above.

```
Definition echelon_extension c f :=
  echelon_recdef (Vl f) extension_to_parts ext_to_prod c.
Definition echelon_can_extension c f :=
  Vl (echelon_extension c f) (slength c).

Lemma echelon_can_extensionE c f:
  echelon_can_extension c f =
  echelon_recdef_last  (Vl f) extension_to_parts ext_to_prod c.

Definition echelon_extension_aux f g a b :=
 Yo (a= \0c) (Vl f b)
    (Yo (b = \0c) (extension_to_parts (Vl g a))
        (ext_to_prod (Vl g a) (Vl g b))).
Definition echelon_extension c f :=
  echelon_recdef c f echelon_extension_aux.
Definition echelon_can_extension c f :=
  Vl (echelon_extension c f) (slength c).

Lemma Eextension_prop1 c f: echelon c ->
  echelon_recdef_prop c f echelon_extension_aux.

Lemma Eextension_prop2 c f (m := slength c)(g :=  echelon_extension c f) :
  echelon c ->
  [/\ fgraph g, domain g = m &
       forall i, i <c m -> Vg g i =
                   echelon_extension_aux  f g (P (Vg c i)) (Q (Vg c i))].
Lemma Eextension_prop c f i (m := slength c)(g :=  echelon_extension c f)
   (n:=  esize c):
  echelon c -> i <c m ->
  let a:= P (Vg c i) in let b := Q (Vg c i) in
  [/\ a = \0c -> [/\ \1c <=c b, b <=c n & Vg g i = (Vl f b)],
      b = \0c -> [/\ \1c <=c a, a <=c i &
          Vg g i = \Pof (Vl g a) ]
    & a <> \0c -> b <> \0c -> [/\ \1c <=c a, a <=c i, \1c <=c b, b <=c i &
   Vg g i = (Vl g a) \ftimes (Vl g b)]].
```

We can define the canonical extension of a tree in a similar but easier way. One has: if c is a scheme, f a family of functions, or whatever, if T is the tree of c, then $\langle f \rangle^c = \mathrm{T}(f)$. More precisely, if c is of length m, and if the trees associated are $\mathrm{T}_1, \ldots, \mathrm{T}_m$, then $g_i = \mathrm{T}_i(f)$ for every i. The important function is $c(f)$, namely g_m, the important tree is $\mathrm{T} = \mathrm{T}_m$, and we have: if c' is another scheme, with tree T′, then if $\mathrm{T} = \mathrm{T}'$, then $\langle f \rangle^c = \langle f \rangle^{c'}$. Recall that, in order to prove $\mathrm{T} = \mathrm{T}'$, it suffices to check $c(\mathrm{E}) = c'(\mathrm{E})$ for a finite family of sets E.

```
Definition tree_extension f x := tree_rec
```

```
  (fun n => Vg f n)
  (fun t => extension_to_parts t)
  (fun t t' => ext_to_prod t t') x.

Lemma tree_extension_prop f:
  [/\ (forall n, natp n -> tree_extension f (Tb n) = Vg f n),
     (forall x, treep x -> tree_extension f (Tp x) =
        \Pof (tree_extension f x))&
     (forall x y, treep x -> treep y ->
        tree_extension f (Tx x y) =
      (tree_extension f x) \ftimes (tree_extension f y))].
Lemma tree_extension_commutes f c
      (t :=echelon_to_trees c)
      (g := Lg (domain c) (fun i => (tree_extension f (Vg t i)))):
   echelon c -> (echelon_extension c f)  = g.
Lemma tree_extension_commmute_bis E c1 c2:
  echelon c1 -> echelon c2 -> \0c <c slength c1 -> \0c <c slength c2 ->
  echelon_to_tree c1 = echelon_to_tree c2 ->
  echelon_can_extension c1 E = echelon_can_extension c2 E.
Lemma can_extension_of_tree t E: treep t ->
   echelon_can_extension (tree_to_echelon t) E = tree_extension E t.
```

Let's now prove some theorems (including CST1, CST2 and CST3). First, we show that, if $f_i \in \mathscr{F}(\mathrm{E}_i;\mathrm{E}'_i)$, then $g_i \in \mathscr{F}(\mathrm{A}_i;\mathrm{A}'_i)$, where the A_i are defined as above. If every f_i is injective, surjective, bijective, identity, so is g_i.

```
Lemma Eextension_prop_fct c E E' f
   (A :=  echelon_value c E)
   (A' :=  echelon_value c E')
   (g :=  echelon_extension c f):
   echelon c ->
   (forall i, i <c (esize c) -> inc (Vg f i) (functions (Vg E i) (Vg E' i))) ->
   forall i,  i <c (slength c) -> inc (Vg g i) (functions (Vg A i) (Vg A' i)).

Lemma Eextension_prop_inj c f (g := echelon_extension c f):
   echelon c ->
   (forall i, i <c (esize c) -> injection (Vg f i)) ->
   (forall i, i <c  (slength c) -> injection (Vg g i)).
Lemma Eextension_prop_surj  c f (g := echelon_extension c f):
   echelon c ->
   (forall i, i <c (esize c) -> surjection (Vg f i)) ->
   (forall i, i <c  (slength c) -> surjection (Vg g i)).
Lemma Eextension_prop_bij_inv c f (g := echelon_extension c f)
   (lif := Lg (esize c) (fun z => inverse_fun (Vg f z)))
   (lig := echelon_extension c lif):
   echelon c ->
   (forall i, i <c (esize c) -> bijection (Vg f i)) ->
   forall i, i <c  (slength c)  ->
        bijection (Vg g i) /\ inverse_fun (Vg g i) = Vg lig  i.
Lemma Eextension_prop_bijset c E E' f
   (A :=  echelon_value c E)
   (A' :=  echelon_value c E')
   (g :=  echelon_extension c f):
   echelon c ->
   (forall i, i <c (esize c) -> inc (Vg f i) (bijections (Vg E i) (Vg E' i))) ->
   forall i,  i <c (slength c) -> inc (Vg g i) (bijections (Vg A i) (Vg A' i)).
```

```
Lemma Eextension_prop_bijsetL c E E' f:
   echelon c ->
   \0c <c slength c ->
   (forall i, i <c esize c -> inc (Vg f i) (bijections (Vg E i) (Vg E' i))) ->
   inc (echelon_can_extension c f)
       (bijections (echelon_of_base c E) (echelon_of_base c E')).
Lemma Eextension_prop_id c f (g :=  echelon_extension c f)
   (is_identity := fun z => z = identity (source z)):
   echelon c ->
   (forall i, i <c esize c -> is_identity (Vg f i)) ->
   forall i, i <c slength c -> is_identity (Vg g i).
Lemma Eextension_prop_idL c f E:
   echelon c ->
   (forall i, i <c esize c -> (Vg f i) = identity (Vg E i)) ->
   \0c <c slength c ->
   (echelon_can_extension c f) = identity (echelon_of_base c E).
```

Consider a third family E''_i, a sequence f'_i such that $f'_i \in \mathscr{F}(\mathrm{E}'_i, \mathrm{E}''_i)$, and construct $g'_i \in \mathscr{F}(\mathrm{A}'_i; \mathrm{A}''_i)$. Since $f'_i \circ f_i \in \mathscr{F}(\mathrm{E}_i, \mathrm{E}''_i)$ one can also define g''_i. It happens that $g''_i = g'_i \circ g_i$. Finally, assume f_i surjective, let f'_i be the inverse of f_i. In this case, g''_i is the identity function, so that g'_i is the inverse of g_i.

```
Lemma Eextension_prop_comp c f f' E E' E'' (m := slength c)
   (n:=  esize c)
   (f'' := Lg n (fun z =>  (Vg f' z) \co (Vg f z)))
   (g := echelon_extension c f)
   (g' := echelon_extension c f')
   (g'':= echelon_extension c f''):
   echelon c ->
   (forall i, i <c n -> inc (Vg f i) (functions (Vg E i) (Vg E' i))) ->
   (forall i, i <c n -> inc (Vg f' i) (functions (Vg E' i) (Vg E'' i))) ->
   forall i, i <c m -> Vg g'' i = (Vg g' i) \co (Vg g i).

Lemma Eextension_prop_composable c f f'
   (g := echelon_extension c f)
   (g' := echelon_extension c f'):
   echelon c ->
   (forall i, i <c esize c ->  (Vg f' i) \coP (Vg f i)) ->
   forall i,i <c (slength c) -> (Vg g' i) \coP (Vg g i).
```

2.3 Transportable relations

« Let $\mathscr{T}$ be a theory which is stronger than the theory of sets, let $x_1, \ldots, x_n$, $s_1 \ldots, s_p$ be distinct letters which are distinct from the constants of $\mathscr{T}$, and let $\mathrm{A}_1 \ldots, \mathrm{A}_m$ be terms in $\mathscr{T}$ in which none of the letters x_i $(1 \leq i \leq n)$ and s_j $(1 \leq j \leq p)$ appears. Let $\mathrm{S}_1, \ldots \mathrm{S}_p$ be echelon construction schemes on $n+m$ terms. Then the relation $\mathrm{T}\{x_1, \ldots, x_n, s_1, \ldots, s_p\}$:

(2.1) “$s_1 \in \mathrm{S}_1(\cdot)$ and $s_2 \in \mathrm{S}_2(\cdot)$ and … and $s_p \in \mathrm{S}_p(\cdot)$”

is called a *typification* of the letters $s_1, \ldots, s_p$.»

« Let $\mathrm{R}\{x_1, \ldots, x_n, s_1, \ldots, s_p\}$ be a relation in $\mathscr{T}$ which contains certain of the letters x_i, s_j (and possibly certain other letters as well). Then R is said to be *transportable (in $\mathscr{T}$) with respect to the typification* T, *the* x_i $(1 \leq i \leq n)$ *being considered as principal base sets and the* A_h

$(1 \leq h \leq m)$ *as auxiliary base sets* if the following condition is satisfied: let $y_1, \ldots, y_n, f_1, \ldots, f_n$ be distinct letters which are distinct from the x_i $(1 \leq i \leq n)$, the s_j $(1 \leq j \leq p)$, the constants of $\mathscr{T}$, and all the letters which appear in R or in the terms A_h $(1 \leq h \leq m)$ and let Id_h $(1 \leq h \leq m)$ denote the identity mapping of A_h onto itself. Then the relation

$$\text{(2.2)} \qquad \text{“T}\{x_1, \ldots, x_n, s_1, \ldots, s_p\} \text{ and } f_1 \in \mathscr{B}(x_1, y_1) \text{ and } \ldots \text{and } f_n \in \mathscr{B}(x_n, y_n)\text{”}$$

implies, in $\mathscr{T}$, the relation

$$\text{(2.3)} \qquad \mathrm{R}\{x_1, \ldots x_n, s_1, \ldots, s_p\} \iff \mathrm{R}\{y_1, \ldots y_n, s'_1, \ldots, s'_p\},$$

where

$$\text{(2.4)} \qquad s'_j = \langle f_1, \ldots, f_n, \mathrm{Id}_1, \ldots, \mathrm{Id}_m \rangle^{\mathrm{S}_j}(s_j) \qquad (1 \leq j \leq p).$$

There is an analogous but simpler definition in the case where there is no auxiliary set.»

For simplicity, we omitted the arguments, in the definition of T, they are, whatever the index, the list $x_1, \ldots, x_n, \mathrm{A}_1, \ldots, \mathrm{A}_m$. We have written $f \in \mathscr{B}(x, y)$ instead of “f is a bijection of x onto y”.

Example: « if $n = p = 2$ and the typification T is “$s_1 \in x_1$ and $s_2 \in x_1$” the relation $s_1 = s_2$ is transportable; the relation $x_1 = x_2$ is not transportable.» There is no auxiliary set in this example ($m = 0$). We have $\mathrm{S}_1(x_1, x_2) = x_1$ and $\mathrm{S}_2(x_1, x_2) = x_1$. From this, we deduce $\mathrm{T}_1 = \mathrm{T}_2 = \mathrm{T}_b(0)$ where T_1 and T_2 are the trees associated to S_1 and S_2.

In this example, transportability means: “$s_1 \in x_1$ and $s_2 \in x_2$ and f_1 is a bijection of x_1 onto y_1 and f_2 is a bijection of x_2 onto y_2” implies $\mathrm{R}(x_1, x_2, s_1, s_2) \iff \mathrm{R}(y_1, y_2, s'_1, s'_2)$. Take for R the relation $x_1 = x_2$. We get $x_1 = x_2 \iff y_1 = y_2$. This is a relation with a lot of free variables, in order to check its validity, we have to quantify over everything. We eliminate s_1 and s_2 by saying that x_1 and x_2 are non-empty, we eliminate f_1 and f_2 by sating that x_1 is equipotent to y_1 and x_2 is equipotent to y_2. For instance, we can take singleton. If $x_1 = \{a_1\}$, $x_2 = \{a_z\}$, etc, we get $a_1 = a_2 \iff b_1 = b_2$, and there are no more conditions; so the relation is false.

Take for R the relation $s_1 = s_2$ and modify our typification to be “$s_1 \in x_1$ and $s_2 \in x_2$” (this makes x_2 useful). We get $s_1 = s_2 \iff s'_1 = s'_2$. We now use the fact that s_1 depends only on T_1, not the unknown S_1, and $\mathrm{T}_1 = \mathrm{T}_b(0)$. This means $s'_1 = f_1(s_1)$, similarly $s'_2 = f_2(s_2)$. Now we get $s_1 = s_2 \iff f_1(s_1) = f_2(s_2)$. Take the same sets as above, $x_1 = \{a_1\}$, etc. Then we get $s_1 = a_1$, etc, thus $a_1 = a_2 \iff b_1 = b_2$. Again the relation is not transportable.

However, from $s_2 \in x_2$ we get $s'_2 = f_1(s_2)$ and the relation becomes $s_1 = s_2 \iff f_1(s_1) = f_1(s_2)$. This holds by injectivity of f_1, and the relation is transportable.

This example explains how to formalise the notion of transportability: we have to quantify over everything: the x_i, s_i, f_i, y_i, the A_i, the sizes n, p, etc. Note that A has a different status than x: it may depend on a parameter t, in particular, since R may depend on t, it may depend on A. As the example shows, the size of S_i should be $\leq n + m$ (it makes no sense to arbitrarily increase the size in order to meet $n + m$.

We first must define the list $x_1, \ldots, x_n, \mathrm{A}_1, \ldots, \mathrm{A}_m$, let's denote it X, that appears in T. This is the concatenation of the lists x and A. We define F, the list $f_1, \ldots, f_n, \mathrm{Id}_1, \ldots, \mathrm{Id}_m$, and show that $\langle \mathrm{F} \rangle^{\mathrm{S}}$ is a bijection $\mathrm{S}(\mathrm{X}) \to \mathrm{S}(\mathrm{Y})$, where Y is like X, with y_i instead of x_i, provided that f_i is a bijection $x_i \to y_i$, and S is a non-trivial echelon of sise $\leq n + m$. In case $n = 0$, $\langle \mathrm{F} \rangle^{\mathrm{S}}$ is the identity function.

```
Definition slist_append x y :=
  let n := slength x in let m := slength y in
  Lg (n +c m) (fun z => Yo (z <c n) (Vg x z) (Vg y (z -c n))).
Definition Typ_with_id f A :=
  slist_append f (Lg (domain A) (fun z => identity (Vg A z))).

Lemma slist_append_list x y: slistp x -> slistp y ->
  slistp (slist_append x y) /\
  slength (slist_append x y) = slength x +c slength y.
Lemma slist_append_val1 x y i: slistp x -> slistp y ->
  i<c slength x -> Vg (slist_append x y) i = Vg  x i.
Lemma slist_append_val2 x y i: slistp x -> slistp y ->
  i<c slength y -> Vg (slist_append x y) ((slength x) +c i) = Vg y i.

Lemma Typ_with_id_prop n S f x y A :
  slistpl x n -> slistpl y n -> slistpl f n ->
  slistp A -> echelon S -> slength S <> \0c -> esize S <=c n +c slength A ->
  (forall i, i <c n -> inc (Vg f i) (bijections (Vg x i) (Vg y i))) ->
   inc (echelon_can_extension S (Typ_with_id f A))
    (bijections (echelon_of_base S (slist_append x A))
                        (echelon_of_base S (slist_append y A)) ).
Lemma Typ_with_id_prop2 n S f x A:
  slistpl x n -> slistpl f n ->
  slistp A -> echelon S -> slength S <> \0c -> esize S <=c n +c slength A ->
  n = \0c ->
    echelon_can_extension S (Typ_with_id f A) =
   identity (echelon_of_base S (slist_append x A)).
```

Let S be the list $S_1,\dots,S_p$. We consider H(n,A,S) to be: n is an integer A and S are lists, for each i, S_i is an echelon of size $\leq n+m$ (where m is the length of A). Now, a typification (n,A,S,x,s) is a bunch of objects such that H(n,A,S) holds, x is a list of length n, s a list of size p, moreover (2.1) holds.

Note: assume that S is empty; so that neither S(X) not $\langle F\rangle^S$ makes sense; in this case we use the empty set instead of S(X). This means that $s \in S(X)$ will be false, and $\langle F\rangle^S$ will not be used. Note also: in what follows, Bourbaki considers only the case $p = 1$. For this reason, we shall give two pairs of definitions: the general case (with a suffix g), and the case $p = 1$, where S and s are not lists.

```
Definition Typ_auxg n A S :=
  [/\ natp n, slistp A, slistp S &
     forall i, inc i (domain S) ->
       echelon (Vg S i) /\ esize (Vg S i) <=c n +c slength A].

Definition Typ_schemeg x A S i:=
  Yo (slength (Vg S i) = \0c) emptyset (echelon_of_base (Vg S i) (slist_append x A)).

Definition Typificationg n A S x s :=
  [/\ Typ_auxg n A S, slistpl x n, slistpl s (slength S) &
   forall i, i <c  slength s -> inc (Vg s i) (Typ_schemeg x A S i)].


Definition Typ_aux n A S :=
  [/\ natp n, slistp A,  echelon S & esize S <=c n +c slength A].

Definition Typ_scheme x A S:=
```

```
  Yo (slength S = \0c) emptyset (echelon_of_base S (slist_append x A)).

Definition Typification n A S x s :=
  [/\ Typ_aux n A S, slistpl x n & inc s (Typ_scheme x A S)].

Definition Typ_hypg n x A S s y f :=
  [/\ Typificationg n A S x s, slistpl y n, slistpl f n &
   forall i, i <c n -> inc (Vg f i) (bijections (Vg x i) (Vg y i))].
Definition Typ_concg (x:Set) A S s y f R :=
  let s' := Lg (domain S)
   (fun i => Vf (echelon_can_extension (Vg S i) (Typ_with_id f A)) (Vg s i))
  in R x s <-> R y s'.
Definition Transportableg n A S R:=
  forall x s y f, Typ_hypg n x A S s y f ->  Typ_concg x A S s y f R.


Definition Typ_hyp n x A S s y f :=
  [/\ Typification n A S x s, slistpl y n, slistpl f n &
   forall i, i <c n -> inc (Vg f i) (bijections (Vg x i) (Vg y i))].
Definition Typ_conc (x:Set) A S s y f R :=
  let s' :=  Vf (echelon_can_extension S (Typ_with_id f A)) s
  in R x s <-> R y s'.
Definition Transportable n A S R:=
  forall x s y f, Typ_hyp n x A S s y f ->  Typ_conc x A S s y f R.
```

The typification is transportable. We have now (a) if $p = 1$, then the two transportability conditions are equivalent. (b) $\mathrm{T}(y, s')$ holds, meaning that the typification is transportable.

```
Lemma transportable_casep1 n A S R (R' := fun x s => R x (Vg s \0c)):
 Transportable n A S R <->
 Transportableg n A (Lg \1c (fun z => S)) R'.
Lemma transportable_aux1 n x A S s y f
  ( s' := Lg (domain S)
   (fun i => Vf (echelon_can_extension (Vg S i) (Typ_with_id f A)) (Vg s i))):
  Typ_hypg n x A S s y f ->
  Typificationg n A S y s'.
Lemma transportable_typificationg n A S:
  Typ_auxg n A S -> Transportableg n A S (Typificationg n A S).
Lemma transportable_typification n A S:
  Typ_aux n A S -> Transportable n A S (Typification n A S).
```

Special case $n = 0$ (no principal base set). Every relation is trxansportable (because $s'_i = s_i$, whatever i). In wat follows, at least one principal base set is required, since otherwise everything becomes trivial.

Special case $p = 0$. « By abuse of language, in the theory of sets $\mathscr{T}_0$ the giving of n district letters $x_1, \ldots, x_n$ (with no typical characterization and no axiom) is considered as a species of structure Σ_0, called *the structure of a set* on the n principle base sets $x_1, \ldots, x_n$.» The next section explains some of the terms. The condition (2.3) simplifies to $\mathrm{R}(x) \iff \mathrm{R}(y)$ (in the code that follows, R has two arguments, the second being the list of the s_i, thus is empty). Note that (2.3) says that there is a bijection $x_i \to y_i$; we rewrite this as $\mathrm{card}(x_i) = \mathrm{card}(y_i)$.

Special case where R is independent of s (see example above with $x_1 = x_2$). Since s is not arbitrary, we assume $\mathrm{R}(x, s) \iff \mathrm{R}(x, s')$ whenever s and s' satisfy $\mathrm{T}(x, s)$ and $\mathrm{T}(x, s')$. The relation becomes transportable when, whenever we have two families such that $\mathrm{card}(x_i) = \mathrm{card}(y_i)$, whenever $\mathrm{T}(x, s)$ and $\mathrm{T}(y, s')$, hold, then $\mathrm{R}(x, s)$ and $\mathrm{R}(y, s')$ are equivalent.

```
Lemma slist_append_empty x: slistp  x -> slist_append x emptyset = x.
Lemma Typ_with_id_empty x: slistp  x -> Typ_with_id x emptyset = x.


Definition equipotent_fam n x y :=
  forall i, i<c n -> cardinal (Vg x i) = cardinal (Vg y i).

Lemma transportable_spec1 n A S R:
   n = \0c -> Typ_auxg n A S ->  Transportableg n A S R.

Lemma transportable_spec_p0 n A S R:
  slength S = \0c ->  Typ_auxg n A S ->
  (Transportableg n A S R <->
    (forall x y, slistpl x n -> slistpl y n -> equipotent_fam n x y ->
      (R x emptyset <-> R y emptyset))).

Lemma transportable_spec3 n A S R:
  (forall x s s', Typificationg n A S x s -> Typificationg n A S x s' ->
                  (R x s <-> R x s')) ->
  Typ_auxg n A S ->
  (Transportableg n A S R <->
    (forall x y, slistpl x n -> slistpl y n -> equipotent_fam n x y ->
       (forall u v, Typificationg n A S x u -> Typificationg n A S y v ->
            (R x u <-> R y v)))).
```

Example. We first say that $(0, n)$ is an echelon of size n. We have $\mathrm{S}(\mathrm{E}) = \mathrm{E}_n$ and $\langle f \rangle^{\mathrm{S}} = f_n$.

```
Definition slist1 a:= Lg \1c (fun z => a).
Lemma slistp_0: slistpl emptyset \0c.
Lemma slist1_prop a (s := slist1 a):
  slistpl s \1c /\ Vg s \0c = a.
Lemma echelon_trivial n (S:= slist1 (J \0c n)): natp n -> n <> \0c ->
   echelon S /\ esize S = n.
Lemma echelon_trivial_value n (S:= slist1 (J \0c n)) E:
  natp n -> n <> \0c ->
  echelon_of_base S E = (Vl E n).
Lemma echelon_trivial_extension n (S:= slist1 (J \0c n)) E:
  natp n -> n <> \0c ->
  echelon_can_extension S E = (Vl E n).
```

We show that the relations considered above are transportable or not.

```
Definition Ex_scheme1 := slist2 (slist1 (J \0c \1c))  (slist1 (J \0c \1c)).
Definition Ex_scheme2 := slist2 (slist1 (J \0c \1c))  (slist1 (J \0c \2c)).

Lemma Ex_typ_aux1: Typ_auxg  \2c emptyset  Ex_scheme1.
Lemma Ex_typ_aux2: Typ_auxg  \2c emptyset  Ex_scheme2.

Lemma Ex_transportable1:
  Transportableg \2c emptyset  Ex_scheme1 (fun _ s => Vg s \0c = Vg s \1c).
Lemma Ex_transportable2:
  ~ Transportableg \2c emptyset  Ex_scheme1 (fun x _ => Vg x \0c = Vg x \1c).
Lemma Ex_transportable3:
  ~ Transportableg \2c emptyset  Ex_scheme2 (fun _ s => Vg s \0c = Vg s \1c).
```

2.4 Species of structures

« Let $\mathcal{T}$ be a theory which is stronger than the theory of sets. A *species of structures* in $\mathcal{T}$ is a text Σ formed of the following assemblies:

(1) a certain number of letters $x_1,\ldots,x_n,s$, distinct from each other and from the constants of $\mathcal{T}$; $x_1,\ldots,x_n$ are calle the *principal base sets* of the species of structure Σ;

(2) a certain number of terms $A_1,\ldots,A_m$ in $\mathcal{T}$ in which none of the letters $x_1,\ldots,x_n,s$ appears, and which are called the *auxiliary base sets* of Σ; Σ possibly contains no auxiliary base sets (but it must contain has at least one principal base set);

(3) a typification $T\{x_1,\ldots,x_n,s\}$:

$$s \in S(x_1,\ldots,x_n,A_1,\ldots A_m),$$

where S is an echelon construction scheme on $n+m$ terms; $T\{x_1,\ldots,x_n,s\}$ is called the *typical characterization of the species of structure* Σ.

(4) a relation $R\{x_1,\ldots,x_n,s\}$ which is *transportable* (in $\mathcal{T}$) with respect to the typification T, the x_i being the principle base sets and the A_h the auxiliary base set; R is called the *axiom* of the species of structures Σ.

The theory $\mathcal{T}_\Sigma$ which has the same axiom schemes as $\mathcal{T}$ and whose explicit axioms are those of $\mathcal{T}$, together with the axiom "T and R", is called the *theory of species of structures* Σ. The constants of $\mathcal{T}_\Sigma$ are therefore the constants of $\mathcal{T}$ and the letters that appear in T or R. One calls *theory of species* Σ the theory $\mathcal{T}_\Sigma$ having the same axiom schemes as $\mathcal{T}$ and whose ecxplicit axioms are that of $\mathcal{T}$ and the axiom "T and R"; the constants of $\mathcal{T}_\Sigma$ are hence the constants of $\mathcal{T}$ and the letters that appear in T or in R.»

« Let $\mathcal{T}'$ be a theory which is stronger than $\mathcal{T}$, and let $E_1,\ldots,E_n,U$ be terms of $\mathcal{T}'$. In the theory $c\mathrm{T}'$, U is said to be a *structure of species* Σ *on the principal base sets* $E_1,\ldots,E_m$ *with* $A_1,\ldots,A_m$ *as auxiliary base sets,* if the relation

$$T\{E_1\ldots,E_n,U\} \text{ and } R\{E_1,\ldots,E_n,U\}$$

is a theorem in $\mathcal{T}'$. When this is so, then for every theorem $B\{x_1,\ldots,x_n,s\}$ in the theory $\mathcal{T}_\Sigma$ the relation $B\{E_1\ldots,E_n,U\}$ is a theorem in $\mathcal{T}'$. In $\mathcal{T}_\Sigma$ the constant s is called the *generic structure of species* Σ.

In the theory $\mathcal{T}'$, the principal base sets $E_1,\ldots,E_n$ are said to be *endowed with the structure* U. Clearly, $U \in S(E_1,\ldots,E_n,A_1,\ldots,A_m)$. The set of elements V of $S(E_1,\ldots,E_n,A_1,\ldots,A_m)$ which satisfy the relation $R\{E_1,\ldots,E_n,V\}$ is therefore the *set of structures of species* $Sigma$ *on* $E_1,\ldots,E_n$, (and it may be empty).»

For us, a species of structure Σ will be: n, A, S, R, where n is an integer, A a list of length m, and S an echelon of length $\leq n+m$, R is transportable. With this definition, $T(x,s)$ becomes equivalent to: x is a length of size n and $s \in S(x)$. Note that x and s are *not* part of Σ.

```
Definition species_of_structure n A S R:=
  [/\ Typ_aux n A S, n <> \0c, slength S <> \0c & Transportable n A S R].

Lemma species_of_structure_typification n A S R x s :
   species_of_structure n A S R ->
   (Typification n A S x s  <->
      (slistpl x n /\ inc s  (echelon_of_base S (slist_append x A)))).
```

We define here a structure of species Σ and the set of all these structures.

```
Definition structure_of_species n A S R E U:=
  species_of_structure n A S R /\ Typification n A S E U.

Definition set_of_structure_of_species  A S R E :=
  Zo (echelon_of_base S (slist_append E A)) (R E).
```

Example. We consider 3 trees, and convert them to echelons. Each tree has size 0, so that each echelon has size 1.

```
Definition Tree_ex1 := (Tp (Tx (Tb \0c)(Tb \0c))).
Definition Tree_ex2 := (Tp (Tx (Tx (Tb \0c)(Tb \0c)) (Tb \0c))).
Definition Tree_ex3 := (Tp (Tp (Tb \0c))).

Definition Echelon_ex1 := tree_to_echelon Tree_ex1.
Definition Echelon_ex2 := tree_to_echelon Tree_ex2.
Definition Echelon_ex3 := tree_to_echelon Tree_ex3.

Definition echelon_s1 c := [/\ echelon c, slength c <> \0c & esize c = \1c].

Lemma tree_echelon_s1 x:
  (treep x /\ tree_size x = \0c) ->  echelon_s1 (tree_to_echelon x).

Lemma Tree_ex1_prop : treep Tree_ex1 /\ tree_size Tree_ex1 = \0c.
Lemma Tree_ex2_prop : treep Tree_ex2 /\ tree_size Tree_ex2 = \0c.
Lemma Tree_ex3_prop : treep Tree_ex3 /\ tree_size Tree_ex3 = \0c.
Lemma Echelon_ex1_prop1: echelon_s1 Echelon_ex1.
Lemma Echelon_ex2_prop1: echelon_s1 Echelon_ex2.
Lemma Echelon_ex3_prop1: echelon_s1 Echelon_ex3.


Lemma Echelon_value_ex1 E (A := Vg E \0c):
  echelon_of_base Echelon_ex1 E = \Po (A \times A) /\
  echelon_can_extension Echelon_ex1 E =  \Pof (A \fimes A).
Lemma Echelon_value_ex2 E (A := Vg E \0c):
  echelon_of_base Echelon_ex2 E = \Po ((A \times A) \times A) /\
  echelon_can_extension Echelon_ex2 E = \Pof ((A \ftimes A) \ftimes A).
Lemma Echelon_value_ex3 E (A := Vg E \0c):
  echelon_of_base Echelon_ex3 E = \Po (\Po A) /\
  echelon_can_extension Echelon_ex3 E = \Pof (\Pof A).
```

Needs to be formalized; Work in progress.

Chapter 3

Inverse limits and direct limits

In this chapter we implement §7 of chapter III (Ordered Sets, Cardinals, Integers) of Book I (Theory of Sets) of the work of Bourbaki [3]. We consider a family of sets $(E_i)_{i\in I}$ and a family of functions $E_i \to E_j$; in case we have a function $f : E_i \to E_j$ and a function $g : E_j \to E_k$, then $g \circ f$ is the function $E_i \to E_k$. The function f exists when $i \leq j$, so that we require $\leq$ to be transitive. We also assume that the function $E_i \to E_i$ is the identity function of E_i, this implies that $\leq$ is reflexive on I, thus is a preorder on I. One can take the product or the quotient, and obtain what is called the "inverse" or "direct" limit. We shall use here the alternate names "projective limit" and "inductive limit" (unless when quoting Bourbaki).

3.1 Inverse limits

« Let I be a preordered set and let $(E_\alpha)_{\alpha\in I}$ be a family of sets indexed by I. For each pair (α,β) of elements of I such that $\alpha \leq \beta$, let $f_{\alpha\beta}$ be a mapping of E_β into E_α. Suppose that the $f_{\alpha\beta}$ satisfy the following conditions:

(LP_I) The relations $\alpha \leq \beta \leq \gamma$ imply $f_{\alpha\gamma} = f_{\alpha\beta} \circ f_{\beta\gamma}$.
(LP_{II}) For each $\alpha \in I$, $f_{\alpha\alpha}$ is the identity mapping of E_α.

