Final Draft of a paper that appeared in LICS 1993
(Eighth Annual IEEE Symposium on Logic in Computer Science, Montreal, Canada, June 19-23, 1993, IEEE Computer Society Press, Los Alamitos, 1993, pp. 222-232)

Rules of Definitional Reflection

Peter Schroeder-Heister
Universität Tübingen
Wilhelm-Schickard-Institut für Informatik
Sand 13, 7400 Tübingen, Germany

Abstract

This paper discusses two rules of definitional reflection: The "logical" version of definitional reflection as used in the extended logic programming language GCLA and the "ω"-version of definitional reflection as proposed by Eriksson and Girard. The logical version is a Left-introduction rule completely analogous to the Left-introduction rules for logical operators in Gentzen-style sequent systems, whereas the ω-version extends the logical version by a principle related to the ω-rule in arithmetic. Correspondingly, the interpretation of free variables differs between the two approaches, resulting in different principles of closure of inference rules under substitution. This difference is crucial for the computational interpretation of definitional reflection.

1 Introduction

Suppose we equip a logical system such as intuitionistic first-order logic with a database $\mathcal{D}$ of clauses of the form

$$a \Leftarrow C$$

(where a is an atom and C an arbitrary formula). In a Gentzen-style sequent calculus, the normal way of handling such clauses would be to add an inference rule like

$$(\vdash\mathcal{D}) \quad \frac{\Gamma \vdash C\sigma}{\Gamma \vdash a\sigma}$$

where σ stands for an arbitrary substitution. If C is sufficiently restricted (e.g., to Horn clauses or hereditary Harrop formulas), this principle is the basis of logic programming understood proof-theoretically (see [10, 15]). It can also be viewed as reading the database as an inductive definition (see [9, 16]), expressing what it means to establish an atom by reference to its defining conditions. If that way one regards $(\vdash\mathcal{D})$ as introducing an atom on the right side of the turnstile, one may look for a corresponding rule $(\mathcal{D}\vdash)$ introducing an atom on the left side of the turnstile, in accordance with the symmetry of Gentzen-style sequent systems. This rule would express the closed character of the database $\mathcal{D}$ seen as a definition ("there is no further clause defining an atom"). Because of this way of reflecting upon the definition as a whole, such a principle has been called "definitional reflection" by Hallnäs [9].

Right-introduction rules for atoms like $(\vdash\mathcal{D})$ form the declarative basis of logic programming languages, in which implications are allowed to occur in clause bodies and hypothetical queries can be evaluated – the most advanced being λ-Prolog [15]. The computational significance of definitional reflection for logic programming lies in the fact that by $(\mathcal{D}\vdash)$ we obtain an approach to negation which is not via a meta-inference like negation by failure. If we interpret "not a" as $a\vdash\bot$, introducing a on the left side of the turnstile means introducing negation. In particular this implies that variable bindings for negated atoms can be computed, provided, of course, that there is an appropriate computational interpretation of definitional reflection.

There is a history of definitional reflection which includes "inversion principles" in logic ([13, 17]). In the context of logic programming, we have proposed such a principle in [10], which has been incorporated into the logic programming language GCLA (see [1]). A version different from that has been proposed by Eriksson [5] and by Girard [8]. It is the aim of this paper to discuss and compare these two versions. For reasons to be explained they are called the "logical version" and the "ω-version", respectively, of definitional reflection. In the following section we first discuss the propositional case, in which both versions do not differ. Certain basic problems such as cut-elimination can already be discussed there. The remaining three sections are dedicated to the differences in the treatment of variables and to computational aspects.

2 The propositional case

As the underlying logic to which we add definitional reflection we choose the following sequent calculus, of which we consider both the version with the structural rule of contraction and the version without contraction. Due to the possible lack of contraction we have to distinguish between two conjunctions, which we denote by $\wedge$ and $\circ$. We use Γ and Δ to denote multisets of formulas, $A, B, C, \ldots$ to denote formulas and $a, b, c, \ldots$ to denote atomic formulas (= propositional constants in this section). Sequents are of the form $\Gamma \vdash A$.

$$(I)\ \frac{}{A \vdash A}$$

$$(\top)\ \frac{}{\vdash \top} \qquad (\bot)\ \frac{}{\bot \vdash C}$$

$$(\vdash\wedge)\ \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad (\wedge\vdash)\ \frac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C} \quad \frac{\Gamma, B \vdash C}{\Gamma, A \wedge B \vdash C}$$

$$(\vdash\circ)\ \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \circ B} \qquad (\circ\vdash)\ \frac{\Gamma, A, B \vdash C}{\Gamma, A \circ B \vdash C}$$

$$(\vdash\vee)\ \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \qquad (\vee\vdash)\ \frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}$$

$$(\vdash\rightarrow)\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \qquad (\rightarrow\vdash)\ \frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \rightarrow B \vdash C}$$

$$(Thin)\ \frac{\Gamma \vdash C}{A, \Gamma \vdash C} \qquad \left[(Contr)\ \frac{\Gamma, A, A \vdash C}{\Gamma, A \vdash C}\right]$$

$$(Cut)\ \frac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}$$

Due to the presence of thinning, it is sufficient to consider just one single constant for *verum* and one for *falsum*. Though we could do without $\top$, it is convenient to have this constant to avoid clauses with empty bodies.

The following results extend to the cases where thinning is lacking and also to classical sequent systems with multiple formulas in the succedent. Since we do not gain much for the questions under consideration, we here avoid the more complicated notation. Contraction-free logic is the basic substructural logic in our context.

A clause has the form

$$a \Leftarrow C$$

for an atom a and a formula C. A finite set of clauses is called a *definition*. Given a definition $\mathcal{D}$, the set of defining conditions for a in $\mathcal{D}$ is $\mathcal{D}(a) := \{C : a \Leftarrow C \in \mathcal{D}\}$. We then extend our logical system by the following pair of rules

$$(\vdash\mathcal{D})\ \frac{\Gamma \vdash C}{\Gamma \vdash a}\ (C \in \mathcal{D}(a))$$

$$(\mathcal{D}\vdash)\frac{\{\Gamma, C \vdash A : a \in \mathcal{D}(a)\}}{\Gamma, a \vdash A}$$

The rule $(\mathcal{D}\vdash)$ is called the rule of definitional reflection. The system obtained is called the logic of definitional reflection over $\mathcal{D}$ and is denoted by $\mathbf{DR}(\mathcal{D})$, if contraction is present, and by $\mathbf{DR}^{\mathrm{cf}}(\mathcal{D})$, if it is contraction-free.

