A Framework for Unification using Powersets of Terms*

P. Eklund, M.A. Galán
Dept of Computing Science
Umeå University
SE-901 87 Umeå, Sweden
{peklund,magalan}@cs.umu.se

J. Medina, M. Ojeda-Aciego, A. Valverde
Dept Matemática Aplicada †
Universidad de Málaga
E-29071 Málaga, Spain
{jmedina,aciego,a_valverde}@ctima.uma.es

Abstract

Many-valued logic programming with generalised terms requires an extended notion of unification in order to handle powersets of terms. In this paper we present substitutions and unifiers in a categorical framework based on powersets of terms as monads. We build upon developments for monad compositions initiated in [5].

1 Introduction

Extensions of logic programming using classical sets of terms, and involving only truth values, lead mostly to lattice-theoretic considerations. Much work has been done in these directions, and developments build upon achievements in many-valued extensions of corresponding propositional calculi [9]. On the other hand, extensions allowing for the use of generalised terms, in particular in form of various powersets of terms, has drawn much less attention as this leads to complications with respect to unification. However, some work has been done, and a typical first step is to allow only for constants to be used in term sets [2, 8].

In this paper we show how to use powersets of terms in their full range, i.e. allowing also to use functions in the operator domain. In order to allow for this, we propose a categorical framework for handling substitutions and unifiers. A categorical approach provides not only a well-founded formalism but also reveals properties of powersets of terms required e.g. for composing substitutions.

Generalised terms based on monad compositions and used in a unification framework require inventiveness concerning the provision of monad compositions. We build upon previous work on monad compositions where we have investigated conditions [5] under which compositions of monads again produce monads. In [6] we showed how composite expressions involving natural transformations could be pictorially represented in order to provide graphical proof support for providing monad compositions. We have also seen how new monads can be constructed from old ones, e.g. by using techniques given by submonads [7]. This, in fact, led to a more formal concept of generalised terms but, in this paper we take a more informal approach in that we exemplify generalised terms, in particular, with conventional powersets of terms, and terms over conventional powersets.

The paper is organised as follows. In Section 2, we provide required definitions and notations of the categorical framework. Section 3 introduces the concept of similarity, and some examples are given for selected functors: the term functor, the powerset term functor and the term powerset functor. Later, in Section 4 the definitions of variable substitutions and unifier are given for generalised terms. Finally, in Section 5 some conclusions and pointers to related work are presented.

*This work has been developed as a cooperation organised within COST 274.

† Partially supported by Spanish DGI project BFM2000-1054-C02-02

2 Definitions and notations

Moving from conventional terms to a categorical framework involving powersets of terms calls for a level of formalism required to fully make use of the categorical machinery.

It is not an easy task to identify situations in which the underlying behavior corresponds to a precise categorical concept. For instance, note that in the classical case, terms over terms are flattened to terms.

This is possible due to a "flattening" operator which embeds terms over terms into terms. The existence of such an operator is not obvious, and usually not credited to the term monad, as done in [11]. In this section we introduce the necessary requirements from category theory that will be used in the sequel.

A monad (or triple, or algebraic theory) over a category $\mathtt{C}$ is written as $\mathbf{\Phi} = (\Phi, \eta, \mu)$, where $\Phi\colon \mathtt{C} \to \mathtt{C}$ is a (covariant) functor, and $\eta\colon id \to \Phi$ and $\mu\colon \Phi \circ \Phi \to \Phi$ are natural transformations, respectively called unit and multiplication, for which $\mu \circ \Phi\mu = \mu \circ \mu\Phi$ and $\mu \circ \Phi\eta = \mu \circ \eta\Phi = id_\Phi$ hold. The Kleisli category $\mathtt{C}_{\mathbf{\Phi}}$ for $\mathbf{\Phi}$ over $\mathtt{C}$ consists of objects in $\mathtt{C}$, and the morphisms are given by $hom_{\mathtt{C}}(X, \Phi Y)$. See [1, 3] for category theoretic notions.

Let **2** be the usual covariant powerset monad $(2, \eta, \mu)$, where $2X$ is the set of subsets of X, $\eta_X(x) = \{x\}$ and $\mu_X(\mathcal{B}) = \bigcup \mathcal{B}$.

The term functor T_Ω, or T for short, with TX being the set of terms over the operator domain Ω and the variable set X, is extended to a monad in the usual way. A strict categorical notation was adopted in [5]. A term $\omega(m_1, \ldots, m_n)$ is in this paper more formally written as $(n, \omega, (m_i)_{i \le n})$.

As stated in the introduction, we will focus essentially on powersets of terms, in particular, we will be concerned with the composed functor $2T$.