By abuse of language, the pair $((E_\alpha),(f_{\alpha\beta}))$ (usually denoted by $(E_\alpha, f_{\alpha\beta})$) is called an *inverse system of sets*, relative to the index set I. »

It is implicit in (LP_I) that the three indices belong to I. So, if r is the graph of $\leq$ on I, by assumption, r is a preorder on I, and it makes sense to consider the family f_{ij} with domain r. This means that $((E_i)_{i\in I},(f_{ij})_{ij\in r})$ is a pair of sets. For simplicity, we shall add to this pair the domains of E and f (i.e., I and r). So, a (projective or inductive) system will be a quadruple, together with the conditions (LP_I), (LP_{II}), the fact that f_{ij} is a function, and that r is a preorder on I. Packing the sets and the relation in a single object S will make lemmas shorter, but S is no more a set: we loose extensionality (two systems with the same data are not always equal), and we cannot consider families of systems (see Exercise 1 for instance).

```
Record projective_system: Type := ProjectiveSystem {
  psE : Set;
  psI : Set;
  psr : Set;
  psf : Set;
  ps_preorder_r: preorder psr;
  ps_substrate_r: substrate psr = psI;
```

```
  ps_fgraph_E: fgraph psE;
  ps_domain_E: domain psE = psI;
  ps_fgraph_f: fgraph psf;
  ps_domain_f: domain psf = psr;
  ps_function_f:
    forall i, inc i psr ->
              function_prop (Vg psf i) (Vg psE (Q i)) (Vg psE (P i));
  ps_compose_f: forall i j k, gle psr i j -> gle psr j k ->
                  Vg psf (J i j) \co  Vg psf (J j k) = Vg psf (J i k);
  ps_identity_f: forall i, inc i psI -> Vg psf (J i i) = identity (Vg psE i)
}.
```

We define the notion of "having the same data"; this is an equivalence relation, and two systems having the same data can be considered equal. Creating a system is not trivial; in general we shall *not* show the value of every field; for this reason, after each definition we shall give a statement of the form: the system so defined is a system on E, I, r and f. We also define the notion of "having the same index" (this obviously means the same r, it implies the same I).

```
Definition projective_system_on S E I r f :=
  [/\  psE S = E, psr S = I, psr S = r & psf S = f].

Definition prl_same_data S S' :=
 [/\ psE S = psE S', psr S = psr S' & psf S = psf S'].

Definition prl_same_index S S' := psr S = psr S'.

Lemma prl_same_dataS S S':
  prl_same_data S S' -> prl_same_data S' S.
Lemma prl_same_dataT S S' S'' :
 prl_same_data S S' -> prl_same_data S' S'' -> prl_same_data S S''.
Lemma prl_same_index_same_I S S':
  prl_same_index S S' -> psI S = psI S'.
```

We start with trivialities.

```
Lemma prl_prop0 S i j: gle (psr S) i j -> inc i (psI S) /\ inc j (psI S).
Lemma prl_prop1 S i: inc i (psI S) -> inc (J i i) (psr S).

Lemma prl_prop2 S i j k: gle  (psr S) i j -> gle (psr S) j k ->
   Vg (psf S) (J i j) \coP Vg (psf S) (J j k).
Lemma prl_prop3 S y i j k (f:= psf S):
  gle (psr S) i j -> gle (psr S) j k -> inc y (Vg (psE S) k) ->
  Vf (Vg f (J i j))  (Vf (Vg f (J j k)) y) = Vf (Vg f (J i k)) y.
Lemma prl_prop4 S i j: gle (psr S) i j ->
  function_prop (Vg (psf S) (J i j)) (Vg (psE S) j) (Vg (psE S) i).
Lemma prl_prop5 S i x: inc i (psI S) -> inc x (Vg (psE S) i) ->
   Vf (Vg (psf S) (J i i)) x = x.
```

« Let $G = \prod_{\alpha\in I} E_\alpha$ be the *product* of the family of sets $(E_\alpha)_{\alpha\in I}$, and let E denote the subset of G consisting in all x which satisfy *each* of the relations

$$\mathrm{pr}_\alpha x = f_{\alpha\beta}(\mathrm{pr}_\beta x) \tag{3.1}$$

for each pair of indices (α,β) such that $\alpha \le \beta$. E is said to be the *inverse limit of the family* $(E_\alpha)_{\alpha\in I}$ *with respect to the family of mappings* $(f_{\alpha\beta})$, and we write $E = \varprojlim(E_\alpha, f_{\alpha\beta})$ or simply

$E = \varprojlim E_\alpha$ [...] The *restriction* f_α of the projection pr_α to E is called the *canonical mapping* of E into E_α, and we have the relation

$$f_\alpha = f_{\alpha\beta} \circ f_\beta \tag{3.2}$$

whenever $\alpha \leq \beta$. »

A priori, pr_i is the function defined on the product G, with values in E_i that extracts the component of index i. This means that $\mathrm{pr}_i x$ (note that parentheses are omitted) is x_i. So the conditions (3.1) and (3.2) just say $x_i = f_{ij}(x_j)$. If I is empty, then $\varprojlim E$ has a single element, the empty sequence.

```
Definition projective_limit S:=
  Zo (productb (psE S)) (fun x => forall i j, gle (psr S) i j
                               -> (Vg x i) = Vf (Vg (psf S) (J i j)) (Vg x j)).

Definition prl_can_fun S i  :=
  Lf (fun x => Vg x i) (projective_limit S) (Vg (psE S) i).

Lemma prl_limitP S x:
  inc x (projective_limit S) <->
  [/\ fgraph x, domain x = psI S,
     forall i, inc i (psI S) -> inc (Vg x i) (Vg (psE S) i)&
   forall i j, gle (psr S) i j -> Vg x i = Vf (Vg (psf S) (J i j)) (Vg x j)].
Lemma prl_proj_ax S i: inc i (psI S) ->
  lf_axiom (fun x => Vg x i) (projective_limit S) (Vg (psE S) i).
Lemma prl_proj_ev S i x: inc i (psI S) -> inc x (projective_limit S) ->
   Vf (prl_can_fun S i) x = Vg x i.
Lemma prl_can_fun_fp S i: inc i (psI S) ->
  function_prop (prl_can_fun S i) (projective_limit S) (Vg (psE S) i).
Lemma prl_can_fun_prop S i j (f := psf S)
      (fi := prl_can_fun S i) (fj := prl_can_fun S j):
   gle (psr S) i j ->
   (Vg f (J i j) \coP fj) /\  fi = (Vg f (J i j)) \co fj.

Lemma projective_limit_Iv S S':
  prl_same_data S S' -> projective_limit S = projective_limit S'.
Lemma prl_can_fun_Iv S S' i: prl_same_data S S' ->
  prl_can_fun S i = prl_can_fun S' i.
Lemma prl_trivial S: psI S = emptyset ->
  projective_limit S = singleton emptyset.
```

Example 1. Assume that r is the diagonal of I, so that $i \leq j$ means $i = j$. Here ($\mathrm{LP_{II}}$) says that all the functions are the identity. In this case, $\varprojlim S$ is the product of the E_i.

```
Section Example1.
Variables E I: Set.
Hypotheses (fgE:fgraph E) (dE: domain E = I).

Definition prl_exa1_system:projective_system.
Lemma prl_exa1_prop:
  projective_system_on prl_exa1_system
  E I (diagonal I)  (Lg (diagonal I) (fun z => identity (Vg E (P z)))).
Lemma prl_exa1_prop2: projective_limit prl_exa1_system = productb E.
End Example1.
```

Example 2. Assume that I is right directed, (E_i) is the constant family with value F, so that the product is F^I, and f_{ij} is the identity of F. Then $\varprojlim S$ is the diagonal of F^I (the set of constant sequences). Proof: whatever i and j there is k such that $i \leq k$ and $j \leq k$. We get $x_i = x_k$ and $x_j = x_k$ so that $x_i = x_j$.

Of course, I right directed has to be interpreted as: the preorder r is right directed. We prefer a mixed statement of the form: whenever x and y belong to I, there is $z \in I$ such that $x \leq z$ and $y \leq z$.

```
Definition right_directed_on r I :=
  forall x y, inc x I -> inc y I ->
              exists z, [/\ inc z I, gle r x z & gle r y z].

Section Example2.
Variables I r F: Set.
Hypotheses (or:preorder r) (sr: substrate r = I)
(rdr:right_directed_on r I).
Definition prl_exa2_system: projective_system.
Lemma prl_exa2_prop:
  projective_system_on prl_exa2_system
  (cst_graph I F) I r (cst_graph r (identity F)).
Lemma prl_exa2_prop2: projective_limit prl_exa2_system = diagonal_graphp F I.
End Example2.
```

Example 3. Let $E_i = I = \mathbf{N}$ and $f_{ij}(x) = x + (j - i)$. This gives a projective system. Let x be in the projective limit. Then $x_i = x_j + (j - i)$. In particular, if $i = 0$, we get $x_0 = x_j + j$. We get a contradiction if $j = x_0 + 1$. So the limit is empty. (For a less trivial example, see exercise 4.)

```
Section Example3.
Let r := Nat_order.
Let f := fun i j => Lf (fun x => x +c (j -c i)) Nat Nat.
Let ffam := Lg r (fun p => f (P p) (Q p)).
Let Efam:= Lg Nat  (fun i => Nat).

Lemma prl_exa3_prop1:
  [/\ preorder r, substrate r = Nat &
      forall i j, gle r i j <-> [/\ natp i, natp j & i <=c j]].
Lemma prl_exa3_prop2 p: inc p r ->
  [/\ natp (P p), natp (Q p) & gle r (P p) (Q p)].
Lemma prl_exa3_prop3 i j: gle r i j -> lf_axiom (csum2^~ (j -c i)) Nat Nat.
Lemma prl_exa3_prop4 i j: gle r i j -> function_prop (f i j) Nat Nat.
Definition prl_exa3_system: projective_system.

Lemma prl_exa3_prop5: projective_system_on prl_exa3_system Efam Nat r ffam.
Lemma prl_exa3_prop6 x: inc x (projective_limit  prl_exa3_system) ->
   (natp (Vg x \0c) /\ forall i, natp i -> Vg x \0c = (Vg x i) +c i).
Lemma prl_exa3_prop7: projective_limit  prl_exa3_system = emptyset.
```

Restrictions. Let J be a subset of I. The set of all (i, j) such that $i \in J$, $j \in J$ and $i \leq j$, ordered by $\leq$, is the preorder induced by r on J; denote it r'. Let E' and f' be the restrictions of E and f to I and r, respectively. Then (E', J, r', f') is a projective system, it is said to be obtained by *restricting* the index set to J. Note that `prl_restr` take as argument the proof that $J \subset I$, from which S and J are deduced.

Note: if we have two systems with the same data (in particular the same I), two proofs that $J \subset I$, then the restrictions have the same data.

```
Definition prl_restr S J (H: sub J (psI S)) : projective_system.
Definition projective_limit_restr S J (H: sub J (psI S)):=
  projective_limit (prl_restr H).

Lemma prl_restr_prop S J (H: sub J (psI S)):
  projective_system_on (prl_restr H)
         (restr (psE S) J) J (induced_order (psr S) J)
         (restr (psf S) (induced_order (psr S) J)).
Lemma prl_restr_Iv2 S S' J (h1: sub J (psI S))(h2: sub J (psI S')):
  prl_same_data S S' ->
  prl_same_data (prl_restr h1) (prl_restr h2).
```

« For each $x \in E$ the element

$$g(x) = (f_\alpha(x))_{\alpha\in J} \tag{3.3}$$

belongs to E' by virtue of (3.2). The mapping $g : E \to E'$ so defined is called *canonical.* »

Here E is $\varprojlim S$, E' is $\varprojlim S'$, the projective limit of the restricted system. Note that $g(x)$ is the functional graph, defined on J that maps i to $f_i(x)$. Since f_i is the restriction of pr_i, we have $f_i(x) = \mathrm{pr}_i x$, and, as mentioned above, $f_i(x) = x_i$. So $g(x)$ is just the restriction of x to J.

```
Definition prl_restr_canonical S J (H: sub J (psI S)):=
  Lf (restr^~J) (projective_limit S) (projective_limit_restr H).

Lemma prl_restr_canonical_ax S J (H: sub J (psI S)) :
   lf_axiom (restr^~J) (projective_limit S) (projective_limit_restr H).

Lemma prl_restr_canonical_fp S J (H: sub J (psI S)):
  function_prop (prl_restr_canonical H) (projective_limit S)
    (projective_limit_restr H).

Lemma prl_restr_canonical_fun_ev S J (H: sub J (psI S)) x:
 inc x (projective_limit S) -> Vf (prl_restr_canonical H)  x = restr x J.
Lemma prl_restr_canonical_fun_ev2 S J (H: sub J (psI S)) x j:
  inc x (projective_limit S) -> inc j J ->
  Vg (Vf (prl_restr_canonical H) x) j = Vg x j.
```

« If J' is a subset of J, and E'' the inverse limit of the family $(E_\alpha)_{\alpha\in J'}$, and if $g' : E' \to E''$ and $g'' : E \to E''$ are the canonical mappings, then by definition we have

$$g'' = g' \circ g. \tag{3.4}$$ »

In the code that follows, we use K and K' instead of J and J'. Assume $K' \subset K \subset I$. We define S' to be S restricted to K, and S'' to be S' restricted to K' (note that H_2, a proof of $K' \subset K$, is also a proof of $K \subset I(S')$, where $I(S')$ is the index set of S'). We can also consider S'' as the restriction of S to K' (since $K'' \subset I$). These two systems have the same index set, thus the same order, thus the same data.

```
Lemma projective_limit_restr_double_Iv S K K'
  (H1:sub K (psI S)) (H2: sub K' K):
  prl_same_data (prl_restr (H2: sub K' (psI (prl_restr H1))))
    (prl_restr (sub_trans H2 H1)).

Lemma projective_limit_restr_double S K K'
```

```
  (H1:sub K (psI S)) (H2: sub K' K):
  projective_limit_restr (H2: sub K' (psI (prl_restr H1))) =
   projective_limit_restr (sub_trans H2 H1).

Lemma prl_restr_canonical_comp S K K'
      (H1:sub K (psI S)) (H2: sub K' K)
      (g :=  prl_restr_canonical H1)
      (g' :=  prl_restr_canonical (H2: sub K' (psI (prl_restr H1))))
      (g'' :=  prl_restr_canonical (sub_trans H2 H1)):
   g' \coP  g /\ g' \co g = g''.
```

3.2 Inverse systems of mappings

Proposition 1. « Let I be an ordered set, let $(E_\alpha, f_{\alpha\beta})$ be an inverse system of sets relative to I, let $E = \varprojlim E_\alpha$ be its inverse limit, and for each $\alpha \in I$ let

$$f_\alpha : E \to E_\alpha$$

be the canonical mapping. For each $\alpha \in I$, let u_α be a mapping of a set F into E_α such that

$$f_{\alpha\beta} \circ u_\beta = u_\alpha \qquad \text{whenever } \alpha \le \beta. \tag{3.5}$$

Then (a) there exists a unique mapping u of F into E such that

$$u_\alpha = f_\alpha \circ u \qquad \text{for all } \alpha \in I; \tag{3.6}$$

(b) the mapping u is injective if and only if, for each pair of distinct elements y, z of F, there exists $\alpha \in I$ such that $u_\alpha(y) \neq u_\alpha(z)$. »

```
Definition prl_map_compat S u F:=
  [/\ fgraph u, domain u = psI S,
   forall i, inc i (domain u) -> function_prop (Vg u i) F (Vg (psE S) i) &
   forall i j, gle (psr S) i j -> Vg (psf S) (J i j) \co (Vg u j) = Vg u i].

Definition prl_map_property S u F g:=
  function_prop g F (projective_limit S) /\
  forall i, inc i (domain u) -> (Vg u i) = (prl_can_fun S i) \co g.

Definition prl_map_val S u :=
  fun y => Lg (psI S) (fun i => Vf (Vg u i) y).

Definition projective_map S u F :=
  Lf (prl_map_val S u) F (projective_limit S).
```

Equation (3.6) says that if $x = u(y)$, then $x_i = u_i(y)$, so that $u(y)$ is $i \mapsto u_i(y)$. This gives uniqueness and a definition of u.

```
Lemma prl_map_property_res1 S u F g i x:
  prl_map_compat S u F -> prl_map_property S u F g ->
  inc i (psI S) -> inc x F -> Vf (Vg u i) x = Vg (Vf g x) i.
Lemma prl_map_unique S u F g g':
  prl_map_compat S u F ->
  prl_map_property S u F g -> prl_map_property S u F g' ->
```

```
  g = g'.
Lemma prl_map_ax S u F :
  prl_map_compat S u F ->
  lf_axiom (prl_map_val S u) F (projective_limit S).

Lemma projective_map_ev S u F x i:
  prl_map_compat S u F -> inc x F -> inc i (psI S) ->
  Vg (Vf (projective_map S u F) x) i = Vf (Vg u i) x.
Lemma prl_map_prop S u F :
  prl_map_compat S u F ->
  prl_map_property S u F (projective_map S u F).
Lemma prl_map_inj S u F :
  prl_map_compat S u F ->
  (injection (projective_map S u F) <->
   (forall y z, inc y F -> inc z F -> y <> z
           -> exists2 i, inc i (psI S) & (Vf (Vg u i) y <> Vf (Vg u i) z))).
```

Corollary 1. « Let $(E_\alpha, f_{\alpha\beta})$ and $(F_\alpha, g_{\alpha\beta})$ be two inverse systems of sets relative to the same index set I; let $E = \varprojlim E_\alpha$, $F = \varprojlim F_\alpha$, and let f_α (resp. g_α) be the canonical mapping of E into E_α (resp. of F into F_α) for each $\alpha \in I$. For each $\alpha \in I$, let u_α be a mapping of E_α into F_α such that

$$u_\alpha \circ f_{\alpha\beta} = g_{\alpha\beta} \circ u_\beta \qquad \text{whenever } \alpha \leq \beta. \tag{3.7}$$

Then there exists a unique mapping $u : E \to F$ such that

$$u_\alpha \circ f_\alpha = g_\alpha \circ u \qquad \text{whenever } \alpha \in I. \tag{3.8}$$

»

Bourbaki uses the following two commutative diagrams instead of formulas

$$\begin{array}{ccc} E_j & \xrightarrow{u_j} & F_j \\ {\scriptstyle f_{ij}}\downarrow & & \downarrow{\scriptstyle g_{ij}} \\ E_i & \xrightarrow[u_i]{} & F_i \end{array} \qquad\qquad \begin{array}{ccc} E & \xrightarrow{u} & F \\ {\scriptstyle f_i}\downarrow & & \downarrow{\scriptstyle g_i} \\ E_i & \xrightarrow[u_i]{} & F_i \end{array}$$

A family satisfying the assumptions is called a *projective systems of mappings*; the mapping u is called the *projective limit* of the family, and denoted by $\varprojlim u_\alpha$. The idea is to apply Proposition 1 to $v_i = u_i \circ f_i$ and the second system.

```
Definition prl_map2_compat S S' u:=
  [/\ fgraph u, domain u = psI S,
   forall i, inc i (psI S) ->
       function_prop (Vg u i) (Vg (psE S) i) (Vg (psE S') i) &
   forall i j, gle (psr S) i j -> (Vg u i) \co (Vg (psf S) (J i j)) =
         (Vg (psf S') (J i j)) \co (Vg u j)].

Definition prl_map2_property S S' u g:=
  function_prop g (projective_limit S) (projective_limit S')
  /\ forall i, inc i (psI S) ->
   (Vg u i) \co (prl_can_fun S i) = (prl_can_fun S') i \co g.

Definition prl_map2_aux S u :=
  Lg (psI S) (fun i => (Vg u i) \co (prl_can_fun S i)).
Definition projective_limit_fun S S' u :=
   projective_map S' (prl_map2_aux S u) (projective_limit S).
```

Corollary 2 says that if we have three systems, S, S′ and S″, then

$$\varprojlim(v_\alpha \circ u_\alpha) = (\varprojlim v_\alpha) \circ (\varprojlim u_\alpha).$$

```
Lemma prl_projective_limit_fun_IV2 S1 S2 x S1' S2' x':
  prl_same_data S1 S1' ->  prl_same_data S2 S2' -> x = x' ->
  projective_limit_fun S1 S2 x = projective_limit_fun S1' S2' x'.

Lemma prl_map2_prop1 S S' u:
  prl_same_index S S' ->  prl_map2_compat S S' u ->
  prl_map_compat S' (prl_map2_aux S u) (projective_limit S).

Lemma prl_map2_prop2 S u i t:
  inc i (psI S) -> inc t (projective_limit S) ->
  function (Vg u i) ->  source (Vg u i) = Vg (psE S) i ->
  Vf (Vg (prl_map2_aux S u) i) t = Vf (Vg u i) (Vg t i).

Lemma prl_map2_unique S S' u g g':
  prl_same_index S S' -> prl_map2_compat S S' u->
  prl_map2_property S S' u g -> prl_map2_property S S' u  g' -> g = g'.

Lemma prl_map2_prop S S' u (g := projective_limit_fun S S' u):
  prl_same_index S S' -> prl_map2_compat S S' u ->
  prl_map2_property S S' u g.

Lemma prl_map2_compat_aux S S' u x i j:
  prl_same_index S S' ->  prl_map2_compat S S' u ->
  inc x (projective_limit S) -> gle (psr S) i j ->
  Vf (Vg u i) (Vg x i) = Vf (Vg (psf S') (J i j)) (Vf (Vg u j) (Vg x j)).

Lemma prl_map_val_aux2 S S' u  (Ha :prl_same_index S S')
      (Hu: prl_map2_compat S S' u)
      (f := projective_limit_fun S S' u) i x:
   inc i (psI S) ->  inc x (projective_limit S) ->
   (Vf (Vg u i) (Vg x i)) = (Vg (Vf f x) i).

Lemma prl_map2_compose S S' S'' u v (F := projective_limit_fun)
  (w:= Lg (psI S) (fun i => (Vg v i) \co (Vg u i))) :
  prl_same_index S S' ->  prl_same_index S' S'' ->
  prl_map2_compat S S' u -> prl_map2_compat S' S'' v ->
  prl_map2_compat S S'' w /\
  F S S'' w = F S' S'' v \co F S S' u.


Lemma prl_map2_prop3 S S' u  (Ha :prl_same_index S S')
      (Hu: prl_map2_compat S S' u)
      (f := projective_limit_fun S S' u):
  function_prop f (projective_limit S)  (projective_limit S') /\
  forall i x,
   inc i (psI S) ->  inc x (projective_limit S) ->
   (Vf (Vg u i) (Vg x i)) = (Vg (Vf f x) i).
```

Product of systems. Consider two systems S and S′ on the same index set I. Recall that if f and g are functions $E \to E'$ and $F \to F'$, then that $f \times g$ is the function $E \times E' \to F \times F'$ that maps (x, x') to $(f(x), g(x'))$. This allows us to define the product of S and S′.

```
Definition prl_product_E S S':=
```

```
   Lg (psI S) (fun i => (Vg (psE S) i) \times (Vg (psE S') i)).
Definition prl_product_f S S' :=
  Lg (psr S) (fun i => (Vg (psf S) i) \ftimes (Vg (psf S') i)).

Definition prl_system_product S S' (sd: prl_same_index S S'): projective_system.
Lemma prl_system_product_prop S S' (sd: prl_same_index S S'):
  projective_system_on (prl_system_product sd)
   (prl_product_E S S') (psI S) (psr S) (prl_product_f S S').
```

Consider the canonical mappings, and their product $f_i \times f'_i$. These functions are compatible with the projections, so that $\varprojlim(f_i \times f'_i)$ is a function $\varprojlim S \times \varprojlim S' \to \varprojlim(S \times S')$. It is not hard to see that it is a bijection.

```
Definition prl_product_can_fun S S' :=
  Lg (psI S) (fun i => (prl_can_fun S i)
                        \ftimes (prl_can_fun S' i)).


Lemma prl_product_can_fun_compat  S S' (sd: prl_same_index S S'):
  prl_map_compat (prl_system_product sd) (prl_product_can_fun S S')
  ((projective_limit S) \times (projective_limit S')).

Lemma prl_product_can_fun_bij  S S' (sd: prl_same_index S S')
  (E:=  projective_limit S) (E' := projective_limit S')
  (f:= projective_map (prl_system_product sd) (prl_product_can_fun S S')
        (E \times E')):
  bijection_prop f
      (E \times E') (projective_limit (prl_system_product sd)).
```

Restricting the sets. « Let $(E_\alpha, f_{\alpha\beta})$ be an inverse system of sets, and for each $\alpha \in I$, let M_α be a subset of E_α. If $f_{\alpha\beta}\langle M_\beta\rangle \subset M_\alpha$ whenever $\alpha \leq \beta$, the M_α are said to form *an inverse system of subsets* of the E_α. Let $g_{\alpha\beta}$ be the mapping of M_β into M_α (where $\alpha \leq \beta$) whose graph is the same as that of the restriction of $f_{\alpha\beta}$ to M_β. Then it is clear that $(M_\alpha, g_{\alpha\beta})$ is an inverse system of sets and that

$$\varprojlim M_\alpha = (\varprojlim E_\alpha) \cap \prod_{\alpha\in I} M_\alpha. \text{»} \tag{3.9}$$

```
Definition prl_subfam_hyp S M:=
  [/\ fgraph M, domain M = psI S,
   forall i, inc i (psI S) -> sub (Vg M i) (Vg (psE S) i) &
   forall i j, gle (psr S) i j ->
     sub (Vfs (Vg (psf S) (J i j)) (Vg M j)) (Vg M i) ].

Definition prl_subfam_fct S M :=
  Lg (psr S) (fun z => restriction2 (Vg (psf S) z) (Vg M (Q z)) (Vg M (P z))).

Lemma prl_subfam_prop1 S M (g := prl_subfam_fct S M):
  prl_subfam_hyp S M ->
  [/\
   forall z, inc z (psr S) ->
      restriction2_axioms (Vg (psf S) z) (Vg M (Q z)) (Vg M (P z)),
   forall i j x, gle (psr S) i j -> inc x (Vg M j) ->
     Vf (Vg g (J i j)) x = Vf (Vg (psf S) (J i j)) x,
   forall i, inc i (psr S)-> function_prop (Vg g i) (Vg M (Q i)) (Vg M (P i)),
```

```
   forall i j k, gle (psr S) i j -> gle (psr S) j k ->
                 Vg g (J i j) \co  Vg g (J j k) = Vg g (J i k) &
   forall i, inc i (psI S) -> Vg g (J i i) = identity (Vg M i)].

Definition projective_system_subsets
    S M (H:prl_subfam_hyp S M) : projective_system.

Lemma prl_subsets_prop S  M (H:prl_subfam_hyp S M) :
  projective_system_on (projective_system_subsets H)
     M  (psI S)  (psr S) (prl_subfam_fct S M).

Lemma prl_subsets_prop_Iv S M
      (H:prl_subfam_hyp S M) (H':prl_subfam_hyp S M) :
  prl_same_data (projective_system_subsets H) (projective_system_subsets H').
Lemma prl_subsets_prop_I2v S S' M
      (H:prl_subfam_hyp S M) (H':prl_subfam_hyp S' M) :
  prl_same_data S S' ->
  prl_same_data (projective_system_subsets H) (projective_system_subsets H').

Lemma prl_subsets_prop2 S  M (H:prl_subfam_hyp S M):
   projective_limit(projective_system_subsets H) =
   projective_limit S \cap (productb M).
```

Proposition 2. « Let $(E_\alpha, f_{\alpha\beta})$ and $(E'_\alpha, f'_{\alpha\beta})$ be two inverse systems of sets relative to I, and let u_α be a mapping of E_α into E'_α for each $\alpha \in I$, such that the u_α form an inverse system of mappings. Let $u = \varprojlim u_\alpha$. Then for each $x' = (x'_\alpha) \in E' = \varprojlim E'_\alpha$, the $\overset{-1}{u}_\alpha(x'_\alpha)$ form an inverse system of subsets of the E_α, and $\overset{-1}{u}(x') = \varprojlim \overset{-1}{u}_\alpha(x'_\alpha)$. » Note that $\overset{-1}{u}_i(x_i)$ is the set of all $z \in E_i$ such that $u_i(z) = x_i$.

```
Definition prl_invim_set u x :=
  Lg (domain u) (fun i => (Vfi1 (Vg u i) (Vg x i))).

Lemma prl_inv_hyp S S' u x:
  prl_same_index S S' ->  prl_map2_compat S S' u ->
  inc x (projective_limit S') ->
  prl_subfam_hyp S (prl_invim_set u x).

Lemma prl_inv_hyp_prop S S' u x
  (Hsb: prl_same_index S S') (Hc:  prl_map2_compat S S' u)
  (Hx: inc x (projective_limit S')):
  (Vfi1 (projective_limit_fun S S' u) x) =
  projective_limit (projective_system_subsets (prl_inv_hyp  Hsb Hc Hx)).
```

Corollary. If every u_i is injective (resp. bijective) so is u. For injectivity, assume $u(a) = u(b)$, and consider $x' = u(a)$. Then b is in the projective limit, each b_i is in $u_i^{-1}(u(a)_i)$, $u_i(b_i) = u_i(a_i)$ so that $b_i = a_i$. Assume every u_i bijective; then $u_i^{-1}(x'_i)$ is a singleton $\{x_i\}$, if x is the family of x_i, then $u(x) = x'$. [For surjectivity, see Exercise 4].

```
Lemma prl_inv_hyp_prop1 S S' u:
  prl_same_index S S' ->  prl_map2_compat S S' u ->
  (forall i, inc i (psI S) -> injection (Vg u i)) ->
  injection (projective_limit_fun S S' u).
```

```
Lemma prl_inv_hyp_prop2 S S' u:
  prl_same_index S S' ->  prl_map2_compat S S' u ->
  (forall i, inc i (psI S) -> bijection (Vg u i)) ->
  bijection (projective_limit_fun S S' u).
```

Direct image. With the same notations as above, consider the sets $M_i = u_i\langle E_i\rangle$ (this is the image of u_i). This family of sets form a projective family of subsets of M_i. If u is the projective limit of u_i, then the image of u is a subset of the projective limit of the M_i. We have not always equality: for instance, assume u_i surjective, so that $M_i = E'_i$ and $\varprojlim M_i = \varprojlim E'_i$. It may happen that u is not surjective.

```
Definition prl_dirim_set u :=
  Lg (domain u) (fun i => Imf (Vg u i)).

Lemma prl_direct_hyp S S' u:
  prl_same_index S S' ->  prl_map2_compat S S' u ->
  prl_subfam_hyp S' (prl_dirim_set u).

Lemma prl_dirim_prop S S' u
  (Hsb: prl_same_index S S') (Hc:  prl_map2_compat S S' u):
  sub (Imf (projective_limit_fun S S' u))
  (projective_limit (projective_system_subsets (prl_direct_hyp  Hsb Hc))).
```

Proposition 3. « Let I be a preordered set, let $(E_\alpha, f_{\alpha\beta})$ be an inverse system of sets relative to I and let $E = \varprojlim E_\alpha$. Let J be a cofinal subset of I such that J is right directed, and let E' be the inverse limit of the inverse system of sets obtained from $(E_\alpha, f_{\alpha\beta})$ by restricting the index set to J. Then the canonical mapping g of E into E' (no. 1, formula (3.3)) is bijective. »

We also assume that, whenever $x \in I$, there is $y \in J$ such that $x \leq y$. Note that "cofinal" implies $J \subset d$ where d is the domain of r (i.e. I).

Bourbaki claims that if f'_i is the canonical mapping $E' \to E_i$, then g is the unique mapping $E \to E'$ such that $f_i = f'_i \circ g$. He uses the criterion of Proposition 1 for injectivity. We proceed directly: given x and y in E, show $x = y$ given that x and y have the same restrictions. Note that x and y are functional graphs with domain I, so it suffices to show $x_i = y_i$. Since J is cofinal, there is $j \in J$ such that $i \leq j$; this implies $x_i = f_{ij}(x_j)$; the result follows since $x_j = y_j$. In order to prove that g is surjective, consider $x \in E'$. This is a functional graph on J and we have to extend it. If $i \in I$, there is $j \in J$ such that $i \leq j$. We want $x_i = f_{ij}(x_j)$. The trick is that the RHS is independent of j.

In the special case where I has a greatest element ω, one can take $J = \{\omega\}$. In this case, E' is equal to the product $\prod_{i\in J} E_i$, hence isomorphic to E_ω. We give a direct proof. The isomorphism is f_ω.

```
Lemma right_directed_ind_prop r J:
  preorder r -> sub J (substrate r) -> right_directed_on r J ->
  right_directed_on (induced_order r J) J.

Lemma prl_rest_can_cofinal_bf S J (H: sub J (psI S)):
  cofinal (psr S) J -> right_directed_on (psr S) J ->
  bijection (prl_restr_canonical H).

Lemma prl_singleton_prop S k
  (f := Lf (Vg ^~ k) (projective_limit S) (Vg (psE S) k)):
```

```
  inc k (psI S) -> (forall i, inc i (psI S) -> gle (psr S) i k) ->
  bijection_prop  f (projective_limit S) (Vg (psE S) k).
```

Remark 1. Let f_i be the canonical projection $\mathrm{E} \to \mathrm{E}_i$, and E'_i its image. Then, the system of the E'_i is a projective system of subsets, the associated functions are surjective, the limits are the same, and

$$\mathrm{E}'_i = f_i\langle \mathrm{E}\rangle \subset \bigcup_{i\leq j} f_{ij}\langle \mathrm{E}_j\rangle. \tag{3.10}$$

Note that the inclusion just says: if $i \leq j$ then $\operatorname{Im} f_i \subset \operatorname{Im} f_{ij}$, where $\operatorname{Im} g$ is the image of g.

```
Definition prl_proj_image S i := Imf (prl_can_fun S i).
Definition prl_proj_image_fam S :=  Lg (psI S) (prl_proj_image S).


Lemma prl_proj_image_prop1 S i j:
  gle (psr S) i j ->
  sub (prl_proj_image S i) (Imf (Vg (psf S) (J i j))).
Lemma prl_proj_image_prop2 S i:
  inc i (psI S) -> sub (prl_proj_image S i)  (Vg (psE S) i).
Lemma prl_proj_image_prop3 S: prl_subfam_hyp S (prl_proj_image_fam S).
Lemma prl_proj_image_fam_fs S (fij' := prl_subfam_fct S (prl_proj_image_fam S)):
  forall ij, inc ij (psr S) -> surjection (Vg fij' ij).
Lemma prl_proj_image_prop4 S:
  projective_limit S =
  projective_limit(projective_system_subsets (prl_proj_image_prop3 S)).
```

Remark 2. « Let I be a *(right) directed* ordered set, let $(\mathrm{E}_\alpha, f_{\alpha\beta})$ be an inverse system of sets relative to I, and for each $\alpha \in \mathrm{I}$, let $u_\alpha : \mathrm{F} \to \mathrm{E}_\alpha$ be a mapping such that the family (u_α) satisfies the formula (3.5). Consider the inverse system $(\mathrm{F}_\alpha, i_{\alpha\beta})$ indexed by I, where $\mathrm{F}_\alpha = \mathrm{F}$ for all $\alpha \in \mathrm{I}$ and $i_{\alpha\beta}$ is the identity mapping of F. Then (no. 1, Example 2) F is canonically identified with $\varprojlim \mathrm{F}_\alpha$. If we consider u_α as a mapping of F_α into E_α, then u_α is an inverse system of mappings, and the mapping $u : \mathrm{F} \to \mathrm{E}$ defined by (3.6) is identified with the inverse limit of this system of mappings. Hence by abuse of language, we write $u = \varprojlim u_\alpha$. »

The context is the following, and we instantiated the result of example 2. Recall that $\varprojlim \mathrm{F}_\alpha$ is the set of all objects of the form c_x, the constant graph that maps every $i \in \mathrm{I}$ to x, for $x \in \mathrm{F}$. (This can be identified with F, in case I is non-empty).

```
Section Remark2.
Variable S : projective_system.
Variables u F: Set.
Hypothesis compat: prl_map_compat S u F.
Hypothesis rdr:right_directed_on (psr S) (psI S).

Definition prl_r2_sf := prl_exa2_system F (ps_preorder_r S) (ps_substrate_r S).

Lemma prl_r2_sf_prop1:
 projective_system_on prl_r2_sf (cst_graph (psI S) F) (psI S) (psr S)
     (cst_graph (psr S) (identity F)).
Lemma prl_r2_sf_prop2:projective_limit prl_r2_sf = diagonal_graphp F (psI S).
```

That u_i is a projective system of mappings is obvious. What we get is: two functions with the same target E, and two sources F, F′, where F′ is the diagonal of F^{I}. If $x \in \mathrm{F}$ then

$u(x) = u'(c_x)$. Note: if I is empty, then the projective limit has a single element, the empty set. We cannot identify F and F′ (since F′ has a single element), but $u(x) = \emptyset$ whatever x.

```
Lemma  prl_r2_sf_prop3: prl_map2_compat prl_r2_sf S u.

Lemma prl_r2_sf_prop4 (lf:= (projective_limit prl_r2_sf))
      (ls:= (projective_limit S))
      (u1 := projective_map S u F)
      (u2 := projective_limit_fun  prl_r2_sf S u):
  [/\ function_prop u2 lf ls, function_prop u1  F ls &
      forall x, inc x F ->  Vf u1 x = Vf u2 (cst_graph (psI S) x)].
End Remark2.
```

Remark 3. « Let I be an ordered set, and let $(E_\alpha, f_{\alpha\beta})$ be an inverse system of sets relative to I. For each finite subset J of I, let F_J be the inverse limit of the (finite) inverse system obtained from $(E_\alpha, f_{\alpha\beta})$ by restricting the index set to J. If J and K are any two finite subsets of I such that $J \subset K$ let g_{JK} denote the canonical mapping (3.3) of F_K into F_J. Then the relation (3.4) shows that (F_J, g_{JK}) is an *inverse system* of sets relative to the *directed* set (with respect to the relation $\subset$) $\mathfrak{F}(I)$ of finite subsets of I. Now for each $J \in \mathfrak{F}(I)$ let $h_J : E \to F_J$ be the canonical mapping (3.3). By virtue of (3.4) and with the abuse of language mentioned in Remark 2, h_J is an *inverse system* of mappings. Put $h = \varprojlim h_J : E \to F = \varprojlim F_J$ and let us show that h is a *bijection* (called *canonical*) … »

The following lemma will also be used in Exercise 4.

```
Lemma finite_subsets_order A (I:= Zo (powerset A) finite_set)
      (r:= sub_order I):
  [/\ order_on r I,
      forall x y, inc x I -> inc y I -> inc (x \cup y) I,
      forall x y, inc x I -> inc y I -> gle r x (x \cup y),
      forall i, inc i A -> inc (singleton i) I &
      right_directed r].
```

We first introduce the set $\mathfrak{F}$, show that it is right directed. Note that, if $i \in I$, then $i \in \{i\}$ and $\{i\} \in \mathfrak{F}$; so that $\bigcup \mathfrak{F} = I$; moreover, if $i \leq j$, there is $J \in \mathfrak{F}$, such that $i \in J$ and $j \in J$ (one can choose $\{i, j\}$).

```
Section Remark3.
Variable S: projective_system.

Definition prl_r3_nI := Zo (powerset (psI S)) finite_set.
Definition prl_r3_nr := sub_order prl_r3_nI.

Lemma prl_r3_sr: substrate prl_r3_nr = prl_r3_nI.
Lemma prl_r3_trans i j k:
  gle prl_r3_nr i j ->  gle prl_r3_nr j k -> gle prl_r3_nr i k.
Lemma prl_r3_nI_stable_union x y:
  inc x prl_r3_nI -> inc y prl_r3_nI -> inc (x \cup y) prl_r3_nI.
Lemma prl_r3_directed_nr: right_directed prl_r3_nr.
Lemma prl_r3_qprop0 i: inc i (psI S) -> inc (singleton i) prl_r3_nI.
Lemma prl_r3_qprop1 i j: gle (psr S) i j ->
     exists J, [/\ inc J  prl_r3_nI, inc i J & inc j J].
```

The technique is the same as in Exercise 1 (see details below); instead of L we have $\mathfrak{F}$. Instead of J_λ we have the identity on $\mathfrak{F}$ (this makes some lemmas simpler).

