Proofs by Resolution and Existential Variables

František Farka
University of St Andrews
Scotland, UK, and
University of Dundee
Scotland, UK
ff32@st-andrews.ac.uk

Many researchers have devoted their effort to establish category-theoretic semantics of logic programming (LP). In particular, there have been two lines of work in operational semantics of LP: Komendantskaya, Power and their collaborators developed lax semantics [3] and Bonchi and Zanasi proposed saturated semantics [1]. Recently, Komendanskaya and Power unified the two approaches [3]. They proposed an interpretation of *term-matching* resolution as theorem proving in LP whereas Bonchi and Zanasi's approach captures problem solving aspect of LP. Following our previous work [2], we are interested in type theoretic formulation of the operational semantics and we strengthen the above claim. In fact, term matching steps in resolution reflect proof steps acting on universally quantified variables whereas unification steps correspond to proof steps acting on existentially quantified variables — the problem solving aspect here corresponds to search for proof witnesses for existential variables.

We work in a proof-relevant language. A *signature* comprises a set of function symbols $\mathscr{F}$, including constant symbols, a set of predicate symbols $\mathscr{P}$ and a set of proof-term symbols $\mathscr{K}$. Also, we assume there is a countably infinite set of variables *Var*. A *substitution*, an application of substitution, a *unifier* and a *matcher* are defined entirely standard.

Definition 1 (Syntax)**.**

$$\begin{array}{rrcl@{\qquad}rrcl} \textit{Terms} & \textit{Ter} & ::= & \textit{Var} \mid \mathscr{F}(\textit{Ter},\ldots,\textit{Ter}) & \textit{Horn flae} & \textit{HC} & ::= & \textit{At} \leftarrow \textit{At},\ldots,\textit{At} \\ \textit{Atomic flae} & \textit{At} & ::= & \mathscr{P}(\textit{Ter},\ldots,\textit{Ter}) & \textit{Programs} & \textit{Prog} & ::= & \mathscr{K} : \textit{HC},\ldots,\mathscr{K} : \textit{HC} \\ \multicolumn{4}{c}{} & \textit{Proof terms} & \textit{PT} & ::= & \mathscr{K}\ \textit{PT}\ldots\textit{PT} \mid \textit{ind}(\textit{Var.Term},\textit{PT}) \\ \multicolumn{4}{c}{} & \textit{Goals} & G & ::= & \forall \textit{Var},\ldots,\textit{Var} \exists \textit{Var},\ldots,\textit{Var.At} \end{array}$$

Our language differs from the standard language of LP in two ways: Horn formulae in our programs are annotated with proof-term constant symbols, and goals are explicitly quantified. We refer to a clause in a program by the symbol that annotates it. Moreover, we denote a sequence of distinct variables, by $\overline{x}$, and a sequence of variables occurring in an atomic formula A by $\mathrm{var}(A)$. Substitution application, set intersection and set difference are extended to variable sequences in the natural way.

The explicit quantification of goals helps us to separate matching and unification steps in our proofs. First, let us introduce proof steps by term matching:

Definition 2 (TM-Resolution)**.**

$$\kappa : A \leftarrow B_1,\ldots,B_n \in P \ \frac{P \vdash e_1 : \forall \overline{w_1} \exists \overline{z_1}.\sigma B_1 \qquad \ldots \qquad P \vdash e_n : \forall \overline{w_n} \exists \overline{z_n}.\sigma B_n}{P \vdash \kappa\ e_1\ \ldots\ e_n : \forall \overline{x} \exists \overline{y}.\sigma A}$$

where $\overline{w_i} = \overline{x} \cap \mathrm{var}(\sigma B_i)$ *and* $\overline{z_i} = \mathrm{var}(\sigma B_i) \setminus w_i$.