To define the natural transformations associated to the composed functor $2T$, our monad construction makes use of the mapping $\sigma_X\colon T2X \to 2TX$, called the *swapper*. Functor compositions require the utility of such a swapping natural transformation in order to arrive at suitable multiplications for the functor composition. Functor compositions being extendable to monads are usually subject to conditions related to this swapper. In [4], a set of such conditions, or distributive laws, were given.

The swapper for the composed functor $2T$ is recursively defined by $\sigma_{X|2X} = id_{2X}$, and by

$$\sigma_X(l) = \{(n, \omega, (m_i)_{i \le n}) \mid m_i \in \sigma_X(l_i)\}.$$

otherwise.

The natural transformation $\eta^{2T}\colon id \to 2T$, defined as $\eta^{2T}_X(x) = \{x\}$, and the natural transformation $\mu^{2T}\colon 2T2T \to 2T$ defined for $R = \{(n_j, \omega_j, (r_{ij})_{i \le n_j}) \mid j \in J\} \in 2T2TX$ as

$$\{(n_j, \omega_j, (m_{ij})_{i \le n_j}) \mid j \in J, m_{ij} \in \sigma_{TX}(r_{ij})\}$$

provide $2T$ with the structure of a monad. See [5] for details.

3 Similarities

There are several references in the literature that include concepts as either fuzzy equality relation, or fuzzy equivalence relation, or similarity relation. We will adopt the latter terminology which is defined below.

For a formal treatment of similarities and equalities used in many-valued predicate logics, see [9].

In the rest of the section, L will denote a completely distributive lattice. Proofs omitted are included in an extended version of this paper.

Definition 3.1 *A similarity on X is a mapping $E\colon X \times X \to L$ satisfying the following axioms,*

$$E(x, x) = 1 \qquad \textit{(reflexivity)}$$

$$E(x, y) = E(y, x) \qquad \textit{(symmetry)}$$

$$E(x, y) \wedge E(y, z) \le E(x, z) \qquad \textit{(transitivity)}$$

for all $x, y, z \in X$.

Let Ω be a set of operations, and let E_Ω be a similarity on Ω. In the following we will use E_Ω in order to define a similarity on TX.

Definition 3.2 *The relation*

$$E_T \colon TX \times TX \to L$$

is defined as follows: For $x_1, x_2 \in X$,

$$E_T(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 = x_2 \\ 0 & \text{otherwise} \end{cases}$$

and further, for terms $t = \big(n, \omega, (m_i)_{i\leq n}\big)$, and $t' = \big(n', \omega', (m'_i)_{i\leq n'}\big)$ we have $E_T(t, t')$

$$\begin{cases} E_\Omega(\omega, \omega') \wedge \bigwedge_{i\leq n} E_T(m_i, m'_i) & \text{if } n = n' \\ 0 & \text{otherwise} \end{cases}$$

Proposition 3.1 *E_T is a similarity on TX.*

For unification we will need a similarity between powersets of terms, and for this purpose we will now use E_T in order to define a similarity on $2TX$.

Definition 3.3 *The relation*

$$E_{2T} \colon 2TX \times 2TX \to L$$

is defined for all $M_1, M_2 \in 2TX$ as,

$$E_{2T}(M_1, M_2) = \bigwedge_{m_1 \in M_1} \bigvee_{m_2 \in M_2} E_T(m_1, m_2) \wedge \bigwedge_{m_2 \in M_2} \bigvee_{m_1 \in M_1} E_T(m_1, m_2)$$

This definition can be seen as a *bi-implication measure*, in the sense of [10].

Proposition 3.2 *The relation E_{2T} is a similarity on $2TX$.*

Remark 3.1 *The choice of a similarity on $2TX$ as in Definition 3.3 is not obvious. In [8], the following pseudosimilarity was used:*

$$E'(M_1, M_2) = \bigwedge_{m_1, m_2 \in M_1 \cup M_2} E'_T(m_1, m_2).$$

Obviously, E' is not reflexive.

A similarity on $T2X$ can now easily be given using the similarity on $2TX$.

Definition 3.4 *The relation*

$$E_{T2} \colon T2X \times T2X \to L$$

is defined as follows:

$$E_{T2}(l_1, l_2) = E_{2T}\Big(\sigma_X(l_1), \sigma_X(l_2)\Big)$$

where $\sigma_X \colon T2X \to 2TX$ is the swapper.

Proposition 3.3 *The relation E_{T2} is a similarity on $T2X$.*

Remark 3.2 *It is an open question whether the functor composition $T2$ can be extended to a monad.*

4 Unifiers

In this section we introduce the concepts of variable substitutions and unifiers in particular within the context of powersets of terms. In the classical situation, variable substitutions are mappings assigning variables to terms, i.e. mappings $\theta \colon X \to TY$. For powersets of terms, a variable substitution should then be viewed as mappings

$$\theta \colon X \to 2TY$$

Given a generalised term $M \in 2TX$ in form of a powerset of terms, the result $M\theta$ of applying a variable substition θ on M is given by

$$M\theta = (\mu_Y^{2T} \circ 2T\theta)(M)$$

i.e. $M\theta$ is kind of a flattening of a set of terms over sets of terms, where μ_Y^{2T} provides the flattening operation.

