A formal framework for synthesis and verification of logic programs*

Alessandro Avellone Mauro Ferrari Camillo Fiorentini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico 39, 20135 Milano–Italy
{avellone,ferram,fiorenti}@dsi.unimi.it

Abstract. In this paper we will present a formal framework, based on the notion of *extraction calculus*, which has been successfully applied to define procedures for extracting information from constructive proofs. Here we will apply such a mechanism to give a proof-theoretic account of SLD-derivations. We show how proofs of suitable constructive systems can be used in the context of deductive synthesis of logic programs, and we state a link between constructive and deductive program synthesis.

1 Introduction

It is well known that formal proofs can be used for program synthesis and program verification, and this essentially depends on the availability of an *information extraction mechanism* allowing to capture in an *uniform way* the implicit algorithmic content of a proof. In this paper we will present a formal framework, based on the notion of *extraction calculus*, which has been devised by the authors [4–7] and successfully applied to define procedures for extracting information from proofs of a great variety of logical systems.

Here we will apply such a mechanism to give a proof-theoretic account of SLD-derivations. We show how proofs of suitable constructive systems can be used in the context of deductive synthesis of logic programs [3], and we state a link between constructive and deductive program synthesis. In particular we will prove that the logic program consisting of the Horn Axioms occurring in a natural deduction proof of a formula $\exists x G(x) \vee \neg\exists x G(x)$ (G atomic), allows us to correctly compute the goal. This result can be used both to define a synthesis method and to study properties of the deductive synthesis process. Finally we will discuss a simple example showing in which sense our framework allows us to treat modularity and derivation (verification) of open programs.

* Extended Abstracts of LOPSTR 2000, Tenth International Workshop on Logic-based Program Synthesis and Transformation, 24-28 July 2000, London, UK. Technical Report Report Series, Department of Computer Science, University of Manchester, ISSN 1361-6161. Report number UMCS-00-6-1. http://www.cs.man.ac.uk/cstechrep/titles00.html

2 The general framework

In this section we will provide a short presentation of our mechanism to extract information from proofs; for a complete discussion see [5–7]. Our extraction mechanism is based on an abstract definition of the notions of proof and calculus allowing to treat extraction from Gentzen, Tableau or Hilbert style calculi.

Here we consider first-order languages $\mathcal{L}_\Sigma$ over a signature Σ, the set of logical symbols is $\{\wedge, \vee, \rightarrow, \neg, \forall, \exists\}$. A (single-conclusion) *sequent* is an expression $\Gamma \vdash A$, where A is a formula (wff for short) and Γ is a finite set of wff's. A *proof* on $\mathcal{L}_\Sigma$ is any finite object π such that: (1) The (finite) set of wff's of $\mathcal{L}_\Sigma$ occurring in π is uniquely determined and nonempty; (2) π proves a sequent $\Gamma \vdash A$, where Γ (possibly empty) is the set of *assumptions* of π, while A is the *consequence* of π. The notation $\pi : \Gamma \vdash A$ means that $\Gamma \vdash A$ is the sequent proved by π, and $\mathrm{dg}(\pi)$ denotes the *degree of* π, that is the maximum among the degrees of the wff's occurring in π, where the degree of a wff is defined as usual.

A *calculus* on $\mathcal{L}_\Sigma$ is a pair $(\mathbf{C}, [\cdot])$, where $\mathbf{C}$ is a recursive set of proofs on the language $\mathcal{L}_\Sigma$ and $[\cdot]$ is a recursive map associating with every proof of the calculus the set of its *relevant* subproofs. We require $[\cdot]$ to satisfy the following natural conditions: (1) $\pi \in [\pi]$; (2) for every $\pi' \in [\pi]$, $[\pi'] \subseteq [\pi]$; (3) for every $\pi' \in [\pi]$, $\mathrm{dg}(\pi') \leq \mathrm{dg}(\pi)$. We remark that any usual single conclusion inference system is a calculus according to our definition. In particular the natural deduction calculi we will use in this paper meet this characterization.