We introduce a system S_J, via the axiom of choice. In case $J \in \mathfrak{F}$, this gives a subsystem, otherwise S. We deduce F_J, g_J and g_{KJ}. We show that (F_J, g_{JK}) is a projective system.

```
Definition prl_r3_systemi J :=
  match (ixm (inc J prl_r3_nI)) with
    | inl hx => prl_restr (prl_r3_prop4 hx)
    | inr _ => S
    end.
Definition prl_r3_Fl J := projective_limit (prl_r3_systemi J).

Definition prl_r3_gi J:= Lf (restr ^~ J) (projective_limit S) (prl_r3_Fl J).
Definition prl_r3_gij ij :=
  Lf (restr ^~ (P ij)) (prl_r3_Fl (Q ij)) (prl_r3_Fl (P ij)).

Lemma prl_r3_res0 i (H: inc i prl_r3_nI):
  prl_r3_Fl i = (projective_limit (prl_restr (prl_r3_prop4 H))).
Lemma rem3prop5a j: inc j prl_r3_nI -> j = psI (prl_r3_systemi j).
Lemma prl_r3_prop5b i (H: sub i (psI S)):
  inc i prl_r3_nI -> prl_same_data (prl_restr H) (prl_r3_systemi i).
Lemma prl_r3_res1 i: inc i prl_r3_nI ->
  function_prop (prl_r3_gi i) (projective_limit S) (prl_r3_Fl i).
Lemma prl_r3_prop5 i j: gle prl_r3_nr i j ->  sub i (psI (prl_r3_systemi j)).
Lemma prl_r3_prop6 i j  (lij: gle prl_r3_nr i j) :
  prl_same_data (prl_restr (prl_r3_prop5 lij)) (prl_r3_systemi i).
Lemma prl_r3_prop6a i j  (lij: gle prl_r3_nr i j) :
  (projective_limit_restr (prl_r3_prop5 lij)) = (prl_r3_Fl i).
Lemma prl_r3_prop7 i j: gle prl_r3_nr i j ->
  lf_axiom (restr^~  i) (prl_r3_Fl j) (prl_r3_Fl i).
Lemma prl_r3_res2 i j: gle prl_r3_nr i j ->
  function_prop (prl_r3_gij (J i j)) (prl_r3_Fl j) (prl_r3_Fl i).
Lemma prl_r3_res3 i: inc i prl_r3_nI -> prl_r3_gij (J i i) = identity (prl_r3_Fl i).
Lemma prl_r3_pr4 i j k: gle prl_r3_nr i j ->  gle prl_r3_nr j k ->
   prl_r3_gij (J i j) \co prl_r3_gij (J j k) = prl_r3_gij (J i k).

Definition prl_r3_F: projective_system.
Lemma prl_r3_F_prop: projective_system_on prl_r3_F
                    (Lg prl_r3_nI prl_r3_Fl) prl_r3_nI prl_r3_nr (Lg prl_r3_nr prl_r3_gij).



Definition prl_r3_restr_fun z:=  Lg prl_r3_nI (fun i => restr z i).
Definition prl_r3_F_can := Lf prl_r3_restr_fun
   (projective_limit S) (projective_limit prl_r3_F).

Lemma prl_r3_F_can_ax1 i z: inc i prl_r3_nI -> inc z (projective_limit S) ->
   inc (restr z i) (prl_r3_Fl i).
Lemma prl_r3_F_can_ax: lf_axiom prl_r3_restr_fun
   (projective_limit S) (projective_limit prl_r3_F).
Lemma prl_r3_F_can_fun: inc prl_r3_F_can
  (functions (projective_limit S) (projective_limit prl_r3_F)).
Lemma prl_r3_F_can_bf: bijection prl_r3_F_can.

End Remark3.
```

3.3 Double Inverse Limit

Assume that we have two preordered sets, I and L, and a projective family S on $I \times L$. Bourbaki denotes the sets by E_α^λ and the functions by $f_{\alpha\beta}^{\lambda\mu}$ where lower indices are in I, upper indices are in L (note the vertical alignments of the indices); the order is the product, so that

$$f_{\alpha\gamma}^{\lambda\nu} = f_{\alpha\beta}^{\lambda\mu} \circ f_{\beta\gamma}^{\mu\nu} \qquad \text{whenever } \alpha \le \beta \le \gamma \text{ and } \lambda \le \mu \le \nu. \tag{3.11}$$

Fix $\lambda \in L$. Define $g_{\alpha\beta}^{\lambda} = f_{\alpha\beta}^{\lambda\lambda}$ so that the previous relation becomes

$$g_{\alpha\gamma}^{\lambda} = g_{\alpha\beta}^{\lambda} \circ g_{\beta\gamma}^{\lambda} \qquad \text{whenever } \alpha \le \beta \le \gamma. \tag{3.12}$$

This allows us to define S^λ a projective system on E_α^λ indexed by I; let F^λ be its projective limit. Fix λ and μ; let $h_\alpha^{\lambda\mu} = f_{\alpha\alpha}^{\lambda\mu}$. This is (again by (3.11)) a projective system of mappings, let denote its limit by $h^{\lambda\mu}$. We have

$$h^{\lambda\nu} = h^{\lambda\mu} \circ h^{\mu\nu} \qquad \text{whenever } \lambda \le \mu \le \nu, \tag{3.13}$$

so that we can define a projective system S' on L. The objective is to prove that $\varprojlim S$ and $\varprojlim S'$ are canonically isomorphic. The techniques are the same as in Exercise 1.

Proposition 4. If $(E_\alpha^\lambda, f_{\alpha\beta}^{\lambda\mu})$ is an inverse system of sets relative to a product $I \times L$ of preordered sets, then (up to a canonical bijection) we have

$$\varprojlim_{\alpha,\lambda} E_\alpha^\lambda = \varprojlim_{\lambda} (\varprojlim_{\alpha} E_\alpha^\lambda). \tag{3.14}$$

We shall define later on the double direct limit; so we start with some common lemmas. They are prefixed by 'pidl'. The two sets I and L are `I1` and `I2`, ordered by `r1` and `r2`. Note: if the two sets are right directed, so is the product; the converse holds provided that no factor is empty.

```
Section DoubleProjInjLimit.
Variables I1 I2 r1 r2: Set.
Hypothesis (or1: preorder r1)(or2: preorder r2)
           (sr1: substrate r1 = I1)(sr2: substrate r2 = I2).

Lemma pidl_or:  preorder_on (prod_of_relation r1 r2) (I1 \times I2).
Lemma pidl_gleP i j: gle (prod_of_relation r1 r2) i j <->
   [/\ pairp i, pairp j, gle r1 (P i) (P j) & gle r2 (Q i) (Q j)].
Lemma pidl_gleP1 i j k l: gle (prod_of_relation r1 r2) (J i j) (J k l) <->
  gle r1 i k /\ gle r2  j l.
Lemma pidl_i1_L a b: gle r2 a b -> inc a I2.
Lemma pidl_i2_L a b: gle r2 a b -> inc b I2.
Fact pidl_i3_L x: inc x r2 -> gle r2  (P x) (Q x).

Lemma pidl_directed:
  right_directed_prop r1 -> right_directed_prop r2 ->
  right_directed_prop (prod_of_relation r1 r2).

Lemma pidl_directed_bis: nonempty I1 -> nonempty I2 ->
  right_directed_prop (prod_of_relation r1 r2) ->
  right_directed_prop r1 /\ right_directed_prop r2.
End DoubleProjInjLimit.
```

We recall that if I and L are two sets, $\leq_I$ and $\leq_L$ are two order (resp. preorder) relations on I and L, then the relation "$\mathrm{pr}_1 a \leq_I \mathrm{pr}_1 b$ and $\mathrm{pr}_2 a \leq_L \mathrm{pr}_2 b$" between two elements a and b of $I \times L$ is an order (resp. preorder) relation on $I \times L$. We consider a system S whose preorder is this relation (so that the index set will be $I \times L$).

```
Section DoubleProjectiveLimit.
Variables I1 I2 r1 r2: Set.
Hypothesis (or1: preorder r1)(or2: preorder r2)
           (sr1: substrate r1 = I1)(sr2: substrate r2 = I2).
Variable S: projective_system.
Hypothesis Sr: psr S = (prod_of_relation r1 r2).
Lemma prl_dl_I: psI S = I1 \times I2.
```

We define now the families E, g for fixed lambda, and the system S^λ. The system is defined only if $\lambda \in L$. For this reason, we define (via the axiom of choice) a system whatever λ, choosing S as default value. This allows us to define F^λ.

```
Definition prl_dl_Elam_fam lam := Lg I1 (fun i => Vg (psE S) (J i lam)).
Definition prl_dl_glam_fam lam :=
  Lg r1 (fun ij => Vg (psf S) (J (J (P ij) lam) (J (Q ij) lam))).


Lemma prl_dl_index_p1 lam i: inc lam I2 -> inc i r1 ->
   gle (psr S) (J (P i) lam) (J (Q i) lam).
Lemma prl_dl_index_p2 lam mu i:  gle r2 lam mu -> inc i I1 ->
   gle (psr S) (J i lam) (J i mu).

Definition prl_dl_S_lambda lam (Hl: inc lam I2) : projective_system.
Lemma prl_dl_S_lambda_prop lam (Hl: inc lam I2) :
  projective_system_on (prl_dl_S_lambda Hl)
     (prl_dl_Elam_fam lam) I1 r1 (prl_dl_glam_fam lam).

Definition prl_dl_system_S_lambda lam :=
  match (ixm (inc lam I2)) with
    | inl hx => (prl_dl_S_lambda hx)
    | inr _ => S
    end.

Definition prl_dl_F_lambda lam :=
  projective_limit (prl_dl_system_S_lambda lam).

Lemma prl_dl_F_lambda_prop lam (Hl: inc lam I2):
  prl_dl_F_lambda lam = projective_limit (prl_dl_S_lambda Hl).
```

Assume now $\lambda \leq \mu$. This gives two systems S^λ and S^μ with the same index set, and a projective system of mappings h_a^{lm}. Corollary 2 gives (3.13).

```
Definition prl_dl_halm_fam lam mu:=
  Lg I1 (fun i => Vg (psf S) (J (J i lam) (J i mu))).

Definition prl_dl_hlm lam mu (H: gle r2 lam mu) :=
  projective_limit_fun (prl_dl_S_lambda (pidl_i2_L sr2 H))
                       (prl_dl_S_lambda (pidl_i1_L sr2 H))
                       (prl_dl_halm_fam lam mu).
```

```
Lemma prl_dl_halm_compat lam mu (H: gle r2 lam mu):
  prl_map2_compat (prl_dl_S_lambda (pidl_i2_L sr2 H))
                  (prl_dl_S_lambda (pidl_i1_L sr2 H)) (prl_dl_halm_fam lam mu).

Lemma prl_dl_hlm_compose l m n
  (Hlm : gle r2 l m) (Hmn: gle r2 m n):
  (prl_dl_hlm Hlm)  \co (prl_dl_hlm Hmn) =
  (prl_dl_hlm (proj33 or2 _ _ _ Hlm Hmn)).
```

We now define h^{lm}, via the axiom of choice. If $x \in r_2$, then $\mathrm{pr}_1 x \leq \mathrm{pr}_2 x$, and we can consider $h^x = h^{\mathrm{pr}_1 x \leq \mathrm{pr}_2 x}$. Assume $l \leq m$, and let x be the pair (l, m). Obviously, $\mathrm{pr}_1 x = l$ and $\mathrm{pr}_2 x = m$ but this does not imply $h^{lm} = h^x$, so this equality is not trivial. Also h^{ii} is the identity function, but we have to work a bit. Finally, defining S' is trivial.

```
Fact prl_dl_i3_L x: inc x r2 -> gle r2  (P x) (Q x).

Definition prl_dl_hlm_gen x :=
  match (ixm (inc x r2)) with
    | inl hx => (prl_dl_hlm (pidl_i3_L or2 hx))
    | inr _ => emptyset
    end.


Lemma prl_dl_hlm_fct lm: inc lm r2 ->
   function_prop (prl_dl_hlm_gen lm)
         (prl_dl_F_lambda (Q lm))(prl_dl_F_lambda (P lm)).
Lemma prl_dl_S_lambda_Iv2 x y  (H1: inc x I2) (H2: inc y I2) : x = y ->
  prl_same_data (prl_dl_S_lambda H1)(prl_dl_S_lambda H2).
Lemma  prl_dl_hml_invariant i j (H:gle r2 i j) :
   prl_dl_hlm H = prl_dl_hlm_gen (J i j).
Lemma prl_dl_hml_id i: inc i I2 ->
   Vg (Lg r2 prl_dl_hlm_gen) (J i i) = identity (prl_dl_F_lambda i).


Definition prl_dl_systemS': projective_system.
Lemma  prl_dl_systemS'_prop: projective_system_on prl_dl_systemS'
     (Lg I2 prl_dl_F_lambda) I2 r2 (Lg r2 prl_dl_hlm_gen).
```

Finally, we identify the two projective limits, via Chapter II, §5, no 5, Proposition 7. The proposition says that a product on a set A, partitioned in A_i is isomorphic to a double product, the inner product being over all A_i, the outer product over the index set of the partition. It does not apply, but we can think of the partition induced on the product by fixing one index. We may however use the same techniques.

Take $x \in \varprojlim \mathrm{S}$, $\lambda \in \mathrm{L}$; $f_\lambda(x)$ will be the functional graph $i \mapsto x_{i\lambda}$, with domain I. The functional graph $\lambda \mapsto f_\lambda(x)$ with domain I belongs to $\varprojlim \mathrm{S}'$.

```
Definition prl_dl_slice x l := Lg I1 (fun i => Vg x (J i l)).
Definition prl_dl_slice2 x := Lg I2 (prl_dl_slice x).
Definition prl_dl_can_iso := Lf prl_dl_slice2
   (projective_limit S) (projective_limit prl_dl_systemS').

Lemma prl_dl_slice_p1 x lam: inc x (projective_limit S) ->
  inc lam I2 -> inc (prl_dl_slice x lam) (prl_dl_F_lambda lam).
```

```
Lemma prl_dl_slice_p2 x: inc x (projective_limit S) ->
  inc (prl_dl_slice2 x) (projective_limit prl_dl_systemS').
Lemma prl_dl_canon_bijection: bijection_prop prl_dl_can_iso
   (projective_limit S) (projective_limit prl_dl_systemS').  (* 58 *)

End DoubleInverseLimit.
```

Corollary 1. Let $(E'^{\lambda}_{\alpha}, f'^{\mu\lambda}_{\alpha\beta})$ be another inverse system of sets relative to $I \times L$ and for each $(\alpha,\lambda) \in I \times L$ let u^{λ}_{α} be a mapping of E^{λ}_{α} into E'^{λ}_{α} such that the u^{λ}_{α} for an an inverse system of mappings. Then

$$\varprojlim_{\alpha,\lambda} u^{\lambda}_{\alpha} = \varprojlim_{\lambda}(\varprojlim_{\alpha} u^{\lambda}_{\alpha}). \tag{3.15}$$

Bourbaki says that « the verification is similar to that of Proposition 4. » We consider a section, with the same assumptions as above. We consider two systems S and S′ and a family u, satisfying (3.7). We assume that S and S′ have the same index set $I \times L$.

```
Section  DoubleInverseLimit2.

Variables I1 I2 r1 r2: Set.
Hypothesis (or1: preorder r1)(or2: preorder r2)
           (sr1: substrate r1 = I1)(sr2: substrate r2 = I2).

Variables S S': projective_system.
Variable u: Set.

Hypothesis Sr: psr S = prod_of_relation r1 r2.
Hypothesis Sr': psr S' = prod_of_relation r1 r2.
Hypothesis compat_u: prl_map2_compat S S' u.

Lemma psr_dl2_SrSr: prl_same_index S S'.
```

We define here S^{λ}, S'^{λ} and u^{λ}. The systems are defined as above, and u^{λ} is a slice of u. Relations (3.7) hold whenever $\lambda \in L$. This allows to define $v^{\lambda} = \varprojlim u^{\lambda}$, and the family $(v^{\lambda})_{\lambda\in L}$. This family satisfies (3.7) for the systems $\varprojlim S^{\lambda}$ and $\varprojlim S'^{\lambda}$.

```
Definition prl_dl2_ulam_fam lam :=  Lg I1 (fun i => Vg u (J i lam)).
Definition prl_dl2_Slambda := (prl_dl_system_S_lambda or1 or2 sr1 sr2 Sr).
Definition prl_dl2_Slambda':= (prl_dl_system_S_lambda or1 or2 sr1 sr2 Sr').


Lemma prl_dl2_res1 lam:inc lam I2 ->
   prl_same_index (prl_dl2_Slambda lam) (prl_dl2_Slambda' lam) /\
   prl_map2_compat (prl_dl2_Slambda lam) (prl_dl2_Slambda' lam)
                   (prl_dl2_ulam_fam lam).

Definition prl_dl2_v lam :=
   projective_limit_fun (prl_dl2_Slambda lam) (prl_dl2_Slambda' lam)
                   (prl_dl2_ulam_fam lam).
Definition prl_dl2_v_fam := Lg I2 prl_dl2_v.

Definition prl_dl2_limlim := (prl_dl_systemS' or1 or2 sr1 sr2 Sr).
Definition prl_dl2_limlim' := (prl_dl_systemS' or1 or2 sr1 sr2 Sr').
```

```
Lemma prl_dl2_res2:
  prl_map2_compat prl_dl2_limlim  prl_dl2_limlim' prl_dl2_v_fam. (* 107 *)
```

We can now define the RHS of (3.15), the arrow at the bottom of the following diagram. It is equal to the LHS (the arrow at the top of the diagram), modulo the canonical bijections of Proposition 4 (the vertical arrows).

$$\begin{array}{ccc} & \varprojlim_{\alpha,\lambda} u_\alpha^\lambda & \\ \varprojlim_{\alpha,\lambda} \mathrm{E}_\alpha^\lambda & \longrightarrow & \varprojlim_{\alpha,\lambda} \mathrm{E}'^\lambda_\alpha \\ \downarrow & & \downarrow \\ \varprojlim_\lambda (\varprojlim_\alpha \mathrm{E}_\alpha^\lambda) & \xrightarrow[\varprojlim_\lambda (\varprojlim_\alpha u_\alpha^\lambda)]{} & \varprojlim_\lambda (\varprojlim_\alpha \mathrm{E}'^\lambda_\alpha) \end{array}$$

```
Lemma prl_dl2_res3    (* 59 *)
  (pl1 := projective_limit_fun S S' u)
  (pl2 := projective_limit_fun prl_dl2_limlim  prl_dl2_limlim' prl_dl2_v_fam)
  (bij1 :=  prl_dl_can_iso or1 or2 sr1 sr2 Sr)
  (bij2 := prl_dl_can_iso or1 or2 sr1 sr2 Sr'):
  [/\ bijection bij1, bijection bij2 & pl2 \co bij1  =  bij2 \co pl1].
```

Corollary 2. Let $(\mathrm{E}_\alpha^\lambda, f_{\alpha\beta}^\lambda)_{\lambda\in\mathrm{L}}$ be a family of inverse systems of sets, relative to I. If $\prod_{\lambda\in\mathrm{L}} f_{\alpha\beta}^\lambda$ denotes the extensions to products (Chapter II, §5, No. 7, Definition 2) of the family of mappings $(f_{\alpha\beta}^\lambda)_{\lambda\in\mathrm{L}}$, then $(\prod_{\lambda\in\mathrm{L}} \mathrm{E}_\alpha^\lambda, \prod_{\lambda\in\mathrm{L}} f_{\alpha\beta}^\lambda)$ is an inverse system of sets relative to I , and (up to a canonical bijection) we have

$$\varprojlim_\alpha \prod_{\lambda\in\mathrm{L}} \mathrm{E}_\alpha^\lambda = \prod_{\lambda\in\mathrm{L}} (\varprojlim_\alpha \mathrm{E}_\alpha^\lambda). \tag{3.16}$$

The assumption is that we have a projective system $\mathrm{S}(l)$ for every set l, and that, when $l \in \mathrm{L}$, the order is r, so that the substrate is I. We order L by the trivial order, then define the system S.

```
Section  DoubleInverseLimit3.

Variables I L r: Set.
Variable fS: Set -> projective_system.

Hypothesis (or: preorder r)(sr: substrate r = I).
Hypothesis fSr: forall l, inc l L -> psr (fS l) = r.


Definition prl_dl3_or := prod_of_relation r (diagonal L).

Lemma prl_dl3_fSI l: inc l L -> psI (fS l) = I.
Lemma prl_dl3_orL: preorder (diagonal L).
Lemma prl_dl3_srL: substrate (diagonal L) = L.
Lemma prl_dl3_osr: preorder_on (prl_dl3_or) (I \times L).

Lemma prl_dl3_or_prop1 p: inc p (prl_dl3_or) ->
```

```
  [/\ pairp p, inc (P p) (I\times L), inc (Q p) (I\times L),
     inc (J (P (P p)) (P (Q p))) r & (Q (P p)) = (Q (Q p))].

Definition prl_dl3_E := Lg (I\times L) (fun p => Vg (psE (fS (Q p))) (P p)).
Definition prl_dl3_f := Lg prl_dl3_or
    (fun p => Vg (psf (fS (Q (Q p)))) (J (P (P p)) (P (Q p)))).

Definition prl_dl3_systemS: projective_system.

Lemma  prl_dl3_systemS_prop: projective_system_on prl_dl3_systemS
   prl_dl3_E (I\times L) prl_dl3_or  prl_dl3_f.
```

Obviously, we can apply Proposition 4. It says that there is a canonical isomorphism between S and S′, for some S′, which is a system on E. If we look at the definition, we see that this is the family F^λ introduced above, i.e. $\varprojlim_\alpha E^\lambda_\alpha$. Moreover, the only functions of S′ are the identity functions (since L is trivially ordered) so that $\varprojlim S' = \prod E$.

```
Lemma prl_dl3_systemS_sr: psr prl_dl3_systemS = prod_of_relation r (diagonal L).
Let iso:= (prl_dl_can_iso or prl_dl3_orL sr prl_dl3_srL prl_dl3_systemS_sr).
Let S':= (prl_dl_systemS' or prl_dl3_orL sr prl_dl3_srL prl_dl3_systemS_sr).
Let E := Lg L(prl_dl_F_lambda or prl_dl3_orL sr prl_dl3_srL prl_dl3_systemS_sr).

Lemma prl_dl3_RHS: (projective_limit S') = productb E.
Lemma prl_dl3_systemS_can:
  bijection_prop iso (projective_limit prl_dl3_systemS) (productb E).
```

Consider now the LHS, the projective limit of S. This is a subset of $\prod_{I\times L}$, identified with $\prod_I \prod_L$, by considering a double family as a family of families. We introduce the function and show that it is injective.

```
Definition prl_dl3_Efam i :=
  Lg L (fun l => Vg (psE (fS l)) i).
Definition prl_dl3_Ep i := productb (prl_dl3_Efam i).

Definition prl_dl3_mod x :=
  Lg I (fun i => (Lg L (fun l => Vg x (J i l)))).

Lemma prl_dl3_mod_p1 x: inc x  (projective_limit prl_dl3_systemS) ->
  inc  (prl_dl3_mod x) (productb (Lg I prl_dl3_Ep)).

Lemma prl_dl3_mod_inj x y:
  inc x  (projective_limit prl_dl3_systemS) ->
  inc y  (projective_limit prl_dl3_systemS) ->
   (prl_dl3_mod x)  = (prl_dl3_mod y) -> x = y.
```

We pretend that the target of the function is the projective system over I of the sets $\prod_L E^\lambda_\alpha$; the functions are the extensions to products of the $f^\lambda_{\alpha\beta}$. We study these extensions. Then prove the result.

```
Definition prl_dl3_ffam :=
  Lg r (fun ij => (ext_map_prod L (fun l => Vg (psE (fS l)) (Q ij))
                                  (fun l => Vg (psE (fS l)) (P ij))
                                  (fun l => (graph (Vg (psf (fS l)) ij))))).
```

```
Lemma prl_dl3_ffam_ax ij: inc ij r ->
   ext_map_prod_axioms L (fun l : Set => Vg (psE (fS l)) (Q ij))
     (fun l => Vg (psE (fS l)) (P ij))
     (fun l => graph (Vg (psf (fS l)) ij)).
Lemma prl_dl3_ffam_fun ij: inc ij r ->
  function_prop (Vg prl_dl3_ffam ij)(prl_dl3_Ep (Q ij)) (prl_dl3_Ep (P ij)).
Lemma  prl_dl3_ffam_id i: inc i I ->
   Vg prl_dl3_ffam (J i i) = identity (prl_dl3_Ep i).

Lemma  prl_dl3_ffam_comp i j k: gle r i j ->  gle r j k ->
   Vg prl_dl3_ffam (J i j) \co Vg prl_dl3_ffam (J j k) =
   Vg prl_dl3_ffam (J i k).

Definition prl_dl3_systemS': projective_system.
Lemma  prl_dl3_systemS'_val: projective_system_on  prl_dl3_systemS'
       (Lg I prl_dl3_Ep) I r prl_dl3_ffam.

Lemma prl_dl3_res (X := (projective_limit prl_dl3_systemS))
      (Y := projective_limit prl_dl3_systemS'):
  bijection_prop (Lf prl_dl3_mod  X Y) X Y.

End  DoubleInverseLimit3.
```

3.4 Conditions for an inverse limit to be non-empty

Proposition 5. Assume that I is directed and has a countable cofinal subset. Assume that every f_{ij} is surjective. Then every f_i is surjective. In particular, if no E_i is empty, then $\varprojlim \mathrm{E}_i$ is nonempty.

Assume $f_{ij} : \mathrm{E}_j \to \mathrm{E}_i$ is surjective. Then E_i is empty if and only if E_j is empty. So, either all E_i are empty, or all are non-empty. In the first case, every f_i (whose target is empty) is thus surjective. So we shall consider the second case.

Since $f_i = f_{ij} \circ f_j$, if f_j is surjective so is f_i, whenever $i \leq j$. Assume that I has a greatest element, say j. It suffices to show that f_j is surjective. Let $x_j \in \mathrm{E}_j$. Define $x_i = f_{ij}(x_j)$, and let x be $i \in \mathrm{I} \mapsto x_i$. Since x is in the projective limit, we have $f_j(x) = x_j$, and the conclusion holds. Assume that there is a finite cofinal set; by induction the finite set has an upper bound, which must be the greatest element.

```
Section ProjectiveLimitNonEmpty1.

Variable S: projective_system.
Hypothesis rdr:right_directed_pre (psr S).
Hypothesis sjf: forall i, inc i (psr S) -> surjection  (Vg (psf S) i).


Definition prl_ne1_some_nonempty :=
  (exists2 i, inc i (psI S) & nonempty(Vg (psE S) i)).

Lemma prl_ne1_allE_ne:  prl_ne1_some_nonempty ->
   forall i, inc i (psI S) -> nonempty(Vg (psE S) i).
Lemma prl_ne1_res1: ~prl_ne1_some_nonempty ->
  forall i, inc i (psI S) -> surjection (prl_can_fun S i).
```

```
Lemma prl_ne1_surj_rec i j  (f := prl_can_fun S):
  gle (psr S) i j  -> surjection (f j) -> surjection  (f i).

Lemma prl_ne1_res2:
  (exists2 j, inc j (psI S) & forall i, inc i (psI S) -> gle (psr S) i j) ->
  forall i, inc i (psI S) -> surjection (prl_can_fun S i).

Lemma prl_ne1_res3 A: cofinal (psr S) A -> finite_set A ->
   forall i, inc i (psI S) -> surjection (prl_can_fun S i).
```

We assume now that there is a countable cofinal set A. The case A finite (or empty) is not considered by Bourbaki, but is dealt with above. Enumerate A as $a_0, a_1, \ldots$. Take for b_{n+1} an upper bound of a_n and b_n. By induction, there exists an increasing function $b : \mathbf{N} \to \mathrm{I}$, whose range is cofinal. Bourbaki says « we need only prove the Proposition for the case $\mathrm{I} = \mathbf{N}$ » (this is not really true, but the idea is there). Let's show that f_k is surjective when $k \in \mathrm{I}$. There is n such that $k \leq b(n)$; it suffices to show that $f_{b(n)}$ is surjective. Bourbaki says « It is clear that it suffices to prove that f_0 is surjective » (this really means: we need only consider $b(j)$ for $j \geq n$. Take $y \in \mathrm{E}_{b(n)}$; define a sequence y_i by induction via $y_i = f_{b(n+i),b(n+i+1)}(y_{i+1})$ and $y_0 = y$. This is possible, since b is increasing and the function that appears there is surjective. By induction $y_i = f_{b(n+i),b(n+j)}(y_j)$ holds if $i \leq j$. If $i \in \mathrm{I}$ there exists j such that $i \leq b(n+j)$. We define $x_i = f_{i,b(n+j)}(y_j)$. If $j \leq k$, we get $x_i = f_{i,b(n+k)}(y_k)$ by the previous remark, so that x_i is independent of the choice of j. This trivially says that the sequence of the x_i belongs to $\varprojlim \mathrm{S}$, and its value by $f_{b(n)}$ is y.

```
Lemma prl_ne1_res4 A: cardinal A = aleph0 -> cofinal (psr S) A ->
   exists f, [/\  function f, source f = Nat,
      forall i, natp i -> gle (psr S) (Vf f i) (Vf f (csucc i)) &
      forall i, inc i (psI S) -> exists2 n, natp n & gle (psr S) i (Vf f n) ].
Lemma prl_ne1_res5: (exists2 A, countable_set A & cofinal(psr S) A ) ->
  forall i, inc i (psI S) -> surjection (prl_can_fun S i). (* 97 *)
End ProjectiveLimitNonEmpty1.
```

Theorem 1. Consider a projective system S on a right directed set I such that there is $\mathfrak{S}$ satisfying properties (i), (ii), (ii) and (iv), described below. Then (a)

$$f_i\langle \mathrm{E}\rangle = \bigcap_{i\leq j} f_{ij}\langle \mathrm{E}_j\rangle \qquad (i \in \mathrm{I}) \tag{3.17}$$

and (b) if no E_i is empty, then E is nonempty. If (iii) is replaced by the weaker condition (iii)', then (b) remains true.

