Formalization of SLD-Resolution in the calculus of inductive c

Mathieu Jaume

Juin 1996

N^o 96-

Formalization of SLD-Resolution in the calculus of inductive constructions

Mathieu Jaume

Abstract

This report presents a full formalization of the operational semantics of definite programs (used in logic programming), given by SLD-Resolution. The variables renaming process used during a SLD-derivation is completely defined in an explicit manner. Furthermore, constructive proofs of two well known lemmas (lifting lemma and switching lemma) are built from this formalization in the calculus of inductive constructions.

Formalisation de la SLD-Résolution dans le calcul des constructions inductives

Résumé

Ce rapport présente une formalisation de la sémantique opérationnelle des programmes définis (utilisés en programmation logique), définie par la SLD-Résolution. Les conditions de renommage des variables mises en jeu lors d'une SLD-dérivation sont complètement explicitées. Enfin, les preuves de deux lemmes classiques (lemme de généralisation et lemme de commutation) sont construites à partir de cette formalisation dans le calcul des constructions inductives.

Introduction

In [11], first order terms and substitutions have been formalized in the calculus of inductive constructions and a proof of the unification theorem have been built from [18]. This report follows on from this work, and is concerned with programming languages semantics. Its aim is to formalize SLD-Resolution and to build the proofs of two classical lemmas in logic programming (lifting and switching lemmas) which are proved in many books. Depending on the reader's level of understanding, some implicit background knowledge is expected in these books. The two proofs we present here have been encoded in the 5.10 version of the proof assistant COQ [7] (INRIA) which allows the interactive construction of formal (or verified) proofs and which is based on the calculus of inductives constructions. This logical framework, through a typed λ-calculus notation, is used to encode natural deduction proofs using the Curry-Howard isomorphism ([4], [9], [6], [12], [13]) which establishes a strong relationship between proofs and typed λ-terms. Note that inductive definitions are also allowed in this proof assistant [16]. In this way, in our proofs, all objects and properties are *explicitely* defined as well as all hypothesis we need for these proofs. Each step of a proof also *explicitely* results from definitions or applications of induction principles, lemmas, theorems or axioms (of course, in this report, we omit some "minor" details). This makes a difference from the initials proofs which can be found in many books on logic programming. As we will see, the main difficulty of this work concerns the definition of the variables renaming process used in a SLD-derivation. Foundations of logic programming can be found in [2], [3], [5], [8], [19], [10], [12], [13], [14], [15], [17] ; most notations used here come from [12] and [2].

1 Terms and substitutions

In this section, we give the main definitions and the unification theorem, formalized in [11]. Next, we prove some "technical" lemmas used in the following.

Given a signature Σ (*i.e.* a set with an arity function ar : $\Sigma \rightarrow \mathbb{N}$) and a countably infinite set X of variable symbols, terms are inductively defined as follows (to get an infinite number of variable and function symbols, we index them over the natural numbers):

Definition 1.1 (Terms) The set of terms, $T_\Sigma[X]$, built over $\Sigma \cup X$, is inductively defined by:

1. If x is a variable symbol in X, then $\mathrm{tv}(x)$ is a term.

2. If f is an n-ary function symbol in Σ, and l a list of n terms, then $\mathrm{tf}(f, l)$ is a term.

This definition also introduces the dependent type $L_n[X]$ of lists of terms of length n. These two sets are specified in the system COQ using the definition of mutually recursive inductive types, as follows:

```
Mutual inductive Term : Set :=
tv : var -> Term |
tf : (f:fun)(list_term (arity Lar f)) -> Term
with list_term : nat -> Set :=
nil : (list_term 0) |
cons: (n:nat)Term -> (list_term n) -> (list_term (S n)).
```

The induction principle generated by this definition is:

Induction on $T_\Sigma[X]$ (*resp.* $L_n[X]$)
Let P be a property on $T_\Sigma[X]$ and P_0 a property on $\Pi_{n\in\mathbb{N}}L_n[X]$. If the following conditions hold:

1. $\forall v \in X \quad P(\mathrm{tv}(v))$
2. $\forall f \in \Sigma \quad \forall l \in L_{\mathrm{ar}(f)}[X] \quad (P_0(\mathrm{ar}(f), l) \Rightarrow P(\mathrm{tf}(f, l)))$
3. $P_0(0, \mathrm{nil})$
4. $\forall n \in \mathbb{N} \quad \forall t \in T_\Sigma[X]$
   $P(t) \Rightarrow \forall l \in L_n[X]\, P_0(n, l) \Rightarrow P_0(n+1, \mathrm{cons}(n, t, l))$

then:

$$\forall t \in T_\Sigma[X] \quad P(t) \quad (\textit{resp.}\ \forall n \in \mathbb{N}\, \forall l \in L_n[X] \quad P_0(n, l))$$

Therefore, when we prove a property on terms, we prove a similar property on lists of terms.

Definition 1.2 (Substitutions) $\mathcal{S}_T[X]$ is the set of functions from X to $T_\Sigma[X]$.

```
Definition subst := var -> Term.
```

Each substitution s in $\mathcal{S}_T[X]$ is associated with two mutually recursive functions, $\mathrm{Subst\_t}(s)$ and $\mathrm{Subst\_l}(s)$, respectively from $T_\Sigma[X]$ to $T_\Sigma[X]$ and from $L_n[X]$ to $L_n[X]$, defined as follows:

1. $\forall x \in X \quad \mathrm{Subst\_t}(s, \mathrm{tv}(x)) = s(x)$
2. $\forall f \in \Sigma \quad \forall l \in L_{\mathrm{ar}(f)}[X] \quad \mathrm{Subst\_t}(s, \mathrm{tf}(f, l)) = \mathrm{tf}(f, \mathrm{Subst\_l}(s, l))$

1. $\mathrm{Subst\_l}(s, \mathrm{nil}) = \mathrm{nil}$
2. $\forall t \in T_\Sigma[X] \quad \forall l \in L_n[X]$

$$\mathrm{Subst\_l}(s, \mathrm{cons}(n, t, l)) = \mathrm{cons}(n, \mathrm{Subst\_t}(s, t), \mathrm{Subst\_l}(s, l))$$

In the following, these two functions will just be written s. Let us now summarize some of the classical properties on substitutions.

The *domain* and the *range* of a substitution s are defined by:

- $\forall x \in X \quad (s(x) \neq \mathrm{tv}(x) \Rightarrow x \in \mathrm{Domain}(s))$
- $\forall x \in X \, \forall y \in X \quad ((y \in \mathrm{Domain}(s) \wedge x \in s(y)) \Rightarrow x \in \mathrm{Range}(s))$

The substitution s is *relevant* to the term t if all variables which occur either in the domain of s or in the terms from the range of s also occur in t. This definition is extended to lists of terms.

The substitution s is *idempotent* when:

$$\forall x \in X \quad s(s(x)) = s(x)$$

We introduce a *preorder* on substitutions as follows:

$$s_1 \leq s_2 \quad \text{if} \quad \exists s \, \forall x \in X \quad s(s_1(x)) = s_2(x)$$

This preorder induces an equivalence relation on substitutions: s_1 and s_2 are called *variants* if $s_1 \leq s_2$ and $s_2 \leq s_1$.