The above inference rule distributes quantified variables of a goal $\forall \overline{x} \exists \overline{y}.\sigma A$ to goals $\forall \overline{w_1} \exists \overline{z_1}.B_1$ to $\forall \overline{w_n} \exists \overline{z_n}.B_n$ in premise. Universally quantified variables that are relevant to a goal B_i are quantified universally, existential variables are quantified existentially and any (existential) variable that does not occur

Submitted to:
CoALP-Ty'16

in the head of clause κ is quantified existentially. This is in concordance with the above referred results of Komendantskaya and Power [3]. In particular, consider the following example.

Example 1. *Let $P_{ListNat}$: be the following program:*

$$\begin{array}{llll} \kappa_1: & \mathrm{nat}(\mathrm{zero}) & \leftarrow & & \kappa_3: & \mathrm{list}(\mathrm{nil})) & \leftarrow \\ \kappa_2: & \mathrm{nat}(\mathrm{s}(x)) & \leftarrow \mathrm{nat}(x) & & \kappa_4: & \mathrm{list}(\mathrm{cons}(x,y))) & \leftarrow \mathrm{nat}(x), \mathrm{list}(y) \end{array}$$

A ground goal $\mathrm{list}(\mathrm{cons}(\mathrm{zero},\mathrm{nil}))$ can be resolved using only the TM-Resolution rule:

$$\frac{\overline{P \vdash \kappa_1 : \mathrm{nat}(\mathrm{zero})} \qquad \overline{P \vdash \kappa_3 : \mathrm{list}(\mathrm{nil})}}{P \vdash \kappa_4\ \kappa_1\ \kappa_3 : \mathrm{list}(\mathrm{cons}(\mathrm{zero},\mathrm{nil}))}$$

However, for other goals full unification is necessary. The Unif-resolution is defined as follows.

Definition 3 (Unif-Resolution)**.**

$$\kappa : A \leftarrow B_1,\ldots,B_n \in P \ \frac{P \vdash e_1 : \forall \overline{w_1}\exists \overline{z_1}.\sigma B_1 \qquad \ldots \qquad P \vdash e_n : \forall \overline{x}.\sigma B_n}{P \vdash \mathit{ind}(\overline{y}.\sigma\overline{y}, \kappa\ e_1,\ldots,e_n) : \forall \overline{x}\exists \overline{y}.A'}$$

if $\sigma \restriction_{\overline{y}} A' = \sigma \restriction_{\overline{y}} A$. Moreover, $w_i = \overline{x} \cap \mathrm{var}(\sigma B_i)$ and $\overline{z_i} = \mathrm{var}(\sigma B_i) \setminus w_i$.

We use $\sigma \restriction \overline{y}$ to denote a restriction of a substitution σ to variables $\overline{y}$. The notation $\overline{y}.\sigma\overline{y}$ indicates binding of variables in $\overline{y}$ given by substitution σ. Using both TM-resolution and Unif-resolution rules allows us to prove *e.g.* $\exists x,y.\,\mathrm{list}(\mathrm{cons}(x,y))$:

$$\frac{\overline{P \vdash \kappa_1 : \mathrm{nat}(\mathrm{zero})} \qquad \overline{P \vdash \kappa_3 : \mathrm{list}(\mathrm{nil})}}{P \vdash \mathit{ind}(x.\mathit{zero}, y.\mathit{nil}, \kappa_4\ \kappa_1\ \kappa_3) : \exists x,y.\,\mathrm{list}(\mathrm{cons}(x,y))}$$

Finally, proofs by TM-Resolution are sound w.r.t. coalgebraic operational semantics of resolution given by Komendantskaya and Power [3]. We extend Lawvere theory $\mathscr{L}_\Sigma^{\mathrm{op}}$ to category $\mathscr{K}_\Sigma^{op}$ with explicit marking of variables by quantifiers and substitutions preserve these markings as in the TM-Resolution rule. Note that there is the obvious forgetful functor $U : At \to G$. We model goals by presheaf $G : \mathscr{K}_\Sigma^{op} \to \mathit{Poset}$. This allows us to state the following theorem:

Theorem 1. *The forgetful functor $U : G \to At$ has the left adjoint F such that given a non-existential logic program P via P_fP_f-coalgebra $p : At \to P_fP_fAt$, the following diagram commutes:*

$$\begin{array}{ccc} G & \xrightarrow{\overline{Fp}} & C(P_fP_f)(G) \\ F \Big\uparrow \Big\downarrow U & & \Big\downarrow C(P_fP_f)(U) \\ At & \xrightarrow[\bar{p}]{} & C(P_fP_f)(At) \end{array}$$

where $C(-)$ denotes the cofree comonad.

References

[1] Filippo Bonchi & Fabio Zanasi (2015): *Bialgebraic Semantics for Logic Programming*. Logical Methods in Computer Science 11(1), doi:10.2168/LMCS-11(1:14)2015.

[2] František Farka, Ekaterina Komendantskaya, Kevin Hammond & Peng Fu (2016): *Coinductive Soundness of Corecursive Type Class Resolution*. In: Pre-proceedings of the 26th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2016).

[3] Ekaterina Komendantskaya & John Power (2016): *Logic programming: laxness and saturation*. Submitted to Journal of Logic and Algebraic Methods in Programming.

A Definitions

Definition 4 (Lawvere theory)**.** *The (opposite)* Lawvere theory *on a signature* Σ *is a category* $\mathscr{L}_\Sigma^{\mathrm{op}}$ *where*

objects *are natural numbers*

morphisms *for any two objects* $m,n \in \mathscr{L}_\Sigma^{\mathrm{op}}$*, the set* $\mathscr{L}_\Sigma^{\mathrm{op}}[n,m]$ *consists of all n-tuples* $< t_1,\ldots,t_n >$ *of terms in variables* $x_1,\ldots,x_m$*. The composition of* $< t_1,\ldots,t_n >: n \to m$ *and* $< s_1,\ldots,s_m >: m \to l$ *is the tuple* $< r_1,\ldots,r_n >: n \to l$ *where* r_i *is a term* t_i *where every variable* x_j *has been replaced by* s_j*.*

An object $n \in \mathscr{L}_\Sigma^{\mathrm{op}}$ intuitively represents an atomic formula in n distinct variables $v_1,\ldots,v_n$ from *Var* and morphisms are substitutions.

Definition 5. *The collection of atoms At based on a signature* Σ *is a presheaf* $At : \mathscr{L}_\Sigma^{op} \to$ **Poset**.

Definition 6. *The category* $\mathscr{K}_\Sigma$ *on a signature* Σ

objects *are pairs of natural numbers*

morphisms *for any two objects* $(m,m'),(n,n;') \in \mathscr{K}_\Sigma$*, the set* $\mathscr{K}_\Sigma[(m,m'),(n,n')]$ *consists of all pairs of n-tuples* $< t_1,\ldots,t_n >$ *of terms in variables* $v_1,\ldots,v_m$ *and* n'*-tuples of terms* $< t'_1,\ldots,t'_n >$*) on terms in variables* $v_1,\ldots,v'_m$*. The composition is defined by replacement per components.*

An object $(n,m) \in \mathscr{K}_\Sigma$ intuitively represents an atomic formula where variables v_1 to v_n are universally quantified and variables v'_1 to v'_m are existentially quantified, *i.e.* goals. Morphism are substitutions that respect quantifiers.

Proposition 1. $\mathscr{K}_\Sigma$ *is isomorphic to* $\mathscr{L}_\Sigma^{\mathrm{op}} \times \mathscr{L}_\Sigma^{\mathrm{op}}$.

Definition 7. *The collection of goals G based on signature* Σ *is* $G : \mathscr{K}_\Sigma \to$ **Poset**

Proposition 2. *There is a forgetfull functor* $U : G \to At$ *from G to At regarded as functor categories.*

Proof. □