Automatic Transformation and Enlargement of Similarity Models for Case-Based Reasoning[*]

Mirjam Minor[‡] and Karsten Schmidt[‡‡]

[‡] Universität Trier, Lehrstuhl für Wirtschaftsinformatik II
54286 Trier, Germany
minor@uni-trier.de

[‡‡] Universität Rostock, Institut für Informatik,
Albert-Einstein-Str.21, 18059 Rostock, Germany
karsten.schmidt@uni-rostock.de

1 Introduction

In this paper, we present a new approach how similarity models for Case-Based Reasoning can be extended by means of operators for relations. A second contribution of the relations approach is that the mathematical model can be transformed into an OWL notation as well as in the proprietary format that is needed to build a Case Retrieval Net[1]

Inside a case-based system, a case c is represented by a set of information entities $e_i \epsilon E$: $c = \{e_1, e_2, ..., e_n\}$. It is compared with a query q that is as well just a set of information entities. A *composite similarity function* for $q, c \subseteq E$ determines a numeric value for the degree of similarity $SIM(q, c)$. SIM is computed by means of a *local similarity function* $sim : E \times E \rightarrow \mathbb{R}$. A very simple example of SIM is the sum of the local values: $SIM(q, c) = \sum_{e_i \epsilon q} \sum_{e_j \epsilon c} sim(e_i, e_j)$.

To provide the values for sim, we need a *similarity model*. In small domains, this might be a table of manually specified similarity values for the pairs of information entities like, for instance, $sim(MPEG-4, XviD) = 0.3$, $sim(XviD, Videocodec) = 0.7$. In real world applications, the similarity model is often too complex to be clearly arranged in a table. We employ more abstract relations instead.

A *similarity type* is a binary relation between two information entities $S \subseteq E \times E$. Each similarity type has a qualitative or quantitative identifier, e.g. "IS_SUCCESSOR", "IS_ABSTRACTION", "LOW_SIMILARITY". A set $\mathfrak{S}$ of similarity types is called a *similarity dictionary*. A *weighting function* $g : \mathfrak{S} \rightarrow \mathbb{R}$ assigns a numeric value to a similarity

[*]This work has been partly funded by the BMBF project URANOS (no. 01M3075)

[1]Mario Lenz and Hans-Dieter Burkhard. Lazy Propagation in Case Retrieval Nets. In Wolfgang Wahlster, editor, 12th European Conf. on Artificial Intelligence (ECAI96), pages 127 131. John Wiley & Sons, 1996.

type.

2 Extending the similarity model by operators

Whether a pair of information entities belongs to a similarity type is empirically motivated. We define operators for relations with the aim to generate new relations that fulfill some formerly specified properties.

Definition (operators for relations)

An *operator* $op : S \mapsto S'$ *for a relation* $S \subseteq E \times E$ generates a new relation S'. We cover only some special cases of operators in the following. For further operators, we refer to future work:

> **Identity**: $id(S) = S$
>
> **Inverse**: $inv(S) = S^{-1}$
>
> **Transitive closure**: $t(S) =< S >$ (The transitive closure $< S >$ of a relation S is the smallest transitive relation including S.)
>
> **Symmetrical closure**: $s(S) = S \cup S^{-1}$
>
> **Enlargement by the inverse of a partner relation**: $e(S) = S \cup f(S)^{-1}$ ($f : \mathfrak{S} \to \mathfrak{S}$ is a function that assigns a partner relation to a relation.)
>
> **Induced brother relation**: $b(S) = (S^{-1} \circ S) \setminus I$ (I is the identity relation, i.e. $b(S) = \{[b, c] \mid b \neq c, \exists a \in E : [a, b], [a, c] \in S\}$)

t, s, and e are supposed to produce relations with the following properties: t extends a relation, e.g. the "IS_ABSTRACTION" relation, to a new transitive relation. s results in a symmetric relation. e can be applied to pairs of relations, e.g. the "IS_PART" and the "HAS_PART" relation, to generate a pair of real inverse relations.

Definition (Self-reflectivity) f is called *self-reflexive* if it holds: $f(S) = R$ iff $f(R) = S$.

Conclusion 1 Is f self-reflexive then holds: $e(S) = [e(f(S))]^{-1}$

The users specify which chains of operators will be applied to which relations (see Table 2). We consider only the eight special cases $\{te, tse, se, e, bte, btse, bse, be\}$.

Chain of operators ω_S

$todo : \mathfrak{S} \to \{te, tse, se, e\}$ is given by the user. We use the abbreviations $\omega_S = todo(S)$,

$$h_S = \begin{cases} t\ , & if\ \omega_S = te \\ ts, & if\ \omega_S = tse \\ s\ , & if\ \omega_S = se \\ id, & if\ \omega_S = e \end{cases}$$

Specification of f (to be used by the operator e)

Identifier	Weight g	h_S	pf	$\mathfrak{S}_{\mathcal{E}}$, Id.
NO_SIMILARITY	0,0	id	-	-
ID	1.0	id	-	-
ABSTRACTION	0.3	t	SPECIFICATION	+, BROTH
SPECIFICATION	0.7	t	ABSTRACTION	-
HAS_PART	0.5	t	IS_PART	-
IS_PART	0.5	t	HAS_PART	-
USES_AS_MEANS	0.3	t	IS_MEANS_FOR	+, MBROTH
IS_MEANS_FOR	0.7	t	USES_AS_MEANS	-
STRONG_SIM	0.75	id	-	-
WEAK_SIM	0.25	id	-	-
MEDIUM_SIM	0.5	id	-	-
BROTH	0.5	id	-	-
MBROTH	0.3	id	-	-

Table 1: Sample specification of the similarity types from the **ExperienceBook** project.