We now establish the following lemmas:

Lemma 1.1 *Let s be a substitution.*

1. *If no variable occurring in the terms from the range of s, also occurs in the domain of s, then s is an idempotent substitution.*
2. *If s is an idempotent substitution, then no variable occurring in the domain of s also occurs in the terms from the range of s.*

PROOF: (1) : Let x be a variable symbol, it suffices to prove:

$$\forall v \in X \quad (v \in s(x) \Rightarrow v \notin \mathrm{Domain}(s))$$

Let v be a variable symbol which occurs in $s(x)$, two cases are possible. If $x \in \mathrm{Domain}(s)$, then, by definition, $v \in \mathrm{Range}(s)$ and, by hypothesis, v doesn't occur in the domain of s. If $x \notin \mathrm{Domain}(s)$, then we have $s(s(x)) = s(x) = \mathrm{tv}(x)$. This settles the claim.
(2) : Let x be a variable symbol which occurs in the terms from the range of s. By definition, there exists a variable symbol v, occurring in the domain of s, such that $x \in s(v)$. If x occurs in the domain of s, then we get $s(s(x)) \neq s(x)$, which is contradictory to the hypothesis. $\diamond$

Lemma 1.2 *Let s and r be two idempotent substitutions and l a list of terms. If r is relevant to $s(l)$, then $\theta = \lambda x.r(s(x))$ is an idempotent substitution.*

Proof: By lemma 1.1.1, it suffices to prove:

$$\forall x \in X \quad (x \in \text{Range}(\theta) \Rightarrow \theta(x) = \text{tv}(x))$$

Let x be a variable symbol which occurs in the terms from the range of θ. By definition, there exists a variable symbol $y \in \text{Domain}(\theta)$ such that $x \in \theta(y)$. Let us prove that x doesn't occur in the domains of r and s. If $x \in \text{Domain}(r)$, then, seeing that r is an idempotent substitution and by lemma 1.1.2, we have $x \notin \text{Range}(r)$. Two cases are possible. If $x \in s(y)$, then x cannot occur in the domain of r (because $x \in r(s(y))$ and $x \notin \text{Range}(r)$) which is contradictory. If $x \notin s(y)$, then x doesn't occur in $r(s(y))$ (because $x \notin \text{Range}(r)$) which is also contradictory. So, we get $x \notin \text{Domain}(r)$. Suppose now that x occurs in the domain of s. By lemma 1.1.2, we have $x \notin \text{Range}(s)$. First, let us prove that $x \notin \text{Range}(r)$. For this, suppose $x \in \text{Range}(r)$. Seeing that r is relevant to $s(l)$, x occurs in $s(l)$. Hence, from $x \in \text{Domain}(s)$, it follows that $x \in \text{Range}(r)$ which is contradictory. Therefore, from $x \in r(s(y))$, we get $x \in s(y)$. Now, we can establish $x \in \text{Range}(s)$ (because x occurs in the domain of s) which is contradictory. This concludes the proof. $\diamond$

We present now the main result, formalized in [11]. First, we introduce the notion of *unification*. Let t_1 and t_2 be two terms. If for a substitution s, we have $s(t_1) = s(t_2)$, then s is called a *unifier* of t_1 and t_2 and t_1 and t_2 are said to be *unifiable*. s is called *minimal* if for each unifier s' of t_1 and t_2, we have $s \leq s'$. A unifier s of t_1 and t_2 is called a *most general unifier* (or *mgu* in short) if it is minimal, idempotent and relevant to t_1 and t_2. These definitions are extended to lists of terms.

Theorem 1.1 (Unification) *Given two terms (resp. two lists of terms) A and B, either A and B cannot be unified or they can and there exists a most general unifier of A and B.*

We terminate this section with two "technical" lemmas used in the following.

Lemma 1.3 *If the two substitutions σ and θ are variants, then there exists a substitution r which satisfies for each term t:*

1. *$\sigma(t) = r(\theta(t))$.*
2. *for each variable x occurring in $\theta(t)$, there exists a variable v such that $r(x) = \text{tv}(v)$ (r can be viewed as a variable renaming).*

PROOF: Let t be a term. By hypothesis:

$$\sigma \leq \theta \Rightarrow \exists \mu_1 \quad \mu_1 \sigma = \theta$$

So, we get $\mu_1 \sigma(t) = \theta(t)$. In the same way:

$$\theta \leq \sigma \Rightarrow \exists \mu_2 \quad \mu_2 \theta = \sigma$$

So, we get $\mu_2 \theta(t) = \sigma(t)$. Hence, r exists: $r = \mu_2$. Moreover, we have $\mu_1(\mu_2(\theta(t))) = \theta(t)$. Now, let us prove that:

$$\forall x \quad (x \in \theta(t) \Rightarrow \exists v \quad \mu_2(x) = \mathrm{tv}(v))$$

If x occurs in the domain of μ_2, then $\mu_1(\mu_2(\theta(x))) = \theta(x)$ and $\mu_2(x) \neq \mathrm{tv}(x)$. So, there exists a variable v occurring in the domain of μ_1, such that $\mu_2(x) = \mathrm{tv}(v)$. If x doesn't occur in the domain of μ_2, then v exists, this is x. $\diamond$

Lemma 1.4 *Let θ and θ_1 be two substitutions, θ_2 an idempotent substitution such that $\theta = \theta_1 \theta_2$ and t a term. If $\theta t = t$ and $\theta_2 t \neq t$, then there exists a variable x occurring in t such that $x \in \mathrm{Range}(\theta)$.*

PROOF: From $\theta_2 t \neq t$, it follows that there exists a variable x, which occurs in t, such that $\theta_2 x \neq x$. By hypothesis ($\theta t = t$), we have $\theta x = x$, so we get $\theta x = \theta_1 \theta_2 x = x$. Seeing that $\theta_2 x \neq x$, there exists a variable v, which occurs in the domain of θ_1, such that $\theta_1 v = x$ and $\theta_2 x = v$. By hypothesis, θ_2 is an idempotent substitution, and we get $\theta_2 x = \theta_2 \theta_2 x$. Therefore, we have $\theta_1 \theta_2 \theta_2 x = x = \theta \theta_2 x$. Now, from $\theta_2 x \neq x$, it follows that $\theta_2 x$ occurs in the domain of θ. This settles the claim. $\diamond$

2 Syntactic objects in logic programming

We present in this section the objects, built from terms, used in logic programming. Assuming for a moment from the reader knowledge of the semantics for first order logic, we give an idea of the next formal definitions.

