An Algebraic Framework for Solving Proportional and Predictive Analogies

Ute Schmid (schmid@informatik.uni-osnabrueck.de)
Dept. of Mathematics/Computer Science, University Osnabrück
49069 Osnabrück, Germany

Helmar Gust (helmar@uos.de), **Kai-Uwe Kühnberger (kkuehnb@uos.de)**
Institute for Cognitive Science, University Osnabrück
49069 Osnabrück, Germany

Jochen Burghardt (jochen@first.fhg.de)
FIRST, 12489 Berlin, Germany

Abstract

We present an approach to analogical reasoning which is inherently dependent on abstraction. While typical cognitive and AI models of analogy perform a *direct* mapping from objects of the base to objects of the target domain, our model performs mapping *via abstraction*. Abstraction is calculated as most specific generalization of the base and the target structure. In contrast to existing models, learning occurs as a side-effect of analogical reasoning. Our approach is based on the formally sound framework of anti-unification. It allows to deal with different kinds of analogy in a uniform way. After a description of the basic ideas of the approach, we will present examples from the domains of proportional and predictive analogy.

Introduction

The ability to use old information to explain new experiences can be considered as a core feature of human intelligence. Therefore, it is not astonishing that analogy is an active area of cognitive science research. There are numerous psychological studies and experiments as well as different computational models of analogical reasoning.

The simplest form of analogical reasoning are proportional analogies of the form *(A : B) :: (C : ?)*. They are studied in verbal settings (*Lungs are to humans as gills are to* [fish]), with geometric figures (Evans, 1968; O'Hara, 1992), and in string domains (Hofstadter et. al., 1995) (*abc : abd :: kji :* [kjj]). Proportional analogies can be formally characterized as isomorphism (Klix, 1993) (see Fig. 1). Analogy is not performed on the patterns themselves, but on descriptions d of them. For each pattern $x \in \{A, B, C, D\}$, there must be found a description $d(x)$, such that $\mu(f(d(A))) = \mu(d(B)) = d(D) = f'(d(C)) = f'(\mu(d(A)))$, where μ is a mapping from one domain to another and f and f' are functions to transform objects or relations holding between objects. That is, f and f' represent the '**:**' operator.

E. g., if we describe patterns *A* = *'abc'*, *B* = *'abd'*, *C* = *'kji'*, and *D* = *'kjj'* simply as "three letters", define μ as mapping the letters by their positions in the strings, and the transformation of *d(abc) : d(abd)* as *f* = *'take last letter and increase'*, we reach the description of pattern *D* by either first transforming *d(A)* into *d(B)* and then mapping *d(B)* to *d(D)*, or first mapping *d(A)* to *d(C)* and then applying the transformation. Note that different descriptions of the patterns result in different solutions for *D*: Look at *'abc'* as ascending sequence of three letters where the 'largest' letter *'c'* is increased and at *'kji'* as an descending sequence of three letters, then *f'* can be constructed as a function where the 'smallest' letter is decreased, resulting in *D* = *'kjh'*. Calculating such redescriptions is crucial for computational models as Copycat (Hofstadter et. al., 1995) or PAN (O'Hara, 1992).

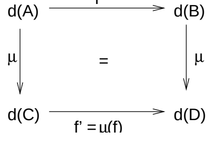


Figure 1: Proportional analogy as isomorphism

Predictive analogy is also known as analogical reasoning and most cognitive models, such as SME (Falkenhainer, Forbus, & Gentner, 1989) or LISA (Hummel & Holyoak, 1997) focus on this kind of analogy. Predictive analogy helps to explain a new domain by observing similarities with a known (base) domain and then transferring further information from the known domain to the new (target) domain (Gentner, 1983). A well-known example is the Rutherford analogy "The atom is like the solar system" (see Fig. 2). A domain is represented as a structure with objects (such as *sun*), attributes (such as *yellow(sun)*), and relations (such as *attracts(sun, planet-i)*). Besides the first-order relations, which are defined over objects, there is a second-order relation, defined over relations: *cause(attracts(sun, planet-i), revolves-around(planet-i, sun))*. All these entities are represented as nodes in the graph. The arcs represent relations between entities or, in other words, their roles (such as S for "subject" or O for "object"). In contrast to proportional analogy, representation are often considered as fixed. Exceptions are incremental mapping approaches (Burstein, 1986; Keane, Ledgeway, & Duff, 1994; Forbus, Ferguson, & Gentner, 1994).