In this statement, E is the projective limit of S and f_i the canonical projection $\mathrm{E} \mapsto \mathrm{E}_i$. Equation (3.17) can be written as $\mathrm{Im}\, f_i = \bigcap_{i\leq j} \mathrm{Im}\, f_{ij}$. Assume $\mathrm{E}_k = \emptyset$ for some index k. In this case E is empty, as well as the RHS of (3.17). There exists j such $i \leq j$ and $k \leq j$. Since E_k is empty, so is E_j as well as $f_{ij}\langle \mathrm{E}_j\rangle$ and the intersection. This means that (a) holds trivially, as well as (b).

```
Section ProjectiveLimitNonEmpty2.

Variable S: projective_system.
Hypothesis rdr: right_directed_prop (psr S).

Definition prl_ne2_res_RHS i :=
  intersectionf  (Zo (psI S) (gle (psr S) i))
```

```
                (fun j =>  (Imf (Vg (psf S) (J i j)))).
Definition prl_ne2_res_a:=
  forall i, inc i (psI S) -> Imf (prl_can_fun S i) = prl_ne2_res_RHS i.

Lemma prl_ne2_intP i j:
  inc j (Zo (psI S) (gle (psr S) i)) <-> (gle (psr S) i j).
Lemma prl_ne2_int_i i: inc i (psI S) -> inc i (Zo (psI S) (gle (psr S) i)).
Lemma prl_ne2_prop_trivial:
   (exists2 i, inc i (psI S) & Vg (psE S) i = emptyset) ->
   prl_ne2_res_a.
```

The assumption is that there exists a functional graph $\mathfrak{S}$ on I, such that $\mathfrak{S}_i$ is formed of subsets of E_i, and every intersection of sets belonging to $\mathfrak{S}_i$ also belongs to $\mathfrak{S}_i$. This is condition (i). The case of the empty intersection is a bit tricky: it has to be interpreted as $E_i \in \mathfrak{S}_i$.

```
Variable FS_fam: Set.

Hypothesis FS_fgraph: fgraph FS_fam.
Hypothesis FS_domain: domain FS_fam = psI S.
Hypothesis FS_range:
  forall i, inc i (psI S) -> sub (Vg FS_fam i) (powerset (Vg (psE S) i)).
Hypothesis FS_whole:
  forall i, inc i (psI S) -> inc  (Vg (psE S) i)(Vg FS_fam i).
Hypothesis FS_inter:
  forall i A, inc i (psI S) -> sub A (Vg FS_fam i) -> nonempty A ->
              inc (intersection A)(Vg FS_fam i).
```

The second condition (ii) is: if a set of subsets $\mathfrak{F} \subset \mathfrak{S}_i$ is such that every finite intersection of sets belonging to $\mathfrak{F}$ is non-empty, then $\bigcap_{M\in\mathfrak{F}} M$ is non-empty. The variant (ii)' says if $\mathfrak{G} \subset \mathfrak{S}_i$ is left directed (with respect to inclusion) and does not contain the empty set, then $\bigcap_{M\in\mathfrak{G}} M$ is non-empty [note that $\mathfrak{G}$ has to be nonempty.] Note: assume (ii) true, and $\mathfrak{G}$ left directed, if A a finite subset of $\mathfrak{G}$, by induction there is $z \in \mathfrak{G}$ such that $z \subset \bigcap A$. This shows (ii)'. Conversely, assume (ii)' true and let $\mathfrak{F}$ be a set satisfying the hypothesis of (ii). Let $\mathfrak{G}$ be the set of finite non-empty intersections of $\mathfrak{F}$. This is a subset of $\mathfrak{F}$ (take singletons) and does not contain the empty set; it is also stable by intersection.

```
Definition prl_ne1_condii i:=
  forall F, sub F (Vg FS_fam i) -> nonempty F ->
            (forall A, sub A F -> finite_set A -> nonempty A ->
                       nonempty (intersection A)) ->
            nonempty (intersection F).
Definition prl_ne1_condii' i:=
  forall F, sub F (Vg FS_fam i) -> nonempty F ->
            ~(inc emptyset F) ->
            (forall x y, inc x F -> inc y F ->
                         exists z, [/\ inc z F, sub z x & sub z y]) ->
            (forall A, sub A F -> finite_set A -> nonempty A ->
                       nonempty (intersection A)) ->
            nonempty (intersection F).

Lemma prl_ne2_prop1 i: inc i (psI S) ->
   (prl_ne1_condii i <-> prl_ne1_condii' i).
```

Condition (iii) says: if $i \leq j$ and $x_i \in \mathrm{E}_i$ we have $f_{ij}^{-1}(x_i) \in \mathfrak{S}_j$. The weaker condition (iii)' is: for each $i \in \mathrm{I}$ and each non-empty set $\mathrm{M}_i \in \mathfrak{S}_i$ there exists $x_i \in \mathrm{M}_i$ such that $f_{ij}^{-1}(x_i) \in \mathfrak{S}_i$ for each $j \geq i$. Recall that $f_{ij}^{-1}(x_i)$ is the set of all x_j (in E_j) such that $f_{ij}(x_j) = x_i$.

```
Definition pr1_ne2_hyp3_aux x a b:=
  inc (Vfi1 (Vg (psf S) (J a b)) x) (Vg FS_fam b).

Definition  pr1_ne2_hyp3_plain:= forall a b  x,
  gle (psr S) a b ->  inc x (Vg (psE S) a)  ->  pr1_ne2_hyp3_aux x a b.

Definition  pr1_ne2_hyp3_weak:= forall a M,
  inc a (psI S) -> inc M (Vg FS_fam a) -> nonempty M ->
  exists2 x, inc x M & forall b, gle (psr S) a b -> pr1_ne2_hyp3_aux x a b.

Lemma pr1_ne2_hyp3_weak_prop:  pr1_ne2_hyp3_plain -> pr1_ne2_hyp3_weak.
```

Condition (iv) is: if $i \leq j$ and $\mathrm{M}_j \in \mathfrak{S}_j$, then $f_{ij}\langle \mathrm{M}_j \rangle \in \mathfrak{S}_i$.

```
Hypothesis FS_prop_iv:
  forall i j M,  gle (psr S) i j ->  inc M (Vg FS_fam j) ->
              inc (Vfs (Vg (psf S) (J i j)) M) (Vg FS_fam i).

Hypothesis prl_ne2_ne: forall i, inc i (psI S) -> nonempty (Vg (psE S) i).
```

Bourbaki proves (b) as follows. Assume every E_i non-empty. Fix i, let $\mathrm{X}_i = f_{ij}\langle \mathrm{E}_j \rangle$. Since E_j is nonempty, it is non-empty. By (iv) it belongs to $\mathfrak{S}_i$. The family is left directed, for if $i \leq j \leq k$ then $\mathrm{X}_k \subset \mathrm{X}_j$, and given two indices j and j', there is an upper bound k. By (ii)' the intersection K_i is non-empty.

Now (a) implies (b) is as follows. First, there is nothing to prove if I is empty. So, let's take an index i. By (a), $\mathrm{K}_i = f_i(\mathrm{E})$, so that K_i non-empty says E non-empty.

```
Lemma prl_ne2_im_of_id i: inc i (psI S) ->
   (Imf (Vg (psf S) (J i i))) = Vg (psE S) i.
Lemma prl_ne2_res4b i: inc i (psI S) ->  prl_ne1_condii' i ->
  nonempty (prl_ne2_res_RHS i).
Lemma prl_ne2_nonempty_bis:
  (forall i, inc i (psI S) -> prl_ne1_condii' i) ->
  prl_ne2_res_a ->
  nonempty (projective_limit S).
```

We consider Σ, the set of families $\mathfrak{A} = (\mathrm{A}_\alpha)_{\alpha \in \mathrm{I}}$ which satisfy the following conditions

$$\mathrm{A}_\alpha \neq \emptyset, \quad \text{and} \quad \mathrm{A}_\alpha \in \mathfrak{S}_\alpha \quad \text{for all} \quad \alpha \in \mathrm{I}; \tag{3.18}$$

$$f_{\alpha\beta}\langle \mathrm{A}_\beta \rangle \subset \mathrm{A}_\alpha \quad \text{whenever} \quad \alpha \leq \beta. \tag{3.19}$$

The proof is divided in four steps. We first notice that Σ is non-empty (recall that E_i is non-empty). We order it by: $\mathrm{A} \leq \mathrm{A}'$ whenever $\mathrm{A}'_i \subset \mathrm{A}_i$ for every index $i \in \mathrm{I}$. Step 1 shows that Σ is an inductive set for this ordering. Let L be a nonempty totally ordered subset of Σ. Let $x_i = \bigcap_{z \in \mathrm{L}} z_i$. By (ii)', $(x_i)_{i \in \mathrm{I}}$ is in Σ, and is an upper bound of L.

```
Definition prl_ne2_sigma :=
  Zo (productb  FS_fam) (fun A =>
      (forall i, inc i (psI S) -> nonempty (Vg A i)) /\
      (forall i j, gle (psr S) i j ->
                   sub (Vfs (Vg (psf S) (J i j)) (Vg A j)) (Vg A i))).

Definition prl_ne2_sigma_le A A' :=
  forall i, inc i (psI S) -> sub (Vg A' i) (Vg A i).
Definition prl_ne2_sigma_order := graph_on prl_ne2_sigma_le prl_ne2_sigma.
Lemma prl_ne2_sigma_osr: order_on prl_ne2_sigma_order prl_ne2_sigma.
Lemma prl_ne2_sigma_ne: nonempty prl_ne2_sigma.

Lemma prl_ne2_sigma_inductive:
  (forall i, inc i (psI S) -> prl_ne1_condii i) ->
  inductive prl_ne2_sigma_order.
```

Step 2: $A_i = f_{ij}\langle A_j\rangle$ whenever $i \le j$ and A is maximal in Σ. Introduce $B_i = \bigcap_{j\ge i} f_{ij}\langle A_j\rangle$. By (iv), $f_{ij}\langle A_j\rangle \in \mathfrak{S}_i$. This is a non-empty set, so by (ii)', the intersection is non-empty; note that the set of $f_{ij}\langle A_j\rangle$ is left directed (for inclusion, fixed i, and j varying in I). This because if $i \le j \le k$ then $f_{ik}\langle A_k\rangle = f_{ij}\langle f_{jk}\langle A_k\rangle\rangle \subset f_{ij}\langle A_j\rangle$; so, if $i \le j$ and $i \le j'$ and k is an upper bound of j and j', $f_{ik}\langle A_k\rangle$ is a subset of $f_{ij}\langle A_j\rangle$ and $f_{ij'}\langle A_{j'}\rangle$. Now

$$f_{ij}\langle B_j\rangle \subset \bigcap_{k\ge j} f_{ij}\langle f_{jk}\langle A_k\rangle\rangle = \bigcap_{k\ge j} f_{ik}\langle A_k\rangle = \bigcap_{l\ge i} f_{il}\langle A_l\rangle = B_i.$$

We deduce $B \in \Sigma$. From $B_i \subset A_i$ (consider $j = i$ in the definition of B_i) we get that $B \ge A$, hence $B = A$ since A is maximal. Put $A_i = B_i$ in $B_i \subset f_{ij}\langle A_j\rangle$. Since the converse inclusion holds we have equality.

```
Lemma prl_ne2_sigma_maximal_prop1 A:
  (forall i, inc i (psI S) -> prl_ne1_condii' i) ->
  maximal prl_ne2_sigma_order A ->
  (forall i j, gle (psr S) i j -> (Vg A i)
                      = Vfs (Vg (psf S) (J i j)) (Vg A j)).
```

Step 3. Let A be as above. We pretend that each A_i is s singleton. We start by choosing $x_i \in A_i$. Let $C_j = f_{ij}^{-1}(x_i)$. We shall assume (iii)', in other words, we can chose x_i such that $C_j \subset \mathfrak{S}_j$. We set $B_j = A_j \cap C_j$ when $i \le j$ and $B_j = A_j$ otherwise. Then $B_j \in \mathfrak{S}_j$ by (i). By the previous result, $x_i \in f_{ij}\langle A_j\rangle$, so $x_i = f_{ij}(x_j)$ for some x_j, and B_j is non-empty. We have $f_{jk}\langle B_k\rangle \subset B_j$. In case $i \le j$ is false, we have $f_{jk}\langle B_k\rangle \subset f_{jk}\langle A_k\rangle$ and $B_j = A_j$, hence the conclusion. Otherwise $i \le j \le k$, and the property is easy. By maximality, B = A since A is maximal. This says: $A_i = A_i \cap C_i$ Now, $C_j = f_{ij}^{-1}(x_i) = \{x_i\}$.

```
Lemma prl_ne2_sigma_maximal_prop2 A:
  (forall i, inc i (psI S) -> prl_ne1_condii' i) ->
  pr1_ne2_hyp3_weak ->
  maximal prl_ne2_sigma_order A ->
  (forall i, inc i (psI S) -> singletonp (Vg A i)).
```

Step 4. Proof of the theorem. We have to show that if (iii) holds, then (a) and (b) are true, and when (iii)' holds then (b) holds. Note that (b) says that the projective limit is non-empty; we have shown above that (a) implies (b); we give here a direct proof. By step 1 and Zorn's lemma, Σ has a maximal element, a family $(A_i)_i$; by step 3, $A_i = \{x_i\}$, and by step 2 $x_i = f_{ij}(x_j)$. This says that the family (x_i) belongs to E.

```
Lemma prl_ne2_sigma_maximal_prop3 A (xi := fun i =>  union (Vg A i)):
  (forall i, inc i (psI S) -> prl_ne1_condii' i) ->
  pr1_ne2_hyp3_weak ->
  maximal prl_ne2_sigma_order A ->
  (forall i, inc i (psI S) -> (Vg A i) = singleton (xi i))
  /\ inc  (Lg (psI S) xi) (projective_limit S).
Lemma prl_ne2_sigma_maximal_ne A:
  (forall i, inc i (psI S) -> prl_ne1_condii' i) ->
  pr1_ne2_hyp3_weak ->
  maximal prl_ne2_sigma_order A ->
  nonempty (projective_limit S).

Lemma prl_ne2_nonempty:
  (forall i, inc i (psI S) -> prl_ne1_condii' i) ->
  pr1_ne2_hyp3_weak ->
  nonempty (projective_limit S).
```

We now prove (a). Fix i. Let $\mathrm{X}_j = f_{ij}\langle \mathrm{E}_j\rangle$. We have $f_i\langle \mathrm{E}\rangle \subset \mathrm{X}_j$ when $i \leq j$ (by definition of the projective limit). Let K be the intersection of the X_j. Then $f_i\langle \mathrm{E}\rangle \subset \mathrm{K}$. It remains to show that $\mathrm{K} \subset f_i\langle \mathrm{E}\rangle$. So take $x_i \in \mathrm{K}$. Define $\mathrm{B}_j = f_{ij}^{-1}(x_i)$. Bourbaki says: «Finally the proof of (4) shows that if $\mathrm{K} \neq \emptyset$ and if we choose an x_α in this set such that $\mathrm{B}_\beta \in \mathfrak{S}_\beta$ whenever $\beta \geq \alpha$, there exists $y \in \mathrm{E}$ such that $f_\alpha(y) = x_\alpha$, which proves our assertion.» We have shown above that if no E_i is empty, then K is non-empty. In this case (iii)' says that we can choose x_i such that $\mathrm{B}_j \in \mathfrak{S}_j$. We shall however assume (iii), it says $\mathrm{B}_j \in \mathfrak{S}_j$, whatever j. We shall prove that, under this condition, x_i is in the range of f_i. So: (iii) says that every element of K is in $f_i\langle \mathrm{E}\rangle$, and (iii)' says that at least one element of K is in $f_i\langle \mathrm{E}\rangle$. We extend the family B by setting $\mathrm{B}_j = \mathrm{E}_j$ for indices that do not satisfy $i \leq j$. By definition of x_i, B_j is non-empty. Clearly, it is in Σ. Since Σ is inductive (Step 1) there exists a maximal element A with $\mathrm{B} \leq \mathrm{A}$. Since (iii)' holds, we know that $\mathrm{A}_i = \{t_i\}$ and $t = (t_i)_i \in \mathrm{E}$. We have $f_i(t) = t_i$. However $\mathrm{A}_i \subset \mathrm{B}_i$ says $t_i \in \mathrm{B}_i$; since f_{ii} is the identity function $\mathrm{B}_i = \{x_i\}$, so that $t_i = x_i$. (total size of the definitions + proof: 600 lines).

```
Lemma prl_ne2_sigma_maximal_ne A:
  (forall i, inc i (psI S) -> prl_ne1_condii' i) ->
  pr1_ne2_hyp3_weak ->
  maximal prl_ne2_sigma_order A ->
  nonempty (projective_limit S).

Lemma prl_ne2_nonempty:
  (forall i, inc i (psI S) -> prl_ne1_condii' i) ->
  pr1_ne2_hyp3_weak ->
  nonempty (projective_limit S).
Lemma prl_ne2_prop:
  (forall i, inc i (psI S) -> prl_ne1_condii' i) ->
  pr1_ne2_hyp3_plain ->
  prl_ne2_res_a.
End ProjectiveLimitNonEmpty2.
```

3.5 Direct limits

«Let I be a *(right) directed* preordered set and let $(\mathrm{E}_\alpha)_{\alpha\in\mathrm{I}}$ be a family of sets indexed by I. For each pair (α,β) of elements of I such that $\alpha \leq \beta$, let $f_{\beta\alpha}$ be a *mapping of* E_α *into* E_β.

Suppose that the $f_{\beta\alpha}$ satisfy the following conditions:

(LI$_{\mathrm{I}}$) The relations $\alpha \leq \beta \leq \gamma$ imply $f_{\gamma\alpha} = f_{\gamma\beta} \circ f_{\beta\alpha}$.
(LI$_{\mathrm{II}}$) For each $\alpha \in \mathrm{I}$, $f_{\alpha\alpha}$ is the identity mapping of E_α.

By abuse of language, the pair $((\mathrm{E}_\alpha), (f_{\beta\alpha}))$ (which is usually written $(\mathrm{E}_\alpha, f_{\beta\alpha})$) is called a *direct system of sets*, relative to the index set I. »

We shall use the term "inductive" system instead of "direct" system, unless quoting Bourbaki. We shall use a record, as in the case of a projective system. We shall identify f_{ji} with $f_{(i,j)}$. This means that, for every $p \in r$ (where r is the preorder of I), f_p is a function $\mathrm{E}_{\mathrm{pr}_1(p)} \to \mathrm{E}_{\mathrm{pr}_2(p)}$.

```
Record inductive_system: Type := InductiveSystem {
  isE : Set;
  isI : Set;
  isr : Set;
  isf : Set;
  is_preorder_r: preorder isr;
  is_substrate_r: substrate isr = isI;
  is_directed_r: right_directed_on isr isI;
  is_fgraph_E: fgraph isE;
  is_domain_E: domain isE = isI;
  is_fgraph_f: fgraph isf;
  is_domain_f: domain isf = isr;
  is_function_f:
    forall p, inc p isr ->
              function_prop (Vg isf p) (Vg isE (P p)) (Vg isE (Q p));
  is_compose_f: forall i  j k, gle isr i j -> gle isr j k ->
                  Vg isf (J j k) \co  Vg isf (J i j) = Vg isf (J i k);
  is_identity_f: forall i, inc i isI -> Vg isf (J i i) = identity (Vg isE i)
}.

Definition inductive_system_on S E I r f :=
  [/\ isE S =  E, isI S = I, isr S = r & isf S = f].

Definition inl_same_data S S' :=
  [/\ isE S = isE S', isr S = isr S' & isf S = isf S'].
```

We start with trivial properties, the same as for projective systems.

```
Lemma inl_same_dataS S S':
  inl_same_data S S' -> inl_same_data S' S.
Lemma inl_same_dataT S S' S'' :
  inl_same_data S S' ->  inl_same_data S' S'' -> inl_same_data S S''.

Lemma inl_same_index_same_I S S':
  inl_same_index S S' -> isI S = isI S'.
Lemma inl_prop0 S i j: gle (isr S) i j -> inc i (isI S) /\ inc j (isI S).
Lemma inl_prop1 S i: inc i (isI S) -> inc (J i i) (isr S).
Lemma inl_prop2 S i j k: gle (isr S) i j -> gle (isr S) j k ->
  Vg (isf S) (J j k) \coP  Vg (isf S) (J i j).
Lemma inl_prop3 S y i j k (f:= isf S):
  gle (isr S) i j -> gle (isr S) j k -> inc y (Vg (isE S) i) ->
  Vf (Vg f (J j k))  (Vf (Vg f (J i j)) y) = Vf (Vg f (J i k)) y.
Lemma inl_prop4 S i j: gle (isr S) i j ->
  function_prop (Vg (isf S) (J i j)) (Vg (isE S) i) (Vg (isE S) j).
```

```
Lemma inl_prop5 S i x: inc i (isI S) -> inc x (Vg (isE S) i) ->
   Vf (Vg (isf S) (J i i)) x = x.
```

« Let G be the set which is the *sum* of the family of sets $(E_\alpha)_{\alpha\in I}$ (Chapter II, §4, no. 8); by abuse of language, we shall identify the E_α with their canonical images in G, and for each $x \in G$ we shall denote by $\lambda(x)$ the unique index $\alpha \in I$ such that $x \in E_\alpha$. » Recall that G is the union of the $E_i \times \{i\}$. The canonical image of E_i is the set of all (x, i) for $x \in E_i$, and $\lambda(x)$ is just the second projection of x. « Let $R\{x, y\}$ denote the following relation between two elements x, y of G: "there exists an element $\gamma \in I$ such that $\gamma \geq \alpha = \lambda(x)$ and $\gamma \geq \alpha = \lambda(y)$ for which $f_{\gamma\alpha}(x) = f_{\gamma\beta}(y)$." »

So we introduce the following definitions

```
Definition inl_sum S := disjointU (isE S).
Definition inl_equiv_rel S x y:=
  exists k, [/\ gle (isr S) (Q x) k,  gle (isr S) (Q y) k   &
   Vf (Vg (isf S) (J (Q x) k)) (P x) =  Vf (Vg (isf S) (J (Q y) k)) (P y) ].
Definition inl_equiv S := graph_on(inl_equiv_rel S)  (inl_sum S) .
```

The relation R is an equivalence relation. We shall denote by $\mathscr{C}_S(t)$ the class of t for this relation. Assume $i \leq j$, $x \in E_i$, and let $y = f_{ji}(x) \in E_j$. Then $\mathscr{C}_S(x, i) = \mathscr{C}_S(y, j)$. (Note that R is the least equivalence satisfying this property).

```
Lemma inl_sumP S x: inc x (inl_sum S) <->
  [/\ pairp x, inc (Q x) (isI S) & inc (P x) (Vg (isE S) (Q x))].

Lemma inl_equiv_reflexive S a: inc a (inl_sum S) -> inl_equiv_rel S a a.
Lemma inl_equiv_esr S: equivalence_on (inl_equiv S) (inl_sum S).

Lemma inl_class_eq S x y:
  inc x (inl_sum S) -> inc y (inl_sum S)  ->
  (class (inl_equiv S) x = class (inl_equiv S) y
     <-> inl_equiv_rel S x y).

Lemma inl_class_eq_bis S i j x y:
  inc i (isI S) -> inc j (isI S) ->
  inc x (Vg (isE S) i) ->  inc y (Vg (isE S) j) ->
  (class (inl_equiv S) (J x i) = class (inl_equiv S) (J y j)
   <-> inl_equiv_rel S (J x i) (J y j)).
Lemma inl_class_of_fij S i j x:
  gle (isr S) i j -> inc x (Vg (isE S) i) ->
  class (inl_equiv S) (J (Vf (Vg (isf S) (J i j)) x) j) =
  class (inl_equiv S) (J x i)/

Lemma inl_equivalence_prop S R:
  equivalence R ->
  (forall i j x,
    gle (isr S) i j -> inc x (Vg (isE S) i) ->
    related R (J x i) (J (Vf (Vg (isf S) (J i j)) x) j)) ->
  forall a b, related (inl_equiv S)  a b -> related R a b.
```

« The quotient set E = G/R is called the *direct limit of the family* $(E_\alpha)_{\alpha\in I}$ *with respect to the family of mappings* $(f_{\beta\alpha})$, and is written $E = \varinjlim(E_\alpha, f_{\beta\alpha})$ or simply $E = \varinjlim E_\alpha$. » We shall use the notation $\varinjlim S$ for the inductive limit of the inductive system S. We shall denote by $\mathscr{R}(t)$

a representative of the class t in the quotient. This is a pair (x,i) such that $i \in \mathrm{I}$, $x \in \mathrm{E}_i$ and $t = \mathscr{C}_{\mathrm{S}}(x,i)$. Clearly, the inductive limit is non-empty if and only if there is an index i such that E_i is non-empty.

```
Definition inductive_limit S := quotient (inl_equiv S).

Lemma inductive_limitP S x:
  inc x (inductive_limit S) <-> classp  (inl_equiv S) x.

Lemma inl_class_in_lim S i x:
   inc i (isI S) -> inc x (Vg (isE S) i) ->
   inc (class (inl_equiv S) (J x i)) (inductive_limit S).

Lemma inductive_limit_hi S x (i := (Q (rep x))) (y := P (rep x)):
  inc x (inductive_limit S) ->
  [/\ inc i (isI S), inc y (Vg (isE S) i) & x = class (inl_equiv S) (J y i)].
Lemma inl_limit_nonempty S:
  (exists2 i, inc i (isI S) & nonempty (Vg (isE S) i)) <->
  nonempty (inductive_limit S).
```

« We denote by f_α the restriction to E_α of the canonical mapping f of G into $\mathrm{E} = \mathrm{G}/\mathrm{R}$; f_α is called the *canonical mapping* of E_α into E. We have the relation

$$f_\beta \circ f_{\beta\alpha} = f_\alpha \qquad \text{when } \alpha \leq \beta. \tag{3.20}$$

»

If $i \in \mathrm{I}$ and $x \in \mathrm{E}_i$, Bourbaki identifies x in E_i and (x,i) in G and defines $f_i(x)$ to be $f((x,i))$. So $f_i(x) = \mathscr{C}_{\mathrm{S}}(x,i)$. Equation (3.20) just says that some functions can be composed and $\mathscr{C}_{\mathrm{S}}(x,i) = \mathscr{C}_{\mathrm{S}}(f_{ji}(x),j)$.

```
Definition inl_can_fun S i :=
  Lf (fun x => class (inl_equiv S) (J x i)) (Vg (isE S) i) (inductive_limit S).

Lemma inl_can_fun_ax S i :
  inc i (isI S) ->
  lf_axiom (fun x => class (inl_equiv S) (J x i)) (Vg (isE S) i)
           (inductive_limit S).

Lemma inl_can_fun_ev S i x:  inc i (isI S) -> inc x (Vg (isE S) i) ->
  Vf (inl_can_fun S i) x = class (inl_equiv S) (J x i).
Lemma inl_can_fun_fp S i: inc i (isI S) ->
  function_prop (inl_can_fun S i)  (Vg (isE S) i) (inductive_limit S).
Lemma inl_can_fun_prop S i j (f := isf S)
      (fi := inl_can_fun S i) (fj := inl_can_fun S j):
   gle (isr S) i j ->
   (fj \coP Vg f (J i j) /\  fi = fj \co (Vg f (J i j))).
```

We show that some quantities depend only on the data of S.

```
Lemma inl_equiv_Iv S S':
  inl_same_data S S' -> inl_equiv S = inl_equiv S'.
Lemma inductive_limit_Iv S S':
  inl_same_data S S' -> inductive_limit S = inductive_limit S'.
Lemma inl_can_fun_Iv S S' i:
  inl_same_data S S' ->
  inl_can_fun S i = inl_can_fun S' i.
```

Example 1. We consider a directed set I and a family of subsets V_i of A, indexed by I, such that $i \leq j$ implies $V_j \subset V_i$. We define E_i to be the set of functions $V_i \to B$. The function f_{ji} will be the restriction (it associates to a function $g : V_i \to B$ the function $g' : V_j \to B$ defined by $g'(x) = g(x)$). This is clearly an inductive system.

```
Section InjExample1.
Variable A B I V r: Set.

Hypotheses (or: preorder r)(sr: substrate r = I) (rdr: right_directed_on r I).
Hypothesis Vprop:
  [/\ fgraph V, domain V = I, (forall i, inc i I -> sub (Vg V i) A) &
    forall i j, gle r i j -> sub (Vg V j) (Vg V i)].

Definition Injex1_E := Lg I (fun i => functions (Vg V i) B).

Definition Injex1_ff p :=
  Lf (fun f => restriction f (Vg V (Q p)))
    (Vg Injex1_E (P p)) (Vg Injex1_E (Q p)).


Lemma Injex1_ff_ax p : inc p r ->
  lf_axiom (fun f => restriction f (Vg V (Q p)))
    (Vg Injex1_E (P p)) (Vg Injex1_E (Q p)).

Lemma Injex1_ff_p p: inc p r ->
   function_prop (Injex1_ff p) (Vg  Injex1_E (P p)) (Vg  Injex1_E (Q p)).

Definition  Injex1_system: inductive_system.
Lemma  Injex1_system_val:
  inductive_system_on Injex1_system Injex1_E I r (Lg r Injex1_ff).
End InjExample1.
```

Example 2. Suppose all the E_i equal to F, and take for f_{ij} the identity. We get an inductive system S; let E be the limit. The relation $\mathscr{C}_S(x,i) = \mathscr{C}_S(y,j)$ simplifies to $x = y$ (since there is k such that $i \leq k$ and $j \leq k$). Let $f(x) = \mathrm{pr}_1 \mathscr{R}(x)$; this induces an injection $\varinjlim S \to F$. It is surjective when I is non-empty. Bourbaki identifies $\varinjlim S$ and F.

```
Section InjExample2.
Variable  F I r: Set.

Hypotheses (or: preorder r)(sr: substrate r = I) (rdr: right_directed_on r I).

Definition  Injex2_system: inductive_system.

Lemma Inj_ex2_val:   inductive_system_on Injex2_system
    (cst_graph I F) I r (cst_graph r (identity F)).
Lemma Inj_ex2_can_prop x y (E := (inl_sum Injex2_system)):
  related (inl_equiv Injex2_system) x y <->
  [/\ inc x E, inc y E & P x = P y].

Lemma Inj_ex2_can_fun (E := (inductive_limit Injex2_system)):
  nonempty I ->
  bijection_prop (Lf (fun z => (P (rep z))) E F) E F.
End InjExample2.
```

Lemma 1. Let S be an inductive system, E the limit, f_i the canonical function. « (i) Let $(x^{(i)})_{1\leq i\leq n}$ be a finite system of elements of E. Then there exists $\alpha \in I$ and a finite system $(x_\alpha^{(i)})_{1\leq i\leq n}$ of elements of E_α such that $x^{(i)} = f_\alpha(x_\alpha^{(i)})$ for $1 \leq i \leq n$. (ii) Let $(y_\alpha^{(i)})_{1\leq i\leq n}$ be a finite system of elements of some E_α. If $f_\alpha(y_\alpha^{(i)}) = f_\alpha(y_\alpha^{(j)})$ for each pair of indices (i, j), then there exists $\beta \geq \alpha$ such that $f_{\beta\alpha}(y_\alpha^{(i)}) = f_{\beta\alpha}(y_\alpha^{(j)})$ for every pair (i, j).»

Note: we replace $1 \leq i \leq n$ by $I \in F$, where F is a finite set. In case $n = 0$, point (ii) becomes trivial and (i) holds when I is non-empty. Bourbaki uses the lemma only in the case $n = 2$; we never use it. The key relation is: every non-empty finite subset of I has an upper bound.

Proof of (i). We consider a family $(X_i)_{i\in K}$ of elements of E. Let $\mathscr{R}(X_i) = (x_i, b_i)$, where $b_i \in I$ and $x_i \in E_{b_i}$. Let $Y_i = f_{ab_i}(x_i)$. Then $X_i = f_a(Y_i)$, provided that $b_i \leq a$. It suffices to choose a large enough.

```
Lemma finite_preorder_directed_bounded r I E:
  preorder r -> substrate r = I -> right_directed_on r I ->
  nonempty E -> finite_set E -> sub E I ->
  exists2 x, inc x I & forall y, inc y E -> gle r y x.

Lemma inl_Lemma5_1i S X (K:= domain X)
  (Y := fun a => Lg K (fun i => Vf (Vg (isf S) (J (Q (rep (Vg X i))) a))
                           (P (rep (Vg X i))))):
  fgraph X -> finite_set K -> nonempty K ->
  (forall i, inc i K -> inc (Vg X i)  (inductive_limit S)) ->
  exists2 a, inc a (isI S) &
      [/\ fgraph (Y a), domain (Y a) = K &
     forall i, inc i K -> Vg X i =  Vf (inl_can_fun S a) (Vg (Y a) i) ].


Definition constant_fun_on f X := forall i j,
  inc i X -> inc j X -> Vf f i = Vf f j.

Lemma inl_Lemma5_1ii S i X:
  inc i (isI S) -> sub X (Vg (isE S) i) -> finite_set X ->
  constant_fun_on (inl_can_fun S i) X ->
  exists2 j, gle (isr S) i j & constant_fun_on (Vg (isf S) (J i j)) X.
```

3.6 Direct systems of mappings

Proposition 6. (This is the equivalent of Proposition 1 for inductive systems). We consider an inductive system S, the inductive limit E, the canonical mapping f_i. For each $i \in I$ we assume there exists a function $u_i : E_i \to F$ such that

$$u_j \circ f_{ji} = u_i \qquad \text{whenever } i \leq j. \tag{3.21}$$

Then (a) there exists a unique mapping u of E into F such that

$$u_i = u \circ f_i \qquad \text{for all } i \in I. \tag{3.22}$$

(b) u is surjective if and only if F is the union of the sets $u_i(E_i)$.
(c) u is injective if and only if for each $i \in I$, the relations $x \in E_i$, $y \in E_i$, $u_i(x) = u_i(y)$ imply that there exists $j \geq i$ such that $f_{ji}(x) = f_{ji}(y)$.

Assume $x \in \mathrm{E}_i$, $y \in \mathrm{E}_j$, $\mathscr{C}_\mathrm{S}(x,i) = \mathscr{C}_\mathrm{S}(y,j)$; there is k such that $i \leq k$, $j \leq k$, $f_{ki}(x) = f_{kj}(y)$ so that $u_i(x) = u_k(f_{ki}(x))$ and $u_j(x) = u_k(f_{kj}(y))$. Thus $u_i(x) = u_j(y)$. So the unique solution is obviously $u(t) = u_{\mathrm{pr}_2(\mathscr{R}(t))}(\mathrm{pr}_1(\mathscr{R}(t)))$. Bourbaki uses Lemma 1 for (c), but a direct proof is simpler. Note that (c) is $\mathrm{F} = \bigcup_{\mathrm{I}\in\mathrm{I}} u_i\langle \mathrm{E}_i\rangle$.