The user specifies the set of partner relations $\mathfrak{S}_{\mathcal{P}} \subseteq \mathfrak{S}$ and the partner function $pf : \mathfrak{S}_{\mathcal{P}} \rightarrow \mathfrak{S}_{\mathcal{P}}$. With those, the function f is built as follows:

$$f(S) = \begin{cases} pf(S), & if\ S \in \mathfrak{S}_{\mathcal{P}} \\ S^{-1} & , else. \end{cases}$$

Parent relations The set of parent relations $\mathfrak{S}_{\mathcal{E}} \subseteq \mathfrak{S}$ is specified by the user.

Definition (Enlargement of a similarity dictionary)

Be $\mathfrak{S}$, $\mathfrak{S}'$ two similarity dictionaries. The operator $Ext : \mathfrak{S} \mapsto \mathfrak{S}'$ is defined as:

$$\begin{aligned} Ext(\mathfrak{S}) = \quad & \{\omega_S(S) \mid S \in \mathfrak{S}\} \quad \cup \\ & \{b\omega_S(S) \mid S \in \mathfrak{S}_{\mathcal{E}}\} \end{aligned}$$

Definition (Consistency of ω_S with f) ω_S is *consistent with* f if holds: $\omega_S = \omega_{f(S)}$

Proposition (Properties of the relations generated by ω_S)

(i) $\omega_S \in \{\boldsymbol{t}e, \boldsymbol{t}s e\} \Rightarrow \omega_S(S)$ is transitive

(ii) $\omega_S \in \{\boldsymbol{t}s e, \boldsymbol{s}e\} \Rightarrow \omega_S(S)$ is symmetric

(iii) Under the conditions $S \in \mathfrak{S}_{\mathcal{P}}$, f self-reflexive, and ω_S consistent with f holds:
$\omega_S(S) = (\omega_{f(S)}(f(S)))^{-1}$

Lemma 1 (i) $t \circ inv = inv \circ t$ (ii) $s \circ inv = inv \circ s$ (iii) $ts \circ inv = inv \circ ts$

Proof of the proposition ad (i) und (ii): as one can see, ad(iii):

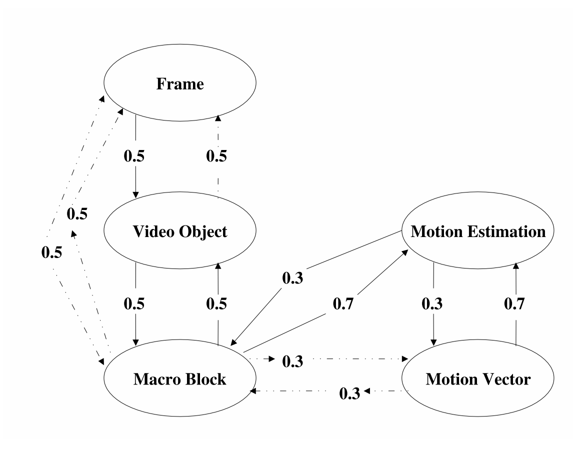

Figure 1: An automatically extended Case Retrieval Net (new and modified edges are dotted).

$$\begin{aligned}\omega_S(S) &= h_S \circ e(S) = h_S \circ (e(f(S)))^{-1} \text{ because of Conclusion 1}\\ &= h_{f(S)} \circ inv \circ e(f(S)) \text{ because } \omega_S \text{ is consistent with } f\\ &= inv(h_{f(S)} \circ e(f(S))) \text{ because of Lemma 1}\\ &= (\omega_{f(S)}(f(S)))^{-1}. \qquad \square\end{aligned}$$

The mathematical description of the similarity model has been transformed into two alternative representations in computer files:

1. A proprietary notation that is helpful to build a Case Retrieval Net (see Fig. 1) lists each pair of information entities explicitely in both directions.
2. An OWL notation can be visualized by ontological tools like Protégé.

The first benefit of the proposed approach is that we can import and export knowledge about similarity from and to domain ontologies in the classical sense. We have implemented this by means of Perl scripts. The danger of ontology pollution is obviously given, but the impacts on the case-based retrieval are relatively small: In case a single local similarity value is too strong or too weak, it is compensated by the other values in the composite similarity function. A critical mass of pollution is reached when an empirically matching case is squeezed out of the first positions of the ranking for a query by overestimated cases. It is future work to investigate empirically when this critical mass is accomplished.

The second and major benefit of our approach is that we can make the modeling process more comfortable for the knowledge engineers. Specifying a desired property of a relation like transitivity is less time-consuming than verifying manually whether all members of a family tree are properly connected. The “NO_SIMILARITY” relationship allows to model exceptions. Protégé is able to generate a visualisation of the similarity model.