Analogical reasoning is modeled as a structure preserving mapping of objects from base to target. After Gentner's (1983) principle of systematicity, mappings of larger structures are preferred. For the Rutherford example, mapping of *sun* to *nucleus* and of *planet-i* to *electron-i* results in a large structural congruence between both domains. Because each object can carry over nodes from the base to the target to which it is connected, the causal explanation why a planet revolves around the sun is transferred to the target domain, resulting in the inference that an electron revolves around the

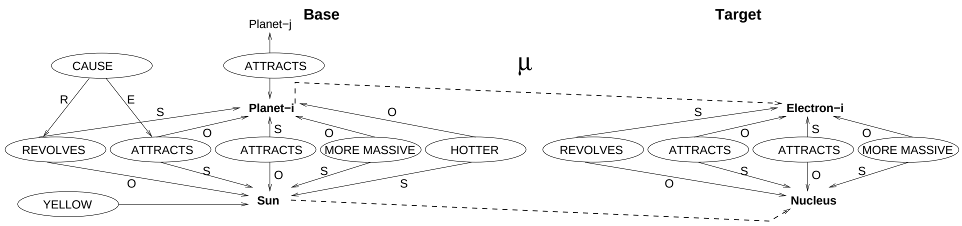


Figure 2: Structure-mapping for the Rutherford analogy (Gentner, 1983, Fig. 1)

nucleus because it is attracted by it. Note that in the Structure Mapping Engine (SME) of Gentner and colleagues (Falkenhainer et al., 1989) nodes representing relations (including attributes as relations with arity one) must be named identically in base and target domain. The systematicity principle translates to the well known problem of finding the greatest common subgraph of two graphs (Jain & Wysotzki, 2002).

Proportional analogy can be viewed as a special case of predictive analogy: consider the upper part in Fig. 1 as base domain with a known relation $f(d(A), d(B))$[1] and consider the lower part as target domain. The relation f' between the patterns of the target domain can be inferred by mapping pattern $d(A)$ onto $d(C)$ and pattern $d(B)$ onto $d(D)$ and transfer $f(d(A), d(B))$ into the target. For the predictive Rutherford analogy, the inferred relation is the *cause* relation. In contrast to proportional analogy, typically not the complete patterns but only a sub-structure of the base domain is mapped to the target (Schmid, Wirth, & Polkehn, 2003). Furthermore, predictive analogy is typically called *between-domain*, while proportional analogies and analogical problem solving (Anderson & Thompson, 1989) are *within-domain* (Vosniadou & Ortony, 1989). Despite the fact, that semantic and possibly also pragmatic information can be included to guide mapping, at least for structure-mapping approaches the distinction between within and between domain analogies is rather artificial in our opinion, since structural alignement relies on the syntactic structure of the representation only.

Common to most cognitive models of analogy is that (a) a structure preserving mapping from objects of the base to the target is performed and that (b) additional information given for the base is transferred to the target. For predictive analogy and analogical problem solving, typically an additional component is proposed: (c) learning by generalization over the common structure of base and target (Gentner, 1989; Anderson & Thompson, 1989). E. g., the abstract concept of a central-force system might be extracted from the commonalities between the solar system and the atom model (Gentner, 1983). Some of the implemented models of analogical and case-based reasoning can be combined with a generalization mechanism (Aamodt & Plaza, 1994; Kuehne, Forbus, Gentner, & Quinn, 2000).

Although mapping is the best worked out process in all models of analogy, in the following, we will introduce yet another approach – anti-unification. In our opinion, the main advantage using anti-unification is, that first and second order mapping, re-representation, and generalization can all be treated with one single mechanism. Furthermore, anti-unification can be applied to the different types of analogy (proportional, predictive, problem solving). The goal of this paper is to give an outline of the formal framework of our approach. We will give references to other papers where we present more details, examples, and information about the algorithms and their implementation. The approach is illustrated with example applications in the domains of proportional and predictive analogies.

Mapping vs. Abstraction

As we described above, the crucial step in analogy is to find a mapping from objects of the base domain to objects of the target domain. An alternative view to mapping is this (see Fig. 3.a): first, identify the common structure (t) of base (t_1) and target (t_2), second, calculate the mapping of objects of the base to objects of the target *via* the common structure.