Example: Let p, q, r, s be propositional constants. Let

$$\begin{aligned}\mathcal{D} = \{&p \Leftarrow q,\\ &p \Leftarrow r,\\ &q \Leftarrow s,\\ &r \Leftarrow s\}.\end{aligned}$$

Then in $\mathbf{DR}(\mathcal{D})$ we can derive $p \vdash s$ in the following way:

$$(\mathcal{D}\vdash)\ \frac{(\mathcal{D}\vdash)\ \dfrac{s \vdash s}{q \vdash s} \quad (\mathcal{D}\vdash)\ \dfrac{s \vdash s}{r \vdash s}}{p \vdash s}\ .$$

There is an obvious symmetry between $(\vdash\mathcal{D})$ and $(\mathcal{D}\vdash)$, which is analogous to the symmetry between the Right-introduction and Left-introduction rules for logical operators. By $(\vdash\mathcal{D})$ an atom can be introduced on the right side of the turnstile, by $(\mathcal{D}\vdash)$ on the left side. As with other pairs of rules in a sequent-style system, this symmetry can be made explicit by a local principle of cut: if an atom a has been introduced on the left and on the right side, respectively, and is then cut away:

$$\frac{(\vdash\mathcal{D})\dfrac{\Gamma \vdash B}{\Gamma \vdash a}\ (B \in \mathcal{D}(a)) \qquad (\mathcal{D}\vdash)\dfrac{\{\Delta, C \vdash A : C \in \mathcal{D}(a)\}}{\Delta, a \vdash A}}{\Gamma, \Delta \vdash A}\ ,$$

then this cut can be reduced to a cut for a defining condition of a:

$$\frac{\Gamma \vdash B \quad \Delta, B \vdash A}{\Gamma, \Delta \vdash A}\ .$$

This step is also called $(\vdash\mathcal{D})/(\mathcal{D}\vdash)$-reduction. Similarly, we speak of $(\vdash *)/(*\vdash)$-reduction for any logical sign $*$.

This is only a *local* principle of cut since $(\vdash\mathcal{D})/(\mathcal{D}\vdash)$-reduction does not guarantee the *global* admissibility of the cut rule for the whole system with its structural postulates. Global cut elimination is a different issue. Suppose $\mathcal{D} = \{p \Leftarrow p\to\bot\}$ for some propositional constant p. Then in $\mathbf{DR}(\mathcal{D})$ we have proofs Π_1:

$$\dfrac{\dfrac{\dfrac{p\vdash p \quad \bot\vdash\bot}{p, p\to\bot\vdash\bot}}{p, p\vdash\bot}\,(\mathcal{D}\vdash)}{p\vdash\bot}\,(Contr)$$

and Π_2:

$$(\vdash\mathcal{D})\,\dfrac{\dfrac{\Pi_1}{\vdash p\to\bot}}{\vdash p},$$

but no proof of $\vdash\bot$, which would result from applying cut to Π_1 and Π_2. This means that the cut rule is not eliminable in $\mathbf{DR}(\mathcal{D})$.

The basic reason is that in order to reduce a cut with the atomic cut-formula p, according to $(\vdash\mathcal{D})/(\mathcal{D}\vdash)$-reduction a cut with the more complex cut-formula $p\to\bot$ is generated. This situation will arise with many definitions $\mathcal{D}$, since a defining condition of an atom a is of greater complexity than a, if it is not itself an atom. However, it can be shown that for certain definitions $\mathcal{D}$, cut is eliminable in $\mathbf{DR}(\mathcal{D})$.

Theorem 1 *If the body C of any clause $a\Leftarrow C$ in $\mathcal{D}$ is implication-free, then cut is eliminable in $\mathbf{DR}(\mathcal{D})$. In particular, for definite Horn clause programs $\mathcal{D}$, where clauses have the form $a\Leftarrow a_1\wedge\ldots\wedge a_n$, $\mathbf{DR}(\mathcal{D})$ enjoys cut-elimination.*

Proof sketch We show that a generalized version of cut, Slaney's [23] multicut rule

$$(mc)\dfrac{\Gamma_1\vdash A_1 \;\ldots\; \Gamma_n\vdash A_n \quad \Delta, A_1,\ldots,A_n\vdash C}{\Gamma_1,\ldots,\Gamma_n,\Delta\vdash C}$$

can be eliminated provided $A_1,\ldots,A_n$ do not contain implication. As the induction measure we use the triple $\langle r,d,l\rangle$, where d and l are, as usual, the degree of the cut-formula and the length of the derivation of a topmost multicut, whereas r is the number of applications of $(\mathcal{D}\vdash)$ and $(Contr)$ above the rightmost premiss of that multicut. Obviously, this number decreases by a $(\vdash\mathcal{D})/(\mathcal{D}\vdash)$ - reduction, even if d does not decrease.

For cut-formulas with implications this method does not work since in a $(\vdash\to)/(\to\vdash)$-reduction premisses of multicuts change sides so that a non-rightmost premiss of a multicut becomes its rightmost premiss after the reduction step. (For a detailed proof of Theorem 1 see [21].)

Other candidates of definitions $\mathcal{D}$ for which cut is eliminable from $\mathbf{DR}(\mathcal{D})$ would be stratified ones, in which defined atoms do not occur in their definitional ancestors (i.e. their defining conditions and definitional ancestors thereof). In general, one may, following Hallnäs [9], call a definition $\mathcal{D}$ *total*, if $\mathbf{DR}(\mathcal{D})$ enjoys cut elimination and *properly partial* if cut is not eliminable. If we pass to a contraction-free system, then every definition is total:

Theorem 2 *Cut is eliminable in $\mathbf{DR}^{\mathrm{cf}}(\mathcal{D})$ for any $\mathcal{D}$.*

Proof sketch As in the proof of Theorem 1, we use a triple $\langle r,d,l\rangle$, where d and l are the degree of the cut-formula and the length of the derivation of a topmost cut. The r-value of a derivation Π ending with an application of cut counts the number of applications of $(\mathcal{D}\vdash)$ in the following way:

$r(\Pi) = 0$ if Π is an axiom.
$r(\Pi) = r(\Pi_1)$
  if Π results from Π_1 by a one-premiss rule
$r(\Pi) = r(\Pi_1) + r(\Pi_2)$
  if Π results from Π_1 and Π_2 by an additive two-premiss rule (only Γ in the conclusion)
$r(\Pi) = max(r(\Pi_1), r(\Pi_2))$
  if Π results from Π_1 and Π_2 by a multiplicative two-premiss rule (Γ and Δ in the conclusion)
$r(\Pi) = max\{r(\Pi_i) : 1\leq i\leq n\} + 1$
  if Π results from $\{\Pi_i : 1\leq i\leq n\}$ by $(\mathcal{D}\vdash)$.

Due to the lack of contraction, r cannot increase during reduction, but decreases if the degree d of the cut-formula does not decrease under $(\vdash\mathcal{D})/(\mathcal{D}\vdash)$-reduction. (For more details see [21].)