Given $\Pi \subseteq \mathbf{C}$, $\mathrm{Seq}(\Pi) = \{\Gamma \vdash A \mid \pi : \Gamma \vdash A \in \Pi\}$ is the set of the *sequents proved in* Π; $\mathrm{Theo}(\Pi) = \{A \mid \ \vdash A \in \mathrm{Seq}(\Pi)\}$ is the set of *theorems proved in* Π, and $[\Pi] = \{\pi' \mid \text{ there exists } \pi \in \Pi \text{ such that } \pi' \in [\pi]\}$ is the *closure under subproofs* of Π in the calculus $\mathbf{C}$. Now, given a set $\mathcal{R}$ of inference rules of the kind

$$\frac{\Gamma_1 \vdash A_1 \quad \ldots \quad \Gamma_n \vdash A_n}{\Delta \vdash B}\,\mathrm{R}$$

we call $\mathcal{R}$ a set of *extraction rules for* $\mathbf{C}$ (*e-rules* for short) if there exists a function $\phi : \mathbf{N} \rightarrow \mathbf{N}$ such that:

1. Every $R \in \mathcal{R}$ can be uniformly simulated in $\mathbf{C}$ w.r.t. ϕ. That is, for every $\pi_1,\ldots,\pi_n$ in $\mathbf{C}$ proving the premises of the rule R, there exists a proof π of the consequence of R such that $\mathrm{dg}(\pi) \leq \max\{\phi(\mathrm{dg}(\pi_1)), \ldots, \phi(\mathrm{dg}(\pi_n))\}$.
2. R is *non-increasing*; that is, the degree of every wff occurring in $\Delta \vdash B$ is bounded by the degree of a wff occurring in the premises.

Condition (1) says that every $R \in \mathcal{R}$ must be an admissible rule for $\mathbf{C}$, and must be simulated in a uniform way (w.r.t. the degrees) in the calculus $\mathbf{C}$. Examples of e-rules for the natural deduction calculus for Intuitionistic Logic are the usual *cut rule* and the *substitution rule*.

Now, given a set $\mathcal{R}$ of e-rules for $\mathbf{C}$ and $\Pi \subseteq \mathbf{C}$, the *extraction calculus for* Π, denoted by $\mathbb{D}(\mathcal{R}, [\Pi])$, is defined as follows:

1. If $\Gamma \vdash A \in \mathrm{Seq}([\Pi])$, then $\tau \equiv \Gamma \vdash A$ is a proof-tree of $\mathbb{D}(\mathcal{R}, [\Pi])$.

2. If $\tau_1 : \Gamma_1 \vdash A_1, \ldots, \tau_n : \Gamma_n \vdash A_n$ are proof-trees of $\mathbb{ID}(\mathcal{R}, [\Pi])$ and

$$\frac{\Gamma_1 \vdash A_1 \quad \ldots \quad \Gamma_n \vdash A_n}{\Delta \vdash B}\,\mathrm{R}$$

is a rule of $\mathcal{R}$, then the proof-tree

$$\tau \equiv \frac{\tau_1 : \Gamma_1 \vdash A_1 \ \ldots \ \tau_n : \Gamma_n \vdash A_n}{\Delta \vdash B}\,\mathrm{R}$$

with root $\Delta \vdash B$ belongs to $\mathbb{ID}(\mathcal{R}, [\Pi])$.

A set of proofs Π is *constructive* if it meets the *disjunction property* ($A \vee B \in \text{Theo}(\Pi)$ implies $A \in \text{Theo}(\Pi)$ or $B \in \text{Theo}(\Pi)$), and the *explicit definability property* ($\exists x A(x) \in \text{Theo}(\Pi)$, implies $A(t/x) \in \text{Theo}(\Pi)$ for some closed term t of the language). A calculus $\mathbf{C}$ is *uniformly constructive* if there exists a set of e-rules $\mathcal{R}$ for $\mathbf{C}$ such that, for every $\Pi \subseteq \mathbf{C}$, $\text{Theo}(\mathbb{ID}(\mathcal{R}, [\Pi]))$ is constructive.