Bourbaki has a complicated argument to show existence: he starts by defining a mapping $v : \mathrm{G} \to \mathrm{F}$ that agrees with u_i on E_i for each $i \in \mathrm{I}$. This is possible as the union is disjoint (Chapter II, §4, no. 7, Proposition 8). In fact, if $x \in \mathrm{E}_i$ then $v(x,i) = u_i(x)$. This function is compatible with the equivalence relation R (Chapter II, §6, no. 5) [if (x,i) and (y,j) are related by the equivalence, then $v(x,i) = v(y,j)$, see above]. So, there is a unique mapping u such that $v = u \circ f$ (where f is the canonical mapping $\mathrm{G} \to \mathrm{G/R}$). If s is a section $\mathrm{G/R} \to \mathrm{G}$ of f, then $u = v \circ s$.

```
Definition inl_map_compat S u F:=
  [/\ fgraph u, domain u = isI S,
   forall i, inc i (isI S) -> function_prop (Vg u i) (Vg (isE S) i) F &
   forall i j, gle (isr S) i j -> (Vg u j) \co Vg (isf S) (J i j) = Vg u j].

Definition inl_map_property S u F g:=
  function_prop g (inductive_limit S) F /\
  forall i, inc i (isI S) -> (Vg u i) = g \co (inl_can_fun S i).

Definition inl_map_val u := fun y => Vf (Vg u (Q (rep y))) (P (rep y)).
Definition inductive_map S u F:=
  Lf (inl_map_val u) (inductive_limit  S) F.


Lemma inl_map_compat0 S u F i j x:
  inl_map_compat S u F -> gle (isr S) i j -> inc x (Vg (isE S) i) ->
  (Vf (Vg u i)) x = Vf (Vg u j) (Vf (Vg (isf S) (J i j)) x).
Lemma inl_map_property_res1 S u F g i x:
  inl_map_compat S u F -> inl_map_property S u F g ->
  inc i (isI S) -> inc x (Vg (isE S) i) ->
  Vf g (class (inl_equiv S) (J x i)) = Vf (Vg u i) x.
Lemma inl_map_unique S u F g g':
  inl_map_compat S u F ->
  inl_map_property S u F g -> inl_map_property S u F g' -> g = g'.

Lemma inl_map_prop  S u F:
  inl_map_compat S u F ->
  inl_map_property  S u F (inductive_map S u F).


Lemma inl_inductive_map_ev  S u F i x:
  inl_map_compat S u F -> inc i (isI S) -> inc x (Vg (isE S) i) ->
  Vf (inductive_map S u F) (class (inl_equiv S) (J x i)) = Vf (Vg u i) x.

Lemma inl_map_surjective  S u F:
  inl_map_compat S u F ->
  (surjection (inductive_map S u F) <->
    F = unionf (isI S) (fun i => Imf (Vg u i))).

Lemma inl_map_injective  S u F:
  inl_map_compat S u F ->
  (injection (inductive_map S u F) <->
   forall i x y, inc i (isI S) ->
```

```
                inc x (Vg (isE S) i) -> inc y (Vg (isE S) i) ->
                Vf (Vg u i) x = Vf (Vg u i) y ->
    exists2 j, gle (isr S) i j &
        Vf (Vg (isf S) (J i j)) x =  Vf (Vg (isf S) (J i j)) y).
```

Remark. If every f_{ij} is injective, so is f_i. In this case, one can identify E_i with $f_i\langle \mathrm{E}_i\rangle$, so consider E as the union of the E_i.

```
Lemma inl_can_fun_inj S:
  (forall p, inc p (isr S) -> injection (Vg (isf S) p)) ->
  (forall i, inc i (isI S) -> injection (inl_can_fun S i)).
```

Converse. Consider a directed set I, an increasing family of sets F_i indexed by I ($i \leq j$ implies $\mathrm{F}_i \subset \mathrm{F}_j$). Take for f_{ij} the canonical injections $\mathrm{F}_i \to \mathrm{F}_j$. This gives an inductive system S. Let $\mathrm{F} = \varinjlim \mathrm{F}_i$.

```
Lemma ci_fp A B: sub A B -> function_prop (canonical_injection A B) A B.

Lemma ci_compose A B C (fAB := canonical_injection A B)
      (fBC := canonical_injection B C)(fAC := canonical_injection A C):
  sub A B -> sub B C -> fBC \co fAB = fAC.

Lemma ci_image A B: sub A B ->
   Imf (canonical_injection A B) = A.

Section InlRemark.

Variables (I r F:Set).

Hypotheses (or: preorder r)(sr: substrate r = I) (rdr: right_directed_on r I).
Hypotheses (fgF: fgraph F) (df: domain F = I).
Hypothesis Fmon: forall i j, gle r i j -> sub (Vg F i) (Vg F j).

Definition inl_remark_f :=
  Lg r (fun p => (canonical_injection (Vg F (P p)) (Vg F (Q p)))).

Definition inl_remark_S: inductive_system.
Lemma inl_remark_S_prop:
   inductive_system_on  inl_remark_S  F I r inl_remark_f.
```

Let u_i be the canonical injection $\mathrm{F}_i \hookrightarrow \bigcup \mathrm{F}_i$. This family of functions satisfies the property of the Proposition, and by (b) and (c), there is is a bijection $u : \varinjlim \mathrm{F}_i \to \bigcup \mathrm{F}_i$. This means that the two sets can be identified. The relation $u_i = u \circ f_i$ allows us to identify f_i with u_i.

```
Definition inl_remark_U := unionb F.
Definition inl_remark_ui :=
  Lg I (fun i => canonical_injection (Vg F i)  inl_remark_U).


Lemma inl_remark_sub i: inc i I ->  sub (Vg F i) inl_remark_U.
Lemma inl_remark_compat: inl_map_compat inl_remark_S inl_remark_ui inl_remark_U.

Lemma inl_remark_bijection:
  bijection_prop (inductive_map inl_remark_S inl_remark_ui inl_remark_U)
```

```
    (inductive_limit inl_remark_S) inl_remark_U.

End InlRemark.
```

Corollary 1. [Compare with corollary 1 of Proposition 1]. « Let $(E_\alpha, f_{\beta\alpha})$ and $(F_\alpha, g_{\beta\alpha})$ be two direct systems of sets relative to the same index set I; let $E = \varinjlim E_\alpha$, $F = \varinjlim F_\alpha$, and for each $\alpha \in I$ let f_α (resp. g_α) be the canonical mapping of E_α (resp. F_α) into E (resp. F). For each $\alpha \in I$ let u_α be a mapping of E_α into F_α such that

$$g_{\beta\alpha} \circ u_\alpha = u_\beta \circ f_{\beta\alpha} \qquad \text{whenever } \alpha \leq \beta. \tag{3.23}$$

Then there exists a unique mapping $u : E \to F$ such that

$$g_\alpha \circ u_\alpha = u \circ f_\alpha \qquad \text{whenever } \alpha \in I. \tag{3.24}$$

A family of functions that satisfies the conditions of Corollary 1 is called a *direct system of mappings* of $(E_\alpha, f_{\beta\alpha})$ into $(F_\alpha, g_{\beta\alpha})$ and the mapping defined by Corollary 1 is called the *direct limit* of the family (u_α) and is written $u = \varinjlim u_\alpha$ when there is no risk af ambiguity. » The associated commutative diagrams are

$$\begin{array}{ccc} E_i & \xrightarrow{u_i} & F_i \\ {\scriptstyle f_{ji}}\downarrow & & \downarrow{\scriptstyle g_{ji}} \\ E_j & \xrightarrow[u_j]{} & F_j \end{array} \qquad\qquad \begin{array}{ccc} E_i & \xrightarrow{u_i} & F_i \\ {\scriptstyle f_i}\downarrow & & \downarrow{\scriptstyle g_i} \\ E & \xrightarrow[u]{} & F \end{array}$$

The idea is to apply Proposition 6 to the functions $g_i \circ u_i$.

```
Definition inl_map2_compat S S' u :=
  [/\ fgraph u, domain u = isI S,
   forall i, inc i (isI S) ->
             function_prop (Vg u i) (Vg (isE  S) i) (Vg (isE S') i) &
   forall i j, gle (isr S) i j ->
     Vg (isf S') (J i j) \co Vg u i =  Vg u j \co Vg (isf S) (J i j) ].

Definition inl_map2_property S S' u g :=
  function_prop g  (inductive_limit S)  (inductive_limit S')
  /\ forall i,  inc i (isI S) ->
    (inl_can_fun S' i) \co (Vg u i) = g \co (inl_can_fun S) i.

Definition inl_map2_aux S u :=
  Lg (psI S) (fun i =>  (inl_can_fun S i) \co (Vg u i)).

Lemma inl_map2_compat_prop0 S S' u x i j:
  inl_same_index S S' ->  inl_map2_compat S S' u ->
  inc x (Vg (isE S) i) ->  gle (isr S) i j ->
  Vf (Vg (isf S') (J i j)) (Vf (Vg u i) x) =
  Vf (Vg u j) (Vf (Vg (isf S) (J i j)) x).

Lemma inl_map2_compat_prop1 S S' u x i j:
  inl_same_index S S' ->  inl_map2_compat S S' u ->
  inc x (Vg (isE S) i) ->  gle (isr S) i j ->
  class (inl_equiv S') (J (Vf (Vg u i) x) i) =
  class (inl_equiv S') (J (Vf (Vg u j) (Vf (Vg (isf S) (J i j)) x)) j).
Lemma inl_map2_prop1 S S' u:
  inl_same_index S S' ->  inl_map2_compat S S' u ->
  inl_map_compat S (inl_map2_aux S' u) (inductive_limit S').
```

We can now define u.

```
Definition inductive_limit_fun S S' u :=
  inductive_map S (inl_map2_aux S' u) (inductive_limit S').


Lemma inl_map2_prop S S' u (g := inductive_limit_fun S S' u):
  inl_same_index S S' ->  inl_map2_compat S S' u ->
  inl_map2_property S S' u g.

Lemma inl_map2_prop2 S  u i t:
  inc i (isI S) -> inc t (source (Vg u i)) ->
  function (Vg u i) ->  target (Vg u i) = Vg (isE S) i ->
  Vf (Vg (inl_map2_aux S u) i) t = class (inl_equiv S) (J (Vf (Vg u i) t) i).

Lemma inl_map2_unique  S S' u g g':
  inl_same_index S S' ->  inl_map2_compat S S' u->
  inl_map2_property S S' u g ->  inl_map2_property S S' u  g' -> g = g'.

Lemma inl_inductive_limit_fun_IV2 S1 S2 x S1' S2' x':
  inl_same_data S1 S1' ->  inl_same_data S2 S2' -> x = x' ->
  inductive_limit_fun S1 S2 x = inductive_limit_fun S1' S2' x'.

Lemma inl_map_val_aux2 S S' u i x (f := inductive_limit_fun S S' u) :
  inl_same_index S S' -> inl_map2_compat S S' u ->
  inc i (isI S) ->  inc x (Vg (isE S) i) ->
  Vf f (class (inl_equiv S) (J x i)) =
  class (inl_equiv S') (J  (Vf (Vg u i) x) i).

Lemma inl_map2_prop3 S S' u (f := inductive_limit_fun S S' u):
  inl_same_index S S' -> inl_map2_compat S S' u ->
  function_prop f (inductive_limit S)  (inductive_limit S') /\
  forall i x,
  inc i (isI S) ->  inc x (Vg (isE S) i) ->
  Vf f (class (inl_equiv S) (J x i)) =
  class (inl_equiv S') (J  (Vf (Vg u i) x) i).
```

Corollary 2. Consider three systems S, S′, S″ and functions $u_i : \mathrm{E}_i \to \mathrm{E}'_i$, $v_i : \mathrm{E}'_i \to \mathrm{E}''_i$. Then $v_i \circ u_i$ is an inductive system of mappings and

$$\varinjlim(v_i \circ u_i) = (\varinjlim v_i) \circ (\varinjlim u_i). \tag{3.25}$$

```
Lemma inl_map2_compose S S' S'' u v (F := inductive_limit_fun)
  (w:= Lg (isI S) (fun i => (Vg v i) \co (Vg u i))) :
  inl_same_index S S' ->  inl_same_index S' S'' ->
  inl_map2_compat S S' u -> inl_map2_compat S' S'' v ->
  inl_map2_compat S S'' w /\
  F S S'' w = F S' S'' v \co F S S' u.
```

Proposition 7. Same assumptions as in Corollary 1. If each u_i is injective (resp. surjective), so is $\varinjlim u_i$.

Bourbaki proves injectivity via Proposition 6 and Lemma 1. The direct proof is as follows. Assume $u(x) = u(y)$. Let's assume that $x = \mathscr{C}_{\mathrm{S}}(x', i)$, so that $u(x) = \mathscr{C}_{\mathrm{S}'}((u_i(x'), i)$. Similarly, $u(y) = \mathscr{C}_{\mathrm{S}'}(u_j(y'), j)$. So, there is k such that $i \leq k$, $j \leq k$ and $f'_{ki}(u_i(x')) = f'_{kj}(u_j(y'))$. This

can be rewritten as $u_k(f_{ki}(x')) = u_k(f_{ki}(y'))$. By injectivity of u_k, (x', i) and (y', j) belong to the same class. Surjectivity: if $y \in \varinjlim \mathrm{S}'$, then there is y' and i such that $y = \mathscr{C}_{\mathrm{S}'}(y', i)$. Since $y' \in \mathrm{E}'_i$ and u_i is surjective, there is x' such that $u_i(x') = y'$. Then $u(\mathscr{C}_{\mathrm{S}}(x', i)) = y$. This is simpler than the argument of Bourbaki.

```
Lemma inl_limit_fun_inj S S' u:
  inl_same_index S S' -> inl_map2_compat S S' u ->
  (forall i, inc i (isI S) ->   injection (Vg u i)) ->
  injection (inductive_limit_fun S S' u).

Lemma inl_limit_fun_surj S S' u:
  inl_same_index S S' -> inl_map2_compat S S' u ->
  (forall i, inc i (isI S) -> surjection (Vg u i)) ->
  surjection (inductive_limit_fun S S' u).
```

An *inductive system of subsets of the* E_i is a family M_i indexed by I such that $\mathrm{M}_i \subset \mathrm{E}_i$ and $f_{ji}\langle \mathrm{M}_i \rangle \subset \mathrm{M}_j$. Let g_{ji} be the restriction of f_{ji} as a function $\mathrm{M}_i \to \mathrm{M}_j$. Then (M_i, g_{ij}) is an inductive system.

```
Definition inl_subfam_hyp S M:=
  [/\ fgraph M, domain M = isI S,
   forall i, inc i (isI S) -> sub (Vg M i) (Vg (isE S) i) &
   forall i j, gle (isr S) i j ->
     sub (Vfs (Vg (isf S) (J i j)) (Vg M i)) (Vg M j) ].

Definition inl_subfam_fct S M :=
  Lg  (isr S) (fun z => restriction2 (Vg (isf S) z) (Vg M (P z)) (Vg M (Q z))).

Lemma inl_subfam_prop1 S M (g := inl_subfam_fct S M):
  inl_subfam_hyp S M ->
  [/\
   forall z, inc z (isr S) ->
      restriction2_axioms (Vg (isf S) z) (Vg M (P z)) (Vg M (Q z)),
   forall i j x, gle (isr S) i j -> inc x (Vg M i) ->
     Vf (Vg g (J i j)) x = Vf (Vg (isf S) (J i j)) x,
   forall i, inc i (isr S)-> function_prop (Vg g i) (Vg M (P i)) (Vg M (Q i)),
   forall i j k, gle (isr S) i j -> gle (isr S) j k ->
                 Vg g (J j k) \co  Vg g (J i j) = Vg g (J i k) &
   forall i, inc i (isI S) -> Vg g (J i i) = identity (Vg M i)].


Definition inductive_system_subsets
           S M (H:inl_subfam_hyp S M) : inductive_system.
Lemma inl_subsets_prop S  M (H:inl_subfam_hyp S M) :
  inductive_system_on (inductive_system_subsets H)
     M  (isI S)  (isr S) (inl_subfam_fct S M).


Lemma inl_subsets_prop_Iv S M
      (H H':inl_subfam_hyp S M) :
  inl_same_data (inductive_system_subsets H) (inductive_system_subsets H').

Lemma inl_subsets_prop_I2v S S' M
      (H:inl_subfam_hyp S M) (H':inl_subfam_hyp S' M) :
  inl_same_data S S' ->
  inl_same_data (inductive_system_subsets H) (inductive_system_subsets H').
```

One can apply Proposition 7, taking for u_i the canonical injection $j_i : \mathrm{M}_i \hookrightarrow \mathrm{E}_i$. This gives an injection $\varinjlim j_i : \varinjlim \mathrm{M}_i \to \varinjlim \mathrm{E}_i$. It maps $\mathscr{C}_{\mathrm{M}}(x,i)$ to $\mathscr{C}_{\mathrm{S}}(x,i)$, for every $x \in \mathrm{M}_i$. So, $\varinjlim \mathrm{M}_i$ can be identified to a subset of $\varinjlim \mathrm{E}_i$.

```
Lemma inl_subfam_compat S M
   (H:inl_subfam_hyp S M) (S' := (inductive_system_subsets H))
   (ji := fun i => canonical_injection (Vg M i) (Vg (isE S) i)):
  inl_map2_compat S' S (Lg (isI S) ji).

Lemma inl_subfam_prop3 S M
   (H:inl_subfam_hyp S M) (S' := (inductive_system_subsets H))
   (ji := fun i => canonical_injection (Vg M i) (Vg (isE S) i))
   (u := (inductive_limit_fun S' S (Lg (isI S) ji))):
  forall i x, inc i (isI S) ->  inc x (Vg M i) ->
        Vf u (class (inl_equiv S') (J x i)) =  class (inl_equiv S) (J x i).

Lemma inl_subfam_prop4 S M
      (H:inl_subfam_hyp S M) (S' := (inductive_system_subsets H))
      (ji := fun i => canonical_injection (Vg M i) (Vg (isE S) i)):
      injection_prop (inductive_limit_fun S' S (Lg (isI S) ji))
                     (inductive_limit S') (inductive_limit S).
```

Corollary. « Let $(\mathrm{E}_\alpha, f_{\beta\alpha})$ and $(\mathrm{E}'_\alpha, f'_{\beta\alpha})$ be two direct systems of sets, let (u_α) be a direct system of mappings, $u_\alpha : \mathrm{E}_\alpha \to \mathrm{E}'_\alpha$, and let $u = \varinjlim u_\alpha$.
(i) Let (M_α) be a direct system of subsets of the E_α. Then $(u_\alpha(\mathrm{M}_\alpha))$ is a direct system of subsets of the E'_α and we have

$$\varinjlim u_\alpha(\mathrm{M}_\alpha) = u(\varinjlim \mathrm{M}_\alpha). \tag{3.26}$$

(ii) Let $(a'_\alpha)_{\alpha\in\mathrm{I}}$ be a family such that $a'_\alpha \in \mathrm{E}'_\alpha$ for each $\alpha \in \mathrm{I}$ and $f'_{\beta\alpha}(a'_\alpha) = a'_\beta$ whenever $\alpha \le \beta$. Then the sets $\overset{-1}{u}{}_\alpha(a'_\alpha)$ form a direct system of subsets of the E_α and we have

$$\varinjlim \overset{-1}{u}{}_\alpha(a'_\alpha) = \overset{-1}{u}(a') \tag{3.27}$$

where a' is the unique element of $\varinjlim \mathrm{E}'_\alpha$ which is the canonical image of a'_α for each $\alpha \in \mathrm{I}$. »

We open a section, introducing S, S′ and u, then two sections, one for (i), and one for (ii).

```
Section InductiveLimitCorollary.

Variables S S': inductive_system.
Variable u: Set.
Hypothesis sii:inl_same_index S S'.
Hypothesis m2c: inl_map2_compat S S' u.
```

For (i), Bourbaki introduces the function $v_i : \mathrm{M}_i \to u_i\langle \mathrm{M}_i\rangle$ that coincides with u_i on M_i. If j'_i is the canonical injection of $u_i\langle \mathrm{M}_i\rangle$ into E'_i, then $u_i = v_i \circ j_i$. We have $u_i\langle \mathrm{M}_i\rangle = v_i\langle \mathrm{M}_i\rangle$, v_i is surjective, so that, by Proposition 7, $\varinjlim v_i$ is surjective. The reality is a bit more complicated.

Define $\mathrm{M}'_i = u_i\langle \mathrm{M}_i\rangle$. This is obviously an inductive systems of subsets of E'_i.

```
Section InductiveLimitCorollary1.

Variable M: Set.
```

```
Hypothesis Mhyp: inl_subfam_hyp S M.

Definition inl_p7c1_M' :=
  Lg (isI S) (fun i => Vfs (Vg u i) (Vg M i)).

Lemma inl_sub_fam_im1: inl_subfam_hyp S' inl_prop7_cor_M'.
```

We define now S_M and S'_M the two inductive systems associated to M_i and M'_i. So $\varinjlim M_i = \varinjlim S_M$ and $\varinjlim u_i\langle M_i\rangle = \varinjlim S'_M$. Let j_i and j'_i be the canonical inclusions $M_i \hookrightarrow E_i$ and $M'_i \hookrightarrow E'_i$. In (3.26), Bourbaki identifies the source and image of $\varinjlim j_i$ and $\varinjlim j'_i$; the correct formula is

$$\varinjlim j'_i\langle\varinjlim S'_M\rangle = \varinjlim u_i\langle\varinjlim j_i\langle\varinjlim S_M\rangle\rangle.$$

Let A be the image of $\varinjlim j_i$ and A′ the image of $\varinjlim j'_i$. The formula becomes $u\langle A\rangle = A'$. Note that A is a subset of the source of u, so that the formula makes sense.

Assume $t \in u\langle A\rangle$. There is $i \in I$ and $x \in M_i$ such that $t = u(\mathscr{C}_S(x,i)) = \mathscr{C}_{S'}(u_i(x),i)$. By definition $u_i(x) \in M'_i$ so that $t \in A'$. Conversely if $t \in A'$ then $t = \mathscr{C}_{S'}(y,i)$ where $y \in M'_i$ so that $y = u_i(x)$ for some $x \in M_i$. So $t = u(\mathscr{C}_S(x,i))$. We conclude by noting that the argument of u belongs to A.

```
Definition inl_p7c1_MS :=  inductive_system_subsets Mhyp.
Definition inl_p7c1_MS':= inductive_system_subsets  inl_sub_fam_im1.

Definition inl_p7c1_ji :=
 Lg (isI S)( fun i => canonical_injection (Vg M i) (Vg (isE S) i)).

Definition inl_p7c1_ji' :=
 Lg (isI S')( fun i => canonical_injection (Vg inl_p7c1_M' i) (Vg (isE S') i)).

Definition inl_p7c1_ji_lim := inductive_limit_fun inl_p7c1_MS S inl_p7c1_ji.
Definition inl_p7c1_ji_lim' := inductive_limit_fun inl_p7c1_MS' S' inl_p7c1_ji'.


Lemma inl_p7c1_ji_prop :
 injection_prop  inl_p7c1_ji_lim
 (inductive_limit  inl_p7c1_MS) (inductive_limit S).
Lemma inl_p7c1_ji'_prop :
 injection_prop inl_p7c1_ji_lim'
 (inductive_limit inl_p7c1_MS') (inductive_limit S').


Lemma inl_prop7_cor_i:
  Imf inl_p7c1_ji_lim' = Vfs (inductive_limit_fun S S' u) (Imf inl_p7c1_ji_lim).
End InductiveLimitCorollary1.
```

Part (ii). We first show that if $M'_i \subset E'_i$ is such that $f'_{ji}\langle M'_i\rangle \subset M'_j$, then the family $u_i^{-1}(M'_i)$ is a direct family of subsets. We then consider $M'_i = \{a'_i\}$ and assume $f'_{ji}(a'_i) = a'_j$. Then whenever i and j belong to I, we have $\mathscr{C}_{S'}(a'_i,i) = \mathscr{C}_{S'}(a_j,j)$. This relation holds in particular when j is the representative of I (note: $i \in I$ says that I is non-empty). We write $N_i = u_i^{-1}(a'_i)$, and let j_i be the canonical injection $N_i \subset E_i$. We define S″ to be the inductive system associated to N_i, and $\varinjlim j_i$ the inductive limit of the canonical injections, this is an injection $\varinjlim S'' \to \varinjlim S$.

```
Definition inl_inv_image_compat Mi:=
```

```
  [/\ fgraph Mi,
   domain Mi = isI S',
   forall i, inc i (isI S') -> sub (Vg Mi i) (Vg (isE S') i) &
   forall p, inc p (isr S') ->
          sub (Vfs (Vg (isf S') p)(Vg Mi (P p))) (Vg Mi (Q p))].
Lemma inl_sub_fam_im2 Mi
  (Mi' := Lg (isI S) (fun i => Vfi (Vg u i) (Vg Mi i))):
  inl_inv_image_compat Mi ->
  inl_subfam_hyp S Mi'.

Definition inl_inv_image_compat1 ai:=
  [/\ fgraph ai,
   domain ai = isI S',
   forall i, inc i (isI S') -> inc (Vg ai i) (Vg (isE S') i) &
   forall p, inc p (isr S') -> Vf (Vg (isf S') p)(Vg ai (P p)) = (Vg ai (Q p))].


Section InductiveLimitCorollary2.

Variable a_fam: Set.
Hypothesis a_fam_prop: inl_inv_image_compat1 a_fam.

Definition inl_p7c2_Ni :=
   Lg (isI S) (fun i => Vfi1 (Vg u i) (Vg a_fam i)).
Definition inl_p7c2_ci :=
  Lg (isI S) (fun i => canonical_injection (Vg inl_p7c2_Ni i) (Vg (isE S) i)).


Lemma inl_sub_fam_im3: inl_subfam_hyp S inl_7c2_Ni.
Lemma inl_sub_fam_im3_val:
  forall i, inc i (isI S) ->
    class (inl_equiv S') (J (Vg a_fam i) i)
    = class (inl_equiv S') (J (Vg a_fam (rep (isI S))) (rep (isI S))).

Definition inl_p7c2_S'' := (inductive_system_subsets inl_sub_fam_im3).
Definition inl_p5c2_ip := inductive_limit_fun inl_p7c2_S'' S inl_p7c2_ci.
```

Let's prove the result:

$$\varinjlim j_i\langle\varinjlim \mathrm{S}''\rangle = u^{-1}(\bar{a})$$

Note that the RHS of this equation is a subset of the source of u, so a subset of $\varinjlim \mathrm{S}$; we pretend that it is the image of $\varinjlim j_i$ (recall that Bourbaki identifies the source $\varinjlim \mathrm{S}''$ with the image).

Consider first $t \in u^{-1}(\bar{a})$. This means $u(t) = \bar{a}$, where $t \in \varinjlim \mathrm{S}$, so that $t = \mathscr{C}_{\mathrm{S}}(x_i, i)$ for some $i \in \mathrm{I}$ and $x_i \in \mathrm{E}_i$. By definition of u, we have $u(t) = \mathscr{C}_{\mathrm{S}'}(u(x_i), i)$. On the other hand, $\bar{a}$ is (by definition) $\mathscr{C}_{\mathrm{S}'}(a_k, k)$ for some $k \in \mathrm{I}$ (as shown above, this is independent of k and we may chose $k = i$). So $\mathscr{C}_{\mathrm{S}'}(u(x_i), i) = \mathscr{C}_{\mathrm{S}'}(a_i, i)$ and there exists j, such that $i \leq j$ and $f'_{ji}(a_i) = f'_{ji}(u_i(x_i))$. This simplifies to $a'_j = u_i(f_{ji}(x_i))$, and says that $y = f_{ji}(x_i)$ belongs to N_j. Now $t = \mathscr{C}_s(x_i, i) = \mathscr{C}_s(y, j) = \varinjlim j_i(\mathscr{C}_{\mathrm{S}''}(y, i))$, so that t is in the image of $\varinjlim j_i$. Converse. Assume that t is in the image, so $t = \mathscr{C}_{\mathrm{S}''}(x_i, i)$ for some $x_i \in \mathrm{N}_i$. In particular $x_i \in \mathrm{E}_i$ and $u_i(x_i) = a_i$. By definition of S'', $t = \mathscr{C}_{\mathrm{S}}(x_i, i)$, hence $t \in \varinjlim \mathrm{S}$. As above $u(t) = \mathscr{C}_{\mathrm{S}'}(u_i(x_i), i) = \mathscr{C}_{\mathrm{S}'}(a_i, i) = a$.

```
Lemma inl_sub_fam_im4:
  injection_prop inl_p7c2_ip (inductive_limit inl_p7c2_S'') (inductive_limit S).
```

```
Lemma inl_prop7_cor_ii
      (a := class (inl_equiv S') (J (Vg a_fam (rep (isI S))) (rep (isI S)))):
  Imf inl_p7c2_ip = Vfi1 (inductive_limit_fun S S' u)  a.

End InductiveLimitCorollary2.
End InductiveLimitCorollary.
```

Remark. Assume that u_i is a family of functions satisfying (3.21), with target E' instead of F. According to example 2, define $\mathrm{E}'_i = \mathrm{E}'$ and take for f_{ji} the identity function. This gives an inductive system S', and $\varinjlim \mathrm{S}'$ can be identified with E'. If u_i is considered as a mapping $\mathrm{E}_i \to \mathrm{E}'_i$, then (u_i) is a direct system of mappings, and the function u defined by (3.22) can be identified with $\varinjlim u_i$.

```
Section InlRemark2.

Variables (S: inductive_system) (u E': Set).
Hypothesis mcu: inl_map_compat S u E'.

Definition inl_rem2_S' := Injex2_system E' (is_preorder_r S)
                                        (is_substrate_r S) (@is_directed_r S).

Lemma inl_rem2_prop1: inl_map2_compat S inl_rem2_S' u.

Lemma inl_rem2_prop2 (u1 := inductive_map S u E')
      (u2:= inductive_limit_fun S  inl_rem2_S' u)
      (can := Lf (fun z => (P (rep z))) (inductive_limit inl_rem2_S') E'):
  nonempty (isI S) ->
  can \coP u2 /\ u1 = can \co u2.
End InlRemark2.
```

Consider a system S and a subset J of I, assumed to be right directed (every pair in J is bounded above by an element of J); this implies that the preorder r' induced on J is right directed. If we restrict the sets and functions of S to J and r' we get an inductive system, it is said to be obtained by *restricting* the index set to J. Let f_i be the canonical mapping; then the $(f_j)_{j\in\mathrm{J}}$ form an inductive system of mappings. Let $g = \varinjlim f_i$. This function is called *canonical*.

```
Definition sub_right_directed J r :=
   sub J (substrate r) /\ (right_directed_on r J).
Definition inl_restr S J (H:sub_right_directed J(isr S)) : inductive_system.

Lemma inl_restr_prop S J (H:sub_right_directed J(isr S)) :
  inductive_system_on  (inl_restr H)
      (restr (isE S) J) J (induced_order (isr S) J)
      (restr (isf S) (induced_order (isr S) J)).

Lemma inl_restr_cf_compat  S J (H:sub_right_directed J(isr S)):
  inl_map_compat (inl_restr H) (Lg J (inl_can_fun S)) (inductive_limit S).

Definition inl_restr_cf  S J (H:sub_right_directed J(isr S)):=
  (inductive_map (inl_restr H) (Lg J (inl_can_fun S)) (inductive_limit S)).

Lemma inl_restr_cf_compat2 S J (H:sub_right_directed J(isr S)):
  function_prop (inl_restr_cf H)
```

```
     (inductive_limit (inl_restr H)) (inductive_limit S).
Lemma inl_restr_cf_ev S J (H:sub_right_directed J (isr S)) i x:
      inc i J ->  inc x (Vg (isE S) i) ->
      Vf (inl_restr_cf H) (class (inl_equiv (inl_restr H)) (Pair.J x i)) =
        class (inl_equiv S) (Pair.J x i).
```

Assume now that $J \subset I$ is right directed, let S' be the system obtained from S by restricting indices to J, and g the canonical functional. Assume now that J' is a right directed subset of the index set of S', and define S'' and g' accordingly. Since J' is a directed subset of I we can define S''' and g'' and we have

$$g'' = g \circ g'. \tag{3.28}$$

```
Lemma sub_right_directed_trans J J' r:
  preorder r ->
   sub_right_directed J r ->
   sub_right_directed J' (induced_order r J) ->
   sub_right_directed J' r.
Lemma inl_restr_canonical_comp S J J'
      (H: sub_right_directed J (isr S))
      (S' := inl_restr H)
      (H': sub_right_directed J' (isr S'))
      (g1 := inl_restr_cf H)(g2 := inl_restr_cf H')
      (g3 := inl_restr_cf (sub_right_directed_trans (is_preorder_r S) H H')):
   g1 \coP g2 /\ g3 = g1 \co g2.
```

Proposition 8. « Let I be a directed set, let $(E_\alpha, f_{\beta\alpha})$ be a direct system of sets relative to I, and let $\varinjlim E_\alpha$ be its direct limit. Let J be a cofinal subset of I, and let E' be the direct limit of the direct system of sets obtained from $(E_\alpha, f_{\beta\alpha})$ by restricting the index set to J. Then the canonical mapping g of E' into E is bijective. »

Note: if i and j belong to J they are bounded above in I by some k, so that there an upper bound $k' \in J$ of i and j. Injectivity: we use Proposition 6; assume $f_i(x) = f_i(y)$, so that for some k, $f_{ki}(x) = f_{ki}(y)$. Since J is cofinal, we may assume $k \in J$, and conclude. Surjectivity: we could use Proposition 6, but the direct proof is shorter. Let $t = \mathscr{C}_S(x, i)$, where $i \in I$, and $j \in J$ such that $i \leq j$. If $y = f_{ji}(x)$, then $t = \mathscr{C}_S(y, j) = g(\mathscr{C}_{S'}(y, j))$.

```
Lemma cofinal_directed S J:
  cofinal (isr S) J -> sub_right_directed J (isr S).
Lemma inl_restr_cofinal S J (H:cofinal (isr S) J)
  (H' :=(cofinal_directed H)):
  bijection (inl_restr_cf H').
```

3.7 Double Direct Limit. Product of Direct Limits

[Compare with section "double inverse limit"]. Assume that we have two directed sets, I and L and an inductive family S on $I \times L$. Bourbaki denotes the sets by E_α^λ and the functions by $f_{\beta\alpha}^{\mu\lambda}$ where lower indices are in I, upper indices are in L; the order is the product, so that[1]

$$f_{\gamma\alpha}^{\nu\lambda} = f_{\gamma\beta}^{\nu\mu} \circ f_{\beta\alpha}^{\mu\lambda} \qquad \text{whenever } \alpha \leq \beta \leq \gamma \text{ and } \lambda \leq \mu \leq \nu. \tag{3.29}$$

[1] The English version of Bourbaki has $f_{\gamma\beta}^{\mu\nu}$ which is a typo.