It is a philosophical question whether one should insist on definitions $\mathcal{D}$ being total and perhaps force totality by requiring the underlying logic to be contraction-free (e.g., linear — this is what Girard [8] proposes). We do not think that definitions have to be total (as we do not think that computable functions have to be total). The declarative semantics of the logic programming language GCLA, for example, is based on the system $\mathbf{DR}(\mathcal{D})$ without cut, in which, as we have seen, cut is not necessarily admissible. To generelly abandon implications in clause bodies and achieve cut elimination in that way, is no viable solution, since implications in clause bodies have proved extremely useful in proof-theoretic extensions of logic programming.

One might add that the idea of definitional reflection itself including the differences between its two versions discussed in the following is entirely independent of what philosophical point of view one takes with respect to cut elimination, and whether one works in a contraction-free system or not (see [11]). Historically, the tension between the availability of implication and contraction, which is reflected in Theorem 1 and Theorem 2, was already observed in the discussion of combinatory completeness and contraction-free logics in the early days of combinatory logic, in particular by Fitch [7] and Curry [4].

3 Definitional reflection with variables: the logical version

The two versions of definitions reflection mentioned in the introduction differ when variables are available. We now consider atoms to be of the form $p(t_1, \dots, t_n)$ for n-ary predicate symbols p (including 0-ary propositional constants) and terms t_i which are either individual variables or of the form $f(t_1, \dots, t_n)$ for n-ary function symbols f (including 0-ary individual constants). We extend our logical system by the usual first-order quantifier rules:

$$(\vdash\forall)\ \frac{\Gamma\vdash A(y)}{\Gamma\vdash\forall x A(x)}\ y\ new \qquad (\forall\vdash)\ \frac{\Gamma, A(t)\vdash C}{\Gamma, \forall x A(x)\vdash C}$$

$$(\vdash\exists)\ \frac{\Gamma\vdash A(t)}{\Gamma\vdash\exists x A(x)} \qquad (\exists\vdash)\ \frac{\Gamma, A(y)\vdash C}{\Gamma, \exists x A(x)\vdash C}\ y\ new$$

When we apply substitutions $\sigma, \theta, \tau, \dots$, it is always assumed that no variable clashes can occur.

As before, definitional clauses are of the form $a \Leftarrow C$, where now a is an atom and C a formula in the new sense. For a definition $\mathcal{D}$, the set $\mathcal{D}(a)$ of defining conditions is redefined as follows:

$$\mathcal{D}(a) \ = \{C\sigma : b \Leftarrow C \in \mathcal{D} \text{ and } a = b\sigma\}\ .$$

With this notion of $\mathcal{D}(a)$, the $(\vdash\mathcal{D})$- and $(\mathcal{D}\vdash)$-rules are formulated exactly as in the previous section, where to the $(\mathcal{D}\vdash)$-rule the following proviso is added:

Proviso for the application of $(\mathcal{D}\vdash)$:
$\mathcal{D}(a\theta) = (\mathcal{D}(a))\theta$ for any substitution θ of variables in a

(actually, $\mathcal{D}(a\theta) \subseteq (\mathcal{D}(a))\theta$ would suffice, since the converse holds anyway). Let now $\mathbf{DR}(\mathcal{D})$ and $\mathbf{DR}^{\mathrm{cf}}(\mathcal{D})$ be the logics of definitional reflection with variables, with and without contraction, respectively.

Example: Let & and $\supset$ be binary function symbols in infix notation. Let

$$\mathcal{D} = \{p(x \& y) \Leftarrow p(x) \circ p(y),\ p(x \supset y) \Leftarrow p(x) \to p(y)\}.$$

Then in $\mathbf{DR}(\mathcal{D})$ we can derive $\vdash p(((x \supset y) \& x) \supset y)$ as follows:

$$\begin{array}{rc} (\to\vdash) & \dfrac{p(x)\vdash p(x) \quad p(y)\vdash p(y)}{p(x)\to p(y), p(x)\vdash p(y)} \\ (\mathcal{D}\vdash) & \dfrac{}{p(x\supset y), p(x)\vdash p(y)} \\ (\circ\vdash) & \dfrac{}{p(x\supset y)\circ p(x)\vdash p(y)} \\ (\mathcal{D}\vdash) & \dfrac{}{p((x\supset y)\& x)\vdash p(y)} \\ (\vdash\to) & \dfrac{}{\vdash p((x\supset y)\& x)\to p(y)} \\ (\vdash\mathcal{D}) & \dfrac{}{\vdash p(((x\supset y)\& x)\supset y)} \end{array}$$

In this way, we can give a kind of truth definition for all logical constants by just giving definitional clauses corresponding to the Right-introduction-rules. The Left-introduction inferences are generated by the deduction mechanism of the $(\mathcal{D}\vdash)$-rule. Our underlying logic is then the "metalogic" in which some object logic is defined. If we permit even variable-binding operators in addition to function symbols, we can give a truth-definition of quantification by a clause like $p((\bigwedge x)y_x) \Leftarrow (\forall x)p(y_x)$, where y_x stands for a term with free variable x (an elegant treatment would use λ-terms). So in a certain sense, $(\mathcal{D}\vdash)$ has the effect of the general schema for generating elimination rules from introduction rules proposed in a natural deduction setting in [19].

In $\mathbf{DR}(\mathcal{D})$ and $\mathbf{DR}^{\mathrm{cf}}(\mathcal{D})$, rules with variables are understood as they are usually understood in formal systems, particularly in proof-theoretic accounts of logic programming: An inference rule

$$\frac{S_1 \quad \dots \quad S_n}{S}$$

for sequents $S_1, \dots, S_n, S$ with variables means that for each ground substitution θ, if $S_1\theta, \dots, S_n\theta$ holds (in some intuitive sense), then so does $S\theta$.

Correspondingly, we define a primitive inference rule

$$\frac{S_1 \quad \dots \quad S_n}{S}$$

to be *strongly closed under substitution*, if for any substitution θ for the variables free in S,

$$\frac{S_1\theta \quad \dots \quad S_n\theta}{S\theta}$$

is itself a primitive inference rule. (The restriction of θ to variables free in S is to cope with eigenvariable conditions.)

Lemma 1 *All inference rules of* $\mathbf{DR}(\mathcal{D})$ *are strongly closed under substitution.*

Proof In the case of $(\mathcal{D}\vdash)$ this is guaranteed by the proviso. We assume that free variables in clauses $b \Leftarrow C$, when used with $(\mathcal{D}\vdash)$, are always (dynamically) chosen different from the variables in Γ, a and C (standardizing apart).