The properties of a uniformly constructive calculus $\mathbf{C}$ assure that, if $\pi : \vdash \exists x A(x) \in \mathbf{C}$, then we can determine a closed term t such that $A(t)$ is provable in $\mathbf{C}$ exploiting the information contained in the proof π by means of the calculus $\mathbb{ID}(\mathcal{R}, [\pi])$. Moreover, such information can be searched in the calculus $\mathbb{ID}(\mathcal{R}, [\pi])$ by means of an enumerative procedure only involving wff's of bounded logical complexity, the bound depending on π. This allows us to define procedures for extracting information from constructive proofs. These procedures are of interest in the fields of program synthesis and formal verification [1, 6].

In [6, 7] a wide family of constructive systems (involving theories formalizing Abstract Data Types) is shown to be uniformly constructive, while in [7] an example of a constructive but not uniformly constructive formal system is provided. However, the proofs of uniform constructivity of all the systems discussed in [6, 7] rely on extraction rules which are not suitable to get efficient proof search in $\mathbb{ID}(\mathcal{R}, [\Pi])$; in particular, all these results involve the cut rule (for a discussion on the complexity of these extraction procedure see [2]). On the other hand, in [5] it is proved that $\mathbb{ID}(\mathcal{R}, [\Pi])$ can be characterized as a goal-oriented calculus for suitable proofs involving Hereditary Harrop Formulae (see [10]). In Section 3, we will show that the SLD rule is sufficient to decide the explicit definability property and the disjunction property for suitable constructive proofs only involving *program clauses* as assumptions; this immediately yields a logic program synthesis method.

2.1 Natural deduction calculi for program extraction

Let us denote with $\mathcal{ND}$ the usual natural deduction calculus for Minimal Logic [12] with the following rules for negation[1]:

$$\frac{\begin{array}{c}\Gamma\\ \vdots\,\pi_1\\ A\end{array} \qquad \begin{array}{c}\Gamma\\ \vdots\,\pi_2\\ \neg A\end{array}}{B}\,\mathit{Contr} \qquad\qquad \frac{\begin{array}{c}[H_1, \ldots, H_n, A]\\ \vdots\,\pi\\ B \wedge \neg B\end{array}}{H_1 \wedge \ldots \wedge H_n \rightarrow \neg A}\,I\neg$$

with $H_1, \ldots, H_n$ atomic or negated wff's. $H_1, \ldots, H_n, A$ are the only undischarged assumptions of π.

[1] We indicate within square brackets the assumptions discharged by the rule application, see [12].

We associate with any proof of $\mathcal{ND}$ a sequent $\Gamma \vdash A$ where: Γ is the set of the assumptions on which π depends, and A is the end-formula of π. A wff A is *provable in* $\mathcal{ND}$ if this calculus contains a proof π of the sequent $\vdash A$ (i.e., π has no undischarged assumptions). Given a theory $\mathbf{T}$, that is a recursively enumerable set of wff's, we denote with $\mathcal{ND}(\mathbf{T})$ the calculus obtained by adding to $\mathcal{ND}$ the rule

$$\frac{}{A}\,{\scriptstyle A\in\mathbf{T}}$$

Given a theory $\mathbf{T}$ on $\mathcal{L}_\Sigma$, we say that $\mathbf{T}$ *admits a cover set* if there exists a finite set $\mathcal{C}$ of terms of $\mathcal{L}_\Sigma$ such that:

1. No term of $\mathcal{C}$ is a variable;
2. For every closed term t of $\mathcal{L}_\Sigma$, there is a term $t' \in \mathcal{C}$ such that $\mathcal{ND}(\mathbf{T})$ proves $t = \theta t'$, for some closed substitution θ (that is, a substitution associating a closed term with each variable).

