Parallel Retrieval from Case Bases

Hugh Osborne and Derek Bridge

Department of Computer Science
University of York
Heslington
York YO1 5DD

Abstract. A method for constructing similarity measures for case based reasoning systems from similarity measures for component features of the cases in the case base is presented. Disjunctive normal forms can be derived for these similarity measures, and cases retrieved by applying the conjunctions from this normal form in parallel. A system that graphically demonstrates the techniques involved is presented. of parallel retrieval is presented.

1 Case Memory Systems

This paper presents a method for parallel retrieval from case bases. A case base will be considered to consist of a case memory and a retrieval mechanism. The case memory will be modelled as a finite subset of a set, Θ, of cases, or tuples, equiped with projection functions for accessing the component elements of these cases. A retrieval request is presented to the system as a pair, consisting of an element, ϑ, of Θ and a similarity measure, σ. The case ϑ, known as the *probe*, represents the "best possible" case. The similarity measure is a function from cases to partial orders over cases, $\sigma :: \Theta \to \Theta \to \Theta \to \mathbf{bool}$. Application of σ to ϑ gives a partial order over Θ which can be used to deliver the maxima from the case memory — the "best fits".

Normally, where cases are description-report pairs, one might expect a probe to be a description, rather than a case. There are, however, situations in which one might like to compare reports in order to discriminate between cases — for example when cosideration of the descriptions alone has failed to discriminate sufficiently between cases. It is for this reason that the probe is taken to be a case. If only descriptions are to be compared σ will be designed to ignore reports. A description-report pair in which both the description and the report are tuples, can, without loss of generality, be seen as a tuple containing the elements of the description and the report. The description and the report can be recovered by the application of suitable projection functions.

This article will concentrate on the similarity measure σ — in particular how to construct similarity measures, and then analyse them in order to enable some degree of parallel evaluation in retrieving "best" cases from the case memory system.

Section 2 will present a means of constructing similarity measures; section 2.1 introducing a repertoire of atomic similarity measures (intended for application to component elements of cases), and section 2.2 providing operations for combining simple similarity measures to form more complex measures. In particular, section 2.3 discusses preferences.

Section **??** discusses retrieval from a case memory system using similarity measures, and shows how this can be done in parallel. Finally, section 3 presents an implementation of a demonstration system for parallel retrieval.

The paper is illustrated throughout by the following example (based on one by Ryan [**?**]):

Example 1 (a)

Consider the problem of providing a guest with a meal. A meal can be characterised by its name, a list of ingredients (in this case just "meat"), some indication of its degree of spiciness, the number of calories it contains, etc., etc.

dish:: (**string**,		*the name of the dish*
	bool,	*does it contain meat*
	spiciness,	*how spicy is it*
	num)	*number of calories*

spiciness::= mild| medium mild| medium|
medium hot| hot| extra hot| killer| suicide

The projections π_{meat}, $\pi_{\text{spiciness}}$ and π_{calories} will select the relevant elements of the list.

Assume that you know how to cook lamb casserole, vegetable biryani and chicken vindaloo, and pasta with tomato chilli sauce. These are specified by the following four cases.

name	*meat*	*spiciness*	*calories*	
("lamb",	True,	mild,	634	),
("vegetable biryani",	False,	hot,	843	),
("chicken vindaloo",	True,	extra hot,	716	),
("pasta",	False,	medium hot,	716	)

Given a guest, called "Mark", who is vegetarian, likes his food medium hot, and is watching his weight a probe can be defined expressing these wishes.

$$\mathsf{mark} = (\text{"Mark"}, \text{False}, \mathsf{medium\ hot}, 0)$$

Cases are then retrieved from the case base by applying a similarity measure to the ideal, giving an ordering over cases, and then taking the maxima of the case base according to this ordering.

---

2 Constructing Similarity Measures

This section presents a repertoire of comparison operations, or similarity measures, and (binary) operations on these similarity measures used to construct new similarity measures.