A formula of the form:

$$\forall \vec{x}\, (A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_m)$$

which can be written:

$$\forall \vec{x}\, (B_1 \wedge \ldots \wedge B_m \Rightarrow A_1 \vee \ldots \vee A_n)$$

where all A_i and B_i are atoms, is a *general clause*. Henceforth, we'll use the following clausal form:

$$A_1, \ldots, A_n \leftarrow B_1, \ldots, B_m$$

If a clause has only one conclusion ($n = 1$), then it is called a *definite clause* ($0 \leq m$):

$$A \leftarrow B_1, \ldots, B_m$$

and when the set of premises of a definite clause is empty ($m = 0$), we call it a *unit clause*:

$$A \leftarrow$$

When the set of conclusions of a general clause is empty ($n = 0$), we call it a *negative clause* or a *definite goal* or a *request*:

$$\leftarrow B_1, \ldots, B_m$$

which can be written:

$$\forall \vec{x}\, (\neg B_1 \vee \ldots \vee \neg B_m) \quad \text{or} \quad \neg \exists \vec{x}\, (B_1 \wedge \ldots \wedge B_m)$$

A *definite program* is a finite set of definite clauses. A *Horn clause* is either a definite clause or a negative clause.

We now consider a pair of disjoint signatures Σ_f and Σ_r. The signature $\Sigma = \Sigma_f \cup \Sigma_r$ where Σ_f is a countably set of function symbols and Σ_r a countably set of relation (or predicate) symbols is called a *predicate calculus signature*.

Definition 2.1 (Atoms) The set $At_\Sigma[X]$, of atoms, is inductively defined as follows: if p is an n-ary predicate symbol in Σ_r and l a list of n terms, then $p(l)$ is an atom.

```
Inductive atom : Set :=
pl : (p:predic)(list_term (arity_p p)) -> atom.
```

We can apply a substitution to an atom as follows:

$$\forall s \in \mathcal{S}_T[X] \quad \forall p \in \Sigma_r \quad \forall l \in L_{\mathrm{ar}(p)}[X] \quad s(p(l)) = p(s(l))$$

Two atoms $a_1 = p_1(l_1)$ and $a_2 = p_2(l_2)$ are unifiable if $p_1 = p_2$ and if there exists a most general unifier of the lists l_1 and l_2.

A request is a finite sequence, possibly empty, of atoms.

Definition 2.2 (Request) $R_\Sigma[X]$ is inductively defined by:

$$R_\Sigma[X] ::= r_\emptyset \mid c_r(At_\Sigma[X], R_\Sigma[X])$$

```
Inductive request : Set :=
true_req : request |
cons_req : atom -> request -> request .
```

The classical functions on lists are defined for requests:

- length of a request

$$\lg(r) = \begin{cases} 0 & \text{if } r = r_\emptyset \\ 1 + \lg(r') & \text{if } r = c_r(a, r') \end{cases}$$

- concatenation of two requests

$$r_1 \bowtie_r r_2 = \begin{cases} r_2 & \text{if } r_1 = r_\emptyset \\ c_r(a, r' \bowtie_r r_2) & \text{if } r_1 = c_r(a, r') \end{cases}$$

- $(n+1)$-*th* atom of a request

$$r_{/n,a} = \begin{cases} a & \text{if } r = r_\emptyset \\ \begin{cases} a' & \text{if } n = 0 \\ (r')_{/n-1,a} & \text{if } n > 0 \end{cases} & \text{if } r = c_r(a', r') \end{cases}$$

- replacing the $(n+1)$-*th* atom of a request by a request

$$r[n \leftarrow r_1] = \begin{cases} r_\emptyset & \text{if } r = r_\emptyset \\ \begin{cases} r_1 \bowtie_r r' & \text{if } n = 0 \\ c_r(a', r'[n-1 \leftarrow r_1]) & \text{if } n > 0 \end{cases} & \text{if } r = c_r(a', r') \end{cases}$$

We can apply a substitution to a request as follows:

$$\forall s \in \mathcal{S}_T[X] \quad s(r) = \begin{cases} r_\emptyset & \text{if } r = r_\emptyset \\ c_r(s(a), s(r')) & \text{if } r = c_r(a, r') \end{cases}$$

Definition 2.3 (Clauses) $C_\Sigma[X] ::= At_\Sigma[X] \times R_\Sigma[X]$.

```
Definition clause : Set := (atom*request) .
```

We can apply a substitution to a clause as follows:

$$\forall c = <a, r> \in C_\Sigma[X] \quad \forall s \in \mathcal{S}_T[X] \quad s(c) = <s(a), s(r)>$$

Definition 2.4 (Definite programs) $P_\Sigma[X]$ is inductively defined by:

$$P_\Sigma[X] ::= P_\emptyset \mid c_p(C_\Sigma[X], P_\Sigma[X])$$

```
Inductive program : Set :=
nil_pgm : program |
cons_pgm : clause -> program -> program .
```

3 SLD-Resolution

Definite programs compute through a combination of two mechanisms: replacement and unification. This form of computing is a specific form of theorem proving, called SLD-Resolution (for Selection Linear Definite) and based on resolution: an inference rule which is particularly well-suited to automation on a computer. This way gives the operational semantics of definite programs which describes what can be executed.

3.1 Resolution and transitions

The resolution principle (or cut rule) is an inference rule, which defines a deductive relation, written $\overset{n,r,C}{\rightarrow}$, on $C_\Sigma[X] \times R_\Sigma[X] \times R_\Sigma[X]$. Let C be the definite clause:

$$\underbrace{A}_{C+} \leftarrow \underbrace{A_1, \ldots, A_p}_{C-}$$

and R the request $\leftarrow L_0, \ldots, L_q$. We suppose here that the clause C is renamed with a variable renaming r such that no variable occurring in $r(C)$ also occurs R. The request R' is obtained from R and $r(C)$ (or R' is a *resolvent* of $r(C)$ and R), if the following conditions hold:

1. L_n is an atom which occurs in R, written $R_{/n}$ $(0 \leq n \leq q)$.
2. θ is a most general unifier of L_n and $r(A)$.
3. R' is the request $\leftarrow \theta(L_0, \ldots, L_{n-1}, r(A_1), \ldots, r(A_p), L_{n+1}, \ldots, L_q)$, written $\theta R[n \leftarrow r(C^-)]$.

This resolution step will be written $R \overset{n,r,C}{\rightarrow} \theta R[n \leftarrow r(C^-)]$. In this way, the resolution principle can be viewed as a rule which moves from a state to another.

Definition 3.1 (Resolution state) A resolution state is a pair $\rho.R$, where ρ is a substitution and R a request.

```
Definition state : Set := (subst * request).
```

We can now define inductively the set Γ of *transitions*. For this, only one constructor is used, which links two states. A predicate on Γ is defined, which allows to consider transitions satisfying the "rule":

$$\frac{\theta R_{/n} = \theta r(C^+)}{\rho.R \overset{n,r,C}{\rightarrow} \theta\rho.\theta R[n \leftarrow r(C^-)]}$$

So, in each resolution step, two choices are made: the choice of the selected atom and the choice of the input clause whose head unifies with the selected atom.

The clause C is renamed with the renaming substitution r, which is an element of a set defined as follows:

Definition 3.2 (Renaming substitutions) $\mathcal{S}_X[X]$ is the set of functions from X to X.

```
Definition rename := var -> var .
```

The classical definitions of properties on substitutions are extended to $\mathcal{S}_X[X]$. We now introduce a predicate S_X on this set.