Let $\mathcal{C} = \{c_1, \ldots, c_n, t_1(x_1^1, \ldots, x_{w_1}^1), \ldots, t_m(x_1^m, \ldots, x_{w_m}^m)\}$ be a cover set for $\mathbf{T}$, where the c_i are constant symbols of $\mathcal{L}_\Sigma$ and the $t_j(x_1^j, \ldots, x_{w_j}^j)$ are terms containing the variables $x_1^j, \ldots, x_{w_j}^j$ $(w_j \geq 1)$. We introduce the following *Induction Rule* related to $\mathcal{C}$:

$$\frac{\ldots\ \begin{array}{c}\Gamma\\ \vdots\,\pi_i\\ A(c_i)\end{array}\ \ldots\ \begin{array}{c}\Gamma,[\Delta_j]\\ \vdots\,\pi_j^1\\ A(t_j(x_1^j,\ldots,x_{w_j}^j))\end{array}\ \ \begin{array}{c}\Gamma\\ \vdots\,\pi_j^2\\ \exists p_1^j\ldots\exists p_{w_j}^j R_j(p_1^j,\ldots,p_{w_j}^j,t_j(x_1^j,\ldots,x_{w_j}^j))\end{array}\ \ldots}{A(x)}\,\mathrm{Ind}^{\mathcal{C}}$$

where, in the premise of the rule, $i = 1, \ldots, n$ and $j = 1 \ldots, m$ and $\Delta_j = \{A(p_1^j), \ldots, A(p_{w_j}^j), R_j(p_1^j, \ldots, p_{w_j}^j, t_j(x_1^j, \ldots, x_{w_j}^j))\}$. The variables p_l^j occurring in the premises of the proofs π_j^1 are called *proper parameters* of the rule application and must not occur free in $\Gamma, A(x)$. The wff's $A(p_1^j), \ldots, A(p_{w_j}^j)$ in the premises of the proofs π_j^1 are the induction hypotheses of the rule, and are discharged by the induction rule application. On the other hand, the wff's $A(c_i)$ in the consequence of the proofs π_i are the base cases of the induction rule. As for the wff $\exists p_1^j \ldots \exists p_{w_j}^j R_j(p_1^j, \ldots, p_{w_j}^j, t_j(x_1^j, \ldots, x_{w_j}^j))$, it allows us to guarantee that the provability of $A(t_j(x_1^j, \ldots, x_{w_j}^j))$ can be reduced to the provability of wff's $A(s_1^j), \ldots, A(s_{w_j}^j)$. In the next section we will give the conditions on R needed to guarantee the correctness of $\mathrm{Ind}^{\mathcal{C}}$.

We point out that the above rule schema can be generalized to the case of wff's $A(x_1, \ldots, x_m)$ with m free variables; in the example of Section 3.1 we will provide an application of the above schema to a wff with two free variables.

We will denote with $\mathcal{ND}^+$ the calculus obtained by adding the rule $\mathrm{Ind}^{\mathcal{C}}$ to $\mathcal{ND}$. The notions of *proof*, *provability*, *subproof* and *depth of a proof* π (denoted by $\mathrm{depth}(\pi)$) in a natural deduction calculus are defined in the usual way ([12]).

3 Constructive proofs in deductive logic program synthesis

In this section we show how proofs of suitable constructive systems $\mathcal{ND}(\mathbf{T})$ can be used in the context of deductive synthesis of logic programs [3], stating a link between constructive and deductive program synthesis. Here we adopt the usual proof-theoretic characterization of $\textsc{sld}^+$-derivations [10, 11]. A *Horn clause* is any wff of the kind $A_1 \wedge \ldots \wedge A_n \rightarrow B$, where $n \geq 0$ and $A_1, \ldots, A_n, B$ are atomic wff's. A *Horn Program* is a finite set of Horn clauses and a *goal* is any wff of the kind $\exists x G(x)$, where G is an atomic wff. We call *neg-axiom* any wff of the kind $K_1 \wedge \ldots \wedge K_n \rightarrow \neg H$ with $K_1, \ldots, K_n$ atomic or negated wff's.