2.1 Atomic Similarity Measures

Some standard similarity measures are defined below, and illustrated by an example applied to the set $\{1,2,3,4,5\}$, with the usual total order $\leq$.

flat: No elements are related (returns the flat partial order).

$$x\ (\mathsf{flat}\ e)\ y = (x = y)$$

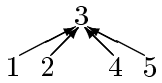
flat 3

1 2 3 4 5

is: Chooses some particular element as being better than all others.

$$x\ (\mathsf{is}\ e)\ y = (x = e) \vee (x = y)$$

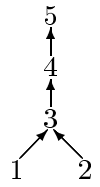
is 3



greater: The given element is a minimum requirement, and anything greater than that is better.

$$x\ (\mathsf{greater}\ e)\ y = (x \geq y \wedge x \geq e) \vee (x = y)$$

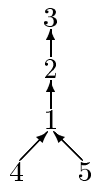
greater 3



less: The given element is a maximum requirement.

$$x\ (\mathsf{less}\ e)\ y = (x \leq y \wedge x \leq e) \vee (x = y)$$

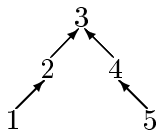
less 3



best: The given element is the best ("break the back" of the underlying total order at this element).

$$x\ (\mathsf{best}\ e)\ y = x \leq y \leq e \vee x \geq y \geq e$$

best 3



Furthermore the inverse function is defined for similarities, where the inverse of a similarity σ returns the inverse of the order returned by σ.

$$\sigma^{-1}\ e\ x\ y = \sigma\ e\ y\ x$$

Example 1 (b)

Recall that the probe for mark was

$$\mathsf{mark} = (\text{"Mark"}, \mathsf{medium\ hot}, 0).$$

Assuming that spiciness is ordered from least spicy to most spicy, Mark's preferences are given by: π_{meat} is, $\pi_{\text{spiciness}}$ best and π_{calories} $\mathsf{greater}^{-1}$, all applied to mark.

The orders generated on the meals in the case base by each of these atomic measures are then:

π_{meat} is mark

vegetable biryani, pasta

↑

chicken vindaloo, lamb casserole

$\pi_{\text{spiciness}}$ best mark

pasta

↗ ↖

lamb casserole vegetable biryani

↑

chicken vindaloo

and

$$\frac{\pi_{\text{calories}}\ \mathsf{greater}^{-1}\ \mathsf{mark}}{\begin{array}{c}\text{lamb casserole}\\ \uparrow \\ \text{pasta, chicken vindaloo}\\ \uparrow \\ \text{vegetable biryani}\end{array}}$$

Note that no one member of the case base is best in all of these orderings. The following section will present a method of combining these orderings to arrive at new orderings which may give a (unique) best fit.

2.2 Connectives

The first obvious candidates for combining orderings are the usual boolean operators. These are presented below. This is followed by a discussion of determining the maxima of the new similarity measures obtained by application of these operators.

A second set of candidates will then be introduced. These are connectives expressing some form of priority — e.g. that Mark considers it to be more important that the meal is vegetarian than that it be medium hot.

Normally one would expect the priority connectives to require one computation of the maxima for each level of priority. It is however possible to reduce both the boolean operators and the priority connectives to a (disjunctive) normal form, which can then not only be computed in one pass, but also be evaluated in parallel. A summary of the method used concludes this section.

Boolean Operators. The usual boolean operators may be applied to similarity measures to form new similarity measures. This is done by "lifting" the point-wise boolean operators to operate on similarities. If $\oplus$ is a binary boolean operator then the lifted operator $\widehat{\oplus}$, acting on similarities σ_1 and σ_2, applied to probe p and cases x and y, is defined by

$$(\sigma_1 \mathbin{\widehat{\oplus}} \sigma_2)\ p\ x\ y = (\sigma_1\ p\ x\ y) \oplus (\sigma_2\ p\ x\ y).$$

Example 1 (c)

The order generated by $(\pi_{\text{meat}} \mathbin{\widehat{\wedge}} \pi^{-1}_{\text{greater}})$ when applied to the probe for Mark and the case memory given is:

$$\frac{(\pi_{\text{meat}} \mathbin{\widehat{\wedge}} \pi^{-1}_{\text{greater}})\ \mathsf{best}\ \mathsf{mark}}{\begin{array}{ccc}\text{pasta} & & \text{lamb casserole}\\ \uparrow & \nearrow & \uparrow \\ \text{vegetable biryani} & & \text{chicken vindaloo}\end{array}}$$

Determining Maxima. Note that if $\oplus$ is a boolean operator other than $\wedge$ then, even if $\sigma_1\ p$ and $\sigma_2\ p$ are partial orders, $(\sigma_1 \mathbin{\widehat{\oplus}} \sigma_2)\ p$ is not necessarily a partial order. It will however be a pre-order. This problem is addressed by extending the definition of maxima to cover pre-orders.

