Relaxed Unification – Proposal

Tony Abou-Assaleh and Nick Cercone

School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada
{*tabouassaleh,ncercone*}*@uwaterloo.ca*

Abstract

Classical unification is strict in the sense that it requires a perfect agreement between the terms being unified. In practice, data is seldom error-free and can contain incorrect information. Classical unification fails when the data is imperfect. Fuzzy unification partially addresses uncertainty but requires the creation of fuzzy databases. We propose *relaxed unification* as a new theory that relaxes the constraints of classical unification without requiring special preprocessing of data. The goal of relaxed unification is to tolerate possible errors and inconsistencies in the data and facilitate reasoning under uncertainty.

Key words: relaxed unification, imperfect data, uncertain reasoning

1 Introduction

The classical unification function takes two terms as input and produces a boolean value indicating whether the unification can be performed successfully. In case of a result of true, the function also returns a substitution that unifies these two terms. The unification fails if the same feature is assigned different values in the objects being unified. This process places rigid constrains on the data requiring it to be correct and consistent. Since real-world data is seldom perfect, the classical unification fails at the encounter of the slightest error. Erroneous data often contains enough information that one can exploit to overcome the errors. In other cases, it is possible to draw approximate or uncertain conclusions.

Fuzzy unification is useful in situations when the exact values of some attributes is not known. To benefit from fuzzy unification, the uncertain values in the data must be described using approximate qualitative attribute values.

Such data is normally stored in fuzzy databases. However, fuzzy unification still requires consistency between the approximate values. A single erroneous instance in the data set causes the whole unification process to fail.

Relaxed unification provides a method for extracting information from imperfect data. To achieve this functionality, we relax the strict true/false result of classical unification and replace it by a real number in the range $[0, 1]$ that indicates the correctness of the unification. A correctness value of 1 would represent a success under the classical unification; any other value would represent a failure. Relaxed unification does not includes a notion of failure; the unification always succeeds and returns a substitution.

2 Related Work

Unification was first introduced by Robinson in 1965 [1]. He presented a straightforward recursive unification algorithm that was refined by Martelli and Montanari [2]. Knight [3] provided an extensive survey of representations, algorithms, and applications of unification. More recently, Keselj [4] introduced an efficient graph unification algorithm along with its implementation. Classical unification forms the basis of logic programming. Its applications range from theorem proving, to expert systems, to natural language processing. Although classical unification has witnessed a great success, its assumption that the data is absolutely true isolated it from any real-world problems that involve uncertainty.

Lee was one of the leading figures who introduced uncertainty into the realm of logic programming [5]. His notion of fuzzy resolution, which was based on the fuzzy set theory [6], has opened the door for fuzzy logic and fuzzy unification theory. Significant work in advancing fuzzy logic was done by Mukaidono [7] amd Baldwin [8].

3 Theory

We first give a concise definition of classical unification and then extend the definition into relaxed unification. The qualifiers *classical* and *relaxed* are omitted when the context clearly dictates the intended meaning.

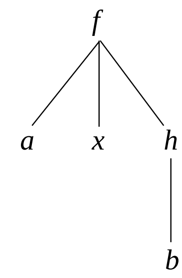

Fig. 1. Example of a tree representation of a classical term

3.1 Classical Unification

Definition 1 (Classical Term) *Let F be a countably infinite set of functions f, where f could be nullary. Let V be an infinite set of variables x. We construct a term algebra $T(F, V)$ of terms t. Each term is either a variable or a function of terms.*

Graph representation is often used when manipulating terms. In many cases, a simple tree representation is sufficient and provides a one-to-one mapping between the symbolic expression and the graph. Figure 1 illustrates the tree representation of the term $t = f(a, x, h(b))$.