We call $(\mathcal{D}\vdash)$ the "logical" version of definitional reflection, since it resembles the Left-introduction rules for logical constants in two crucial respects:
1. $(\mathcal{D}\vdash)$ is the exact counterpart of $(\vdash\mathcal{D})$ needed to ensure a $(\vdash\mathcal{D})/(\mathcal{D}\vdash)$-reduction. This reduction is formulated exactly as in the propositional part in section 2.
2. As in the rules for logical operators, there is just one formula being introduced by $(\mathcal{D}\vdash)$, whereas the side formulas in Γ and C remain unchanged.
This will both be different with the version considered in the next section.

We also speak of *local* definitional reflection since the $(\mathcal{D}\vdash)$-rule is a rule for introducing single *atoms* rather then predicates on the left side of the turnstile. For example, if $\mathcal{D} = \{p(1) \Leftarrow q, p(2) \Leftarrow q\}$, we can derive $p(1) \vdash q$ and $p(2) \vdash q$ by

$$(\mathcal{D}\vdash)\ \frac{q \vdash q}{p(1) \vdash q} \quad \text{and} \quad (\mathcal{D}\vdash)\ \frac{q \vdash q}{p(2) \vdash q},$$

but not $p(x) \vdash q$. Even if we add the clause $p(x) \Leftarrow r$ to $\mathcal{D}$, neither $p(x) \vdash q$ nor $p(x) \vdash r$ is derivable, since for $p(x)$ the proviso for the application of $(\mathcal{D}\vdash)$ is not fulfilled: Whereas $p(x)$ depends on r as its defining condition, substituting x with 1 or 2 gives a different defining condition (so closure under substitution as expressed by the proviso would be violated).

Theorem 3 *The cut-elimination results of Theorem 1 and Theorem 2 also hold for definitional reflection with variables.*

Proof sketch To deal with quantifiers one has to use closure under substitution (Lemma 1).

Computationally, in a logically restricted system, $(\vdash\mathcal{D})$ corresponds to the resolution rule (when understood as a proof search and not as a refutation principle). By unifying a with heads of definitional clauses one proceeds from $\Gamma \vdash a$ backwards to $\Gamma\sigma \vdash C\sigma$ for some $C\sigma \in \mathcal{D}(a\sigma)$. Similarly, with $(\mathcal{D}\vdash)$, one proceeds from $\Gamma, a \vdash A$ backwards to $\{\Gamma\sigma, C\sigma \vdash A\sigma : C\sigma \in \mathcal{D}(a\sigma)\}$. Here σ is a joint unifier of a with a maximal set of heads of unifiers with the additional constraint that $a\sigma$ fulfils the proviso $\mathcal{D}(a\sigma\theta) = (\mathcal{D}(a\sigma))\theta$ for any substitution θ of variables in $a\sigma$. (For particular algorithms to compute such unifiers in the environment of the logic programming language GCLA, where $(\mathcal{D}\vdash)$ is implemented, see [2]).

So, computationally, one passes from the conclusion of $(\vdash\mathcal{D})$ or $(\mathcal{D}\vdash)$ to a substitution instance of its premiss by means of unification. However, declaratively, there is just instantiation, i.e., substitution, but no unification. This is different with another version of definitional reflection.

4 The ω-version of definitional reflection

We now consider the following rule of definitional reflection, which has been proposed by Eriksson [5] and Girard [8].

$(\mathcal{D}\vdash)_\omega$:

$$\frac{\{\Gamma\sigma, C\sigma \vdash A\sigma : \sigma = mgu(a,b) \text{ for some } b \Leftarrow C \in \mathcal{D}\}}{\Gamma, a \vdash A},$$

with the proviso: The variables free in $\Gamma, a \vdash A$ must be different from those in the $b \Leftarrow C$ above the line, i.e., we always assume that variables in clauses are standardized apart. The systems $\mathbf{DR}_\omega(\mathcal{D})$ and $\mathbf{DR}_\omega^{\mathrm{cf}}(\mathcal{D})$ are obtained from $\mathbf{DR}(\mathcal{D})$ and $\mathbf{DR}^{\mathrm{cf}}(\mathcal{D})$, respectively, by replacing $(\mathcal{D}\vdash)$ with $(\mathcal{D}\vdash)_\omega$.

The motivation behind $(\mathcal{D}\vdash)_\omega$ is that to prove an atom, it is sufficient to prove all substitution instances of its defining conditions, which are obtained by considering all *mgu*s of the atom with the heads of clauses in $\mathcal{D}$. Underlying is the reading of the variables of a sequent as universally quantified from outside, i.e., $\Gamma \vdash A$ is intuitively interpreted as: "for each ground substitution σ, $\Gamma\sigma \vdash A\sigma$ holds". This idea is obviously related to the ω-rule in arithmetic. (See also the philosophical appendix below.)

It is important to realize that *mgu*s are used in the premisses of $(\mathcal{D}\vdash)_\omega$. Simple unifiers would not work: By using *mgu*s, the premisses of $(\mathcal{D}\vdash)_\omega$ refer to the defining conditions of *all* possible instances of a. Therefore, as Eriksson [5] has pointed out, in the higher-order case, where there is no unique *mgu*, *complete* sets of unifiers have to be considered.

This is different from $(\mathcal{D}\vdash)$, where just substitutions and not unifiers are used. To make this point more obvious, we write $(\mathcal{D}\vdash)$ in the following form (which is just a notational variant of the formulation given in the previous section):

$$(\mathcal{D}\vdash)\ \frac{\{\Gamma, C\sigma \vdash A : a = b\sigma \text{ for some } b{\Leftarrow}C \in \mathcal{D}\ \}}{\Gamma, a \vdash A}\ .$$

The different behavior of $(\mathcal{D}\vdash)$ and $(\mathcal{D}\vdash)_\omega$ may be illustrated by the example of the previous section with $\mathcal{D} = \{p(1){\Leftarrow}q, p(2){\Leftarrow}q\}$. In contradistinction to $\mathbf{DR}(\mathcal{D})$, in $\mathbf{DR}_\omega(\mathcal{D})$ we can derive $p(x)\vdash q$:

$$(\mathcal{D}\vdash)_\omega\ \frac{q\vdash q}{p(x)\vdash q}\ .$$

We simply use that q is the defining condition of each defined substitution instance of $p(x)$ (namely of both $p(1)$ and $p(2)$).

The standard example discussed both by Eriksson [5] and Girard [8] is equality. Let $\mathcal{D} = \{x = x{\Leftarrow}\top\}$. Then all axioms of general equality can be derived, for example transitivity:

$$(\mathcal{D}\vdash)_\omega\ \frac{(Thin)\ \dfrac{x_2 = x_3\vdash x_2 = x_3}{\top, x_2 = x_3\vdash x_2 = x_3}}{x_1 = x_2, x_2 = x_3\vdash x_1 = x_3}$$

In the last step we use that the substitution $[x_2/x_1, x_2/x]$ is an *mgu* of $x_1 = x_2$ with the head $x = x$ of $x = x{\Leftarrow}\top$. Furthermore, the freeness axioms of Clark's equality theory [3] can be derived.