The maxima of a partial order $\sqsubset$ are usually defined as

Definition 1.

$$\sqcap\ \sqsubset\ S = \{x \in S | \forall y \in S : x \sqsubset y \Rightarrow x = y\}. \tag{1}$$

[maxdef]

Since, for a partial order, $(x \sqsubset y \Rightarrow x = y) \equiv (x \sqsubset y \Rightarrow y \sqsubset x)$ this is equivalent to

Definition 2.

$$\sqcap \sqsubset S = \{x \in S | \forall y \in S : x \sqsubset y \Rightarrow y \sqsubset x\} \quad (2)$$
[maxaltdef]

and this will be taken as the definition of the "maxima" of any relation, i.e. if $\oplus$ is a relation then

Definition 3.

$$\sqcap \oplus S = \{x \in S | \forall y \in S : x \oplus y \Rightarrow y \oplus x\} \quad (3)$$
[maxreldef]

which is equivalent to the definition of the maxima of the partial order $\sqsubseteq$ over the equivalence classes generated by $\sqsubseteq$.

This definition has the following properties:

$$\sqcap (\widehat{\frown} \oplus) = \sqcap \oplus^{-1}$$
$$\sqcap (\oplus^1 \mathbin{\widehat{\vee}} \oplus^2) \subseteq (\sqcap \oplus^1) \mathbin{\widehat{\cap}} (\sqcap \oplus^2)$$

Indeed, if $\oplus^1 \mathbin{\widehat{\vee}} \oplus^2$ is a partial order, then

$$\sqcap (\oplus^1 \mathbin{\widehat{\vee}} \oplus^2) = (\sqcap \oplus^1) \mathbin{\widehat{\cap}} (\sqcap \oplus^2)$$

2.3 Two More Connectives

Two more connectives will be useful, known as *preferences*. One similarity measure is prefered to another if the second is only applied to discriminate between maxima of the first. The notation $s_1 \gg s_2$ is used for "s_1 is prefered to s_2". An analogous notation is used for the inverse: $s_1 \ll s_2$, with $s_1 \ll s_2 \equiv s_2 \gg s_1$. The maxima of the preference of two relations are defined by:

Definition 4. If s_1 and s_2 are similarity measures, i is a probe, and S is a case base, then

$$\sqcap ((s_1 \gg s_2)\ i)\ S = \sqcap\ (s_2\ i)\ (\sqcap\ (s_1\ i)\ S)$$

The maxima of $((s_1 \gg s_2)\ i)$ are the maxima $s_1\ i\ S$, if there is no more than one maximum in $s_1\ i\ S$, and the maxima $s_2\ i\ S'$, where S' is the set of maxima $s_1\ i\ S$, otherwise.

This definition implies that two computations are necessary to determine $\sqcap\ (s_1 \gg s_2)\ i)\ S$, one to find $\sqcap\ (s_1\ i)\ S$, and a second applying $\sqcap\ (s_2\ i)$ to this result. However, $(s_1 \gg s_2)\ i$ can be expressed in terms of the standard boolean operators.

$$s_1 \gg s_2 = s_1 \mathbin{\widehat{\vee}} (s_1 \mathbin{\widehat{\wedge}} s_1^{-1} \mathbin{\widehat{\wedge}} s_2)$$

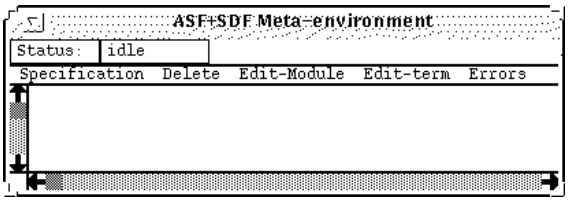
3 An Implementation

A demonstration prototype of the system described above has been implemented. The syntax of cases, probes and similarity measures was defined using the ASF+SDF [?] Meta-environment developed at the *Centrum voor Wiskunde en Informatica* and the University of Amsterdam, as was the transformation of similarity measures to disjunctive normal form. The disjunctive normal form is then passed to Gadget Gofer [?], a dialect of Gofer developed at the University of York for parallel programming of graphical user interfaces.