Definition 2 (Classical Substitution) *A substitution σ is a mapping from variables to terms, e.g. $\sigma = \{x \mapsto h(a)\}$. The application of a substitution σ to a term t, denoted as $t\sigma$, is defined as,*

$$t\sigma = \left\{ \begin{array}{ll} u & \text{if } x = t \text{ and } x \mapsto u \in \sigma \\ f(t_1\sigma, \ldots, t_n\sigma) & \text{if } t = f(t_1, \ldots, t_n) \\ t & \text{otherwise} \end{array} \right\} .$$

Definition 3 (Classical Unification) *Two terms, t and u, are unifiable if and only if there exists a unifying substitution σ such that $t\sigma = u\sigma$. The term resulting from the unification is denoted $t \sqcup u$.*

3.2 Relaxed Unification

Definition 4 (Relaxed term) *Let F be a countably infinite set of functions f, where f could be nullary. Let V be an infinite set of variables x. We construct a term algebra $T(S_F, V)$ of terms t, where S_F is a set of all subsets of F. Each term is either a variable or a set of functions of terms.*

A relaxed term is represented as a directed rooted graph. A relaxed term can contain shared nodes, which are represented symbolically by suffixing a boxed

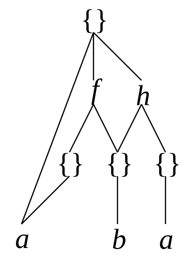

Fig. 2. Example of a graph representation of a relaxed term

index to the shared terms. For instance, the term

$$t = \{a\boxed{1}, f(\{a\boxed{1}\}, \{b\}\boxed{2}), h(\{b\}\boxed{2}, \{a\})\}$$

corresponds to the graph in figure 2. Edges from sets to functions are labelled with the function name and edges from functions to sets are labelled with the parameter index (not shown in the figure). Variables are represented in the graph as empty sets. For each variable x, a mapping from x to the corresponding node, $u = \{\}$ is added to the substitution.

Definition 5 (Path) *A path to a node u, denoted as $\pi(u)$, is a sequence of edges that connects the root of the graph to u. The set of all paths from the root to u is denoted as $\Pi(u)$.*

Definition 6 (Relaxed Substitution) *A substitution σ is a mapping from variables to nodes and from nodes to graphs representing terms. The application of a substitution σ to a term t, denoted as $t\sigma$, is defined as*

$$t\sigma = \left\{ \begin{array}{ll} u & \textit{if } x = t \textit{ and } x \mapsto u \in \sigma \\ u & \textit{if } \Pi(v) \cap \Pi(t) \neq \phi \textit{ and } v \mapsto u \in \sigma \\ f(t_1\sigma, \ldots, t_n\sigma) & \textit{if } t = f(t_1, \ldots, t_n) \\ \{t_1\sigma, \ldots, t_n\sigma\} & \textit{if } t = \{t_1, \ldots, t_n\} \\ t & \textit{otherwise} \end{array} \right\} .$$

Definition 7 (Relaxed Unification) *Two terms, t and u, are always unifiable with a unifying substitution σ such that $t\sigma = u\sigma$.*

A high-level algorithm for the function RelaxUnify is provided below. It takes the roots of the terms to be unified as arguments u and v and returns the root of the unified graph t and a substitution σ.

global σ : Substitution := ϕ
function Unify(u : Node, v : Node) $\Rightarrow$ (t : Node, σ : Substitution)
begin

if $u = \{u_1, u_2, \ldots, u_n\}$ **and** $v = \{v_1, v_2, \ldots, v_m\}$ for $n, m \geq 0$ **then**
  $t :=$ merge u and v unifying edges with identical labels
  if $t \neq u$ **then** $\sigma := \sigma \cup \{u \mapsto t\}$
  if $t \neq v$ **then** $\sigma := \sigma \cup \{v \mapsto t\}$
else
  $u = f(u_1, u_2, \ldots, u_n)$, $v = f(v_1, v_2, \ldots, v_n)$ for $n \geq 0$
  $t := f(t_1, t_2, \ldots, t_n)$ where $t_i := \text{Unify}(u_i, v_i)$
end if
return (t, σ)
end

An external user-defined evaluation function is used to determine the correctness value of the resulting term. This function can be as simple as the number of elementary sets divided by the number of none-empty sets in the term. A smoother and slightly more sophisticated function is to compute the average of the inverse of the size of none-empty sets.