Definition 3.3 (S_X) A renaming substitution r in $\mathcal{S}_X[X]$ satisfies the predicate S_X if:

1. $\forall x \in \mathrm{Domain}(r) \quad \forall y \in \mathrm{Domain}(r) \quad (x \neq y \Rightarrow r(x) \neq r(y))$
2. No variable occurring in the domain of r also occurs in the range of r (r is idempotent).

We now define, more precisely, the notion of valid transitions.

Definition 3.4 (Valid transitions) The transition:

$$\rho.R \overset{n,r,C}{\rightarrow} \theta\rho.\theta R[n \leftarrow r(C^-)]$$

is called a valid transition, if $\theta R[n \leftarrow r(C^-)]$ is a resolvent of $r(C)$ and R and if the renaming r used, satisfies the following conditions (called standardization apart):

H1 r satisfies the predicate S_X.

H2 r renames all the variables which occur in the clause C and only these variables.

H3 No variable occurring in the range of r also occurs in the domain of ρ.

H4 $r(C)$ does not have a variable in common with R.

Furthermore, ρ and R must satisfy:

H5 $\rho R = R$

In order to be able to build derivations from valid transitions, we prove that the final state of a valid transition satisfies the condition H5. For this, we first prove the following lemma.

Lemma 3.1 *If $\rho.R \overset{n,r,C}{\rightarrow} \theta\rho.\theta R[n \leftarrow r(C^-)]$ is a valid transition, then $\rho\theta R[n \leftarrow r(C^-)] = \theta R[n \leftarrow r(C^-)]$.*

PROOF: It suffices to prove:

$$\forall x \in X \quad (x \in \theta R[n \leftarrow r(C^-)] \Rightarrow x \notin \text{Domain}(\rho))$$

Let x be a variable symbol which occurs in $\theta R[n \leftarrow r(C^-)]$, two cases are possible. Either $x \in R[n \leftarrow r(C^-)]$, or $x \in \text{Range}(\theta)$. In these two cases, seeing that θ is a most general unifier of $R_{/n}$ and $r(C^+)$, we know that θ is relevant to these two atoms. Hence, either $x \in R$, or $x \in r(C)$. In the first case, by condition H5, we can conclude. In the second case, by conditions H2 and H3, we can also conclude. $\diamond$

We are now in position to prove the following lemma.

Lemma 3.2 *If* $\rho.R \overset{n,r,C}{\rightarrow} \theta\rho.\theta R[n \leftarrow r(C^-)]$ *is a valid transition, then* $\theta\rho\theta R[n \leftarrow r(C^-)] = \theta R[n \leftarrow r(C^-)]$.

PROOF: By lemma 3.1, we have:

$$\theta\rho\theta R[n \leftarrow r(C^-)] = \theta\theta R[n \leftarrow r(C^-)]$$

By definition, θ is an idempotent substitution, and we get:

$$\theta\rho\theta R[n \leftarrow r(C^-)] = \theta R[n \leftarrow r(C^-)]$$

which concludes the proof. $\diamond$

3.1.1 SLD-Resolution and derivations

A *finite valid SLD-Derivation* is a finite sequence of valid transitions of the form:

$$\rho_0.R_0 \overset{n_0,r_0,C_0}{\rightarrow} \ldots \overset{n_{k-1},r_{k-1},C_{k-1}}{\rightarrow} \rho_k.R_k$$

also written $\rho_0.R_0 \overset{*}{\rightarrow} \rho_k.R_k$. Therefore, a SLD-derivation is defined by:

- a finite sequence $R_0, R_1, \ldots, R_k$ of requests.
- a finite sequence $r_0C_0, r_1C_1, \ldots, r_{k-1}C_{k-1}$ of variants of clauses from a definite program P.
- a finite sequence $\rho_0, \rho_1, \ldots, \rho_k$ of substitutions.

such that for all $i \geq 1$:

- R_i is a resolvent of R_{i-1} and $r_{i-1}C_{i-1}$.
- r_iC_i does not have a variable in common with $R_0, r_0C_0, \ldots, r_{i-1}C_{i-1}$.

First, we introduce the syntactical definition of derivations.

Definition 3.5 (Derivations) The set $\mathcal{D}_l$ is iductively defined, from Γ, as follows:

$$\mathcal{D}_l ::= d_t^l(\Gamma) \mid d_c^l(\mathcal{D}_l, \Gamma)$$

```
Inductive deriv:Set :=
deriv_init : trans -> deriv |
deriv_cons : deriv -> trans -> deriv.
```

Next, we define a valid composable pair of transitions.

Definition 3.6 The two transitions:

$$e_i^1 \xrightarrow{t_1} e_f^1 \quad \text{and} \quad e_i^2 \xrightarrow{t_2} e_f^2$$

form a valid composable pair of transitions if they are valid and if $e_f^1 = e_i^2$.

As we remarked earlier, the derivation

$$\rho_0.R_0 \overset{n_0, r_0, C_0}{\rightarrow} \ldots \overset{n_{k-1}, r_{k-1}, C_{k-1}}{\rightarrow} \rho_k.R_k$$

is valid if each $r_i C_i$ is a clause such that $r_i C_i$ does not have any variables which already occur in the input clauses used in the derivation up to R_{i-1}. So we introduce a function on $\mathcal{D}_l$ which computes the sequence of variables which occur in the input clauses used in a derivation.

Definition 3.7 (ϑ) The function ϑ on $\mathcal{D}_l$ is recursively defined by:

1. if $d = d_t^l(t_0)$, then $\vartheta(d) = \text{var}(r(c))$ where r and c are respectively the renaming and the clause used in t_0.

2. if $d = d_c^l(d_0, t_0)$, then $\vartheta(d) = \vartheta(d_0) \bowtie_v \text{var}(r(c))$ where r and c are respectively the renaming and the clause used in t_0 ($\bowtie_v$ is a concatenation function on lists of variables).

We now define valid derivations.

Definition 3.8 (Valid derivations) Valid derivations are defined by a recursive predicate on $\mathcal{D}_l$ as follows:

1. The derivation $d = d_t^l(t)$ is valid if t is a valid transition.

2. The derivation:

$$\underbrace{\underbrace{e_i = \rho.R \xrightarrow{*} e_k}_{d_0} \xrightarrow{t_0} e_f}_{d = d_c^l(d_0, t_0)}$$

   is valid if the following conditions hold:

   - d_0 is a valid derivation.

- the pair of transitions formed by the last transition of d_0 and t_0 is a valid composable pair of transitions.
- the renaming r used in t_0 satisfies:

$$\forall x \in X \quad ((x \in \vartheta(d_0) \vee x \in R) \Rightarrow x \notin \mathrm{Range}(r))$$

We now establish the three following lemmas on variables which occur in a valid derivation. Henceforth, s_{id} stands for the substitution $\lambda x.\mathrm{tv}(x)$.