Given a set of proofs Π, we call *Extended Horn Program of* Π the set $\mathcal{P}^+_\Pi$ of Horn clauses and neg-axioms occurring in $\mathrm{Theo}(\Pi)$. $\textsc{sld}^+$ is the rule:

$$\frac{\vdash \theta H_1 \quad \ldots \quad \vdash \theta H_n \qquad \overline{\vdash H_1 \wedge \ldots \wedge H_n \rightarrow K}}{\vdash \theta K}\,\textsc{sld}^+$$

where $H_1 \wedge \ldots \wedge H_n \rightarrow K$ is a Horn clause or a neg-axiom, and θ is an arbitrary substitution (if $n = 0$ we obtain, as a special case, the substitution rule for atomic axioms). We do not require θ to be a *most general unifier*, but this does not affect our treatment because any SLD proof using arbitrary substitutions is an instance of a SLD proof using mgu's [11]. Besides $\textsc{sld}^+$, we will also use the following rules to treat identity:

$$\frac{}{\vdash t = t}\,\textsc{id}_1 \qquad \frac{\vdash t = t' \quad \vdash A(t)}{\vdash A(t')}\,\textsc{id}_2$$

where t and t' are terms. We remark that the above rules are e-rules according to the definition of Section 2.

Given a set of proofs Π of $\mathcal{ND}^+(\mathbf{T})$ we simply denote with $\mathbb{D}(\Pi)$ the extraction calculus $\mathbb{D}(\{\textsc{id}_1, \textsc{id}_2, \textsc{sld}^+, \textsc{cov}\}, [\Pi])$, where $\textsc{cov}$ is a set of e-rules allowing to prove equalities of the kind $t = \theta t'$ with t' belonging to the cover set of $\mathbf{T}$, any time that $\mathcal{ND}(\mathbf{T})$ proves $t = \theta t'$[2]. Now, we will study the proof-theoretic properties of $\mathbb{D}(\Pi)$ and of the associated program $\mathcal{P}^+_\Pi$ using the following notion of evaluation.

Definition 1 (Evaluation). *Given a set of proofs Π (on $\mathcal{L}_\Sigma$) and a wff A of $\mathcal{L}_\Sigma$, A is* evaluated *in Π (in symbols $\Pi \rhd A$) if, for any closed substitution θ, one of the following inductive conditions holds:*

1. *θA is an atomic or a negated wff and $\vdash \theta A \in \mathrm{Seq}(\Pi)$;*
2. *$\theta A \equiv B \wedge C$ and $\Pi \rhd B$ and $\Pi \rhd C$;*
3. *$\theta A \equiv B \vee C$ and either $\Pi \rhd B$ or $\Pi \rhd C$;*
4. *$\theta A \equiv B \rightarrow C$ and, if $\Pi \rhd B$ then $\Pi \rhd C$;*
5. *$\theta A \equiv \forall x B(x)$ and, for every closed term t of $\mathcal{L}_\Sigma$, $\Pi \rhd B(t)$;*

[2] As an example, in the case of Peano Arithmetic $\textsc{cov}$ contains the axioms for sum and product.

6. $\theta A \equiv \exists x B(x)$ *and* $\Pi \triangleright B(t)$ *for some closed term* t *of* $\mathcal{L}_\Sigma$.

A set Γ of wff's is evaluated in Π ($\Pi \triangleright \Gamma$) if $\Pi \triangleright A$ holds for every $A \in \Gamma$. The following fact can be proved:

Theorem 1. *Let* Π *be any set of proofs of* $\mathcal{ND}^+(\mathbf{T})$ *and let* H *be an Horn clause or a neg-axiom. If there exists a proof* $\pi : \vdash H$ *in the closure under substitution of* $[\Pi]$*, then* $\mathbb{D}(\Pi) \triangleright H$.

This immediately guarantees that any wff in the Extended Horn Program $\mathcal{P}^+_\Pi$ of a set of proofs $\Pi \subseteq \mathcal{ND}^+(\mathbf{T})$ is evaluated in $\mathbb{D}(\Pi)$.

Now, we give the condition needed to guarantee the correctness of the $\mathrm{Ind}^{\mathcal{C}}$ rule in a theory $\mathbf{T}$. Given a set of proofs of $\Pi \subseteq \mathcal{ND}^+(\mathbf{T})$, $\mathrm{Ind}^{\mathcal{C}}$ *is applied in the adequate context in* Π if there exist a set of e-rules $\mathcal{R}$ a classical model $\mathfrak{M}$ of $\mathbf{T}$ and a binary relation $\prec$ in $\mathfrak{M}$ such that, for every wff $R(x_1, \ldots, x_n, y)$ occurring in the $\mathrm{Ind}^{\mathcal{C}}$ rule:

A1 If $\mathbb{D}(\Pi) \triangleright R(s_1, \ldots, s_n, t)$, with $s_1, \ldots, s_n, t$ closed terms, then $\mathfrak{M} \models s_1 \prec t$,…,$\mathfrak{M} \models s_n \prec t$;
A2 The $\prec$ relation is well founded in $\mathfrak{M}$.