Note that variable substitutions can be defined more generally over monads (F, η^F, μ^F). Indeed for an object $A \in FX$, and a variable substitution $\theta \colon X \to FY$, we will have

$$A\theta = (\mu_Y^F \circ F\theta)(A)$$

Definition 4.1 *The composition of two substitutions $\theta_1 \colon X \to 2TY$ and $\theta_2 \colon Y \to 2TZ$ is given by*

$$\theta_1\theta_2 = \mu_Z^{2T} \circ 2T\theta_2 \circ \theta_1$$

i.e. the composition in the Kleisli category $\mathtt{Set}_{\mathbf{2T}}$ *for the powerset monad* **2T** *over the category of sets.*

Given $M_1, M_2 \in 2TX$, let $[M_1; M_2]$ represent an *equation* over $2TX$. In order to define generalised unifiers for equations we will assume the existence of similarities.

Definition 4.2 *A unifier of the equation* $[M_1; M_2]$ *over* $2TX$ *is a substitution,* $\theta\colon X \to 2TY$*, such that* $E_{2T}(M_1\theta, M_2\theta)$ *equals*

$$\sup\{E_{2T}(M_1\vartheta, M_2\vartheta) \mid \vartheta \text{ is a substitution}\}.$$

It might be possible that the supremum above could not be attained by any substitution. The particular features of the lattice or the underlying application might require a weaker version of the definition, as follows:

Let θ be a substitution, and $[M_1; M_2]$ an equation over $2TX$. We say that θ is a unifier if $E_{2T}(M_1, M_2) \leq E_{2T}(M_1\theta, M_2\theta)$, that is, if the substitution increases the similarity degree.

5 Conclusions and future work

We have shown how generalised terms, as given by powersets of terms, can be handled in equational settings involving substitutions and unifiers. The utility of categorical techniques as provided by monads is obvious and indeed encouraging for further investigations on more elaborate compositions and categorical techniques for unification as initiated in [11].

In further work it is important to merge our efforts with developments, such as in [2], that have focused more on semantic aspects of many-valued logic programming. These developments have a rather specialised use of terms as they typically restrict to using powersets of constants instead of generalised terms in their full range. However, restricting to powersets of constants seems more to be a struggle with unification than with proof procedural issues, and there are no indications that the specialised use of terms is enforced by the semantic developments. The procedural issues being important it is equally worthwhile to underline the importance of further studies on monad compositions.

References

[1] J. Adámek, H. Herrlich, G. Strecker, *Abstract and Concrete Categories*, John Wiley & Sons, 1990.

[2] T. Alsinet, L. Godo, *A complete calculus for possibilistic logic programming with propositional variables*, Proc. Uncertainty in AI conference, 2000, pp. 1–10.

[3] M. Barr, C. Wells, *Toposes, Triples and Theories*, Springer-Verlag, 1985.

[4] J. Beck, *Distributive laws*, Seminars on Triples and Categorical Homology Theory, 1966/67, Lect. Notes in Mathematics 80, pp. 119–140. Springer, 1969.

[5] P. Eklund, M.A. Galán, M. Ojeda-Aciego, A. Valverde, *Set functors and generalised terms*, Proc. IPMU 2000, 8th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference, 2000, 1595-1599.

[6] P. Eklund, M.A. Galán, J. Medina, M. Ojeda Aciego, A. Valverde, *A graphical approach to monad composition*, Electronic Notes in Theoretical Computer Science **40** (2001). (www.elsevier.nl/locate/entcs/volume40.html)

[7] P. Eklund, M.A. Galán, J. Medina, M. Ojeda-Aciego, A. Valverde, *Composing submonads*, Proc. 31st IEEE Int. Symposium on Multiple-Valued Logic (ISMVL 2001), 2001, 367-372.

[8] F. Formato, G. Gerla, M.I. Sessa, Similarity-based unification, *Fundamenta Informaticae* 40 (2000), 393-414.

[9] P. Hájek, *Metamathematics of Fuzzy Logic*, Kluwer Academic Publishers, 1998.

[10] Ruspini, E., On the semantics of fuzzy logic, *International Journal of Approximate Reasoning* 5, pp. 45-88, 1991.

[11] D.E. Rydeheard, R.M. Burstall, *A categorical unification algorithm*, Proc. Category Theory and Computer Programming, 1985, LNCS 240, Springer-Verlag, 1986, 493-505.