4 Application

A detective is investigating a crime. He has a list of suspects stored in a knowledgebase using the format of

$$\textit{suspect}(Name, Sex, Age, Eye\_Colour, Hair\_Colour, Height) \ .$$

Let the knowledgebase contain the information below.

$$\textit{suspect}(\{\text{“\textit{John}”}\}, \{\textit{male}\}, \{\textit{old}\}, \{\textit{black}\}, \{\textit{dark}\}, \{\textit{tall}\})$$
$$\textit{suspect}(\{\text{“\textit{Alice}”}\}, \{female\}, \{\textit{young}\}, \{\textit{green}\}, \{\textit{dark}\}, \{\textit{short}\})$$
$$\textit{suspect}(\{\text{“\textit{Mike}”}\}, \{\textit{male}\}, \{\textit{young}\}, \{\textit{brown}\}, \{\textit{light}\}, \{\textit{short}\})$$

An eyewitness describes the person who committed the crime as being a tall young man with brown eyes and light hair. The query that incorporates this information can be expressed as

$$?\ \textit{suspect}(X, \{\textit{male}\}, \{\textit{young}\}, \{\textit{brown}\}, \{\textit{dark}\}, \{\textit{short}\}) \ ,$$

where X is a variable that we would like to instantiate. Querying the knowledgebase using classical unification returns a failure. None of the stored suspects matches the witness' description. However, the detective knows that eyewitnesses can make mistakes and their data is not reliable. He wishes to determine which suspect best fits the description. The answer is easily found using relaxed unification. We relax unify the query with each item in the knowledgebase, which produces the following terms.

$$suspect(\{\text{“}John\text{”}\}, \{male\}, \{young, old\}, \{black, brown\}, \{dark\},$$
$$\{tall, short\})$$
$$suspect(\{\text{“}Alice\text{”}\}, \{female, male\}, \{young\}, \{green, brown\}, \{dark\},$$
$$\{short\})$$
$$suspect(\{\text{“}Mike\text{”}\}, \{male\}, \{young\}, \{brown\}, \{light, dark\}, \{short\})$$

Let the correctness function be the number of elementary sets in the term divided by the number of none-empty sets. John receives a score of 3/6; Alice receives 4/6; and Mike receives 5/6. Since Mike is the closest match, the detective concludes that Mike is the prime suspect.

In the example, we treated mismatches in different attributes as being equally important. In practice, it is often desirable to assign different weights to attributes. We intend to investigate weighted and probabilistic relaxed unification in our future work.

5 Conclusion

We present a novel unification method that we call relaxed unification. This method targets problems where the data can be inaccurate or uncertain and attempts to find the best matching term as opposed to a perfect fit. Any system that uses unification on imperfect domains can benefit from this approach. In particular, this theory was design to overcome limitations in unification grammars that are used for natural language processing.

6 Future Work

We are currently working on the representation, algorithm, and correctness function. These three elements are crucial to the development of a fully functional system that uses relaxed unification.

Our next step in developing the theory is assigning weights to features and probabilities to values, which would allow for a more accurate data representation and manipulation.

References

[1] J.A. Robinson, A Machine-Oriented Logic Based on the Resolution Principle, *J. ACM* **12** (1) 23-41 (1965).

[2] A. Martelli and U. Montanari, An Efficient Unification Algorithm, *ACM Trans. on Programming Languages and Systems* **4** (2) 258-282 (1982).

[3] K. Knight, Unification: A Multidisciplinary Survey, *ACM Computing Surveys* **21** (1) 93-124 (1989).

[4] V. Keselj and N. Cercone, A Graph Unification Machine for NL Parsing, *Technical Report CS-2002-01*, Department of Computer Science, University of Waterloo, Ontario, Canada, (2002).

[5] R.C.T. Lee, Fuzzy Logic and the Resolution Principle, *J. ACM* **19** (1) 109 - 119 (1972).

[6] H.-J. Zimmermann, *Fuzzy Set Theory and its Applications*, Kluwer Academic, Boston (1996).

[7] M. Mukaidono, Fuzzy Inference of Resolution Style, In *Fuzzy Set and Possibility Theory*, (Edited by R.R Yager), pp. 224-231, Pergamon Press, (1982).

[8] J.F. Baldwin, Evidential Support Logic Programming, *Fuzzy Sets and Systems* **24** 1-26 (1987).