$(\mathcal{D}\vdash)_\omega$ is not strongly closed under substitution, as the following example shows: Suppose $\mathcal{D} = \{p(x){\Leftarrow}q(x),\ q(x){\Leftarrow}r(1)\}$. Then

$$\frac{q(x)\vdash r(1)}{p(z)\vdash r(1)}$$

is an instance of $(\mathcal{D}\vdash)_\omega$, whereas

$$\frac{q(x)\vdash r(1)}{p(1)\vdash r(1)}$$

ist not. But, of course,

$$\frac{q(1)\vdash r(1)}{p(1)\vdash r(1)}$$

is again an instance of $(\mathcal{D}\vdash)_\omega$. The reason is obvious: Due to the universal reading of variables according to $(\mathcal{D}\vdash)_\omega$, there is no "vertical" connection between variables. An inference rule

$$\frac{S_1\ \ \dots\ \ S_n}{S}$$

for sequents $S_1, \dots, S_n, S$ with variables means that if for each i $(1 \le i \le n)$ and each ground substitution θ_i, $S_i\theta_i$ holds (in some intuitive sense), then for each ground substitution θ, $S\theta$ holds.

Correspondingly, we call a primitive inference rule

$$\frac{S_1\ \ \dots\ \ S_n}{S}$$

weakly closed under substitution, if for any substitution θ for the variables free in S, there are substitutions $\theta_1, \dots, \theta_n$ such that

$$\frac{S_1\theta_1\ \ \dots\ \ S_n\theta_n}{S\theta}\ ,$$

or a rule obtained from that by deleting some of the premisses $S_i\theta_i$, is itself a primitive inference rule.

Lemma 2 *All inference rules of* $\mathbf{DR}_\omega(\mathcal{D})$ *are weakly closed under substitution.*

Proof by induction on the length of derivations. We just have to consider $(\mathcal{D}\vdash)_\omega$. Suppose its i-th premiss is $\Delta\sigma, C\sigma\vdash A\sigma$. If $a\theta$ and $b\theta$ are not unifiable then this premiss is not needed for the derivation of $\Delta\theta, a\theta\vdash A\theta$. Otherwise let $\sigma' := mgu(a\theta, b)$. Then $\theta\sigma'$ is a unifier of a and b (since variables are standardized apart), so that $\theta\sigma' = \sigma\tau$ for some τ. Take τ to be the θ_i we are looking for.

As a corollary we have closure of derivations under substitutions, which is essentially what we need.

Lemma 3 *If* $\Gamma\vdash A$ *is derivable in* $\mathbf{DR}_\omega(\mathcal{D})$ *or* $\mathbf{DR}^{\mathrm{cf}}_\omega(\mathcal{D})$*, then there is a derivation of* $\Gamma\theta\vdash A\theta$ *in* $\mathbf{DR}_\omega(\mathcal{D})$ *or* $\mathbf{DR}^{\mathrm{cf}}_\omega(\mathcal{D})$*, respectively, which is not longer than the original one and which does not use any inference rules beyond those used in the original derivation.*

Consider the last example. Then in $(\mathcal{D}\vdash)_\omega$ we have the derivation

$$\frac{\dfrac{r(1)\vdash r(1)}{q(x)\vdash r(1)}}{p(z)\vdash r(1)}$$

and for the substitution $[1/z]$ the derivation

$$\frac{\dfrac{r(1)\vdash r(1)}{q(1)\vdash r(1)}}{p(1)\vdash r(1)}$$

where we have to use $[1/x]$ at the intermediate step. Viewed as a tree whose nodes are labelled with the names of the inferences used, after substitution one obtains the same tree with perhaps some branches missing.

It is not trivial to extend the cut elimination results from the logical version of definitional reflection to the present case, since, unlike $(\mathcal{D}\vdash)$, $(\mathcal{D}\vdash)_\omega$ is not a Left-introduction rule in the genuine sense, i.e., not a rule dual to $(\vdash\mathcal{D})$. There is no straightforward $(\vdash\mathcal{D})/(\mathcal{D}\vdash)_\omega$-reduction, since in $(\mathcal{D}\vdash)_\omega$ the side-formulas Γ and A do not remain unchanged when a is introduced on the left side of the turnstile. Rather, they occur substituted above and unsubstituted below the inference line. Due to the σ occurring in the premisses of $(\mathcal{D}\vdash)_\omega$ on both sides, $(\mathcal{D}\vdash)_\omega$ can even be used to introduce an atom on the right side.

However, by decomposing $(\mathcal{D}\vdash)_\omega$ into $(\mathcal{D}\vdash)$ and some other rule, we can carry over the methods used for cut-elimination with $(\mathcal{D}\vdash)$ to the present case. In that way we also obtain additional insight into the relationship between $(\mathcal{D}\vdash)_\omega$ and $(\mathcal{D}\vdash)$. Let (ω) be the following rule.

(ω):

$$\frac{\{\Gamma\sigma, a\sigma \vdash A\sigma : \sigma = mgu(a,b) \text{ for some } b{\Leftarrow}C \in \mathcal{D}\}}{\Gamma, a \vdash A}$$

Then the following holds:

Theorem 4 *In the system with or without contraction, but without cut, $(\mathcal{D}\vdash)_\omega$ is interadmissible with $(\mathcal{D}\vdash)$ and (ω), i.e., $(\mathcal{D}\vdash)_\omega$ is admissible if $(\mathcal{D}\vdash)$ and (ω) are primitive rules, and both $(\mathcal{D}\vdash)$ and (ω) are admissible, if $(\mathcal{D}\vdash)_\omega$ is a primitive rule.*

Proof Let $(\mathcal{D}\vdash)$ have the alternative form given at the beginning of this section. We first show that $(\mathcal{D}\vdash)_\omega$ is admissible if $(\mathcal{D}\vdash)$ and (ω) are present. Consider a topmost application of $(\mathcal{D}\vdash)_\omega$. Suppose $\sigma = mgu(a,b)$ for some $b{\Leftarrow}C \in \mathcal{D}$. Suppose furthermore that $a\sigma = b'\sigma'$ for some $b'{\Leftarrow}C' \in \mathcal{D}$. Since variables in a and b' are different, $\sigma \cup \sigma'$ is a unifier of a and b', so there is an mgu θ of a and b' such that $\sigma \cup \sigma' = \theta\theta'$ for some θ'. Then $\Gamma\theta, C'\theta \vdash A\theta$ is among the premisses of $(\mathcal{D}\vdash)_\omega$. Therefore by Lemma 3, $\Gamma\sigma, C'\sigma' \vdash A\sigma$ is derivable without $(\mathcal{D}\vdash)_\omega$ (again, notice that clauses variables are standardized apart). By $(\mathcal{D}\vdash)$ we obtain $\Gamma\sigma, a\sigma \vdash A\sigma$, if we can make sure that the proviso $\mathcal{D}(a\sigma\tau) = (\mathcal{D}(a\sigma))\tau$ is fulfilled for any substitution τ of variables in $a\sigma$. For that, suppose $a\sigma\tau = b''\tau''$ for some τ'' and some $b''{\Leftarrow}C'' \in \mathcal{D}$. Then a and b'' are unifiable and every element of $\mathcal{D}(a\sigma\tau)$ is in $(\mathcal{D}(a\sigma))\tau$. Since we have $\Gamma\sigma, a\sigma \vdash A\sigma$ for any mgu σ of a with some $b{\Leftarrow}C \in \mathcal{D}$, we obtain $\Gamma, a \vdash A$ by (ω).