Now, the following fact can be proved:

Theorem 2. *Let* Π *be a set of proofs of* $\mathcal{ND}^+(\mathbf{T})$ *where* $\mathrm{Ind}^{\mathcal{C}}$ *is applied in the adequate context and* $\mathbb{D}(\Pi) \triangleright \mathbf{T}$. *For every* $\pi : \Gamma \vdash A$ *belonging to the closure under substitution of* $[\Pi]$*, if* $\mathbb{D}(\Pi) \triangleright \Gamma$ *then* $\mathbb{D}(\Pi) \triangleright A$.

This theorem has interesting applications in the context of deductive synthesis of logic programs. Here we will consider two significant examples.

The first application is to the halting of a synthesis process (see [9]) $\mathcal{P}_0 \subseteq \mathcal{P}_1 \subseteq \ldots \subseteq \mathcal{P}_k \subseteq \ldots$ in the context of an axiomatization of a specification framework $\mathcal{F}$ [8]. Let $\mathcal{D}_k$, for each k, be the specification of the predicates of $\mathcal{P}_k$, and let π_k be a proof of $\mathcal{ND}^+(\mathbf{T})$ where $\mathbf{T}$ consists of all the Horn clauses of $\mathcal{P}_k$ and neg-axioms classically provable in $\mathcal{F} \cup \mathcal{D}_k$. Clearly, any wff θA provable in $\mathbb{D}([\pi_k])$ is a computed answer for $\mathcal{P}_k \cup \{\leftarrow A\}$. In contrast, if $\mathbb{D}([\pi_k])$ proves a closed instance of $\theta\neg\exists z G(x,z)$ then $\leftarrow \theta G(x,z)$ cannot have any answer by the partial correctness of $\mathcal{P}_k$. Hence, if $\pi : \vdash \forall x(\exists z G(x,z) \vee \neg\exists z G(x,z))$, then $\mathcal{P}_k$ will solve all the closed goals $\theta\exists z G(x,z)$ provable in $\mathcal{ND}^+(\mathbf{T})$. This means that $\mathcal{P}_k$ is correct with respect to its specification $\mathcal{D}_k$: whenever $\theta\exists z G(x,z)$ has an answer in $\mathcal{F} \cup \mathcal{D}_k$ the closed answer is computed by $\mathcal{P}_k$. Therefore, we can halt the synthesis process. Moreover, if the above condition is not satisfied we can get heuristic information about the way to enrich the specification.

The second application is in the context of modular program synthesis. Indeed Theorem 1 applies to sequents of the form $\Gamma \vdash A$. Now, let $\mathcal{P}_k$ be an *open program* ([8]) in a program synthesis process $\mathcal{P}_0 \subseteq \mathcal{P}_1 \subseteq \ldots \subseteq \mathcal{P}_k \subseteq \ldots$, that contains in the body of its clauses open predicates, to be computed by other programs. By $\Gamma \vdash G$ we can represent the following property of $\mathcal{P}_k$: $\mathcal{P}_k$ solves G when it is compound with programs $\mathcal{Q}$'s that solve Γ. This allows us to guarantee the compositionality of the synthesis process.