Lemma 3.3 *If $d : s_{id}.R \overset{*}{\to} \sigma.R'$ is a valid derivation, then σ is an idempotent substitution.*

PROOF: We proceed by induction on the derivation d.
For $d = d_t^l(t)$:

$$s_{id}.R \overset{n,r,C}{\to} \sigma.R'$$

By definition, σ is an idempotent substitution.
For $d = d_c^l(d_0, t_0)$:

$$s_{id}.R \overset{*}{\to} \theta_0.R_0 \overset{n,r,C}{\to} \theta\theta_0.R'$$

By induction hypothesis, θ_0 is an idempotent substitution. By definition, θ is a most general unifier of $R_{0_{/n}}$ and $r(C^+)$, hence θ is relevant to the list of terms l obtained by concatenation of lists of terms coming from these two atoms. By lemma 3.2, $\theta_0 R_0 = R_0$ and by condition H3 on t_0, $\theta_0 r(C^+) = r(C^+)$. Therefore, θ is also relevant to the list $s(l)$. Consequently, by lemma 1.2, $\theta\theta_0$ is an idempotent substitution. $\diamond$

Lemma 3.4 *If $d : \mu.R \overset{*}{\to} \theta\mu.R'$ is a valid derivation, then if x is a variable symbol occurring in R', then x occurs either in R, or in $\vartheta(d)$.*

PROOF: By induction on d
For $d = d_t^l(t)$:

$$\mu.R \overset{n,r,C}{\to} \theta\mu.\theta R[n \leftarrow r(C^-)]$$

If $x \in \theta R[n \leftarrow r(C^-)]$, two cases are possible. In the first case, $x \in \theta R$, either $x \in R$ which allows to conclude, or x occurs in the terms from the range of θ. In this case, because, by definition, θ is relevant to $R_{/n}$ and $r(C^+)$, either $x \in R_{/n}$, or $x \in r(C^+)$, which settles the claim. If $x \in \theta r(C^-)$, then either $x \in r(C^-)$ which concludes the proof, or $x \in \mathrm{Range}(\theta)$ and a similar proof can be obtained.
For $d_c^l(d_0, t_0)$:

$$\mu.R_0 \overset{*}{\to} \rho\mu.R \overset{n,r,C}{\to} \theta\rho\mu.\theta R[n \leftarrow r(C^-)]$$

Let x be a variable occurring in $\theta R[n \leftarrow r(C^-)]$, two cases are possible: either $x \in \theta R$, or $x \in \theta r(C^-)$. In the first case, two subcases are possible. If $x \in R$, then the induction hypothesis allows to conclude. If $x \in \mathrm{Range}(\theta)$, then, because θ is relevant to $R_{/n}$ and $r(C^+)$, either $x \in R_{/n}$ and then $x \in R$ which allows to conclude by induction hypothesis, or $x \in r(C^+)$ which also concludes the proof. If $x \in \theta r(C^-)$, then, either $x \in r(C^-)$ which concludes the proof, or $x \in \text{Range}(\theta)$ and a similar proof can be obtained. $\diamond$

Lemma 3.5 *If $d : s_{id}.R \overset{*}{\rightarrow} \sigma.R'$ is a valid derivation and if x is a variable symbol occurring either in the domain of σ, or in the terms from the range of σ, then either $x \in R$, or $x \in \vartheta(d)$.*

PROOF: By induction on d, in the same manner of the above lemma. $\diamond$

4 Lifting lemma

Lemma 4.1 (Lifting) *If $s_{id}.\eta R \overset{*}{\rightarrow} \rho.R_1$ is a valid derivation, where:*

- *η is a substitution relevant to R*
- *no variable occurring in the range of a renaming used in this derivation also occurs in R*

then there exists a substitution σ such that $\sigma \leq \rho\eta$ and such that:

$$s_{id}.R \overset{*}{\rightarrow} \sigma.R_2$$

is a valid derivation where for a request R_f we have:

$$\rho\eta R_f = R_1 \quad \text{and} \quad \sigma R_f = R_2$$

```
Lemma lifting : (d:deriv)(r:request)(eta:subst)
(Deriv_ok d)
->
(Fst (state_init_d d))=([x:var](tv x))
->
(Snd (state_init_d d))=(subst_req eta r)
->
(over_under_r eta r)
->
((t:trans)(IS_IN_D t d)->
 ((x:var)(IS_IN_LV x (var_req r)) -> ~(rrange (sr_trans t) x)))
->
(Ex [d0:deriv]
  ( ((t1,t2:trans)(IS_IN_D t1 d)->(IS_IN_D t2 d0)->
                (p_trans t1)=(p_trans t2))                    /\
  (Deriv_ok d0)                                               /\
  (list_var_c_d d0)=(list_var_c_d d)                          /\
  (Fst (state_init_d d0))=([x:var](tv x))                     /\
  (Snd (state_init_d d0))=r                                   /\
  (Ex [rf:request]
((Snd (state_end_d d0))=(subst_req (Fst (state_end_d d0)) rf) /\
 (Snd (state_end_d d))=
 (subst_req (Fst (state_end_d d)) (subst_req eta rf))))       /\
 (less_subst_t (Fst (state_end_d d0))
          ([x:var](Subst_t (Fst (state_end_d d)) (eta x)))))).
```

PROOF: We proceed by induction on the derivation coming from $s_{id}.\eta R$. Initial step:

$$s_{id}.\eta R \overset{n,r,C}{\rightarrow} \rho.\rho\eta \underbrace{R[n \leftarrow r(C^-)]}_{R_f}$$

By definition, we have $\rho\eta R_{/n} = \rho r(C^+)$ and, by hypothesis, we have $\eta r(C) = r(C)$. Hence, we get $\rho\eta R_{/n} = \rho\eta r(C^+)$. Therefore, $R_{/n}$ and $r(C^+)$ are unifiable and there exists a most general unifier σ of these two atoms. So, we get $\sigma \leq \rho\eta$ and we can easily prove the validity of the following derivation:

$$s_{id}.R \overset{n,r,C}{\rightarrow} \sigma.\sigma \underbrace{R[n \leftarrow r(C^-)]}_{R_f}$$

Inductive step:

$$\underbrace{s_{id}.\eta R_0 \overset{*}{\rightarrow} \mu.R_1}_{d_0} \overset{n,r,C}{\rightarrow} \theta\mu.\theta R_1[n \leftarrow r(C^-)]$$

By induction hypothesis, there exists a substitution ρ, such that $\rho \leq \mu\eta$ and such that:

$$s_{id}.R_0 \overset{*}{\rightarrow} \rho.R_2$$

is a valid derivation and where, for a request R', we have:

$$\mu\eta R' = R_1 \quad \text{and} \quad \rho R' = R_2$$

Since $\rho \leq \mu\eta$, there exists a substitution ε such that $\varepsilon\rho = \mu\eta$. By lemma 3.3, ρ is an idempotent substitution. Let us first prove that $\mu\eta r(C) = r(C)$ and $\rho r(C) = r(C)$.
If x is a variable which occurs in the domain of $\mu\eta$, then either x occurs in the domain of μ, or in the domain of η. In the first case, then, by lemma 3.5, either x occurs in R_0, or in $\vartheta(d_0)$, and consequently, by hypothesis, x cannot occur in $r(C)$. In the second case, by definition of the renaming process used in a valid derivation, x cannot occur in $r(C)$, this settles $\mu\eta r(C) = r(C)$.
Since $\mu\eta r(C) = r(C)$, we get $\varepsilon\rho r(C) = r(C)$. Suppose $\rho r(C) \neq r(C)$, by lemma 1.4, there exists a variable x which occurs in $r(C)$ and in the terms from the range of $\varepsilon\rho = \mu\eta$. By condition H2, $x \in r(C)$ and then x occurs in the range of r. Two cases are now possible: either $x \in \text{Range}(\mu)$, or $x \in \text{Range}(\eta)$. In the first case, by lemma 3.5, x occurs either in ηR_0, or in $\vartheta(d_0)$ which is contradictory to the renaming process used. In the second