Conversely, suppose $(\mathcal{D}\vdash)_\omega$ is given and the premisses of $(\mathcal{D}\vdash)$ of the form $\Gamma, C\sigma \vdash A$, where $a = b\sigma$ for some $b{\Leftarrow}C \in \mathcal{D}$, i.e. $C\sigma \in \mathcal{D}(a)$, have been derived. Then

$(*)$ $\Gamma\sigma, C\sigma \vdash A\sigma$ with $\sigma = mgu(a,b)$ (standardizing clause variables apart).

Now suppose $\theta = mgu(a,b')$ for some $b'{\Leftarrow}C' \in \mathcal{D}$, and only $a\theta$ but not a itself is an instance of b'. Then due to the proviso $\mathcal{D}(a\theta) = (\mathcal{D}(a))\theta$ we have the following: $C'\theta = C\sigma\theta$ for some $C\sigma$ of the kind considered in $(*)$. But $(*)$ implies $\Gamma\sigma\theta, C\sigma\theta \vdash A\sigma\theta$ by closure under substitution (Lemma 3). Thus from the sequents $(*)$ all premisses of $(\mathcal{D}\vdash)_\omega$ are derivable.

Finally, we have to show that (ω) is admissible, if $(\mathcal{D}\vdash)_\omega$ is given. Consider a premiss $\Gamma\sigma, a\sigma \vdash A\sigma$ of (ω), where $\sigma = mgu(a,b)$ for some $b{\Leftarrow}C \in \mathcal{D}$. Then according to the following Lemma 4, $a\sigma$ is either obtained by (I) or by $(Thin)$ or by $(\mathcal{D}\vdash)_\omega$. The first two cases are trivial, and in the last case $\Gamma\sigma, C\sigma \vdash A\sigma$ is among the premisses of $\Gamma\sigma, a\sigma \vdash A\sigma$, since σ is already an mgu of a and b.

Lemma 4 *Any derivation of $\Gamma, a \vdash A$ in $\mathbf{DR}_\omega(\mathcal{D})$ or $\mathbf{DR}^{\mathrm{cf}}_\omega(\mathcal{D})$ can be transformed into one in which a is introduced in the last step either by (I) or by $(Thin)$ or by $(\mathcal{D}\vdash)_\omega$.*

Proof by induction on the length of derivations. The only critical case is the one where in the last step $(\mathcal{D}\vdash)_\omega$ is applied introducing some a'. That is, we have two successive applications of $(\mathcal{D}\vdash)_\omega$, one with respect to a and one with respect to a'. Every branch of such a derivation then ends as follows:

$$\cfrac{\cfrac{\Gamma\sigma_i\sigma_{ij}, C_i\sigma_i\sigma_{ij}, C_j\sigma_i\sigma_{ij} \vdash A\sigma_i\sigma_{ij}}{\Gamma\sigma_i, a\sigma_i, C_i\sigma_i \vdash A\sigma_i}}{\Gamma, a, a' \vdash A},$$

where $1 \leq i,j \leq n$, $\mathcal{D} = \{b_1{\Leftarrow}C_1, \ldots, b_n{\Leftarrow}C_n\}$, $\sigma_i = mgu(a', b_i)$, and $\sigma_{ij} = mgu(a\sigma_i, b_j)$. Thus $\sigma_i\sigma_{ij}$ is the mgu of the equation system $\{a' = b_i, a = b_j\}$. Since the order of steps in which to solve the equation system is irrelevant, we can interchange the last two inferences of the given derivation.

Theorem 5 *The cut elimination results of Theorem 1 and Theorem 2 extend to definitional reflection with $(\mathcal{D}\vdash)_\omega$.*

Proof sketch By Theorem 4, we only have to deal with additional effects due to the presence of (ω). The crucial case is the following:

$$\dfrac{\dfrac{\Gamma \vdash B}{\Gamma \vdash a} \quad \dfrac{\{\Delta\sigma, a\sigma \vdash A\sigma : \sigma = mgu(a,b) \text{ for } b \Leftarrow C \in \mathcal{D}\}}{\Delta, a \vdash A}}{\Gamma, \Delta \vdash A} .$$

Since $B \in \mathcal{D}(a)$, we have $a = b\sigma$ for some $b \Leftarrow B \in \mathcal{D}$, i.e., the mgu of a and b only instantiates variables in b. Thus $\Delta, a \vdash A$ is among the premisses of $(\mathcal{D}\vdash)$.

All other cases of (ω) being involved in cuts can be dealt with by permutative reductions in combination with closure under substitution (Lemma 3, which obviously holds for (ω), too).

There is a close relationship between $(\mathcal{D}\vdash)_\omega$ and the completion on a definition. Suppose the definitional clauses for p in $\mathcal{D}$ have the following form:

$$\begin{array}{rcl} p(t_{11}, \ldots, t_{1m}) & \Leftarrow & C_1 \\ & \vdots & \\ p(t_{n1}, \ldots, t_{nm}) & \Leftarrow & C_n \ . \end{array}$$

Then, if we add to $\mathcal{D}$ the single equality clause

$$x = x \Leftarrow \top,$$

the following axiom can be derived in $\mathbf{DR}^{\mathrm{cf}}_\omega(\mathcal{D})$.