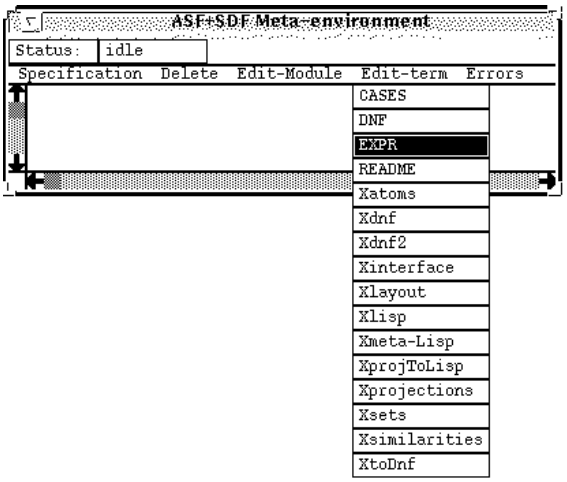
3.1 Calling the system

The system is called by `maxima` <*filename*>. A small window will pop up called "`Centaur menu`". Selecting "`ASF+SDF`" from the "`View`" pull-down menu will start the ASF+SDF system.

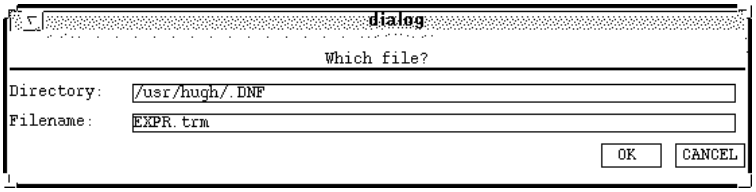
The ASF+SDF system will automatically load the modules defining the case base language. The root window of the system will appear.



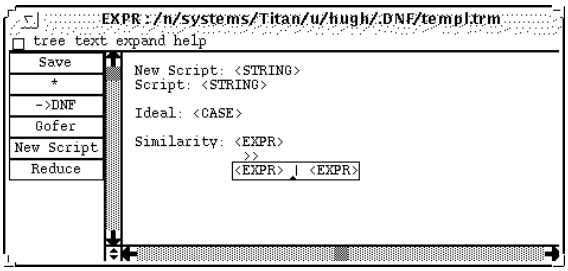
A probe and a similarity measure (together known as a "script") can be constructed by selecting "`EXPR`" in the pull-down "`EDIT-TERM`" menu.



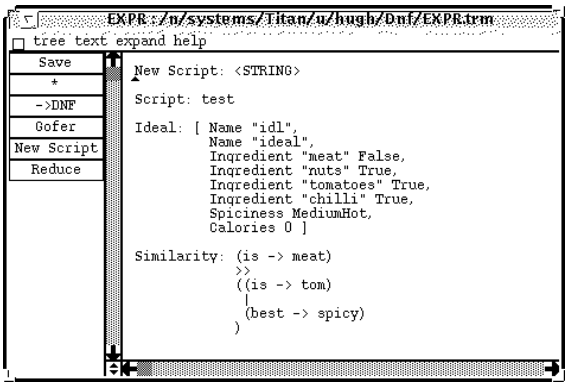
A pop-up box asks for confirmation of the choice.