case, by hypothesis, x occurs in R_0 which induces the same contradiction. This settles $\rho r(C) = r(C)$.
By definition, θ satisfies:

$$\theta R_{1_{/n}} = \theta r(C^+)$$

Hence:

$$\theta\mu\eta R'_{/n} = \theta r(C^+)$$

Since $\mu\eta r(C) = r(C)$, we get:

$$\theta\mu\eta R'_{/n} = \theta\mu\eta r(C^+)$$

Therefore:

$$\theta\varepsilon\rho R'_{/n} = \theta\varepsilon\rho r(C^+)$$

Since $\rho r(C) = r(C)$, we get:

$$\theta\varepsilon\rho R'_{/n} = \theta\varepsilon r(C^+)$$

and now:

$$\theta\varepsilon R_{2_{/n}} = \theta\varepsilon r(C^+)$$

Consequently, $R_{2_{/n}}$ and $r(C^+)$ are unifiable and there exists a most general unifier σ of these two atoms such that $\sigma \leq \theta\varepsilon$. We can now easily prove that the following derivation:

$$s_{id}.R_0 \overset{*}{\rightarrow} \rho.R_2 \overset{n,r,C}{\rightarrow} \sigma\rho.\sigma R_2[n \leftarrow r(C^-)]$$

is valid and we also get:

$$\sigma R_2[n \leftarrow r(C^-)] = \sigma\rho R'[n \leftarrow r(C^-)]$$

and

$$\theta R_1[n \leftarrow r(C^-)] = \theta\mu\eta R'[n \leftarrow r(C^-)]$$

Furthermore, since $\sigma \leq \theta\varepsilon$, there exists a substitution ν such that $\nu\sigma = \theta\varepsilon$ and we get:

$$\theta\mu\eta = \theta\varepsilon\rho = \nu\sigma\rho$$

which settles $\sigma\rho \leq \theta\mu\eta$. $\diamond$

For the $R_1 = r_\emptyset$ case, there is:

Corollary 4.1 *If $s_{id}.\eta R \overset{*}{\rightarrow} \rho.r_\emptyset$ is a valid derivation, then for a substitution $\sigma \leq \rho\eta$, the derivation $s_{id}.\eta R \overset{*}{\rightarrow} \sigma.r_\emptyset$ is also valid.*

5 Switching lemma

At each resolution step of a derivation, an atom and a clause must be selected. The following well known lemma ensures that the non-determinism in the choice of atom does not matter: this is called "don't care" non-determinism (however, the choice of input clause is done by a "don't know" non-determinism).

Lemma 5.1 (Switching) *If during a valid derivation, two atoms A and B are successively selected, then they can also be selected in the reverse order and the derived states are the same up to renaming of variables.*

```
Lemma switching : (d:deriv)(t1:trans)(t2:trans)
(Deriv_ok (deriv_cons (deriv_cons d t1) t2)) ->
(le (plus (n_trans t1)
    (Length_r (body_c (c_trans t1)))) (n_trans t2))->
(Fst (state_init_d d))=([x:var](tv x)) ->
(Ex [t3:trans]
  (Ex [t4:trans]
    ((Deriv_ok (deriv_cons (deriv_cons d t3) t4))              /\
     (n_trans t3)=
     (S (minus (n_trans t2) (Length_r (body_c (c_trans t1)))))  /\
     (n_trans t4)=(n_trans t1)                                  /\
     (Ex [r:subst]
      ( (Snd (state_end_d (deriv_cons (deriv_cons d t1) t2)))=
        (subst_req r (Snd (state_end_d
                           (deriv_cons (deriv_cons d t3) t4)))) /\
       ((x:var)
        (IS_IN_LV x (var_req
            (Snd (state_end_d (deriv_cons (deriv_cons d t3) t4)))))
              -> (Ex [v:var] (r x)=(tv v)))))))).
```

Proof: The proof presented here follows exactly the formal proof developped in the proof assistant COQ. First, we proceed by induction on the two transitions, next we prove some "cut-lemmas" which allow to conclude.

INDUCTION ON TRANSITIONS

The derivation can be written: $d_c^l(d_c^l(d,t_1),t_2)$

$$\underbrace{s_{id}.R_0 \overset{*}{\to} \rho.R}_{d} \quad \overset{\substack{t_1:\\ n_1,r_1,C_1}}{\longrightarrow} \quad \theta\rho.\theta R_1 \quad \overset{\substack{t_2:\\ n_2,r_2,C_2}}{\longrightarrow} \quad \sigma\theta\rho.\sigma R_2$$

where $\theta R_1 = \theta R[n_1 \leftarrow r_1(C_1^-)]$ and $R_2 = (\theta R_1)[n_2 \leftarrow r_2(C_2^-)]$.
Furthermore, we suppose the first selected atom is before the second in the request R, so we have:

$$n_1 + \lg(C_1^-) \leq n_2$$

“CUT-LEMMAS”
In the switching lemma proof, we're going to use many times the same assertions. Instead of proving them every time, we first prove them using the `Cut` tactic. In this way, we can use them at any time during the proof without proving them again. Hypothesis of these cut-lemmas are the same as the switching lemma's hypothesis.

Cut-Lemma 5.1 $\rho r_2 C_2 = r_2 C_2$

PROOF: Let us prove that if z is a variable which occurs in the domain of ρ, then z doesn't occur in r_2C_2. If $z \in \text{Domain}(\rho)$, then, by lemma 3.5, either z occurs in R_0 or, in $\vartheta(d)$. In these two cases, by definition of the renaming process used in a valid derivation, z cannot occur in r_2C_2. This settles the claim. $\diamond$

Cut-Lemma 5.2 *$\sigma\theta$ is a unifier of $R_{/n_2-\lg(C_1^-)+1}$ and $r_2C_2^+$ and there exists a most general unifier μ of these two atoms such that $\mu \leq \sigma\theta$ (i.e. there exists a substitution ε such that $\varepsilon\mu = \sigma\theta$).*