The inductive limit will be denoted by E or $\varinjlim_{\alpha,\lambda} E^\lambda_\alpha$. Fix $\lambda \in L$. Define $g^\lambda_{\beta\alpha} = f^{\lambda\lambda}_{\beta\alpha}$ so that the previous relation becomes

$$g^\lambda_{\gamma\alpha} = g^\lambda_{\gamma\beta} \circ g^\lambda_{\beta\alpha} \qquad \text{whenever } \alpha \le \beta \le \gamma. \tag{3.30}$$

This allows us to define S^λ an inductive system on E^λ_α indexed by I, let F^λ be the inductive limit, and g^λ_α be canonical mapping. Fix λ and μ; let[2] $h^{\mu\lambda}_\alpha = f^{\mu\lambda}_{\alpha\alpha}$. This is (again by (3.29)) an inductive system of mappings, let's denote its limit by $h^{\mu\lambda}$. We have

$$h^{\nu\lambda} = h^{\nu\mu} \circ h^{\mu\lambda} \qquad \text{whenever } \lambda \le \mu \le \nu, \tag{3.31}$$

so that we can define an inductive system S' on L. The objective is to prove that $\varprojlim S$ and $\varprojlim S'$ are canonically isomorphic.

```
Section DoubleInductiveLimit.

Variables I1 I2 r1 r2: Set.
Hypothesis (or1: preorder r1)(or2: preorder r2)
           (sr1: substrate r1 = I1)(sr2: substrate r2 = I2)
           (dr1: right_directed_on r1 I1) (dr2: right_directed_on r2 I2).


Variable S : inductive_system.
Hypothesis Sr: isr S = prod_of_relation r1 r2.

Lemma inl_dl_I: isI S = I1 \times I2.
```

We define here S^λ and F^λ.

```
Definition inl_dl_Elam_fam lam := Lg I1 (fun i => Vg (isE S) (J i lam)).
Definition inl_dl_glam_fam lam :=
  Lg r1 (fun ij => Vg (isf S) (J (J (P ij) lam) (J (Q ij) lam))).

Lemma inl_dl_index_p1 lam i: inc lam I2 -> inc i r1 ->
   gle (isr S) (J (P i) lam) (J (Q i) lam).
Lemma inl_dl_index_p2 lam mu i:  gle r2 lam mu -> inc i I1 ->
   gle (isr S) (J i lam) (J i mu).

Definition inl_dl_S_lambda lam (Hl: inc lam I2) : inductive_system.

Lemma inl_dl_S_lambda_prop lam (Hl: inc lam I2) :
  inductive_system_on (inl_dl_S_lambda Hl)
     (inl_dl_Elam_fam lam) I1 r1 (inl_dl_glam_fam lam).

Definition inl_dl_system_S_lambda lam :=
  match (ixm (inc lam I2)) with
    | inl hx => (inl_dl_S_lambda hx)
    | inr _ => S
    end.

Definition inl_dl_F_lambda lam :=
  inductive_limit (inl_dl_system_S_lambda lam).
```

[2] The English Edition of Bourbaki has $h^{\mu\lambda}_\alpha = f^{\lambda\mu}_{\alpha\alpha}$, which is a typo.

```
Lemma inl_dl_F_lambda_prop lam (Hl: inc lam I2):
  inl_dl_F_lambda lam = inductive_limit (inl_dl_S_lambda Hl).
```

We introduce $h_\alpha^{\mu\lambda}$, the limit $h^{\mu\lambda}$ and prove (3.31).

```
Definition inl_dl_halm_fam lam mu:=
  Lg I1 (fun i => Vg (isf S) (J (J i lam) (J i mu))).


Definition inl_dl_hlm lam mu (H: gle r2 lam mu) :=
  inductive_limit_fun (inl_dl_S_lambda (pidl_i1_L sr2 H))
                      (inl_dl_S_lambda (pidl_i2_L sr2 H))
                      (inl_dl_halm_fam lam mu).


Lemma inl_dl_halm_compat lam mu (H: gle r2 lam mu):
  inl_map2_compat (inl_dl_S_lambda (pidl_i1_L sr2 H))
                  (inl_dl_S_lambda (pidl_i2_L sr2 H))
                  (inl_dl_halm_fam lam mu).

Lemma inl_dl_hlm_compose l m n
  (Hlm : gle r2 l m) (Hmn: gle r2 m n):
  (inl_dl_hlm Hmn) \co (inl_dl_hlm Hlm)  =
  (inl_dl_hlm (proj33 or2 _ _ _ Hlm Hmn)).
```

We now define $h^{\mu\lambda}$ everywhere via the axiom of choice, then S′.

```
Definition inl_dl_hlm_gen x :=
  match (ixm (inc x r2)) with
    | inl hx => (inl_dl_hlm (pidl_i3_L or2 hx))
    | inr _ => emptyset
    end.

Lemma inl_dl_hlm_fct lm: inc lm r2 ->
   function_prop (inl_dl_hlm_gen lm)
         (inl_dl_F_lambda (P lm))(inl_dl_F_lambda (Q lm)).

Lemma inl_dl_S_lambda_Iv2 x y  (H1: inc x I2) (H2: inc y I2) : x = y ->
  inl_same_data (inl_dl_S_lambda H1)(inl_dl_S_lambda H2).

Lemma  inl_dl_hml_invariant i j (H:gle r2 i j) :
   inl_dl_hlm H = inl_dl_hlm_gen (J i j).


Lemma inl_dl_hml_id i: inc i I2 ->
   Vg (Lg r2 inl_dl_hlm_gen) (J i i) = identity (inl_dl_F_lambda i).
Definition inl_dl_systemS': inductive_system.

Lemma inl_dl_systemS'_prop: inductive_system_on inl_dl_systemS'
     (Lg I2 inl_dl_F_lambda) I2 r2 (Lg r2 inl_dl_hlm_gen).
```

So far, we followed the proof of the projective case. We consider now u_α^λ, the composition of the canonical mapping $E_\alpha^\lambda \to F^\lambda$ and $F^\lambda \to F$; we show that it forms a system of mappings, and that its limit is the desired bijection. We have

$$u_\beta^\mu \circ f_{\beta\alpha}^{\mu\lambda} = h^\mu \circ g_\beta^\mu \circ f_{\beta\alpha}^{\mu\lambda} = h^\mu \circ g_\beta^\mu \circ f_{\beta\alpha}^{\mu\mu} \circ f_{\alpha\alpha}^{\mu\lambda} = h^\mu \circ g_\beta^\mu \circ g_{\beta\alpha}^\mu \circ f_{\alpha\alpha}^{\mu\lambda} = h^\mu \circ g_\alpha^\mu \circ f_{\alpha\alpha}^{\mu\lambda} =$$

$$= h^{\mu} \circ g_{\alpha}^{\mu} \circ h_{\alpha}^{\mu\lambda} = h^{\mu} \circ h^{\mu\lambda} \circ g_{\alpha}^{\lambda} = h^{\lambda} \circ g_{\alpha}^{\lambda} = u_{\alpha}^{\lambda}.$$

Justifications for the equalities: by definition of u_{β}^{α}; by the properties of f; by definition of $g_{\beta\alpha}^{\mu}$; by the properties of g_x^{μ}; by definition of $h_{\alpha}^{\mu\lambda}$; by Corollary 1, with $u_i = h_i^{\mu\lambda}$; by the properties of h^x; by definition.

```
Definition inl_dl_fg i l :=
   (inl_can_fun (inl_dl_system_S_lambda l) i).

Lemma inl_dl_fg_prop1 i l (H:inc l  I2):
  inl_dl_fg i l =  (inl_can_fun (inl_dl_S_lambda H) i).
Lemma inl_dl_fg_fp i l  (Hi: inc i I1) (Hl: inc l I2):
  function_prop (inl_dl_fg i l)
     (Vg (isE S) (J i l)) (inductive_limit (inl_dl_S_lambda Hl)).

Lemma inl_dl_fh_cp p
      (h := inl_can_fun inl_dl_systemS' (Q p)) (g:= inl_dl_fg (P p) (Q p)):
  inc p (isI S) ->
  h \coP g /\
  function_prop (h \co g) (Vg (isE S) p) (inductive_limit (inl_dl_systemS')).

Definition inl_dl_fu :=
  Lg (isI S)  (fun  p => (inl_can_fun inl_dl_systemS' (Q p))
                           \co (inl_dl_fg (P p) (Q p))).

Lemma inl_dl_fu_compat:
  inl_map_compat S inl_dl_fu  (inductive_limit (inl_dl_systemS')).

Lemma inl_dl_bijection: bijection_prop
     (inductive_map S inl_dl_fu (inductive_limit inl_dl_systemS'))
     (inductive_limit S) (inductive_limit inl_dl_systemS').

End DoubleInductiveLimit.
```

Proposition 9. Bourbaki expresses the previous result by saying that, up to a canonical bijection, we have

$$\varinjlim_{\alpha,\lambda} \mathrm{E}_{\alpha}^{\lambda} = \varinjlim_{\lambda} (\varinjlim_{\alpha} \mathrm{E}_{\alpha}^{\lambda}). \tag{3.32}$$

Corollary. Let $(\mathrm{E}'^{\lambda}_{\alpha}, f'^{\mu\lambda}_{\alpha\beta})$ be another direct system of sets relative to $\mathrm{I} \times \mathrm{L}$, and for each $(\alpha, \lambda) \in \mathrm{I} \times \mathrm{L}$ let u_{α}^{λ} be a mapping of $\mathrm{E}_{\alpha}^{\lambda}$ into $\mathrm{E}'^{\lambda}_{\alpha}$ such that the u_{α}^{λ} form a direct system of mappings. Then

$$\varinjlim_{\alpha,\lambda} u_{\alpha}^{\lambda} = \varinjlim_{\lambda} (\varinjlim_{\alpha} u_{\alpha}^{\lambda}). \tag{3.33}$$

Let's introduce all variables and assumptions, and show that S and S′ have the same index set.

```
Section  DoubleDirectLimit2.

Variables I1 I2 r1 r2: Set.
Hypothesis (or1: preorder r1)(or2: preorder r2)
```

```
          (sr1: substrate r1 = I1)(sr2: substrate r2 = I2).
Hypothesis (dr1: right_directed_on r1 I1) (dr2: right_directed_on r2 I2).

Variables S S': inductive_system.
Variable u: Set.
Hypothesis Sr: isr S = prod_of_relation r1 r2.
Hypothesis Sr': isr S' = prod_of_relation r1 r2.
Hypothesis compat_u: inl_map2_compat S S' u.

Lemma inl_dl2_SrSr: inl_same_index S S'.
```

Introduce S^λ, S'^λ and u^λ. Whenever $\lambda \in L$, equations (3.23) hold.

```
Definition inl_dl2_ulam_fam lam :=  Lg I1 (fun i => Vg u (J i lam)).
Definition inl_dl2_Slambda := (inl_dl_system_S_lambda or1 or2 sr1 sr2 dr1 Sr).
Definition inl_dl2_Slambda' := (inl_dl_system_S_lambda or1 or2 sr1 sr2 dr1 Sr').

Lemma inl_dl2_res1 lam:inc lam I2 ->
   inl_same_index (inl_dl2_Slambda lam) (inl_dl2_Slambda' lam) /\
   inl_map2_compat (inl_dl2_Slambda lam) (inl_dl2_Slambda' lam)
                   (inl_dl2_ulam_fam lam).
```

This means that we can define $v^\lambda = \varinjlim u^\lambda$, and equations (3.23) hold; so that we can define $\varinjlim v^\lambda$.

```
Definition inl_dl2_v lam :=
   inductive_limit_fun (inl_dl2_Slambda lam) (inl_dl2_Slambda' lam)
                   (inl_dl2_ulam_fam lam).
Definition inl_dl2_v_fam := Lg I2 inl_dl2_v.
Definition inl_dl2_limlim := (inl_dl_systemS' or1 or2 sr1 sr2 dr1 dr2 Sr).
Definition inl_dl2_limlim' := (inl_dl_systemS' or1 or2 sr1 sr2 dr1 dr2 Sr').

Lemma inl_dl2_res2:
  inl_map2_compat inl_dl2_limlim  inl_dl2_limlim' inl_dl2_v_fam. (* 120 *)
```

What we have is not (3.33), but the following commutative diagram, where the vertical arrows are the canonical bijections hidden in (3.33).

$$\begin{array}{ccc}
 & \varinjlim_{\alpha,\lambda} u_\alpha^\lambda & \\
\varinjlim_{\alpha,\lambda} \mathrm{E}_\alpha^\lambda & \longrightarrow & \varinjlim_{\alpha,\lambda} \mathrm{E}'^\lambda_\alpha \\
\downarrow & & \downarrow \\
\varinjlim_\lambda (\varinjlim_\alpha \mathrm{E}_\alpha^\lambda) & \underset{\varinjlim_\lambda (\varinjlim_\alpha u_\alpha^\lambda)}{\longrightarrow} & \varinjlim_\lambda (\varinjlim_\alpha \mathrm{E}'^\lambda_\alpha)
\end{array}$$

```
Lemma inl_dl2_res3  (* 81 *)
  (bij1 := (inductive_map S (inl_dl_fu or1 or2 sr1 sr2 dr1 dr2 Sr)
            (inductive_limit (inl_dl_systemS' or1 or2 sr1 sr2 dr1 dr2 Sr))))
  (bij2 := (inductive_map S' (inl_dl_fu or1 or2 sr1 sr2 dr1 dr2 Sr')
            (inductive_limit (inl_dl_systemS' or1 or2 sr1 sr2 dr1 dr2 Sr'))))
  (pl1 := inductive_limit_fun S S' u)
```

```
  (pl2 := inductive_limit_fun inl_dl2_limlim  inl_dl2_limlim' inl_dl2_v_fam):
  [/\ bijection bij1, bijection bij2 & pl2 \co bij1  =  bij2 \co pl1].

End  DoubleDirectLimit2.
```

Proposition 10. « Let $(E_\alpha, f_{\beta\alpha})$ and $(E'_\alpha, f'_{\beta\alpha})$ be two direct systems of sets, both relative to the same directed set I. Let $E = \varinjlim E_\alpha$, $E' = \varinjlim E'_\alpha$, and let $f_\alpha : E_\alpha \to E$, $f'_\alpha : E'_\alpha \to E'$ denote the canonical mappings, for each $\alpha \in I$. Then $(E_\alpha \times E'_\alpha, f_{\beta\alpha} \times f'_{\beta\alpha})$ is a direct system of sets, $(f_\alpha \times f'_\alpha)$ is a direct system of mappings and $\varinjlim(f_\alpha \times f'_\alpha)$ is a bijection

$$\varinjlim(E_\alpha \times E'_\alpha) \to (\varinjlim E_\alpha) \times (\varinjlim E'_\alpha). \tag{3.34}$$ »

The first claim is obvious.

```
Definition inl_product_E S S':=
   Lg (isI S) (fun i => (Vg (isE S) i) \times (Vg (isE S') i)).
Definition inl_product_f S S' :=
   Lg (isr S) (fun i => (Vg (isf S) i) \ftimes (Vg (isf S') i)).

Definition inl_system_product S S' (sd: inl_same_index S S'): inductive_system.

Lemma inl_system_product_prop S S' (sd: inl_same_index S S'):
  inductive_system_on (inl_system_product sd)
   (inl_product_E S S') (isI S) (isr S) (inl_product_f S S').
```

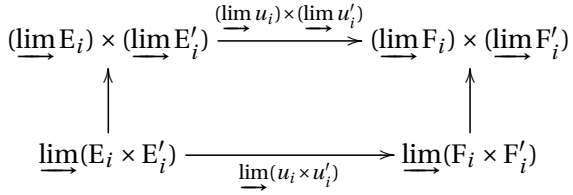
The second property is obvious as well. Injectivity. We use Proposition 6. So consider two elements, with the same value. This means that there is an index i, elements x, y in E_i, elements x', y' in E'_i such tat $f_i(x) = f_i(y)$ and $f'_i(x') = f'_i(y')$. The first relation is $\mathscr{C}_S(x,i) = \mathscr{C}_S(y,i)$ and says $f_{ki}(x) = f_{ki}(y)$. The second relation is similar (with, perhaps a different k), but we can obviously take the same. Surjectivity. Consider an element of the product. This is a pair of classes, $\mathscr{C}_S(x,i)$ and $\mathscr{C}_{S'}(x',i')$. We may assume $i = i'$. Now $\mathscr{C}_S((x,x'),i)$ is the desired result.

```
Definition inl_product_can_fun S S' :=
  Lg (isI S) (fun i => (inl_can_fun S i) \ftimes (inl_can_fun S' i)).

Lemma inl_product_can_fun_compat  S S' (sd: inl_same_index S S'):
  inl_map_compat (inl_system_product sd) (inl_product_can_fun S S')
  ((inductive_limit S) \times (inductive_limit S')).

Lemma inl_product_can_fun_bij  S S' (sd: inl_same_index S S')
  (E:=  inductive_limit S) (E' := inductive_limit S')
  (f:= inductive_map (inl_system_product sd) (inl_product_can_fun S S')
         (E \times E')):
   bijection_prop f
       (inductive_limit (inl_system_product sd)) (E \times E').
```

Corollary. « Let $(F_\alpha, g_{\beta\alpha})$ and $(F'_\alpha, g'_{\beta\alpha})$ be two direct systems of sets relative to I, and for each $\alpha \in I$ let $u_\alpha : E_\alpha \to F_\alpha$, $u'_\alpha : E'_\alpha \to F'_\alpha$ be mappings such that (u_α) and (u'_α) are two direct systems of mappings. Then $(u_\alpha \times u'_\alpha)$ is a direct system of mappings, and (up to a canonical bijection) we have

$$\varinjlim(u_\alpha \times u'_\alpha) = (\varinjlim u_\alpha) \times (\varinjlim u'_\alpha). \tag{3.35}$$ »

The result is straightforward. What we prove is that the following diagram is commutative, where the vertical arrows are the bijections defined in Proposition 10.




```
Section InjectiveProductMap.

Variables (SE SE' SF SF': inductive_system).
Variables u u': Set.
Hypotheses (si1:inl_same_index SE SE')
           (si2:inl_same_index SE SF)
           (si3:inl_same_index SF SF').
Hypotheses (cu:inl_map2_compat SE SF u) (cu':inl_map2_compat SE' SF' u').

Definition inl_prod_SEE := inl_system_product si1.
Definition inl_prod_SFF := inl_system_product si3.

Definition inl_prod_uu:= Lg (isI SE) (fun i => (Vg u i) \ftimes (Vg u' i)).

Lemma inl_prod_si4: inl_same_index SE' SF'.
Lemma inl_prod_uu_prop:inl_map2_compat inl_prod_SEE inl_prod_SFF inl_prod_uu.


Lemma inl_prod__uu_comp
  (E := inductive_limit SE)
  (E' := inductive_limit SE')
  (F := inductive_limit SF)
  (F' := inductive_limit SF')
  (EE:= inductive_limit (inl_system_product si1))
  (FF:= inductive_limit (inl_system_product si3))
  (lu:= inductive_limit_fun SE SF u)
  (lu':= inductive_limit_fun SE' SF' u')
  (luu:= inductive_limit_fun inl_prod_SEE inl_prod_SFF inl_prod_uu)
  (idEE := inductive_map (inl_system_product si1) (inl_product_can_fun SE SE')
         (E \times E'))
  (idFF := inductive_map (inl_system_product si3) (inl_product_can_fun SF SF')
         (F \times F')):
  [/\ bijection_prop idEE EE (E \times E'),
   bijection_prop idFF FF (F \times F'),
   function_prop (lu \ftimes lu') (E \times E') (F \times F'),
   function_prop luu EE FF &
   (lu \ftimes lu') \co idEE = idFF \co luu].
End InjectiveProductMap.
```

3.8 Exercises

The 1956 edition of Bourbaki [1] defines inductive and projective limits as no. 11 and 12 of §1, instead of §7, as applications of no. 10 (directed sets). Moreover conditions ($\mathrm{LI}_{\mathrm{II}}$) and

(LP_{II}) are missing. Exercises 24 and 29 explain that adding these constrains yields canonically isomorphic objects.

Let's consider a projective system E, I, r, f, but without the condition that f_{ii} is the identity function, and define $\varprojlim \mathrm{E}$ as the subset of $\prod \mathrm{E}_i$ formed of all sequences $(x_i)_{i\in \mathrm{I}}$ such that $x_i = f_{ij}(x_i)$ whenever $i \leq j$.

```
Variables E I r f: Set.
Hypothesis
  (preorder_r:  preorder r)
  (substrate_r:  substrate r = I)
  (fgraph_E: fgraph E)
  (domain_E: domain E = I)
  (fgraph_f: fgraph f)
  (domain_f: domain f = r)
  (function_f:
    forall i, inc i r ->
               function_prop (Vg f i) (Vg E (Q i)) (Vg E (P i)))
  (compose_f: forall i  j k, gle r i j -> gle r j k ->
                  Vg f (J i j) \co  Vg f (J j k) = Vg f (J i k)).
Definition noid_projlim :=
  Zo (productb E) (fun x => forall  i j, gle r i j
                               -> (Vg x i) = Vf (Vg f (J i j)) (Vg x j)).
```

Let's introduce $\mathrm{E}'_i = f_{ii}\langle \mathrm{E}_i\rangle$ (the image of f_{ii}). If $i \leq j$ then $f_{ii} \circ f_{ij} = f_{ij} \circ f_{jj} = f_{ij}$. This relation says that the E'_i form a projective system S′ of subsets of the E_i, and equation (3.9) reads $\varprojlim \mathrm{S}' = \varprojlim \mathrm{E} \cap \prod \mathrm{E}'_i$. If $x \in \varprojlim \mathrm{E}$ then $x_i = f_{ii}(x_i)$ whenever $i \in \mathrm{I}$, hence $\varprojlim \mathrm{E} \subset \prod \mathrm{E}'_i$. So $\varprojlim \mathrm{E} = \varprojlim \mathrm{S}'$, and we shall prove this relation (without establish (3.9) in a context where (LP_{II}) is missing. Notre that, if $x_i \in \mathrm{E}'_i$, relation $f_{ii} \circ f_{ii} = f_{ii}$ says $f_{ii}(x_i) = x_i$, so that S′ satisfies (LP_{II}).

```
Definition noid_E  :=  Lg I (fun i => Imf (Vg f (J i i))).
Definition noid_f := Lg r (fun z => restriction2 (Vg f z)
   (Vg noid_E (Q z)) (Vg noid_E (P z))).

Lemma noid_prop0 i: inc i I -> sub (Vg noid_E i) (Vg E i).
Lemma noid_prop1 (M:= noid_E) (g := noid_f):
  [/\
   forall z, inc z r ->
      restriction2_axioms (Vg f z) (Vg M (Q z)) (Vg M (P z)),
   forall i j x, gle r i j -> inc x (Vg M j) ->
     Vf (Vg g (J i j)) x = Vf (Vg f (J i j)) x,
   forall i, inc i r -> function_prop (Vg g i) (Vg M (Q i)) (Vg M (P i)),
   forall i j k, gle r i j -> gle r j k ->
                 Vg g (J i j) \co  Vg g (J j k) = Vg g (J i k) &
   forall i, inc i I -> Vg g (J i i) = identity (Vg M i)].

Definition noid_proj_system: projective_system.
Lemma noid_prop2: projective_system_on noid_proj_system  noid_E I r noid_f.
Lemma noid_prop3 : projective_limit(noid_proj_system) = noid_projlim.
ProjectiveLimitNoId.
```

Consider now the case of an inductive limit. The assumptions are the same as above (we assume however that I is right directed). The set E'_i is defined as above, while g_{ij} is the restriction (the other way around).

```
Section InductiveLimitNoId.
Variables E I r f: Set.
Hypothesis
  (preorder_r:  preorder r)
  (substrate_r:  substrate r = I)
  (directed_r: right_directed_on r I)
  (fgraph_E: fgraph E)
  (domain_E: domain E = I)
  (fgraph_f: fgraph f)
  (domain_f: domain f = r)
  (function_f:
    forall p, inc p r ->
              function_prop (Vg f p) (Vg E (P p)) (Vg E (Q p)))
  (compose_f: forall i  j k, gle r i j -> gle r j k ->
                  Vg f (J j k) \co  Vg f (J i j) = Vg f (J i k)).

Definition noid_E'  :=  Lg I (fun i => Imf (Vg f (J i i))).
Definition noid_g := Lg r (fun z => restriction2 (Vg f z)
   (Vg noid_E' (P z))  (Vg noid_E' (Q z))).
```

The following lemmas are proved as above. In particular we can define an inductive system S based on the sets E'_i.

```
Lemma noid_prop5a i: inc i I -> sub (Vg noid_E' i) (Vg E i).
Lemma noid_prop5b z: inc z r ->
       restriction2_axioms (Vg f z) (Vg noid_E' (P z)) (Vg noid_E' (Q z)).
Lemma noid_prop5c i j x: gle r i j -> inc x (Vg noid_E' i) ->
     Vf (Vg noid_g (J i j)) x = Vf (Vg f (J i j)) x.
Lemma noid_prop5d i: inc i r ->
  function_prop (Vg noid_g i) (Vg noid_E' (P i)) (Vg noid_E' (Q i)).
Lemma noid_prop5e i: inc i I -> Vg noid_g (J i i) = identity (Vg noid_E' i).
Lemma noid_prop5f i j k:  gle r i j -> gle r j k ->
  Vg  noid_g (J j k) \co  Vg noid_g (J i j) = Vg noid_g (J i k).
Lemma noid_prop5g y i j k:
  gle r i j -> gle r j k -> inc y (Vg E i) ->
  Vf (Vg f (J j k))  (Vf (Vg f (J i j)) y) = Vf (Vg f (J i k)) y.

Definition noid_ind_system: inductive_system.
Lemma noid_prop6: inductive_system_on noid_ind_system  noid_E' I r noid_g.
```

Let's define an equivalence relation on the disjoint union of the E_i. If $x \in \mathrm{E}_i$, then $\mathscr{C}_\mathrm{E}(x,i)$ will denote the class of x for this relation; if $x \in \mathrm{E}'_i$ then $\mathscr{C}_\mathrm{S}(x,i)$ denotes the class of S. Assume $x \in \mathrm{E}'_i$, $y \in \mathrm{E}'_j$; then $\mathscr{C}_\mathrm{S}(x,i) = \mathscr{C}_\mathrm{S}(y,j)$ is equivalent to $\mathscr{C}_\mathrm{E}(x,i) = \mathscr{C}_\mathrm{E}(y,j)$ (both equalities say that there is an upper bound k of i and j such that $f_{ki}(x) = f_{kj}(y)$, where f can be replaced by g). This equation means that we can define an injection $\phi : \mathscr{C}_\mathrm{S}(x,i) \mapsto \mathscr{C}_\mathrm{E}(x,i)$ Assume $x \in \mathrm{E}_i$, and let $x' = f_{ii}(x)$. Then $\mathscr{C}_\mathrm{E}(x,i) = \mathscr{C}_\mathrm{E}(x',i)$. This says that $\mathscr{C}_\mathrm{E}(x,i)$ is in the image of ϕ.

```
Definition noid_inl_sum := disjointU E.
Definition noid_inl_equiv_rel x y:=
  exists k, [/\ gle r (Q x) k,  gle r (Q y) k   &
   Vf (Vg f (J (Q x) k)) (P x) =  Vf (Vg f (J (Q y) k)) (P y) ].
Definition noid_inl_equiv  := graph_on noid_inl_equiv_rel noid_inl_sum.
Definition noid_limit := quotient noid_inl_equiv.
```

```
Lemma noid_inl_sumP x: inc x noid_inl_sum <->
  [/\ pairp x, inc (Q x) I & inc (P x) (Vg E (Q x))].
Lemma noid_inl_equiv_reflexive a: inc a noid_inl_sum -> noid_inl_equiv_rel a a.
Lemma noid_inl_equiv_esr: equivalence_on noid_inl_equiv noid_inl_sum.

Lemma noid_inl_class_eq x y:
  inc x noid_inl_sum -> inc y noid_inl_sum ->
  (class noid_inl_equiv x = class noid_inl_equiv y
     <-> noid_inl_equiv_rel x y).

Lemma noid_inl_class_ii i x (y := Vf (Vg f (J i i)) x):
  inc i I -> inc x (Vg E i) ->
  [/\ inc y (Vg noid_E' i),
     inc (J x i) noid_inl_sum, inc (J y i) noid_inl_sum  &
  class noid_inl_equiv (J x i) = class noid_inl_equiv (J y i)].

Lemma noid_inl_class_compat i j x y (R := (inl_equiv noid_ind_system) ):
  inc i I -> inc j I -> inc x (Vg noid_E' i) -> inc y (Vg noid_E' j) ->
   (class R (J x i) = class R (J y j) <->
    class noid_inl_equiv (J x i) =  class noid_inl_equiv (J y j)).
```

We now formally define ϕ and show that it is a bijection, which is rather trivial.

```
Definition noid_can x :=  class noid_inl_equiv (J (P (rep x)) (Q (rep x))).
Lemma noid_inl_prop7 (A := inductive_limit noid_ind_system) (B:= noid_limit):
  bijection_prop (Lf noid_can A B) A B.
End InductiveLimitNoId.
```

Exercise 1. « Let I be a directed set, let $(J_\lambda)_{\lambda\in L}$ be a family of subsets of I, indexed by a directed set L, such that (i) for each $\lambda \in L$, J_λ is directed with respect to the induced ordering; (ii) the relation $\lambda \leq \mu$ implies $J_\lambda \subset J_\mu$; (iii) I is the union of the family (J_λ). Let $(E_\alpha, f_{\alpha\beta})$ be an inverse system of sets relative to I, let E be its inverse limit, and for each $\lambda \in L$ let F_λ be the inverse limit of the system obtained from $(E_\alpha, f_{\alpha\beta})$ by restricting the index set to J_λ. For $\lambda \leq \mu$, let $g_{\lambda\mu}$ be the canonical mapping of F_μ into F_λ. Show that $(F_\lambda, g_{\lambda\nu})$ is an inverse system of sets relative to L, and define a canonical bijection of $F = \varprojlim F_\lambda$ onto E. »

It happens that all conditions are not necessary; for instance, we only need preorders, and L has to be directed. We denote by $i \leq_I j$ the order relation on I, and by $i \leq_L j$ the relation on L. So the context will be the following:

```
Variables I rI L rL Jf: Set.
Variable S: projective_system.

Hypothesis rS: (psr S = rI).
Hypotheses (HIp :preorder rI) (HIs: substrate rI = I).
Hypotheses (HLp :preorder rL) (HLs: substrate rL = L)
           (HLd:right_directed_pre rL).
Hypothesis (HJg: fgraph Jf) (HJd: domain Jf = L) (HJI: unionb Jf = I)
   (HJm: forall i j, gle rL i j -> sub (Vg Jf i) (Vg Jf j)).
```

We start with some trivial properties. The last lemma says under which condition the order induced on J_i by $\leq_I$ would make it a directed set.

```
Lemma ex1_prop1 i: inc i L -> sub (Vg Jf i) I.
Lemma ex1_prop2 i j: gle rL i j -> inc i L /\ inc j L.
Lemma ex1_prop3: I = psI S.
Lemma ex1_prop4 i: inc i L -> sub (Vg Jf i) (psI S).

Lemma ex1_preorder i (ri:= induced_order rI (Vg Jf i)):  (* 7 *)
   ( forall k, inc k L -> forall i j, inc i (Vg Jf k) ->  inc j (Vg Jf k) ->
     exists t, [/\  inc t (Vg Jf k), gle rI i t & gle rI j t])  ->
   inc i L ->
  [/\ preorder ri, substrate ri = (Vg Jf i) & right_directed_pre ri].
```

We define now the projective system S_i obtained by restricting indices to J_i, the projective limit F_i, the canonical function $g_i : E \to F_i$, and the functions $g_{ij} : F_j \to F_i$. Assume $i \in L$ so that $J_i \subset I$; in this case we can define S_i; in the general case, we use a version of the axiom of choice that says: it is decidable whether or not $i \in L$, and we can do case analysis in the definition.

```
Definition ex1_systemi i:=
  match (ixm (inc i L)) with
    | inl hx => (prl_restr (ex1_prop4 hx))
    | inr _ => S
end.

Definition ex1_Fl i := projective_limit(ex1_systemi i).
Definition ex1_gij ij :=
  Lf (restr ^~ (Vg Jf (P ij))) (ex1_Fl (Q ij)) (ex1_Fl (P ij)).
```

The proof of the following lemmas is the following: unfold the definitions, exposing the `match`, then do a case analysis on `(ixm (inc j L))`; in the true case, the result is obvious, in the false case we have a contradiction with $i \in L$.

```
Lemma ex1_res0 i (H: inc i L):
  ex1_Fl i = (projective_limit (prl_restr (ex1_prop4 H))).
Lemma ex1_prop5a j: inc j L -> (Vg Jf j) = psI (ex1_systemi j).
Lemma ex1_prop5b i (H: sub (Vg Jf i) (psI S)):
  inc i L -> prl_same_data (prl_restr H) (ex1_systemi i).
```

A non-trivial property is `ex1_prop6`: if $i \leq_L j$, then S_i has the same data as the system obtained by restricting the indices of S_j to J_i (we have to apply `ex1_prop5b` twice, then the property of double restriction). We deduce that g_{ij} is a function $M_J \to M_i$, which is the identity when $i = j$. A bit more complicated is that $g_{ij} \circ g_{jk} = g_{ij}$. By `ex1_res`, the functions g_{ij} and g_{jk} can be composed; so that it suffices to prove $g_{ij}(g_{jk}(x)) = g_{ij}(x)$. it suffices to unfold g, and use `ex1_prop7`.

```
Lemma ex1_prop5 i j: gle rL i j ->  sub (Vg Jf i) (psI (ex1_systemi j)).
Lemma ex1_prop6 i j  (lij: gle rL i j) :
  prl_same_data (prl_restr (ex1_prop5 lij)) (ex1_systemi i).  (* 7 *)
Lemma ex1_prop6a i j  (lij: gle rL i j) :
  (projective_limit_restr (ex1_prop5 lij)) =  (ex1_Fl i).
Lemma ex1_prop7 i j: gle rL i j ->
  lf_axiom (restr^~ (Vg Jf i)) (ex1_Fl j) (ex1_Fl i). (* 2 *)
Lemma ex1_res2 i j: gle rL i j ->
   function_prop (ex1_gij (J i j)) (ex1_Fl j) (ex1_Fl i)).
```

```
Lemma ex1_res3 i: inc i L -> ex1_gij (J i i) = identity (ex1_Fl i). (* 2 *)
Lemma ex1_pr4 i j k: gle rL i j ->   gle rL j k ->
   ex1_gij (J i j) \co ex1_gij (J j k) = ex1_gij (J i k). (* 16 *)
```

We can now define the projective system with the functions g_{ij}, let's call it S', and the projective limit E'. If x is in the product of the E_i, and $i \in L$ we denote by $(x)_i$ the restriction of x to J_i, and by (x) the functional graph $x \mapsto x_i$. Moreover, we denote by ϕ the function $x \mapsto (x)$. If $x \in E$, then $(x)_i \in M_i$, and $\phi(x) \in E'$ (this is rather simple).

```
Definition ex1_F: projective_system.  (* 15 *)
Lemma ex1_F_prop: projective_system_on ex1_F (Lg L ex1_Fl) L rL (Lg rL ex1_gij).

Definition ex1_restr_fun z:=  Lg L (fun i => restr z (Vg Jf i)).
Definition ex1_F_can := Lf ex1_restr_fun
   (projective_limit S) (projective_limit ex1_F).

Lemma ex1_F_can_ax1 i z: inc i L -> inc z (projective_limit S) ->
   inc (restr z (Vg Jf i)) (ex1_Fl i).  (* 2 *)
Lemma ex1_F_can_ax: lf_axiom ex1_restr_fun
   (projective_limit S) (projective_limit ex1_F).  (* 12 *)
Lemma ex1_F_can_fun: function_prop ex1_F_can
 (projective_limit S) (projective_limit ex1_F).  (* 2 *)
```

Showing that ϕ is injective is easy, because $(x)_i(k) = x(k)$ whenever $k \in J_i$ (where $t(k)$ is the value of t at k). In order to show that ϕ is surjective, we consider y and look for x such that $(x)_i = y(i)$, so $x(k) = y(i)(k)$; for every k in I there exists i such that $k \in J_i$, but it is not unique. Claim 1: if $i \leq_L i''$, then $y(i)$ is the restriction of $y(i'')$ to J_i (obvious by definition). Claim 2: if $k \in J_i$ and $k \in J_{i'}$, then $y(i)(k) = y(i'')(k)$. In fact, there is $i'' \in L$ such that $i \leq_L k''$ and $i' \leq_L k''$. By claim 1, $y(i)(k)$ and $y(i')(k)$ are equal to $y(i'')(k)$. We can define (via the axiom of choice) a function $i(k)$ such that $k \in J_{i(k)}$, and a function x by $x(k) = y(i(k))(k)$. Claim 3: if $k \in J_i$, then $x(k) = y(i)(k)$ (obvious). Claim 4: $y(i)$ belongs to the projective limit obtained from S, by restricting indices to J_i, obvious by definition; this means $y(i) \in \prod_{k \in J_i} E_k$ and $f_{ab}(y(i)(b)) = y(i)(a)$ whenever $a \leq_I b$. Claim 5: $x \in \prod E_k$ (by claim 4). Claim 6, $\phi(x) = y$. Obviously, both terms are functional graphs defined on L; it suffices to show $\phi(x)(i) = y(i)$. Both quantities are functional graphs with domain J_i, it suffices to show $\phi(x)(i)(k) = y(i)(k)$. By definition of ϕ, the LHS is $x(k)$; the result holds by claim 3. Finally $x \in E$. This means: $f_{ab}(x(b)) = x(a)$, whenever $a \leq_I b$. This is $f_{ab}(y(i(b))(b)) = y(i(a))(a)$. Take i such that $i(a) \leq_L i$ and $i(a) \leq_L i$, and use claim 2 twice. We get $f_{ab}(y(i)(b)) = y(i)(a)$, which holds by claim 4.

```
Lemma ex1_F_can_bf: bijection ex1_F_can. (* 75 *)
```

Exercise 2. « Let $(E_\alpha, f_{\alpha\beta})$ be an inverse system of sets relative to a directed index set, let $E = \varprojlim E_\alpha$ and let $f_\alpha : E \to E_\alpha$ be the canonical mapping for each α. Show that, if all the $f_{\alpha\beta}$ are injective, then f_α is injective. »

Proof. Consider x and y in E such that $f_i(x) = f_i(y)$; this means $x_i = y_i$. In order to show $x = y$, it suffices to show $x_j = y_j$ for every j. Assume the order *right directed*, so that for some k, we have $i \leq k$ and $j \leq k$. Since x and y are in the limit, the assumption becomes $f_{ik}(x_k) = f_{ik}(y_k)$ and the conclusion becomes $f_{jk}(x_k) = f_{kk}(y_k)$. By injectivity $x_k = y_k$, the conclusion follows.

```
Lemma Exercise7_2 S:
  right_directed  (psr S) ->
  (forall i j, gle (psr S) i j -> injection (Vg (psf S) (J i j))) ->
  forall i, inc i (psI S) -> injection (prl_can_fun S i). (* 20 *)
```

Exercise 3. « Let $(E_\alpha, f_{\alpha\beta})$ and $(F_\alpha, g_{\alpha\beta})$ be two inverse systems of sets relative to the same index set I. For each $\alpha \in I$, let u_α be a mapping of E_α into F_α, such that the u_α form an inverse system of mappings. Let $G_\alpha \subset E_\alpha \times F_\alpha$ be the graph of u_α. Show that (G_α) is an inverse system of subsets of $E_\alpha \times F_\alpha$ and that its inverse limit may be canonically identified with the graph of $u = \varprojlim u_\alpha$. »

Note that the main Bourbaki text does not define the product of the two systems, but we have considered it. Proving that (G_i) is a projective system of subsets is straightforward.

```
Variables S S': projective_system.
Variable (u:Set).
Hypothesis same_I: (prl_same_index S S').
Hypothesis (Hu: prl_map2_compat S S' u).

Lemma ex3_prl_subfm_hyp (S'' := prl_system_product  same_I):
  prl_subfam_hyp S'' (Lg (psI S) (fun i => graph (Vg u i))). (* 37 *)


Definition ex3limit_graphs := projective_system_subsets ex3_prl_subfm_hyp.
Definition ex3_gl_val x :=
  Lg (psI S) (fun i => (J (Vg (P x) i) (Vg (Q x) i))).

Lemma ex3_gl_val_ax:  (* 44 *)
  lf_axiom ex3_gl_val  ex3_graphs_limit (projective_limit ex3limit_graphs).
Lemma ex3_gl_val_bf (E := (projective_limit ex3limit_graphs))
       (f:= Lf ex3_gl_val  ex3_graphs_limit  E):
  bijection_prop f ex3_graphs_limit E. (* 62 *)
```

Exercise 4. « Let I be a non-empty directed set with no greatest element, and let F be the set of all sequences $x = (\alpha_1, \alpha_2, \ldots, \alpha_{2n-1}, \alpha_{2n})$ of an even number ≥ 2 of elements of I with the following properties: (i) $\alpha_{2i-1} < \alpha_{2i}$ for $1 \leq i \leq n$; (ii) $\alpha_{2i-1} \not\leq \alpha_{2j-1}$ for $1 \leq j < i \leq n$. The set F is not empty. Put $r(x) = \alpha_{2n-1}$, $s(x) = \alpha_{2n}$. The integer n is called the *length* of x.