$(Comp_p)$:

$$\begin{array}{ll} \vdash & (\forall x_1 \ldots x_m)(p(x_1, \ldots, x_m) \leftrightarrow \\ & (\exists \overline{y_1})(x_1 = t_{11} \circ \ \ldots \ \circ x_m = t_{1m} \circ C_1) \\ & \vee \\ & \vdots \\ & \vee \\ & (\exists \overline{y_n})(x_1 = t_{n1} \circ \ \ldots \ \circ x_m = t_{nm} \circ C_n)), \end{array}$$

where $\overline{y_i}$ contains the variables free in the i-th clause for p. It can actually be shown that $\mathbf{DR}^{\mathrm{cf}}_\omega(\mathcal{D} \cup \{x = x \Leftarrow \top\})$ is equivalent to $\mathbf{COMP(\mathcal{D})}$, where $\mathbf{COMP(\mathcal{D})}$ results from contraction-free first-order logic (without $(\vdash\mathcal{D})$ and $(\mathcal{D}\vdash)$) by adding
1. a sequent-style version of Clark's [3] equality theory (see, e.g., [18]) and
2. for each predicate p in the definition $\mathcal{D}$ Right- and Left-introduction rules $(\vdash p)$ and $(p\vdash)$ which, unlike $(\vdash\mathcal{D})$ and $(\mathcal{D}\vdash)$, are uniform in the arguments of p.
This equivalence, which is elaborated in [22], does not rely on the cut rule.

Since cut can be eliminated in $\mathbf{DR}^{\mathrm{cf}}_\omega(\mathcal{D})$, from $(Compl_p)$ we may infer the following principle for $\mathbf{DR}^{\mathrm{cf}}_\omega(\mathcal{D})$.

(Fix_p):

$\vdash p(x_1, \ldots, x_m)$ is derivable iff for some i,
$\vdash x_1 = t_{i1} \circ \ \ldots \ \circ x_m = t_{im} \circ C_i$ is derivable.

This result was called "fixpoint theorem" by Girard [8], because it allows one to define a predicate by any disjunction of defining conditions.

Unlike $(\mathcal{D}\vdash)_\omega$, the rule $(\mathcal{D}\vdash)$ does not imply $(Compl_p)$. It would just permit us to derive

$$a \dashv\vdash C_1 \vee \ \ldots \ \vee C_n$$

if a is defined by

$$\begin{array}{rcl} a & \Leftarrow & C_1 \\ & \vdots & \\ a & \Leftarrow & C_n \ . \end{array}$$

This again shows the different view of variables underlying $(\mathcal{D}\vdash)$ and $(\mathcal{D}\vdash)_\omega$.

5 Computational issues

If we want to extend logic programming by definitional reflection, then the logical rule $(\mathcal{D}\vdash)$ is more appropriate then $(\mathcal{D}\vdash)_\omega$. The idea of *successively* computing an answer substitution, which is central to logic programming, is bound to *strong* closure under substitution, which holds for $(\mathcal{D}\vdash)$ but not for $(\mathcal{D}\vdash)_\omega$. If we are searching an answer substitution σ such that, given a query $\Gamma \vdash A$, the sequent $\Gamma\sigma \vdash A\sigma$ is derivable, we compose σ step by step as $\sigma = \sigma_1 \ldots \sigma_n$. Here any σ_i, which is being computed at a certain stage, is viewed at the same time as (part of the) substitution of the original query. This possibility of interchanging substitution and computation, also called lifting, crucially depends on a "vertical" connection between variables, which according to Lemma 1 is available for $(\mathcal{D}\vdash)$, but not for $(\mathcal{D}\vdash)_\omega$. So, for computational reasons, $(\mathcal{D}\vdash)_\omega$ is not suitable as the basic rule of definitional reflection in an extended logic programming language.

However, $(\mathcal{D}\vdash)_\omega$ can play a central role in the modelling of a certain kind of quantification in such a language. Besides quantification in the "logical" sense in which the introduction of $\forall$ on the right side and the introduction of $\exists$ on the left side of the turnstile is reduced to the schematic derivability with eigenvariables, there may be "ω-quantification" in the sense that $\forall x A$ is reduced to $A\sigma$ for all substitutions σ. This corresponds to the intuition behind (ω), which, according to Theorem 4, is the characteristic part of $(\mathcal{D}\vdash)_\omega$.

To allow for both definitional reflection with the computation of answer aubstitutions in the sense of $(\mathcal{D}\vdash)$ and an interpretation of quantification in the sense of (ω), Eriksson ([5]) has proposed a system with the following general rule which we call $(\mathcal{D}\vdash)_{gen}$. We distinguish between existential variables $x, y, z, \ldots$ and universal variables $x^*, y^*, z^*, \ldots$ ("parameters"). In the terminology of the present paper, existential variables are those for which strong closure under substitution holds and therefore answer substitutions can be computed successively (in principle). Universal variables are those for which only weak closure under substitution holds. The primitive rules for quantifiers are now formulated with universal variables x^* and y^* rather than x and y. Then $(\mathcal{D}\vdash)_{gen}$ is the following rule.

$(\mathcal{D}\vdash)_{gen}$:

$$\frac{\{\Gamma\sigma^*, C\sigma^* \vdash A\sigma^* : \sigma^* = mgu(a,b) \text{ for } b{\Leftarrow}C \in \mathcal{D}\}}{\Gamma, a \vdash A},$$

where σ^* is a substitution of universal variables (including variables in clauses), and where we have as a proviso that for all substitutions θ for existential variables in a, $\mathcal{D}(a\theta) = (\mathcal{D}(a))\theta$. (Eriksson's rule is even more general, allowing for higher-order definitions. For details see [5].) Obviously, both $(\mathcal{D}\vdash)$ and $(\mathcal{D}\vdash)_\omega$ are special cases of $(\mathcal{D}\vdash)_{gen}$: $(\mathcal{D}\vdash)$ is the case without universal variables and $(\mathcal{D}\vdash)_\omega$ the case without existential variables.

It is obvious that this rule adds much expressive power to extended logic programming languages, of course with many algorithmic problems of how to efficiently compute bindings at applications of $(\mathcal{D}\vdash)_{gen}$.

Philosophical Appendix Why is it that the ω-rule in arithmetic

$$\frac{\Gamma[1/x]\vdash A[1/x] \quad \Gamma[2/x]\vdash A[2/x] \quad \ldots}{\Gamma\vdash A}$$

is infinitary, but (ω) is finitary? Basically because of the *absurdity principle*, according to which $\Gamma, a\vdash A$ holds, if a is not defined by the definition $\mathcal{D}$.

This can be seen as follows: Suppose we want to justify (ω) by showing that, if for each ground substitution, each premiss of (ω) holds, then for each ground substitution its conclusion holds, i.e., for each ground θ : $\Gamma\theta, a\theta\vdash A\theta$. We have to argue as follows: Either $a\theta$ is defined, i.e., $a\theta = b\sigma$ for some $b{\Leftarrow}C \in \mathcal{D}$: Then $\Gamma\theta, a\theta\vdash A\theta$ is a substitution instance of a premiss of (ω). Or $a\theta$ is not defined: Then $\Gamma\theta, a\theta\vdash A\theta$ holds anyway by the absurdity principle.

Therefore, if one does not accept the absurdity principle for philosophical reasons, one cannot accept $(\mathcal{D}\vdash)_\omega$.