PROOF: By definition (transition t_2), we have:

$$\sigma\theta R_{1_{/n_2}} = \sigma r_2(C_2^+)$$

By condition H3 on t_2, we have $\theta\rho r_2(C_2) = r_2(C_2)$. Hence:

$$\sigma\theta R_{1_{/n_2}} = \sigma\theta\rho r_2(C_2^+)$$

By cut-lemma 5.1, we get:

$$\sigma\theta R_{1_{/n_2}} = \sigma\theta r_2(C_2^+)$$

From $\theta R_1 = \theta R[n_1 \leftarrow r(C_1^-)]$ and $n_1 + \lg(C_1^-) \leq n_2$, it follows that:

$$\theta R_{1_{/n_2}} = \theta R_{/n_2-\lg(C_1^-)+1}$$

and now:

$$\sigma\theta R_{/n_2-\lg(C_1^-)+1} = \sigma\theta r_2 C_2^+$$

which allows to conclude. $\diamond$

Cut-Lemma 5.3 $\mu r_1 C_1 = r_1 C_1$

PROOF: Let us prove that if z is a variable which occurs in $r_1(C_1)$, then z does not occur in the domain of μ. By cut-lemma 5.2, μ is relevant to:

$$R_{/n_2-\lg(C_1^-)+1} \quad \text{and} \quad r_2(C_2^+)$$

Hence, z occurs in one of these two atoms. If z occurs in R, then, by condition H4 on t_1, z cannot occur in $r_1(C_1)$ which is contradictory. Else, if z occurs in $r_2(C_2^+)$, then z occurs in the range of r_2, and z cannot occur in $r_1(C_1)$. This settles the claim. $\diamond$

Cut-Lemma 5.4 *The substitution ε is a unifier of:*

$$r_1(C_1^+) \quad \text{and} \quad (\mu R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)])_{/n_1}$$

and there exists a most general unifier ν of these two atoms such that $\nu \leq \varepsilon$ (i.e. there exists a substitution ξ such that $\xi\nu = \varepsilon$).

PROOF: By definition (transition t_1), we have:

$$\theta R_{/n_1} = \theta r_1(C_1^+)$$

From $n_1 + \lg(C_1^-) \leq n_2$, it follows that:

$$R_{/n_1} = (R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)])_{/n_1}$$

and now:

$$\theta(R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)])_{/n_1} = \theta r_1(C_1^+)$$

Hence:

$$\sigma\theta(R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)])_{/n_1} = \sigma\theta r_1(C_1^+)$$

By cut-lemma 5.2, we have $\varepsilon\mu = \sigma\theta$, so:

$$\varepsilon\mu(R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)])_{/n_1} = \varepsilon\mu r_1(C_1^+)$$

By cut-lemma 5.3, we get:

$$\varepsilon\mu(R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)])_{/n_1} = \varepsilon r_1(C_1^+)$$

thus concluding the proof. ◇

EXISTENCE AND VALIDITY

By cut-lemma 5.2, we can build the transition t_3, seeing that μ is a most general unifier of $R_{/n_2 - \lg(C_1^-)+1}$ and $r_2(C_2^+)$. In the same manner, by cut-lemma 5.4, we can build the transition t_4, seeing that ν is a most general unifier of $(\mu R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)])_{/n_1}$ and $r_1(C_1^+)$:

$$t_3: \quad \rho.R \overset{n_2 - \lg(C_1^-) + 1, r_2, C_2}{\longrightarrow} \mu\rho.\mu R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)]$$

$$t_4: \quad \mu\rho.\underbrace{\mu R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)]}_{R'} \overset{n_1, r_1, C_1}{\longrightarrow} \nu\mu\rho.\nu R'[n_1 \leftarrow r_1(C_1^-)]$$

Now, let us prove that t_3 and t_4 are valid transitions (we omit here the immediate conditions H1 and H2).

Condition H3:

transition t_3: Immediate by cut-lemma 5.1.
transition t_4: By cut-lemma 5.3, we have $\mu r_1 C_1 = r_1 C_1$ and by condition H3 on t_1, we have $\rho r_1 C_1 = r_1 C_1$. Hence, we get $\mu\rho r_1 C_1 = \mu r_1 C_1 = r_1 C_1$.
Condition H4:
transition t_3: Let x be a variable which occurs in R. By lemma 3.4, either x occurs in R_0, or in $\vartheta(d)$. In these two cases, by definition of a valid transition, x cannot occur in $r_2 C_2$.
transition t_4: Let x be a variable occurring in $\mu R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)]$. Two cases are possible: either x occurs in the terms from the range of μ, or in $R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)]$. In these two cases, by cut-lemma 5.2 and by definition, either x occurs in R, or in $r_2 C_2$. In this first case, by lemma 3.4, either x occurs in R_0, or in $\vartheta(d)$. Therefore, in all of these cases, x cannot occur in $r_1 C_1$.
Condition H5:
transition t_3: Immediate by lemma 3.2 (seeing that, by hypothesis, the last transition of d is a valid transition).
transition t_4: Immediate by lemma 3.2 (seeing that, by the above, t_3 is a valid transition).

LINK BETWEEN THE TWO DERIVED STATES
Let us prove there exists a renaming r such that:

$$r(\sigma R_2) = \nu R'[n_1 \leftarrow r_1(C_1^-)]$$

For this, we first prove the following cut-lemmas.

Cut-Lemma 5.5 *$\nu\mu \leq \sigma\theta$.*

PROOF: By cut-lemmas 5.2 and 5.4, we have $\varepsilon\mu = \sigma\theta$ and $\xi\nu = \varepsilon$. Hence, we get $\xi\nu\mu = \sigma\theta$, which settles the claim. ⋄

Cut-Lemma 5.6 *$\theta \leq \nu\mu$ (i.e. there exists a substitution ζ such that $\zeta\theta = \nu\mu$).*

PROOF: By cut-lemma 5.4, we have:

$$\nu\mu R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)]_{/n_1} = \nu r_1(C_1^+)$$

Applying cut-lemma 5.3, we get:

$$\nu\mu R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)]_{/n_1} = \nu\mu r_1(C_1^+)$$

From $n_1 + \lg(C_1^-) \leq n_2$, it follows that:

$$R_{/n_1} = (R[n_2 - \lg(C_1^-) + 1 \leftarrow r_2(C_2^-)])_{/n_1}$$

and now:

$$\nu\mu R_{/n_1} = \nu\mu r_1(C_1^+)$$

Therefore, $\nu\mu$ is a unifier of $R_{/n_1}$ and $r_1(C_1^+)$. By hypothesis, θ is a most general unifier of these two atoms (transition t_1), which settles $\theta \leq \nu\mu$. ⋄

Cut-Lemma 5.7 $\theta r_2 C_2 = r_2 C_2$

PROOF: Let us prove that if x is a variable which occurs in $r_2(C_2)$, then x does not occur in the domain of θ. By hypothesis, θ is relevant to $R_{/n_1}$ and $r_1(C_1^+)$, hence, x occurs in one of these two atoms. If x occurs in $R_{/n_1}$ then, by lemma 3.4, x occurs either in R_0, or in $\vartheta(d)$ and then x cannot occur in the range of r_2 (*i.e.* cannot occur in $r_2(C_2)$). If x occurs in $r_1(C_1^+)$, then x cannot occur in $r_2(C_2)$. This concludes the proof. $\diamond$

Cut-Lemma 5.8 *$\sigma \leq \zeta$ (i.e. there exists a substitution δ such that $\delta\sigma = \zeta$).*