(a) For each $\alpha \in I$, let E_α be the set of all $x \in F$ such that $r(x) = \alpha$. Then E_α is non-empty. For $\alpha \leq \beta$ in I, we define a mapping $f_{\alpha\beta}$ of E_β into the set of all finite sequences of elements of I, as follows: if

$$x = (\alpha_1, \alpha_2, \ldots, \alpha_{2n-1}, \alpha_{2n}) \in E_\beta,$$

let j be the least index such that $\alpha \leq \alpha_{2j-1}$; then

$$f_{\alpha\beta}(x) = (\alpha_1, \alpha_2, \ldots, \alpha_{2j-2}, \alpha, \alpha_{2j}).$$

Show that $f_{\alpha\beta}(E_\beta) = E_\alpha$, and that $(E_\alpha, f_{\alpha\beta})$ is an inverse system of sets relative to I.

(b) Show that if $x_\alpha \in E_\alpha$ and $x_\beta \in E_\beta$ are such that there exists an index γ for which $\gamma \geq \alpha$ and $\gamma \geq \beta$, and an element $x_\gamma \in E_\gamma$ for which $x_\alpha = f_{\alpha\gamma}(x_\gamma)$ and $x_\beta = f_{\beta\gamma}(x_\gamma)$, then, provided also that x_α and x_β have the same length, we have $s(x_\alpha) = s(x_\beta)$.

(c) Deduce from (b) that, if $E = \varprojlim E_\alpha$ is not empty and if $y = (x_\alpha) \in E$, then the set of elements $s(x_\alpha)$ is countable and cofinal in I.

(d) Let I be the set of all finite subsets of an uncountable set A, ordered by inclusion. Show that I has no countable cofinal subset, and hence deduce from (c) and example of an inverse system of sets $(E_\alpha, f_{\alpha\beta})$ in which the E_α are non-empty and the $f_{\alpha\beta}$ are surjective, but for which $E = \varprojlim E_\alpha = \emptyset$.

(e) Deduce from (d) an example of an inverse system of mappings $u_\alpha : E_\alpha \to E'_\alpha$ such that each u_α is surjective but $\varprojlim u_\alpha$ is not surjective (let each E'_α consist of a single element). »

Answer. We first introduce some properties (H) of I and its order r. Then we show that there is an example satisfying a stronger conditions (H′). We then prove (a), (b) and (c) in a context where (H) holds, then prove (d), (e). Note that (H) holds if I is the set of finite subsets of an infinite set, ordered by inclusion (if $x \in I$, it is a strict subset of A, so that there is t in A not in I). Assume B cofinal. Then $\bigcup B = A$ (if $x \in A$, then $\{x\}$ is bounded above by an element y of B, so $x \in y$). Since elements of B are finite, A and B have the same cardinal (in fact, we show a weaker statement: if B is countable, so is A). As an example, we can take $A = \mathfrak{P}(\mathbf{N})$.

```
Definition ex4_prop_IR I r:=
  [/\ nonempty I, order r,substrate r = I, right_directed r &
   forall x, inc x I -> ~(greatest r x)].

Definition ex4d_orderI A := Zo (powerset A) finite_set.
Definition ex4d_orderr A := sub_order (ex4d_orderI A).
Definition uncountable_set x := ~ (countable_set x).

Lemma ex4d_orderIr_prop1 A: infinite_set A ->
       ex4_prop_IR  (ex4d_orderI A)(ex4d_orderr A).  (* 11 *)

Lemma uncountable_set_infinite x: uncountable_set x -> infinite_set x.

Lemma ex4d_orderIr_prop2 A z: uncountable_set A ->
  cofinal (ex4d_orderr A) z -> countable_set z -> False. (* 16 *)
Lemma ex4d_orderIr_prop3 (A:= powerset Nat):
  ex4_prop_IR (ex4d_orderI A)(ex4d_orderr A) /\
  forall z, cofinal (ex4d_orderr A) z -> ~ countable_set z. (* 6 *)
```

If $x \in I$, there is y not smaller than x; if $x \leq z$ and $y \leq z$ we cannot have $x = z$; so $x < z$. There are x_0 and x_1 such that $x_0 < x_1$, this will say that F is non-empty.

```
Section Exercise4.
Variable I r: Set.
Hypothesis ex4H:ex4_prop_IR I r.

Lemma ex4_or_prop0 i j: gle r i j -> inc i I /\ inc j I.
Lemma ex4_no_greater x: inc x I -> exists y, glt r x y. (* 7 *)
Lemma ex4_or_prop1: exists x y, glt r x y.
```

Instead of the Bourbaki definition of F, we shall use a variant; an element will be a list of pairs, say (a_i, b_i), where indices start with zero. So, $r(x) = a_{n-1}$ and $s(x) = b_{n-1}$. Moreover

$\alpha_{2i-1} = a_i$ and $\alpha_{2i} = b_i$. The conditions become: $a_i < b_i$ for $i < n$ and $a_i \not\leq a_j$ for $j < i < n$, where n is the length of the list. Note that F is the set of all z, such that (a) z is a functional graph, its domain is a subset of **N**, its range is a subset of I × I, and (b) the domain of z is a non-zero integer n and some other property holds (recall that n is equal to the set of integers $< n$, so is a subset of **N**).

```
Definition ex4_seq_prop1 s n:=
  forall i, i <c n -> glt r (P (Vg s i)) (Q (Vg s i)).
Definition ex4_seq_prop2 s n:=
  forall i j, j <c i -> i <c n -> ~(gle r (P (Vg s i)) (P (Vg s j))).

Definition ex4_seqp s n :=
  [/\ natp n, fgraph s, domain s = n,
      ex4_seq_prop1 s n & ex4_seq_prop2 s n].

Definition ex4_F :=
  Zo (sub_fgraphs Nat (coarse I)) (fun z => exists2 n, n <> \0c & ex4_seqp z n).

Definition ex4_last x := cpred (domain x).
Definition ex4_fct_r x := P (Vg x (ex4_last x)).
Definition ex4_fct_s x := Q (Vg x (ex4_last x))

Lemma ex4_inF_hi x (n := domain x): inc x ex4_F ->
  [/\ n <> \0c, ex4_seqp x n  & forall i, i<c n -> pairp (Vg x i)]. (* 5 *)
Lemma ex4_length_prop1 x (n := ex4_last x):  inc x ex4_F ->
  natp n /\ domain x = csucc n.

Lemma ex4_inF x n : n <> \0c -> ex4_seqp x n ->
  (forall i, i<c n -> pairp (Vg x i)) ->
  inc x ex4_F.  (* 7 *)
Lemma ex4_fct_r_in_I x: inc x ex4_F ->
 [/\ inc (ex4_fct_r x) I, inc (ex4_fct_s x) I &
     glt r (ex4_fct_r x) (ex4_fct_s x)]. (* 4 *)
Lemma ex4_F_special i: inc i I ->
    exists2 x, inc x ex4_F & ex4_fct_r x = i. (* 12 *)
Lemma ex4_F_nonempty: nonempty ex4_F.
```

We now define E_α and $f_{\alpha\beta}$. This function is obtained by truncating its argument to the j first terms (operation that obvious yields an element of F) then modifying the value of r. This yields an element of F provided that the new value is $\leq r(x)$ (in order to preserve $r < s$) but not $\leq$ to the other a_i.

```
Definition ex4_setEi i := Zo ex4_F (fun z => ex4_fct_r z =  i).
Lemma ex4_setEi_nonempty i: inc i I -> nonempty(ex4_setEi i).

Definition ex4_modify_r x i:=
  Lg (domain x) (fun z => Yo (z = ex4_last x) (J i (Q (Vg x z))) (Vg x z)).

Lemma ex4_F_stable_restr x m: inc x ex4_F -> m <=c domain x -> m <> \0c ->
  inc (restr x m) ex4_F.  (* 12 *)

Lemma ex4_F_stable_modify_r x i:
  inc x ex4_F -> gle r i (ex4_fct_r x) ->
  (forall k,  k <c cpred (ex4_length x) -> ~ gle r i (Vg x (cdouble k))) ->
  inc (ex4_modify_r x i) ex4_F. (* 25 *)
```

```
Lemma ex4_modify_r_r x i:
  inc x ex4_F -> ex4_fct_r (ex4_modify_r x i)  = i. (* 4 *)
```

If i is the new value to insert, we take for j the first index such that $i \leq a_j$. It exists when $i \leq r(x)$. This is the correct way.

```
Definition ex4_indexj x a:=
  intersection (Zo (domain x) (fun j => gle r a (P (Vg x j)))).
Definition ex4_function_fv a x :=
  ex4_modify_r (restr x (csucc (ex4_indexj x a))) a.

Lemma ex4_indexj_correct x a (j := ex4_indexj x a):
  inc x ex4_F -> gle r a (ex4_fct_r x) ->
  [/\ j <c (domain x),  gle r a (P (Vg x j)) &
      forall k, k <c (domain x) ->  gle r a (P (Vg x k)) -> j <=c k]. (* 12 *)
Lemma ex4_indexj_idem x:
  inc x ex4_F -> ex4_indexj x (ex4_fct_r x) = ex4_last x. (* 9 *)
Lemma  ex4_function_f_prop1 x a (y := ex4_function_fv a x):
  inc x ex4_F -> gle r a (ex4_fct_r x) ->
  inc y ex4_F /\  (ex4_fct_r y)  = a.  (* 17 *)
Lemma ex4_function_f_ax a b: gle r a b ->  (* 2 *)
  lf_axiom (ex4_function_fv a)  (ex4_setEi b) (ex4_setEi a).
```

We now define $f_{\alpha\beta}$. In the special case $\alpha = \beta$ we have $j = n-1$. This means that the restriction is a no-op. Moreover the modification is trivial as well. So f_{ii} is the identity function. In order to prove (LP_I), we consider $i \leq j \leq k$, and x with $r(x) = k$, truncate to length b, modify with j, truncate to length c, modify with i; we can also directly truncate to truncate to length a, and modify with i. It is rather easy to show $a = c$, hence the conclusion. In order to show that $f_{\alpha\beta}$ is surjective, we take $y \in E_\alpha$ and c such that $\beta < c$. Extend y so that $x_i = y_i$ for $i < n$ and $x_n = (\beta, c)$. Then $f_{\alpha\beta}(y) = x$.

```
Definition ex4_function_f ab :=
  Lf (ex4_function_fv (P ab)) (ex4_setEi (Q ab))   (ex4_setEi (P ab)).
Definition ex4_function_f_fam := Lg r  ex4_function_f.

Lemma ex4_function_f_fun a b: gle r a b ->
  function_prop (ex4_function_f (J a b)) (ex4_setEi b)  (ex4_setEi a). (* 2 *)

Lemma ex4_function_f_id a: inc a I ->
  (ex4_function_f (J a a)) = identity (ex4_setEi a). (* 14 *)
Lemma ex4_compose_f i j k (psf :=  ex4_function_f_fam):
  gle r i j -> gle r j k ->
  Vg psf (J i j) \co  Vg psf (J j k) = Vg psf (J i k). (* 68 *)
Lemma ex4_function_f_sf a b:gle r a b ->
  surjection (ex4_function_f (J a b)). (* 61 *)

Definition ex4_system: projective_system. (* 16 *)
Lemma  ex4_system_prop: projective_system_on ex4_system
   (Lg I ex4_setEi) I r ex4_function_f_fam.
```

Assume $a \leq c$, $b \leq c$, $z \in F_c$, $x = f_{ab}(z)$ and $y = f_{bc}(z)$. This means that x and y are obtained by restricting z to size j and j' and modifying r. If x and y have the same length, then $j = j'$. As s is not modified, we have $s(x) = s(y)$. This shows (b).

Take $x \in \varprojlim \mathrm{E}$, and S the set of all $s(x_i)$ for $i \in \mathrm{I}$. If $i \in \mathrm{I}$, we have $i = r(x_i) < s(x_i)$. So, S is cofinal in I. Let S_n be the set of all elements t of S such that $t = s(x_i)$ and x_i has length n. By (b), S_n has at most one element; so is finite, thus countable. Thus, S, being a countable union of countable sets is countable. This shows (c).

```
Lemma ex4_propb a b c x y z: gle r a c -> gle r b c -> inc z (ex4_setEi c) ->
  x = Vf (ex4_function_f (J a c)) z  -> y = Vf (ex4_function_f (J b c)) z ->
  domain x = domain y ->
  ex4_fct_s x = ex4_fct_s y.  (* 13 *)

Lemma ex4_propc x (s := fun_image I (fun z => ex4_fct_s (Vg x z))):
  inc x  (projective_limit ex4_system) ->
  countable_set s /\ cofinal r s.  (* 40 *)
Lemma ex4_propc1: nonempty (projective_limit ex4_system) ->
  exists2 s, countable_set s & cofinal r s.
End Exercise4.
```

Point (d) becomes trivial. Let S' the system described in Example 2, where $\mathrm{F} = \{0\}$. The projective limit is the diagonal of a given set; it is non-empty. For u_i we take the constant function zero. Now $\varprojlim u_i$ is a function whose source is empty and target is non-empty. It cannot be surjective. This terminates the proof.

```
Lemma ex4d (S:= (ex4_system (proj1 ex4d_orderIr_prop3))):
  [/\ (forall i, inc i (psI S) -> nonempty (Vg (psE S) i)),
      (forall ij, inc ij (psr S) -> surjection (Vg (psf S) ij)) &
      (projective_limit S) = emptyset].  (* 12 *)

Lemma ex4e (S := (ex4_system (proj1 ex4d_orderIr_prop3)))
      (S' := (prl_exa2_system \1c (ps_preorder_r S) (ps_substrate_r S)))
      (u:= (Lg (psI S) (fun z => (Lf (fun i => \0c) (Vg (psE S) z) \1c)))):
   [/\  prl_same_index S S',  prl_map2_compat S S' u,
  (forall i, inc i (psI S) -> surjection (Vg u i)) &
   ~(surjection (projective_limit_fun S S' u))]. (* 61 *)
```

Exercise 5. « Let I be a directed set and let $(\mathrm{E}_\alpha)_{\alpha\in\mathrm{I}}$ be a family of lattices such that each E_α, endowed with the opposite ordering is Noetherian (§6, no. 5). For each pair (α,β) of indices in I such that $\alpha \le \beta$ let $f_{\alpha\beta} : \mathrm{E}_\beta \to \mathrm{E}_\alpha$ be an *increasing* mapping, and suppose that $(\mathrm{E}_\alpha, f_{\alpha\beta})$ is an inverse system of sets relative to I. For each $\alpha \in \mathrm{I}$ let G_α be a non-empty subset of E_α such that (i) no two distinct elements of G_α are comparable, (ii) $f_{\alpha\beta}(\mathrm{G}_\beta) = \mathrm{G}_\alpha$ whenever $\alpha \le \beta$, (iii) for each $\alpha \le \beta$ and each $x_\alpha \in \mathrm{G}_\alpha$, $f_{\alpha\beta}^{-1}(x_\alpha)$ has a greatest element $\mathrm{M}_{\alpha\beta}(x_\alpha)$ in E_β, (iv) whenever $\alpha \le \beta$, if $h_\beta \in \mathrm{E}_\beta$ is such that exists $y_\beta \in \mathrm{G}_\beta$ such that $y_\beta \le h_\beta$, then for each $x_\alpha \in \mathrm{G}_\alpha$ such that $x_\alpha \le f_{\alpha\beta}(h_\beta)$ there exists $x_\beta \in \mathrm{G}_\beta$ such that $x_\beta \le h_\beta$ and $x_\alpha = f_{\alpha\beta}(x_\beta)$. Under these conditions the inverse limit of the inverse system of subsets (G_α) is *not empty*. The proof runs as follows:

(a) Let J be a *finite* subset of I. A family $(x_\alpha)_{\alpha\in\mathrm{J}}$, where $x_\alpha \in \mathrm{G}_\alpha$ for all $\alpha \in \mathrm{J}$, is said to be *coherent* if it satisfies the following two conditions: (i) if $\alpha \in \mathrm{J}$, $\beta \in \mathrm{J}$, $\alpha \le \beta$, then $x_\alpha = f_{\alpha\beta}(x_\beta)$; (ii) for each upper bound γ of J in I there exists $x_\gamma \in \mathrm{G}_\gamma$ such that $x_\alpha = f_{\alpha\gamma}(x_\gamma)$ for all $\alpha \in \mathrm{J}$. Show that for each upper bound γ of J in I, the set $\bigcap_{\alpha\in\mathrm{J}} f_{\alpha\gamma}^{-1}(x_\alpha)$ has a greatest element equal to $\inf_{\alpha\in\mathrm{J}}(\mathrm{M}_{\alpha\gamma}(x_\alpha))$; furthermore the intersection of G_γ and $\bigcap_{\alpha\in\mathrm{J}} f_{\alpha\gamma}^{-1}(x_\alpha)$ is the set (non-empty

by hypothesis) of all $y_\gamma \in G_\gamma$ such that

$$y_\gamma \leq \inf_{\alpha\in J}(M_{\alpha\gamma}(x_\alpha))$$

(use condition (i)).

(b) Let J be any subset of I. A family $x_J = (x_\alpha)_{\alpha\in J}$ where $x_\alpha \in G_\alpha$ for all $\alpha \in J$, is said to be *coherent* if every finite subfamily of x_J is coherent. If $J \neq I$ and if $\beta \in I - J$, show that there exists $x_\beta \in G_\beta$ such that the family $x_{J\cup\{\beta\}} = (x_\alpha)_{\alpha\in J\cup\{\beta\}}$ is coherent. (Using (a) and condition (iv), show that for every finite subset F of J, if γ is an upper bound of $F\cup\{\beta\}$, then $f_{\beta\gamma}(G_\gamma \cap \bigcap_{\alpha\in F} f_{\alpha\gamma}^{-1}(x_\alpha))$ is the (non-empty) set of all $y_\beta \in G_\beta$ which are $\leq f_{\beta\gamma}(\inf_{\alpha\in F}(M_{\alpha\gamma}(x_\alpha)))$. Using the fact that E_β endowed with the opposite ordering is Noetherian, show next that there exists a finite subset F_0 of J and an upper bound γ_0 of $F_0\cup\{\beta\}$ such that for each finite subset F of J and each upper bound γ of $F\cup\{\beta\}$ we have

$$f_{\beta\gamma}(\inf_{\alpha\in F}(M_{\alpha\gamma}(x_\alpha)) \geq f_{\beta\gamma_0}(\inf_{\alpha\in F_0}(M_{\alpha\gamma_0}(x_\alpha)).$$

Prove then that every element $x_\beta \in F_\beta$ which is $\leq f_{\beta\gamma_0}(\inf_{\alpha\in F_0}(M_{\alpha\gamma_0}(x_\alpha))$ satisfies the required conditions.

(c) Finally, complete the proof by showing that there exists a coherent family whose index set is the whole of I. (Order the set of coherent families x_J by the relation "x_J is a subfamily of x_K", and apply (b) and Zorn's lemma.) »

We consider a context, formed of a projective system S, two sets representing the family of lattices (L_i), and the family of sets (G_i). Both families are indexed by I, the index set of S. In what follows, we shall denote by $\leq_i$ the order relation of L_i; the substrate is E_i and whenever x and y belong to E_i, the quantities $\max(x,y)$ and $\min(x,y)$ are defined as the maximum and minimum of x and y for $\leq_i$. We shall also assume that every non-empty subset of E_i has a minimal element. If $i \leq j$ (where $\leq$ is the ordering of S), then f_{ij} is a function $E_j \to E_i$; we assume it increasing for the order relations $\leq_j$ and $\leq_i$.

```
Section Exercise5.

Variable S: projective_system.
Variable Er Gf: Set.

Hypothesis rdr:  right_directed_prop (psr S).
Hypothesis fgEr: fgraph Er.
Hypothesis dEr: domain Er = psI S.
Hypothesis lEr: forall i, inc i (psI S) -> lattice (Vg Er i).
Hypothesis sEr: forall i, inc i (psI S) -> substrate (Vg Er i) = Vg (psE S) i.
Hypothesis sen: forall i X, inc i (psI S) -> sub X (Vg (psE S) i) ->
   nonempty X -> exists a,  minimal (induced_order (Vg Er i) X) a.
Hypothesis fm: forall p, inc p (psr S) ->
     increasing_fun (Vg (psf S) p) (Vg Er (Q p))  (Vg Er (P p)).
```

We introduce 8 assumptions G_1, G_2, G_3, G_4, G_5, G_6, G_7, and G_8. Assumptions G_1, G_2, G_4 and G_6 say that the family (G_i) can be considered as a projective system S′ of subsets of the E_i, the objective is to show $\varprojlim S' \neq \emptyset$. Assumption G_3 says that no G_i is empty (which is obviously necessary). Assumption G_5 says that no two elements of G_i are comparable (for $\leq i$). Assumption G_7 is condition (iii) and assumption G_8 is condition (iv).

```
Hypothesis ex5_G1: fgraph Gf.
```

```
Hypothesis ex5_G2: domain Gf = psI S.
Hypothesis ex5_G3: forall i, inc i (psI S) -> nonempty (Vg Gf i).
Hypothesis ex5_G4: forall i, inc i (psI S) -> sub (Vg Gf i) (Vg (psE S) i).
Hypothesis ex5_G5: forall i x y,
    inc i (psI S) ->  inc x (Vg Gf i) -> inc y (Vg Gf i) -> x <> y ->
    ~ (ocomparable (Vg Er i) x y).
Hypothesis ex5_G6: forall i j, gle (psr S) i j ->
    Vfs (ex5_f i j) (Vg Gf j) = Vg Gf i.
Hypothesis ex5_G7:forall i j x, gle (psr S) i j ->  inc x (Vg Gf i) ->
  has_greatest (induced_order (Vg Er j) (Vfi1 (ex5_f i j) x)).
Hypothesis ex5_G8: forall i j h x,  gle (psr S) i j -> inc h (Vg (psE S) j) ->
   (exists2 y, inc y (Vg Gf j) & gle  (Vg Er j) y h) ->
   inc x (Vg Gf i)  -> gle (Vg Er i) x (Vf (ex5_f i j) h) ->
   exists x', [/\ inc x' (Vg Gf j),  gle  (Vg Er j) x' h &
       x = Vf (ex5_f i j) x'].

Lemma ex5_Gsubfams: prl_subfam_hyp S Gf.

Definition ex5_S' :=projective_system_subsets (ex5_Gsubfams).
```

Let $X_{ij}(x) = f_{ij}^{-1}(x)$ be the set of all t such that $f_{ij}(t) = x$. If $i \leq j$ this is a subset of E_j. Assumption G_7 says that, if $x \in G_i$, X has a greatest element $M_{ij}(x)$.

```
Definition ex5_X i j x := Vfi1 (ex5_f i j) x.
Definition ex5_M i j x :=
  the_greatest (induced_order (Vg Er j) (ex5_X i j x)).


Lemma ex5_Gij_prop1 i j x: gle (psr S) i j -> inc x (Vg Gf j) ->   (* 4 *)
  inc (Vf (ex5_f i j) x) (Vg Gf i).
Lemma ex5_Gij_prop2 i j y: gle (psr S) i j -> inc y (Vg Gf i) ->   (* 4 *)
  exists2 x, inc x (Vg Gf j) & y =  Vf (ex5_f i j) x.

Lemma ex5_Xij_pr i j x: gle (psr S) i j ->
   forall t, inc t (ex5_X i j x) <->
     (inc t (Vg (psE S) j) /\ x = Vf (ex5_f i j) t).
Lemma ex5_Xij_pr2 i j x:
  gle (psr S) i j -> sub (ex5_X i j x) (Vg (psE S) j).
Lemma ex5_Mij_pr1 i j x (M:= ex5_M i j x):
  gle (psr S) i j ->  inc x (Vg Gf i) ->
  inc M (ex5_X i j x) /\
  forall t, inc t  (ex5_X i j x) -> gle (Vg Er j) t M. (* 7 *)
Lemma ex5_Mij_pr2 i j x (M:= ex5_M i j x): (* 3 *)
  gle (psr S) i j ->  inc x (Vg Gf i) ->
  Vf (ex5_f i j) M = x /\
  forall t, inc t (Vg (psE S) j) -> Vf (ex6_f i j) t = x -> gle (Vg Er j) t M.
```

We introduce

$$Y_{Jk}(x) = \bigcap_{i \in J} X_{ik}(x_i) = \bigcap_{i \in J} f_{ik}^{-1}(x_i).$$

We assume that k is an upper bound of J in I. We have $Y_{Jk}(x) \subset E_k$. If J is non-empty then $Y_{Jk}(x)$ is the set of all $t \in E_k$ such that $x_i = f_{ik}(t)$ whenever $i \in J$.

```
Definition ex5_Y J k x := intersectionf J (fun i => ex5_X i k (Vg x i)).
Definition ex5_inY J k x t :=
```

```
   forall i, inc i J -> Vg x i = Vf (ex5_f i k) t.
Definition ex5_upper_bd J k :=
  inc k (psI S) /\ (forall i, inc i J -> gle (psr S) i k).

Lemma ex5_Y_prop1 J k x: ex5_upper_bd J k ->
   sub (ex5_Y J k x) (Vg (psE S) k). (* 5 *)
Lemma ex5_Y_prop2 J k x: nonempty J -> ex5_upper_bd J k -> (* 5 *)
  forall t, inc t (ex5_Y J k x) <-> (inc t (Vg (psE S) k) /\ ex5_inY J k x t).
```

We introduce now

$$m_{\mathrm{J}k}(x) = \inf_{i\in\mathrm{J}}(\mathrm{M}_{ik}(x_i)).$$

Assume that J is a non-empty finite subset of I, k an upper bound of J and $x \in \prod_{\mathrm{J}} \mathrm{G}_i$, in other terms, x is a functional graph, with domain J, and $i \in \mathrm{J}$ implies $x_i \in \mathrm{G}_i$. So $i \in \mathrm{J}$ implies $i \leq k$ and $\mathrm{M}_{ik}(x_i) \in \mathrm{E}_k$. Since the number of terms in the inf is finite and E_k is a lattice, it follows that m is effectively

```
Definition ex5_mij_J J k x := (fun_image J (fun i => ex5_M i k (Vg x i))).
Definition ex5m J k x:= infimum (Vg Er k) (ex5_mij_J J k x).
Definition ex5_fneI J := [/\ sub J (psI S), finite_set J & nonempty J].
Definition ex5_prodG J x :=  [/\ fgraph x, domain x = J &
                 forall i, inc i J -> inc (Vg x i) (Vg Gf i)].

Section Exercise5_prop_m.
Variables J k x: Set.
Hypothesis (mp1: ex5_fneI J) (mp2: ex5_upper_bd J k)(mp3: ex5_prodG J x).

Lemma ex5m_prop1: sub (ex5_mij_J J k x) (substrate (Vg Er k)). (* 4 *)
Lemma ex5m_prop2: has_infimum (Vg Er k) (ex5_mij_J J k x). (* 4 *)
Lemma ex5m_prop3 y:
    (gle (Vg Er k) y (ex5m J k x) <->
     (forall i, inc i J -> gle (Vg Er k) y (ex5_M i k (Vg x i)))). (* 7 *)
Lemma ex5m_prop4: inc (ex5m J k x) (Vg (psE S) k). (* 2 *)
Lemma ex5m_prop3_bis i: inc i J ->
  gle (Vg Er k)  (ex5m J k x) (ex5_M i k (Vg x i)).  (* 3 *)
End Exercise5_prop_m.
```

We say that x is *J-coherent* if (C_1): J is a non-empty finite subset of I; (C_2): $x \in \prod_{i\in\mathrm{J}} \mathrm{G}_j$ (in particular, this says that J is the domain of x); (C_3): $x_i = f_{ij}(x_j)$ whenever $i \leq j$ in J; and C_4: for every upper bound k of J, $\mathrm{G}_k \cap \mathrm{Y}_{\mathrm{J}k}(x)$ is non-empty.

```
Definition ex5_coherent1 J x :=
  [/\ ex5_fneI J, ex5_prodG J x,
   forall i j, inc i J -> inc j J -> gle (psr S) i j ->
             Vg x i = Vf (ex5_f i j) (Vg x j) &
   forall k, ex5_upper_bd J k -> nonempty ((Vg Gf k) \cap (ex5_Y J k x))].
```

Let x be J-coherent, k an upper bound of J. We pretend that $m_{\mathrm{J}k}(x) = \max \mathrm{Y}_{\mathrm{J}k}(x)$. Obviously $y \in \mathrm{Y}$ implies $y \leq_k m$, so that it suffices to show $m \in \mathrm{Y}$. Take $i \in \mathrm{J}$. By definition of m, we have $m \leq_k \mathrm{M}_{ij}(x_i)$, so that $f_{ik}(m) \leq_i f_{ik}(\mathrm{M}_{ij}(x_i)) = x_i$. By assumption Y is non-empty, so that there is $x' \in \mathrm{Y}$; hence $x' \leq_k m$ and $x_i = f_{ik}(x') \leq_i f_{ik}(m)$. By antisymmetry, $f_{ik}(m) = x_i$. We then show

$$\mathrm{Y}_{\mathrm{F}k}(x) \cap \mathrm{G}_k = \{t \in \mathrm{G}_k, t \leq_k m_{\mathrm{F}k}(x)\}. \tag{3.36}$$

Obviously, the LHS is a subset of the RHS. So, assume $t \leq_k m$. If $i \in \mathrm{J}$ then $m \leq_k \mathrm{M}_{ik}(x_k)$, so that $f_{ik}(t) \leq_i f_{ik}(\mathrm{M}_{ik}(x_k)) = x_i$. Now, $f_{ik}(t) \in \mathrm{G}_i$ since $t \in \mathrm{G}_k$. Apply property G_5 to $f_{ik}(t) \leq_i x_i$; we get $f_{ik}(t) = x_i$. Since this holds for every i we have $t \in \mathrm{Y}$.

```
Lemma ex5_res1a J k x:
  ex5_coherent1 J x  -> ex5_upper_bd J k ->
  greatest (induced_order (Vg Er k) (ex5_Y J k x)) (ex5m J k x). (* 25 *)
Lemma ex5_res1b J k x:
  ex5_coherent1 J x  -> ex5_upper_bd J k ->
  (Vg Gf k) \cap (ex5_Y J k x) =
  Zo (Vg Gf k) (fun y => gle (Vg Er k) y (ex5m J k x)). (* 22 *)
```

We say that x is *coherent* if $x \in \prod_{\mathrm{J}} \mathrm{G}_i$ and, whenever K is a finite non-empty subset of the domain of x, the restriction of x to K is K-coherent.

The objective here is to show that, if $j \notin \mathrm{J}$, there is an extension of x to $\mathrm{J} \cup \{j\}$ that makes it coherent. We first consider the case $\mathrm{J} = \emptyset$. The problem simplifies to: there is $t \in \mathrm{G}_j$, such that, whenever $j \leq k$, there is $t' \in \mathrm{G}_k$ such that $f_{jk}(t') = t$. This follows from $f_{jk}\langle \mathrm{G}_k \rangle = \mathrm{G}_j \neq \emptyset$.

```
Definition finite_ne_sub K J := [/\ finite_set K, nonempty K & sub K J].
Definition ex5_coherent2 x :=
  [/\  ex5_coh2 (domain x) x, sub (domain x) (psI S) &
  forall K, finite_ne_sub K (domain x) -> ex5_coherent1 K (restr x K) ].

Definition ex5_extend x j a := (x +s1 (J j a)).
Definition ex5_extend_prop x j x':= ex5_coherent2 (ex5_extend x j x').

Lemma ex5_res2 x j:
  ex5_coherent2 x -> inc j (psI S) -> domain x = emptyset ->
  exists x', ex5_extend_prop x j x'.  (* 29 *)
```

The general case being non-trivial, we consider a section where x and j are fixed, J is the non-empty domain of x, x is coherent, $j \in \mathrm{I} - \mathrm{J}$.

```
Section Exercise5b.
Variables j x: Set.
Let J' := domain x.
Hypothesis coh2: ex5_coherent2 x.
Hypothesis jJ : inc j (psI S) /\ ~ (inc j J').
Hypothesis Jne: nonempty J'.
```

We denote by $\mathrm{H}(\mathrm{F}, k)$ the property that F is a non-empty finite subset of J, k an upper bound of F and $j \leq k$. We have

$$f_{jk}\langle \mathrm{G}_k \cap \mathrm{Y} \rangle = \{y \in \mathrm{G}_j, y \leq_j f_{jk}(m)\} \neq \emptyset \qquad (\mathrm{Y} = \mathrm{Y}_{\mathrm{F}k}(x), m = \max \mathrm{Y}). \tag{3.37}$$

Write the equation as $f_{jk}\langle \mathrm{A} \rangle = \mathrm{B}$, and note that A satisfies (3.36), it is the non-empty set of all $t \in \mathrm{G}_k$ such that $t \leq_k m$. Assume $y \in \mathrm{B}$. Apply condition G_8 with $h_\beta = m$ (for y_β we can take any element of A). From $y \leq_j f_{jk}(m)$ we get: there is $t \in \mathrm{A}$ such that $y = f_{jk}(t)$. Conversely, if $y = f_{jk}(t)$ and $t \leq_k m$, we get $f_{jk}(t) \leq_j f_{jk}(m)$ and $y \in \mathrm{B}$.

```
Definition ex5_b_prop F k :=
  [/\ finite_ne_sub F J',  ex5_upper_bd F k &  gle (psr S) j k].
```

```
Lemma ex5_res3 F k (f := (ex5_f j k)) (* 22 *)
      (T:= Vfs f ((Vg Gf k) \cap (ex5_IY F k (restr x F)))):
  ex5_b_prop F k ->
  nonempty T /\
  T = Zo (Vg Gf j) (fun t => gle (Vg Er j) t (Vf f (ex5m F k (restr x F)))).
```

Let $V(F,k) = f_{jk}(m_{Fk}(x))$. We pretend that V has a minimum, under the condition that $H(F,k)$ holds. Proof. Let A be the set of all V; since $m_{Fk}(x) \in E_k$ we have $A \subset E_j$. Next, A is non-empty (there is some $i \in J$, and, since I is right directed, there is k such that $i \leq k$ and $j \leq k$, so that $H(\{j\},k)$ holds). So, by assumption A has a minimal element y. Take any element $y' \in A$. There is $y'' \in A$ such that $y'' \leq_j y$ and $y'' \leq_j y'$ (proof below). By minimality, $y'' = y$, so $y \leq_j y'$. So y is the least element of A.