Historical Appendix The rule $(\mathcal{D}\vdash)$ is related to the attempts to formulate a general schema for elimination rules in natural deduction (see [19]). The (infinitary) variable-free formulation was given 1986 by Hallnäs [9] in his theory of partial inductive definitions. The version with variables including the proviso was developed by Hallnäs and the author in 1987 [10]. We also considered $(\mathcal{D}\vdash)_\omega$ at that time, but rejected it, since it did not seem to us appropriate for a logic programming language. Its first serious consideration was by Eriksson in a talk in May 1990 at Chalmers University (Göteborg) and at the ELP-Workshop in Stockholm in January 1991 [5], and is investigated in detail in his thesis [6] as a special case of $(\mathcal{D}\vdash)_{gen}$. He gives a semantics of this rule in terms of infinitary (variable-free) partial inductive definitions in Hallnäs' sense, extends it by an induction principle and proves completeness with respect to that semantics. In August 1991 Girard presented a rule equivalent to $(\mathcal{D}\vdash)_\omega$ in a talk at the German workshop on Artificial Intelligence and discussed it in [8]. He considers $(\mathcal{D}\vdash)_\omega$ as a rule in the framework of linear logic and motivates it in relation to logic programming, in particular negation as failure. Cut elimination for the case of linear logic is also mentioned.

Acknowledgements Since I have been working closely with Lars-Henrik Eriksson and Lars Hallnäs, some of their ideas may have entered this paper without their authorship being mentioned in each case. Further thanks are due to Dale Miller and the referees for helpful comments and suggestions. This work was supported by DFG grant Schr 275/8-1 and by Esprit Basic Research Working Group 7232 (GENTZEN).

References

[1] Aronsson, M., Eriksson, L.-H., Gäredal, A., Hallnäs, L. & Olin, P. The programming language GCLA: A definitional appraoch to logic programming. *New Generation Computing*, **4** (1990), 381–404.

[2] Aronsson, M. Implementational issues in GCLA: *A*-sufficiency and the definiens operation. In: E. Lamma & P. Mello (Eds.), *Extensions of Logic Programming. Third International Workshop, ELP-92, Bologna, February 1992, Proceedings.* Springer LNCS, Berlin 1993.

[3] Clark, K. L. Negation as failure. In: Gallaire, H. & Minker, J. (Eds.), *Logic and Data Bases*, New York 1978, 293–322.

[4] Curry, H. B. The inconsistency of certain formal logics. *Journal of Symbolic Logic*, **7** (1942), 115–117.

[5] Eriksson, L.-H. A finitary version of the calculus of partial inductive definitions. In: Eriksson, L.-H., Hallnäs, L. & Schroeder-Heister, P. (Eds.), *Extensions of Logic Programming. Second International Workshop, ELP-91, Stockholm, January 1991, Proceedings.* Springer LNCS, Vol. 596, Berlin 1992, 89–134.

[6] Eriksson, L.-H. *Finitary Partial Inductive Definitions and General Logic.* Ph. D. thesis, Royal Institute of Technology, Stockholm 1993.

[7] Fitch. F. B. A system of formal logic without an analogue to the Curry *W* operator. *Journal of Symbolic Logic*, **1** (1936), 92–100.

[8] Girard, J.-Y. A fixpoint theorem for linear logic. In: P. Lincoln (Ed.), *Linear Logic Mailing List*, linear@cs.stanford.edu, 5 February 1992.

[9] Hallnäs, L. Partial inductive definitions. *Theoretical Computer Science*, **87** (1991), 115-142. Previous versions published as SICS Research Reports 86005 (1986) and 86005C (1988), and in: Avron, A. et al. (Eds.), Workshop on General Logic, Dept. of Computer Science, University of Edinburgh, Report ECS-LFCS-88-52 (1988).

[10] Hallnäs, L. & Schroeder-Heister, P. A proof-theoretic approach to logic programming. I. Clauses as rules. *Journal of Logic and Computation*, **1** (1990), 261–283; II. Programs as definitions, *ibid.* **1** (1991), 635–660. Previous version published at SICS Research Report 88005, 1988.

[11] Hallnäs, L. & Schroeder-Heister, Jean Yves Girard's "A fixpoint theorem in linear logic". In: P. Lincoln (Ed.), *Linear Logic Mailing List*, linear@cs.stanford.edu, 19 February 1992.

[12] Hermes, H. Zum Inversionsprinzip der operativen Logik. In: A. Heyting (ed.), *Constructivity in Mathematics*, Amsterdam: North-Holland, 1961, 62-68.

[13] Lorenzen, P. *Einführung in die operative Logik und Mathematik.* Springer: Berlin 1955, 2nd ed. Berlin 1969.

[14] Martin-Löf, P. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In: Fenstad, J. E. (Ed.), *Proceedings of the Second Scandinavian Logic Symposium*, North-Holland: Amsterdam 1971, 179–216.

[15] Nadathur, G. & Miller, D. An overview of λ-Prolog. In: Kowalski, R. & Bowen, K. (Eds.), *Fifth International Conference on Logic Programming*, MIT Press 1988, 810–827.

[16] Paulson, L. C. & Smith, A. W. Logic programming, functional programming, and inductive definitions. In: P. Schroeder-Heister (Ed.), *Extensions of Logic Programming. International Workshop, Tübingen, FRG, December 1989, Proceedings.* Springer LNCS, Vol. 475, Berlin 1991, 283–309.

[17] Prawitz, D. *Natural Deduction: A Proof-Theoretical Study.* Almqvist & Wiksell: Stockholm, 1965.

[18] Sahlin, D., Franzén, T. & Haridi, S. An intuitionistic predicate logic theorem prover, *Journal of Logic and Computation* (to appear).

[19] Schroeder-Heister, A natural extension of natural deduction, *Journal of Symbolic Logic*, **49** (1984), 1284–1300.

[20] Schroeder-Heister, P. Hypothetical reasoning and definitional reflection in logic programming. In: P. Schroeder-Heister (Ed.), *Extensions of Logic Programming. International Workshop, Tübingen, FRG, December 1989, Proceedings.* Springer LNCS, Vol. 475, Berlin 1991, 327–340.

[21] Schroeder-Heister, P. Cut-elimination in logics with definitional reflection. In: D. Pearce &

H. Wansing (Eds.), *Nonclassical Logics and Information Processing. International Workshop, Berlin 1990, Proceedings.* Springer LNCS, Vol. 619, Berlin 1992, 146–171.

[22] Schroeder-Heister, P. Definitional reflection and the completion. *Extensions of Logic Programming. Fourth International Workshop, ELP-93, St. Andrews, March 1993.*

[23] Slaney, J. Solution to a problem of Ono and Komori. *Journal of Philosophical Logic*, **18** (1989), 103–111.