PROOF: Since t_3 is a valid transition, we have:

$$\mu R_{/n_2 - \lg(C_1^-)+1} = \mu r_2(C_2^+)$$

Hence:

$$\nu\mu R_{/n_2 - \lg(C_1^-)+1} = \nu\mu r_2(C_2^+)$$

Applying cut-lemma 5.6, we get:

$$\zeta\theta R_{/n_2 - \lg(C_1^-)+1} = \zeta\theta r_2(C_2^+)$$

and by cut-lemma 5.7 we get:

$$\zeta\theta R_{/n_2 - \lg(C_1^-)+1} = \zeta r_2(C_2^+)$$

Therefore, ζ is a unifier of $\theta R_{/n_2 - \lg(C_1^-)+1}$ and $r_2(C_2^+)$. From $n_1 + \lg(C_1^-) + 1 \leq n_2$, it follows that:

$$\theta R_{/n_2 - \lg(C_1^-)+1} = \theta R[n_1 \leftarrow r_1 C_1^-]_{n_2} = \theta R_{1/n_2}$$

By hypothesis (transition t_2), σ is a most general unifier of these two atoms, this leads to the conclusion that $\sigma \leq \zeta$. $\diamond$

Cut-Lemma 5.9 *$\sigma\theta \leq \nu\mu$.*

PROOF: By cut-lemma 5.6, we have $\nu\mu = \zeta\theta$. Hence, by cut-lemma 5.8, we get $\nu\mu = \delta\sigma\theta$ which settles the claim. $\diamond$

By cut-lemmas 5.5 and 5.9, we can establish that the two substitutions $\nu\mu$ and $\sigma\theta$ are variants. Let us now relate the two derived states. First, we have:

$$\begin{array}{ll}
\sigma R_2 & \\
= \sigma(\theta R_1)[n_2 \leftarrow r_2 C_2^-] & \\
= \sigma\theta R_1[n_2 \leftarrow \theta r_2 C_2^-] & \\
= \sigma\theta R_1[n_2 \leftarrow r_2 C_2^-] & \text{(by cut-lemma 5.7)} \\
= \sigma(\theta R[n_1 \leftarrow r_1 C_1^-])[n_2 \leftarrow r_2 C_2^-] & \\
= \sigma\theta(R[n_1 \leftarrow r_1 C_1^-])[n_2 \leftarrow \theta r_2 C_2^-] & \\
= \sigma\theta(R[n_1 \leftarrow r_1 C_1^-])[n_2 \leftarrow r_2 C_2^-] & \text{(by cut-lemma 5.7)}
\end{array}$$

Moreover:

$$\begin{aligned}
&\nu R'[n_1 \leftarrow r_1 C_1^-] \\
&= \nu(\mu R[n_2 + \lg(C_1^-) + 1 \leftarrow r_2 C_2^-])[n_1 \leftarrow r_1 C_1^-] \\
&= \nu\mu R[n_2 + \lg(C_1^-) + 1 \leftarrow r_2 C_2^-][n_1 \leftarrow \mu r_1 C_1^-] \\
&= \nu\mu R[n_2 + \lg(C_1^-) + 1 \leftarrow r_2 C_2^-][n_1 \leftarrow r_1 C_1^-] \qquad \text{(by cut-lemma 5.3)} \\
&= \nu\mu(R[n_1 \leftarrow r_1 C_1^-])[n_2 \leftarrow r_2 C_2^-]
\end{aligned}$$

Therefore, by lemma 1.3, there exists a renaming r, which satisfies the claim. This concludes the lifting lemma's proof. $\diamond$

Conclusion

SLD-Resolution defines a relation satisfying several important properties. The logical ones are soundness and completeness. Two others are particular to this form of computation: lifting and switching lemmas. In order to formalize SLD-Resolution, we constantly used the formal development coming from [18] and [11], notably the definition of terms and the unification theorem on these terms. Theories library of the proof assistant COQ have been also used, thus showing the interest that a proofs library be provided. The main difficulty of this development was to define explicitely the way clauses have to be renamed in a SLD-Derivation, and to find the additional conditions on this renaming we have to assume in the lifting lemma's proof. The switching lemma's proof is rather tedious but is not difficult to obtain.

Like SLD-Resolution for clauses, β-reduction defines a relation for λ-terms and it may be interesting to compare properties of these relations. Whereas β-reduction is confluent, SLD-Resolution just satisfies the "switching" property which is weaker than confluence. Furthermore, the renaming process used in a SLD-derivation can be viewed as an implicit α-conversion: the renaming is made explicit and allows a full formalization of a SLD-derivation along the lines of the the calculus of explicit substitutions [1] for β-reduction.

These two proofs are made from objects which are syntactically defined. Interpretations of these objects are now under consideration, in order to prove the soundness and completeness of SLD-Resolution in the calculus of inductive constructions.

References

[1] M. Abadi, L. Cardelli, P.L. Curien, and J.J. Levy. Explicit substitutions. *Journal of functional programming*, 1(4):375–416, 1991.

[2] K.R. Apt. Logic programming. In *Handbook of theoretical computer science*, pages 493–574. Ed. J. van Leeuwen, 1990.

[3] K.R. Apt and M.H. van Emden. Contributions to the theory of logic programming. *J. ACM.*, 29(3):841–862, july 1982.

[4] H.P. Barendregt. Lambda calculi with types. In *Handbook of Logic in Computer Science*, volume II. Oxford University Press, 1992.

[5] A. Colmerauer, H. Kanoui, and M. Van Caneghem. Prolog, bases théoriques et développements actuels. *TSI*, 2(4), 1983.

[6] T. Coquand. An introduction to type theory. In *Logique et Informatique: une introduction*, pages 117–135. INRIA, b. courcelle edition, 1991.

[7] C. Cornes, J. Courant, J.C. Filliâtre, G. Huet, P. Manouryand, C. Paulin-Mohring, C. Munoz, C. Murthy, C. Parent, A. Saibi, and B. Werner. *The Coq Proof Assistant Reference Manual Version 5.10.* INRIA-CNRS-ENS, 1995.

[8] K. Doets. *From logic to logic programming.* Foundations of Computing Series. MIT Press, 1994.

[9] J.Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types.* Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[10] C.J. Hogger. *Essentials of Logic Programming.* Clarendon Press Oxford, 1990.

[11] M. Jaume. Unification des termes du premier ordre dans le calcul des constructions inductives. Research Report 96-58, CERMICS, April 1996.

[12] R. Lalement. *Computation as Logic.* Prentice Hall International Series in Computer Science, 1993.

[13] R. Lassaigne and M. de Rougemont. *Logique et fondements de l'informatique.* Hermès, 1993.

[14] J.W. Lloyd. *Foundations of logic programming.* Springer Verlag , second edition, 1987.

[15] J. Maluszynski and U. Nilsson. *Logic programming and Prolog.* second edition, J.Wiley and sons, 1995.

[16] C. Paulin-Mohring. Inductive definitions in the system coq, rules and properties. Research Report 92-49, LIP, ENS-Lyon, December 1992.

[17] J. Robinson. A machine oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, jan. 1969.

[18] J. Rouyer. Développement de l'algorithme d'unification dans le calcul des constructions avec types inductifs. Research Report 1795, INRIA, Lorraine, Novembre 1992.

[19] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *J. ACM*, 23(4):733–742, october 1976.