3.1 An example

Let $\mathcal{P}_n$ be the program we obtain after n steps of the synthesis process:

$$\begin{array}{ll}\mathcal{P}_n: & (P1)\ gcd(S(x),0,S(x)) \leftarrow \qquad (P2)\ gcd(0,S(y),S(y)) \leftarrow \\ & (P3)\ gcd(S(x),S(y),z) \leftarrow sum(p,S(x),S(y)), gcd(S(x),p,z) \\ & (P4)\ gcd(S(x),S(y),z) \leftarrow sum(p,S(y),S(x)), gcd(p,S(y),z)\end{array}$$

whose predicates are introduced in $\mathcal{D}_n$ by extensional definitions of Peano Arithmetic as follows $sum(x,y,z) \leftrightarrow z = x+y$ and $gcd(x,y,z) \leftrightarrow \exists w_1 \exists w_2 (z * w_1 = x \wedge z * w_2 = y) \wedge \forall v(\exists v_1 \exists v_2 (v * v_1 = x \wedge v * v_2 = y) \rightarrow v \leq z)$. We sketch the proof π of the wff $\forall x \forall y (\exists z\ gcd(x,y,z) \vee \neg \exists z\ gcd(x,y,z))$ which guarantees the computability of the gcd in the calculus $\mathcal{ND}^{+}(\mathbf{T})$, where $\mathbf{T} = \mathcal{P}_n \cup \mathcal{D}_n$.

$$A(x,y) \equiv \neg(x=0 \wedge y=0) \rightarrow \exists z gcd(x,y,z)$$
$$R(z_1,z_2,x,y) \equiv (sum(z_1,y,x) \wedge z_2 = y) \vee (sum(z_2,x,y) \wedge (z_1 = x))$$

$\boxed{\pi \equiv}$

$$\dfrac{\begin{matrix}(x=0\wedge y=0)\vee\\ \neg(x=0\wedge y=0)\end{matrix} \qquad \dfrac{\begin{matrix}[x=0\wedge y=0]\\ \vdots\ \pi'\\ \neg\exists z\ gcd(x,y,z)\end{matrix}}{\exists z\ gcd(x,y,z)\vee\neg\exists z\ gcd(x,y,z)}\vee I \qquad \dfrac{\dfrac{\begin{matrix}\vdots\ \pi''\\ A(x,y)\end{matrix}\quad [\neg(x=0\wedge y=0)]}{\exists z\ gcd(x,y,z)}\rightarrow E}{\exists z\ gcd(x,y,z)\vee\neg\exists z\ gcd(x,y,z)}\vee I}{\exists z\ gcd(x,y,z)\vee\neg\exists z\ gcd(x,y,z)}\vee E$$

$\boxed{\pi'' \equiv}$

$$\dfrac{\begin{matrix}\\ \\ \vdots\ \pi_1\\ A(0,0)\end{matrix} \quad \begin{matrix}[A(z_1,z_2)],\\ [(S(z_1)=S(x)\wedge z_2=0)]\\ \vdots\ \pi_2^1\\ A(S(x),0)\end{matrix} \quad \begin{matrix}\vdots\ \pi_2^2\\ \exists z_1\exists z_2(S(z_1)=S(x)\wedge z_2=0)\end{matrix} \quad \ldots \quad \begin{matrix}[A(z_1,z_2)],\\ [R(z_1,z_2,S(x),S(y))]\\ \vdots\ \pi_4^1\\ A(S(x),S(y))\end{matrix} \quad \begin{matrix}\vdots\ \pi_4^2\\ \exists z_1\exists z_2 R(z_1,z_2,S(x),S(y))\end{matrix}}{A(x,y)}\mathrm{Ind}^{\mathcal{C}}{}_R$$

Proof π'' above ends with an application of the generalization of the $\mathrm{Ind}^{\mathcal{C}}$ rule to the two variables wff $A(x,y)$, where $\mathcal{C} = \{0, S(x)\}$ and $R(z_1,z_2,x,y)$ is $(sum(z_1,y,x) \wedge z_2 = y) \vee (sum(z_2,x,y) \wedge (z_1 = x))$. In π'' we have omitted the proofs π_3^1 and π_3^2 regarding the case $A(0,S(x))$ which are symmetrical to proofs π_2^1 and π_2^2. It is easy to check that the occurrence of $\mathrm{Ind}^{\mathcal{C}}$ in π'' as applied in the adequate context; the relation $\prec$ involved in Conditions (A1) and (A2) is the usual "less than" relation between pairs of natural numbers.