Claim 1: $V(F',k) \leq V(F,k)$ when F is a subset of F' and $H(F',k)$ holds. Since f_{jk} is increasing, it follows from $m_{F'k}(x) \leq_k m_{Fk}(x)$, and this holds since m is the greatest element of Y. Claim 2: $V(F,k') \leq V(F,k)$ when $k \leq k'$. We use here (LP_I): if $i \in J$ or $i = j$, then $i \leq k \leq k'$ so that $f_{ik'} = f_{ik} \circ f_{kk'}$. As f_{jk} is increasing, it suffices to show $f_{kk'}(m_{Fk'}(x)) \leq_k m_{Fk}(x)$. Write $m = m_{Fk'}(x)$ for simplicity. As $m_{Fk}(x)$ is the greatest element of some Y, if suffices to show $f_{kk'}(m) \in Y$, hence, whenever $i \in F$, $f_{ik}(f_{kk'}(m)) = x_i$. Now $f_{ik'}(m)) = x_i$ holds because m is in some Y. Claim 3: y'' exists. Assume $y = V(F,k)$, $y' = V(F',k')$, take $F'' = F \cup F'$, and for k'' an upper bound of k and k'. Define $y'' = V(F'',k'')$. That $H(F'',k'')$ holds is clear so $y'' \in A$. We have $V(F'',k'') \leq V(F,k'') \leq V(F,k)$ and $V(F'',k'') \leq V(F',k'') \leq V(F',k')$.

```
Lemma ex5_res4: exists F0 k0,
    ex5_b_prop F0 k0 /\
     forall F k, ex5_b_prop F k ->
     gle (Vg Er j) (Vf (ex5_f j k0) (ex5m F0 k0 (restr x F0)))
         (Vf (ex5_f j k) (ex5m F k (restr x F))). (* 123 *)
```

Take (F_0,k_0) minimizing V under the constraint H. By equation (3.37), if $H(F,k)$ holds, then $f_{jk}\langle G_k \cap Y\rangle = \{y \in G_j, y \leq V\}$. Moreover, this set is non-empty; so that we can take an element $x' \in G_j$ such that $x' \leq V_0$. Whenever $H(F,k)$ holds, we have $x' \leq V$; so that there exists $u \in G_k$, $x' = f_{jk}(u)$ and $u \in Y_{Fk}(x)$. Example: assume $i \leq j$, where $i \in J$. Then $H(\{i\},j)$ holds. As $j = k$ we have $x' = u \in Y$. This says $f_{ij}(x') = x_i$. Assume on the contrary $j \leq i$, so that $H(\{i\},i)$ holds. Now $x' = f_{ji}(u)$ and $u \in Y$ says $x_i = f_{ii}(x) = u$. So $x_i = f_{ij}(x')$.

Extend now x to $J' = J \cup \{j\}$ by defining $x_j = x'$. The previous discussion says: whenever $i \leq k$ are two indices in J' we have $x_i = f_{ik}(x_k)$, since this holds when the indices are in J. We have to show that for every finite non-empty subset K of J', the extension is K-coherent. Conditions (C_1) and (C_2) are easy. Condition C_3 holds as well.

Let's show (C_4), and consider an upper bounder k of K. Assume first $K = \{j\}$. The result follows from assumption G_6. Assume $j \notin K$. The result follows as x is K-coherent. So, assume $K = K' \cup \{j\}$, where K' is non-empty. So $H(K',k)$ holds, and there is u such that $u \in G_k$, $x' = f_{jk}(u)$ and $u \in Y_{K'k}(x)$.

```
Lemma ex5_res5: exists x', ex5_extend_prop x j x'.  (* 118 *)

End Exercise5b.
```

Obviously G forms a projective system of subsets of S, and $\varprojlim G$ is nonempty if there is a coherent system x with domain I (recall that $x \in \prod_I G_i$ and $x_i = f_{ij}(x_j)$ holds). Consider the set of all functional graphs whose domain is a subset of I and whose range is a subset of $\bigcup_i G_i$.

This allows us to define the set T of all coherent graphs, and order it by inclusion. Let X be a totally ordered subset of T, and $x = \bigcup X$. Each $t \in X$ is a functional graph, so that x is a graph. Assume $a \in x$, $b \in x$, a and b have the same first projections. Assume $a \in t_a$, $b \in t_b$. Since X is totally ordered, we have $t_a \subset t_b$ or $t_b \subset t_a$. In the first case, a and b are in the functional graph t_b, so that $a = b$. The same holds in the second case; so that x is a functional graph. Assume $a \in t$, $t \in X$, and let i be the first component of a. We have $x(i) = t(i)$. Note that every i in the domain of x has this form. In particular $x(i) \in G_i$. This shows $x \in \prod G_i$. Consider now a non-empty finite subset K of the domain of x. By finite induction there is $t \in X$ such that K is a subset of the domain of t (assume $K = \{a, b\}$, a is in the domain of t_a, b is in the domain of t_b; one of t_a and t_b is the greatest for inclusion, let's says it is t_a; then K is a subset of t_a). The restriction of x to K is K-coherent since this is the restriction of t to K. It follows $x \in X$. By a corollary of Zorn's lemma, T has a maximal element x. The previous result shows that x can be extended when the domain is not I. So x has domain I and this concludes the proof.

```
Lemma ex5_result: nonempty (projective_limit  ex5_S'). (* 80 *)
```

Exercise 6. « Let I be a directed set, and let $(J_\lambda)_{\lambda \in L}$ be a family of subsets of I satisfying the conditions of Exercise 1. Let $(E_\alpha, f_{\beta\alpha})$ be a direct system of sets indexed by I, let $E = \varinjlim E_\alpha$, and for each $\lambda \in L$ let F_λ be the direct limit of the direct system obtained from $(E_\alpha, f_{\beta\alpha})$ by restricting the index set to J_λ. Whenever $\lambda \leq \mu$, let $g_{\mu\lambda}$ be the canonical mapping of F_λ into F_μ (no. 6). Show that $(F_\lambda, g_{\mu\lambda})$ is a direct system of sets relative to L and define a canonical bijection of E onto $F = \varinjlim F_\lambda$. »

Assumptions are as in Exercise 1, except that we need J_i to be right directed for $j \in L$.

```
Section Exercise6.

Variables I rI L rL Jf: Set.
Variable S: inductive_system.

Hypothesis rS: (isr S = rI).
Hypotheses (HIs: substrate rI = I).
Hypotheses (HLp :preorder rL)
           (HLs: substrate rL = L)
           (HLd:right_directed_on rL L).
Hypothesis (HJg: fgraph Jf)
           (HJd: domain Jf = L)
           (HJI: unionb Jf = I)
           (HJm: forall i j, gle rL i j -> sub (Vg Jf i) (Vg Jf j))
           (HJrd: forall j, inc j L -> right_directed_on rI (Vg Jf j)).
```

Some trivial properties. In particular, if $i \in L$, then $J_i \subset I$; moreover, J_i is right directed, so that we can define S_i; we generalize the definition by using the axiom of choice. We define $F_i = \varinjlim S_i$. We give a lemma that explicits the fields of S_i, in case $i \in L$. Assume $i \leq_L j$; from $J_i \subset J_j$ we deduce that J_i is a right directed subset of the substrate of the order of S_j. The system formed from S_j by restricting the index set to J_i has the same data as S_i.

```
Lemma ex6_prop1 i: inc i L -> sub (Vg Jf i) I.
Lemma ex6_prop2 i j: gle rL i j -> inc i L /\ inc j L.
```

```
Lemma ex6_prop3: I = isI S.
Lemma ex6_prop4 i: inc i L -> sub_right_directed (Vg Jf i) (isr S). (* 2 *)


Definition ex6_systemi i:=
  match (ixm (inc i L)) with
    | inl hx => (inl_restr (ex6_prop4 hx))
    | inr _ => S
end.
Definition ex6_Fl i := inductive_limit (ex6_systemi i).

Lemma ex6_res0 i (H: inc i L):
  ex6_Fl i = inductive_limit (inl_restr (ex6_prop4 H)).

Lemma ex6_prop5a i (H:inc i L):
  inl_same_data (ex6_systemi i) (inl_restr (ex6_prop4 H)).
Lemma ex6_prop5b i (Si := (ex6_systemi i)):  inc i L ->
      [/\ isE Si = restr (isE S) (Vg Jf i),
       isI Si = Vg Jf i,
       isr Si = induced_order (isr S) (Vg Jf i)&
       isf Si =  restr (isf S) (induced_order (isr S) (Vg Jf i))].
Lemma ex6_prop5c i j: gle rL i j ->
  sub_right_directed (Vg Jf i) (isr (ex6_systemi j)). (* 10 *)
Lemma ex6_prop5d i j (H:gle rL i j):
  inl_same_data (ex6_systemi i) (inl_restr (ex6_prop5c H)). (* 6 *)
Lemma ex6_prop6a i (H:inc i L):
  inl_equiv (ex6_systemi i) =  inl_equiv (inl_restr (ex6_prop4 H)).
Lemma ex6_prop6b  i j (H:gle rL i j):
  inl_equiv (ex6_systemi i) = inl_equiv (inl_restr (ex6_prop5c H)).
Lemma ex6_res1 i j (H: gle rL i j):
  ex6_Fl i = (inductive_limit (inl_restr (ex6_prop5b H))).
```

We define here g_{ji} as some inductive map. Assume $i \leq_{\mathrm{L}} j$, so that i and j belong to L. Then g_{ji} is the canonical function $\mathrm{S}_i \to \mathrm{S}_j$. It is the identity function when $i = j$. Moreover, it satisfies $(\mathrm{LI}_{\mathrm{I}})$ so that one can define an inductive system with the F_i and g_{ij}, let's call it S'.

```
Definition ex6_gij ij :=
  inductive_map (ex6_systemi (P ij))
                (Lg (Vg Jf (P ij))  (inl_can_fun (ex6_systemi (Q ij))))
                (inductive_limit (ex6_systemi (Q ij))).

Lemma ex6_gij_prop1 i j (H:gle rL i j):
   ex6_gij (J i j) = inl_restr_cf (ex6_prop5c H).

Lemma ex6_res2 i j: gle rL i j ->
  function_prop (ex6_gij (J i j))  (ex6_Fl i) (ex6_Fl j). (* 2 *)
Lemma ex6_res3 i: inc i L -> ex6_gij (J i i) = identity (ex6_Fl i).
Lemma ex6_res4 i j k: gle rL i j ->  gle rL j k ->
   ex6_gij (J j k) \co ex6_gij (J i j) = ex6_gij (J i k).


Definition ex6_F: inductive_system.
Lemma ex6_F_prop: inductive_system_on ex6_F (Lg L ex6_Fl) L rL (Lg rL ex6_gij).
```

Let's define a canonical injection $f : \varinjlim \mathrm{S}' \to \varinjlim \mathrm{S}$. If $x \in \varinjlim \mathrm{S}'$, there exists $j \in \mathrm{J}$, $i \in \mathrm{J}_j$ and $y \in \mathrm{E}_i$ such that $x = \mathscr{C}_{\mathrm{S}'}(\mathscr{C}_{\mathrm{S}_j}(y,i),j)$. Write this as $x = \mathrm{W}(y,i,j)$. Note that the value of

W does not change if j is replaced by j' when $j \leq_{\mathrm{L}} j'$. So we can define $f(x) = \mathscr{C}_{\mathrm{S}}(y,i)$. This belongs to $\varinjlim \mathrm{S}$ since $i \in \mathrm{I}$. Assume $f(x) = f(x')$ so $\mathscr{C}_{\mathrm{S}}(y,i) = \mathscr{C}_{\mathrm{S}}(y',i')$. There is i'' such that $f_{i''i}(y) = f_{i''i'}(y')$. This implies $\mathscr{C}_{\mathrm{S}_k}(y,i) = \mathscr{C}_{\mathrm{S}_k}(y',i')$, whenever k is big enough (there is j'' such that $i'' \in \mathrm{J}_{j''}$; we need $i'' \leq_{\mathrm{L}} k$). One deduces $\mathrm{W}(y,i,k) = \mathrm{W}(y',i',k)$. If moreover $i' \leq_{\mathrm{L}} k$ and $i' \leq_{\mathrm{L}} k$, one deduces $x = x'$ so that f is injective.

Consider now an element of $\varinjlim \mathrm{S}$, say $y = \mathscr{C}_{\mathrm{S}}(z,i)$. Since $i \in \mathrm{I}$, there is j such that $i \in \mathrm{J}_j$; take $x = \mathrm{W}(z,i,j)$. Obviously $x \in \varinjlim \mathrm{S}'$. We pretend $f(x) = y$. Recall that $x = \mathrm{W}(z',i',j')$ and $f(x) = \mathrm{C}_{\mathrm{S}}(z',i')$, so our objective becomes $\mathscr{C}_{\mathrm{S}}(z,i) = \mathscr{C}_{\mathrm{S}}(z',i')$, where i', j', z', are defined by the axiom of choice. Write $x_1 = \mathscr{C}_{\mathrm{S}_j}(y,i)$ so that $x = \mathscr{C}_{\mathrm{S}'}(x_1,j)$. We also have $x = \mathscr{C}_{\mathrm{S}'}(x'_1,j')$. This says that there is j'' such that $j \leq_{\mathrm{L}} j''$, $j' \leq_{\mathrm{L}} j''$ and $g_{j''j'}(x_1) = g_{j''j'}(x'_1)$. By definition of g, and rewriting x_1, x'_1, this is $\mathscr{C}_{\mathrm{S}}(z,i) = \mathscr{C}_{\mathrm{S}}(z',i')$, where S should be replaced by $\mathrm{S}_{j''}$; but, given the definition of S_j, we may omit the index and conclude.

```
Definition ex6_fct x :=
  let i := (Q (rep (P (rep x)))) in let y := P (rep (P (rep x))) in
  class (inl_equiv S) (J y i).
Definition ex6_F_val y i j :=
  class (inl_equiv ex6_F) (J (class (inl_equiv (ex6_systemi j)) (J y i)) j).
Definition ex6_fct x :=
  let i := (Q (rep (P (rep x)))) in let y := P (rep (P (rep x))) in
  class (inl_equiv S) (J y i).


Lemma ex6_F_prop1 x  (* 4 *)
  (j := Q (rep x)) (i := (Q (rep (P (rep x))))) (y := P (rep (P (rep x)))):
  inc x (inductive_limit ex6_F) ->
  [/\ inc j L, inc i (Vg Jf j),  inc y (Vg (isE S) i) & x = ex6_F_val y i j].

Lemma ex6_F_prop2 y i j j':  (* 15 *)
  inc i (Vg Jf j) -> inc y (Vg (isE S) i) -> gle rL j j' ->
   ex6_F_val y i j =  ex6_F_val  y i j'.

Lemma ex6_fct_ax:
   lf_axiom ex6_fct (inductive_limit ex6_F) (inductive_limit S). (* 3 *)
Lemma ex6_fct_fi: injection ex6_iso.  (* 23 *)
Lemma ex6_fct_fs: surjection ex6_iso.  (* 40 *)
mma ex6_fct_bp:
  bijection_prop  ex6_iso (inductive_limit ex6_F) (inductive_limit S).
End Exercise6.
```

Exercise 7. « Let I be a directed set and let $(\mathrm{E}_\alpha, f_{\beta\alpha})$ be a direct system of sets relative to I. For each $\alpha \in \mathrm{I}$, let $f_\alpha : \mathrm{E}_\alpha \to \mathrm{E} = \varinjlim \mathrm{E}_\alpha$ be the canonical mapping. In each E_α, let R_α be the equivalence relation $f_\alpha(x) = f_\alpha(y)$. Show that, whenever $\alpha \leq \beta$, the mapping $f_{\beta\alpha}$ is compatible with the equivalence relations R_α and R_β. Let $\mathrm{E}'_\alpha = \mathrm{E}_\alpha/\mathrm{R}_\alpha$, and let $f'_{\beta\alpha}$ be the mapping of E'_α into E'_β induced by $f_{\beta\alpha}$ on passing to the quotients. Show that $f'_{\beta\alpha}$ is injective and that $(\mathrm{E}'_\alpha, f'_{\beta\alpha})$ is a direct system of sets, and define a canonical bijection of E onto $\varinjlim \mathrm{E}'_\alpha$. »

Obviously R_i is an equivalence relation on E_i. Write $x \equiv_i y$ when x and y are related by R_i. This is the same as $\mathscr{C}_{\mathrm{S}}(x,i) = \mathscr{C}_{\mathrm{S}}(y,i)$. It clearly implies $f_{ji}(x) \equiv_j f_{ji}(y)$. This is the compatibility property.

```
Section Exercise7.
Variable S : inductive_system.

Definition ex7_eqv i := equivalence_associated  (inl_can_fun S i).

Lemma ex7_eqv_prop1 i: inc i (isI S) ->
   equivalence_on (ex7_eqv i) (Vg (isE S) i). (* 3 *)
Lemma ex7_eqv_prop2 i: inc i (isI S) ->  forall x y,  (* 7 *)
  related (ex7_eqv i) x y <->
  [/\ inc x (Vg (isE S) i), inc y (Vg (isE S) i) &
   class (inl_equiv S) (J x i) = class (inl_equiv S) (J y i)].
Lemma ex7_eqv_prop3 i j: gle (isr S) i j ->
  compatible_with_equivs (Vg (isf S) (J i j)) (ex7_eqv i)(ex7_eqv j). (* 12 *)
```

We define here E'_i the quotient and f'_{ji}, the functions on the quotients. We denote by $\mathscr{C}_i(x)$ the class of R_i, so that $f'_{ji}(x) = \mathscr{C}_j(f_{ji}(\mathscr{R}(x))$, where $\mathscr{R}(x)$ is a representative of x (an element of E_i whose class is x). If $i = j$ we have $f'_{ji}(x) = \mathscr{C}_i(\mathscr{R}(x))$ so that f'_{ij} is the identity function. If $f'_{ji}(x) = f'_{ji}(y)$ then $\mathscr{C}_j(f_{ji}(\mathscr{R}(x))) = \mathscr{C}_j(f_{ji}(\mathscr{R}(y)))$. This gives $\mathscr{C}_{\mathrm{S}}(f_{ji}(\mathscr{R}(x)), j) = \mathscr{C}_{\mathrm{S}}(f_{ji}(\mathscr{R}(y)), j)$, then $\mathscr{C}_{\mathrm{S}}(\mathscr{R}(x), i) = \mathscr{C}_{\mathrm{S}}(\mathscr{R}(y), i)$ and $\mathscr{C}_i(\mathscr{R}(x)) = \mathscr{C}_i(\mathscr{R}(y))$. This says $x = y$ and f'_{ji} is injective.

```
Definition ex7_Ei i := quotient (ex7_eqv i).
Definition ex7_fij ij:=
  fun_on_quotients  (ex7_eqv (P ij)) (ex7_eqv (Q ij)) (Vg (isf S) ij).

Lemma ex7_fij_prop1 i j: gle (isr S) i j ->
   function_prop (ex7_fij i j) (ex7_Ei i) (ex7_Ei j). (* 8 *)
Lemma ex7_fij_ev i j x: gle (isr S) i j -> inc x (ex7_Ei i) ->  (* 4 *)
    Vf (ex7_fij i j) x =  class (ex7_eqv j) (Vf (Vg (isf S) (J i j)) (rep x)).
Lemma ex7_fij_ev_bis i j x: gle (isr S) i j -> inc x (Vg (isE S) i) ->
     Vf (ex7_fij i j) (class (ex7_eqv i) x) =
    class (ex7_eqv j) (Vf (Vg (isf S) (J i j)) x). (* 12 *)
Lemma ex7_fij_prop2 i: inc i (isI S) -> (* 7 *)
   (ex7_fij i i) = identity (ex7_Ei i).
Lemma ex7_fij_prop3 i j k: gle (isr S) i j -> gle (isr S) j k ->  (* 28 *)
   ex7_fij j k \co ex7_fij i j = ex7_fij i k.
Lemma ex7_fij_prop4 i j : gle (isr S) i j -> injection  (ex7_fij i j). (* 16 *)
```

We can define an inductive system with f'_{ji} and E'_i. Let's call it S'. Assume $x \in \varinjlim \mathrm{S}'$, so that $x = \mathscr{C}_{\mathrm{S}'}(z, i)$, where $z \in \mathrm{E}'_i$; we have $z = \mathscr{C}_i(z')$, where $z' = \mathscr{R}(z) \in \mathrm{E}_i$. Then $x \mapsto \mathscr{C}_{\mathrm{S}}(z', i)$ is a canonical bijection $\varinjlim \mathrm{S}' \to \varinjlim \mathrm{S}$.

```
Definition ex7_Ei_fam := Lg (isI S) ex7_Ei.
Definition ex7_fij_fam := Lg (isr S) (fun ij => ex7_fij (P ij) (Q ij)).

Lemma ex7_fij_prop1' p: inc p (isr S) -> (* 5 *)
   function_prop (Vg ex7_fij_fam p) (Vg ex7_Ei_fam (P p)) (Vg ex7_Ei_fam (Q p)).

Definition ex7_system: inductive_system.  (* 17 *)
Lemma ex7_system_val: inductive_system_on ex7_system
  ex7_Ei_fam (isI S) (isr S) ex7_fij_fam.

Definition ex7_fct x :=
  class (inl_equiv S) (J (rep (P (rep x))) (Q (rep x))).
```

```
Definition  ex7_iso :=
  Lf ex7_fct (inductive_limit ex7_system) (inductive_limit S).

Lemma ex7_can_val_bj :  (* 53 *)
  bijection_prop ex7_iso (inductive_limit ex7_system) (inductive_limit S).

End Exercise7.
```

Exercise 8. «Let $(E_\alpha, f_{\beta\alpha})$ and $(F_\alpha, g_{\beta\alpha})$ be two direct systems of sets, both indexed by the same directed set I. For each $\alpha \in I$, let u_α be a mapping of E_α into F_α such that the u_α form a direct system of mappings. Let $G_\alpha \subset E_\alpha \times F_\alpha$ be the graph of u_α. Show that (G_α) is a direct system of subsets of $E_\alpha \times F_\alpha$ and that its direct limit may be canonically identified with the graph of $u = \varinjlim u_\alpha$.»

The first claim is easy (same proof as Exercise 3). Assume that t is an element of the graph of u. So that is $i \in I$, $x' \in E_i$ such that $x = \mathscr{C}_S(x', i)$, $y = \mathscr{C}_{S'}(u_i(x'), i)$ and $t = (x, y)$. This means that $(x', u_i(x')) \in G_i$.

```
Section Exercise8.

Variables S S': inductive_system.
Variable (u:Set).
Hypothesis same_I: (inl_same_index S S').
Hypothesis (Hu: inl_map2_compat S S' u).

Lemma ex8_inl_subfm_hyp (S'' := inl_system_product same_I):
  inl_subfam_hyp S'' (Lg (isI S) (fun i => graph (Vg u i))). (* 35 *)


Definition ex8limit_graphs := inductive_system_subsets ex8_inl_subfm_hyp.
Definition ex8_graphs_limit := graph (inductive_limit_fun S S' u).
Definition ex8_gl_val t :=
  let i := (Q (rep (P t))) in let x := (P (rep (P t))) in
  class (inl_equiv ex8limit_graphs) (J (J x (Vf (Vg u i) x)) i).

Lemma ex8limit_graphs_prop t (i := (Q (rep (P t)))) (x := (P (rep (P t)))):
  inc t ex8_graphs_limit ->
  [/\ inc i (isI S), inc x (Vg (isE S) i) &
        t = J (class (inl_equiv S) (J x i))
              (class (inl_equiv S') (J (Vf (Vg u i) x) i))]. (* 8 *)

Lemma ex8_gl_val_ax:
  lf_axiom ex8_gl_val ex8_graphs_limit (inductive_limit ex8limit_graphs). (* 5 *)

Lemma ex8_gl_val_bf (E := (inductive_limit ex8limit_graphs))
       (f:= Lf ex8_gl_val ex8_graphs_limit  E):
  bijection_prop f ex8_graphs_limit  E.  (* 73 *)

End Exercise8.
```

Exercise 9. « Let I be an *arbitrary* preordered set, and let $(E_\alpha)_{\alpha\in I}$ be a family of sets indexed by I. For each pair of indices (α,β) such that $\alpha \le \beta$, let $f_{\beta\alpha}$ be a mapping of E_α into E_β , and suppose that these mappings satisfy conditions (LI_I) and (LI_{II}). Let G be the set which is the sum of the family E_α and (with the notations of no. 5) let $R\{\!\{x,y\}\!\}$ be the relation "$\lambda(x) = \alpha \le \lambda(y) = \beta$ and $y = f_{\beta\alpha}(x)$" between two elements x, y of G. Let R' be the equivalence relation on G whose graph is the smallest of the graphs of equivalence relations which contain the graph of R (Chapter II, §6, Exercise 10). The set $E = G/R'$ is called the *direct limit* of the family E_α with respect to the family of mappings $(f_{\beta\alpha})$, and we write $E = \varinjlim E_\alpha$. When the index set I is *directed*, show that the definition agrees with that given in no. 5. In the general case, the restriction to E_α of the canonical mapping of G into E_α is called the canonical mapping of E_α into E and is denoted by f_α. Suppose we are given, for each $\alpha \in I$, a mapping u_α of E_α into F such that $u_\beta \circ f_{\beta\alpha} = u_\alpha$ whenever $\alpha \le \beta$; show that there exists a unique mapping u of E into F such that $u = u_\alpha \circ f_\alpha$ for each $\alpha \in I$. »

The context will be the following:

```
Section Exercise9.

Variables (E I r f: Set).
Hypothesis (or: preorder r) (sr:  substrate r = I)
  (fgE:fgraph E) (dE: domain E = I)
  (fgf: fgraph f) (df: domain f = r)
  (function_f:
    forall p, inc p r ->
              function_prop (Vg f p) (Vg E (P p)) (Vg E (Q p)))
  (compose_f: forall i  j k, gle r i j -> gle r j k ->
                  Vg f (J j k) \co  Vg f (J i j) = Vg f (J i k))
  (identity_f: forall i, inc i I -> Vg  f (J i i) = identity (Vg E i)).
```

We consider a set G and two relations R and R'. The first relation is proposed by Bourbaki, the second relation is the equivalence of an inductive system. Obviously, R implies R'. We consider the least equivalence relation s on G such that $R(x,y)$ implies $s(x,y)$. Note that R is reflexive on G, but not symmetric, so that we replace the relation by "x and y are in G, $R(x,y)$ or $R(y,x)$"; this does not change s. Exercise 10 of Chapter 2, implemented in the main text gives an explicit form for s (See part I of this report). In particular, $R'(x,y)$ implies $s(x,y)$. Assume I right directed; we can consider the inductive system defined by I, r, E, f and all the axioms; it follows that R' is an equivalence relation on G. In this case, s is equal to the graph of R' on G.

```
Definition ex9_G := disjointU E.
Definition ex9_rel x y:=
  gle r (Q x) (Q y) /\ P y = Vf (Vg f (J (Q x) (Q y))) (P x).

Definition ex9_srel x y :=
   exists k, [/\ gle r (Q x) k,  gle r (Q y) k &
   Vf (Vg f (J (Q x) k)) (P x) =  Vf (Vg f (J (Q y) k)) (P y) ].

Lemma ex9G_P x: inc x ex9_G <->
  [/\ pairp x, inc (Q x) I & inc (P x) (Vg E (Q x))]. (* 2 *)
Lemma ex9_propa x y: inc x ex9_G -> inc y ex9_G ->
  ex9_rel x y -> ex9_srel x y.  (* 3 *)


Definition ex9_rels x y:=
```

```
  [/\ inc x ex9_G, inc y ex9_G & ex9_rel x y \/  ex9_rel y x].
Definition ex9_rels_ext := chain_equivalence ex9_rels ex9_G.


Lemma ex9_propb: reflexive_re  ex9_rels ex9_G. (* 6 *)
Lemma ex9_propc: symmetric_r  ex9_rels.
Lemma ex9_propd: (forall x y, ex9_rels x y -> inc x ex9_G).
Lemma chain_equivalence_eq: equivalence_on ex9_rels_ext ex9_G.
Lemma ex9_rels_ext_minimal:
  ex9_rels_ext  = eqv_smallest ex9_G ex9_rels.
Lemma ex9_prope: sub (graph_on ex9_srel ex9_G) ex9_rels_ext.  (* 13 *)
Lemma ex9_propf: (* 3 *)
  right_directed_on r I -> equivalence_on (graph_on ex9_srel ex9_G) ex9_G.

Lemma ex9_rels_special: (* 10 *)
  right_directed_on r I ->  ex9_rels_ext = graph_on  ex9_srel ex9_G.
```

Let's consider the quotient of G by s. In the special case where I is right directed, and S is the inductive system mentioned above, this quotient is $\varinjlim$S. In the general case, we can define the canonical function.

```
Definition ex9_quo := quotient ex9_rels_ext.
Definition ex9_can_fun i :=
  Lf (fun x => class  ex9_rels_ext (J x i)) (Vg E i) ex9_quo.

Lemma ex9_quoP x: inc x ex9_quo <-> classp ex9_rels_ext x.
Lemma ex9_propg (h: right_directed_on r I)   (* 4 *)
  (S := InductiveSystem or sr h fgE dE fgf df function_f compose_f identity_f):
  ex9_quo = inductive_limit S.
Lemma ex9_can_fun_ax i: inc i I  ->
   lf_axiom (fun x => class ex9_rels_ext (J x i)) (Vg E i) ex9_quo. (* 3 *)
Lemma ex9_can_fun_fp i: inc i I ->
     function_prop (ex9_can_fun i) (Vg E i)  ex9_quo. (* 2 *)
```

Assume now that we have a family of functions $u_i : \mathrm{E}_i \to \mathrm{F}$, satisfying some properties. If $x = (x', i)$ and $y = (y', j)$ then $\mathrm{R}(x, y)$ implies $u_i(x') = u_j(y')$. Write this as $\mathrm{R}(x, y) \implies r(x, y)$. Note that r is an equivalence relation, so that, if s is the least equivalence that extends R, we have $s(x, y) \implies r(x, y)$. Let X be an equivalence class of s, and (x', i) an element of the class; then $u_i(x')$ depends only of X. We denote it by u(X). This defines a function u such that that $u_i = u \circ f_i$. Conversely, if this holds, then $u(\mathscr{C}_s(x, i)) = u_i(x)$ whenever $i \in \mathrm{I}$ and $x \in \mathrm{E}_i$. This implies uniqueness.

```
Variables (u F: Set).

Hypotheses
  (fgu:fgraph u)
  (du: domain u = I)
  (function_u: forall i, inc i I -> function_prop (Vg u i) (Vg E i) F)
  (compose_u: forall i j, gle r i j -> (Vg u j) \co Vg f (J i j) = Vg u i).

Definition ex9_map_property  g:=
  function_prop g ex9_quo F /\
  forall i, inc i I -> (Vg u i) = g \co (ex9_can_fun i).
```

```
Lemma ex9_map_property_res1 g i x:
  ex9_map_property g ->
  inc i I -> inc x (Vg E i) ->
  Vf g (class ex9_rels_ext (J x i)) = Vf (Vg u i) x. (* 6 *)
Lemma ex9_map_unique g g':
  ex9_map_property g -> ex9_map_property g' -> g = g'. (* 8 *)



Definition ex9_map_val := fun y => Vf (Vg u (Q (rep y))) (P (rep y)).
Definition ex9_map :=  Lf ex9_map_val ex9_quo F.

Lemma ex9_map_ax : lf_axiom  ex9_map_val ex9_quo F. (* 4 *)

Lemma ex9_map_aux x y: related ex9_rels_ext x y  ->
  Vf (Vg u (Q x)) (P x) = Vf (Vg u (Q y)) (P y). (* 13 *)

Lemma ex9_map_prop: ex9_map_property ex9_map. (* 19 *)
End Exercise9.
```

Bibliography

[1] N. Bourbaki. *Éléments de mathématiques, I les structures fondamentales de l'analyse, Livre 1 Théorie des ensembles, Chapitre 3 Ensembles ordonnés, cardinaux, nombres entiers.* Hermann, 1956.

[2] N. Bourbaki. *Éléments de mathématiques, I les structures fondamentales de l'analyse, Livre 1 Théorie des ensembles, Chapitre 4 structures.* Hermann, 1957.

[3] N. Bourbaki. *Elements of Mathematics, Theory of Sets.* Springer, 1968.

[4] N. Bourbaki. *Éléments de mathématiques, Théorie des ensembles.* Diffusion CCLS, 1970.

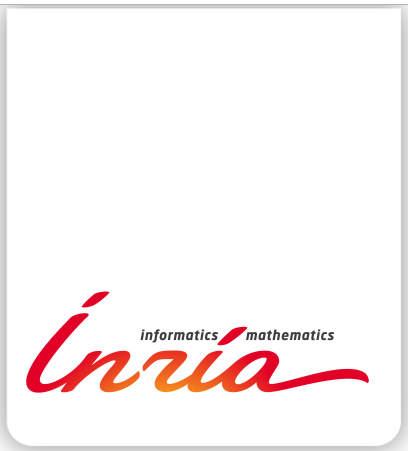
[5] N. Bourbaki. *Elements of Mathematics, Algebra I.* Springer, 1989.

[6] François Garillot, Georges Gonthier, Assia Mahboubi, and Laurence Rideau. Packaging Mathematical Structures. Technical report, March 2009. in: Theorem Proving in Higher Order Logics, Lecture Notes in Computer Science 5674, https://hal.inria.fr/inria-00368403.

[7] José Grimm. Implementation of Bourbaki's Elements of Mathematics in Coq: Part One, Theory of Sets. Research Report RR-6999, INRIA, 2009. http://hal.inria.fr/inria-00408143/en/.

[8] José Grimm. Implementation of Bourbaki's Elements of Mathematics in Coq: Part Two; Ordered Sets, Cardinals, Integers. Research Report RR-7150, INRIA, 2009. http://hal.inria.fr/inria-00440786/en/.

Contents



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399