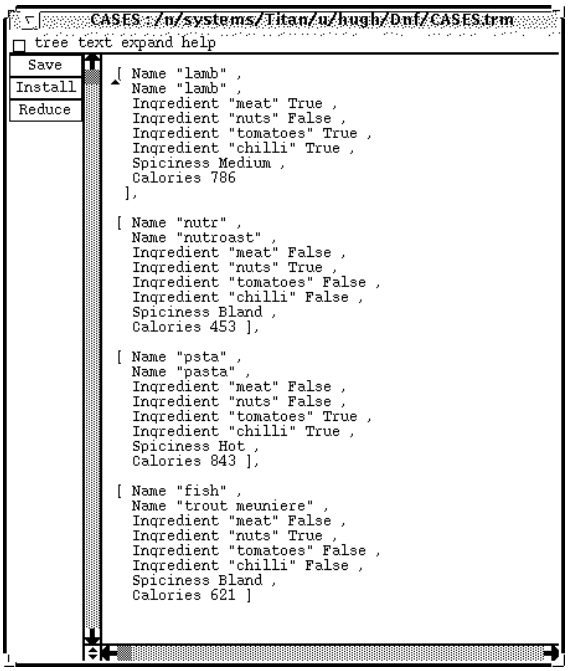
Upon confirmation, a syntax directed editor (a "term editor") for a script will be created. At some point in the development it could look like this.



A final specification could be:



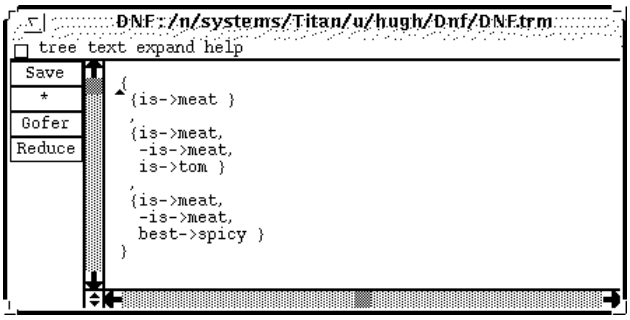
A term editor for the case memory can also be created by selecting "CASES" from the "EDIT-TERM" menu.



The case memory can be installed by first clicking on the "Save" button and then on the "Install" button.

The other two options in the "EDIT-TERM" menu are "README", which will pop up a window containing on-line documentation, and "DNF" which will pop up an editor for the most recent disjunctive normal form. The parallel retrieval mechanism can also be called from a disjunctive normal form term editor.

Once a script has been fully developed it can be installed by clicking on the "Save" button of the script term editor. If a case memory has also been installed the evaluation part of the system can now be called. This is done by clicking the "*" button of the "EXPR" term editor. The system will then compute the disjunctive normal form of the similarity measure provided (this will be available by way of the "DNF" option of the "EDIT-TERM" pull-down menu), and call Gadget Gofer. The disjunctive normal form of the example used here is (the minus sign indicates the inverse of the similarity measure):

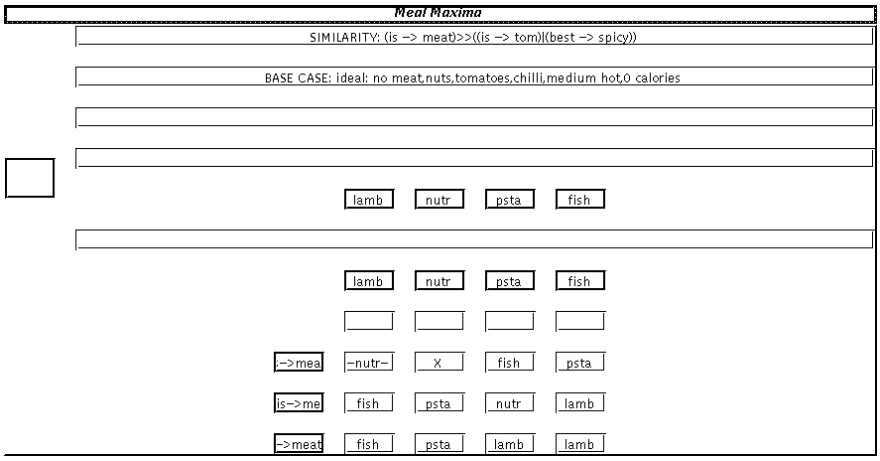


The "`->DNF`" button of the script term editor will compute the disjunctive normal form without calling Gadget Gofer. Similarly, if the disjunctive normal form has already been computed the "`Gofer`" button will start the parallel retrieval without recomputing the disjunctive normal form.