Proofs π', π_1, π_2^1 (using the axiom P1), π_2^2, π_3^1 (using axiom P2), π_3^2 and π_4^2 are trivial. Here below we outline the proof π_4^1.

$$\dfrac{R(z_1,z_2,S(x),S(y)) \qquad \dfrac{\dfrac{\dfrac{\begin{matrix}[sum(z_1,S(y),S(x))\wedge S(y)=z_2]\\ \vdots\\ \neg(z_1=0\wedge z_2=0)\end{matrix}\quad A(z_1,z_2)}{\exists z gcd(z_1,z_2,z)}\rightarrow E \qquad \dfrac{\begin{matrix}P4,\\ [gcd(z_1,z_2,q)],\\ [sum(z_1,S(y),S(x))\wedge S(y)=z_2]\\ \vdots\\ gcd(S(x),S(y),q)\end{matrix}}{\exists z gcd(S(x),S(y),z)}\exists I}{\exists z gcd(S(x),S(y),z)}\exists E}{A(S(x),S(y))}\rightarrow I \qquad \begin{matrix}P3,\\ [A(z_1,z_2)],\\ [sum(z_2,S(x),S(y))\wedge z_1=S(x)]\\ \vdots\\ A(S(x),S(y))\end{matrix}}{A(S(x),S(y))}\vee E$$

Since all the undischarged assumptions of the above proofs belong to $\mathbf{T}$, we can deduce that $\mathcal{P}_n$ is a correct program with respect to the specification $\mathcal{D}_n$ having a program to compute the *sum* predicate.

4 Conclusions

In this paper, we have proposed the formal framework of extraction calculi to study the proof-theory of SLD derivations and we applied it to show that, under some assumptions, it is possible to assure that the Horn Axioms occurring in a natural deduction proof of a goal allow to compute the goal. Finally, we have briefly discussed how to use the formal framework to deduce properties about the synthesis process. Now, our aim is to study in this framework the cases of "negation as failure" and "disjunctive logic programming".

References

1. M. Benini. *Verification and Analysis of Programs in a Constructive Environment.* PhD thesis, Dipartimento di Scienze dell'Informazione, Università di Milano, 1999.
2. S. Buss and G. Mints. The complexity of the disjunction and existential properties in intuitionistic logic. *Annals of Pure and Applied Logic*, 99(3):93–104, 1999.
3. Y. Deville and K. Lau. Logic program synthesis. *Journal of Logic Programming*, 19(20):321–350, 1994.
4. M. Ferrari. *Strongly Constructive Formal Systems.* PhD thesis, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Italy, 1997. Available at `http://homes.dsi.unimi.it/~ferram`.
5. M. Ferrari, C. Fiorentini, and P. Miglioli. Goal oriented information extraction in uniformly constructive calculi. In *Proceedings of WAIT'99: Workshop Argentino de Informática Teórica*, pages 51–63, 1999.
6. M. Ferrari, C. Fiorentini, and P. Miglioli. Extracting information from intermediate T-systems. Technical Report 252-00, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, 2000. Presented at *IMLA99: Intuitionistic Modal Logics and Application*, Trento, July 6, 1999. Available at `http://homes.dsi.unimi.it/~ferram`.
7. M. Ferrari, P. Miglioli, and M. Ornaghi. On uniformly constructive and semiconstructive formal systems. Submitted to *Annals of Pure and Applied Logic*, 1999.
8. K.-K. Lau and M. Ornaghi. On specification frameworks and deductive synthesis of logic programs. In Logic Program Synthesis and Transformation. *Proceedings of LOPSTR'94*. Springer-Verlag, 1994.
9. Kung-Kiu Lau, Mario Ornaghi, and Sten-Åke Tärnlund. The halting problem for deductive synthesis of logic programs. In P. Van Hentenryck, editor, *Logic Programming - Proceedings of the Eleventh International Conference on Logic Programming*, pages 665–683. The MIT Press, 1994.
10. D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.
11. A. Momigliano and M. Ornaghi. Regular search spaces as a foundation of logic programming. In R. Dyckhoff, editor, *Proceedings of the 4th International Workshop on Extensions of Logic Programming*, volume 798 of *LNAI*, pages 222–254, Berlin, 1994. Springer.
12. D. Prawitz. *Natural Deduction.* Almquist and Winksell, 1965.