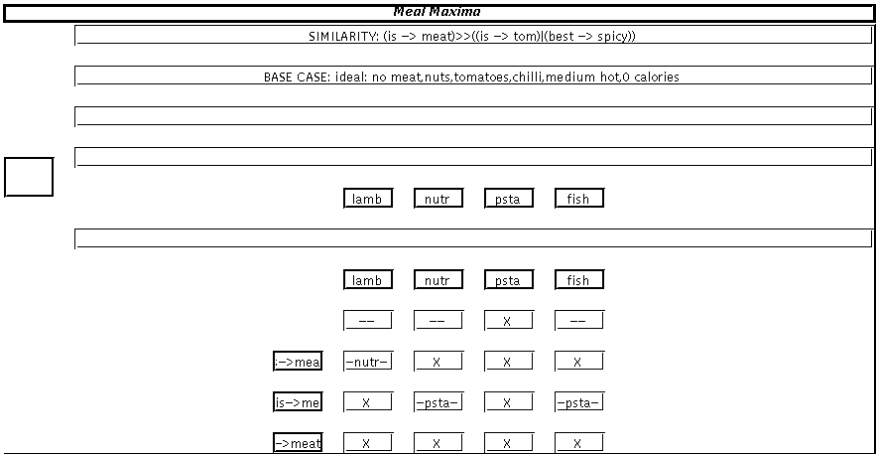
Gadget Gofer will create a window containing a demonstration of the parallel retrieval system. The top half of the window contains various explanatory sub-windows. The top two sub-windows show the similarity measure used, and the base case (probe) it is applied to. The grid in the bottom half is the demonstration of the system. The bottom four rows represent evaluation of the disjunctive normal form. The topmost of these rows will contain the final result of the evaluation. The remaining three rows shows the evaluation of each of the terms in the disjunctive normal form. Clicking on the left most button of one of these rows will cause the corresponding expression to be shown in the third explanatory sub-window. The expression being evaluated in the second row is, for example

$$\pi_{\mathrm{meat}}\ \mathsf{is} \mathbin{\widehat{\wedge}} \pi_{\mathrm{meat}}\ \mathsf{is}^{-1} \mathbin{\widehat{\wedge}} \pi_{\mathrm{tomato}}\ \mathsf{is}.$$

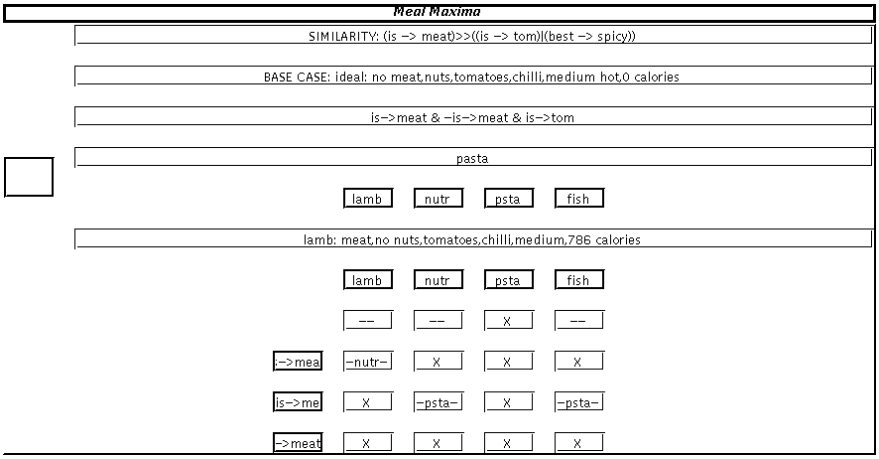
In the first of these rows the lamb casserole has already been eliminated as a candidate (since the nut roast contains no meat, and is therefore better). The nut roast has been selected as a possible candidate, since it is as good as, or better, than all the other cases. Evaluation is still in progress for all the other cases and terms.



The final result is obtained by taking the intersection of the results over all the terms. In this case the pasta has been selected as the best result.



The remaining two explanatory sub-windows give further information about the cases. Clicking once on one of the buttons immediately below these sub-windows will give the name of the case (here illustrated by "pasta"), clicking twice will give the full specification of the case (as shown here by the lamb casserole).



This article was processed using the LaTeX macro package with LLNCS style