E. g., t_1 = *attracts(sun, planet-i)* and t_2 = *attracts (nucleus, electron-i)* can be generalized to t = *attracts(central-body, orbiter)* where '*central-body*' and '*orbiter*' are variables. By calculating the abstraction, the mapping $\varphi = \{$(*sun*,*nucleus*) $\mapsto$ *central-body*, (*planet-i*,*electron-i*) $\mapsto$ *orbiter*$\}$ is obtained. Thereby, the necessary substitutions of variables by constants are also known: $\sigma_1 = \{$ *central-body* $\mapsto$ *sun*, *orbiter* $\mapsto$ *planet-i* $\}$ and $\sigma_2 = \{$ *central-body* $\mapsto$ *nucleus*, *orbiter* $\mapsto$ *electron-i* $\}$. Now the mapping of *sun* onto *nucleus* can be performed by applying first φ and then σ_2.

In structure mapping as well as in mapping via abstraction, the greatest common sub-structure of base and target must be identified. But in mapping via abstraction, the common structure is not forgotten but represented explicitly. Instead of mapping one object directly onto another, it is mapped via the *role* it is playing in the common structure. For example, the sun plays the role of an object, which is a central body, attracting another object. Mapping via abstraction in general is more efficient than direct mapping because there is

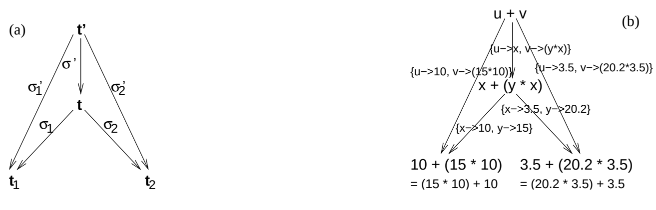


Figure 3: Mapping via abstraction and instantiation

[1]Note that each function $f : Dom \rightarrow Codom$ corresponds to a relation $f(Dom, Codom)$.

no need to try out different possible mappings. In the LISA model (Hummel & Holyoak, 1997) mapping is constrained by semantic features associated with the objects. In anti-unification, the constraints are given by the problem structure, or, in other words, by the *functional role* the objects play in a given structure. Note that these roles are *not* predefined but inferred from the given structures as intrinsic part of the anti-unification process.

The mapping procedure just described is known as calculating the most special generalization (Plotkin, 1969) or as anti-unification (Reynolds, 1970). It is important, that the *most special* generalization is calculated. Only then all structral commonalities are captured in the abstract representation. Formally, a generalization of t_1 and t_2 can be characterized as most specific, if for each other generalization t' holds, that there exists a substitution σ' such that $t\sigma' = t$ (see Fig. 3.a).

Anti-unification (AU) is defined for clauses (that is logical formulae) as well as for terms. In the following, we only deal with terms. For further illustration of AU, look at the following example (see Fig. 3.b):

$t_1 = 10 + (15 \cdot 10)$ and $t_2 = 3.5 + (20.2 \cdot 3.5)$.

The most special generalization is $t = x + (y \cdot x)$ with $\varphi = \{(10, 3.5) \mapsto x, (15, 20.2) \mapsto y\}$ and it holds that $t_1 = t\sigma_1$ for $\sigma_1 = \{x \mapsto 10, y \mapsto 15\}$ and $t_2 = t\sigma_2$ for $\sigma_2 = \{x \mapsto 3.5, y \mapsto 20.2\}$. Another generalization is $t' = u + v$. t_1 can be obtained from t' by $\sigma_1' = \{u \mapsto 10, v \mapsto (15 \cdot 10)\}$, and t_2 by $\sigma_2' = \{u \mapsto 3.5, v \mapsto (20.2 \cdot 3.5)\}$.[2] This generalization t' is not as special as t, it "forgets", that the second operand of the addition is the product of two numbers where the second number is equal to the first operand.

Sometimes, for constructing a suitable abstraction, it might be necessary to include knowledge about the domain. E. g., for $t_2' = (20.2 \cdot 3.5) + 3.5$ and t_1 as above, only the overly general anti-instance $u + v$ can be obtained. But we know, that addition is commutative and therefore, we can rewrite t_2' into its original form t_2. Knowledge about the equality of terms can be represented by an equational theory. The laws for addition constitute such a theory:

$x + 0 \stackrel{E}{=} x \qquad x + y \stackrel{E}{=} y + x \qquad x + (y + z) \stackrel{E}{=} (x + y) + z.$

If we use knowledge in form of equational background theories for rewriting terms before syntactical AU is performed, we speak of E–generalization. Models for solving proportional analogies, such as Copycat (Hofstadter et. al., 1995) or PAN (O'Hara, 1992), allow for an arbitrary sequence of rewritings of the initial representation (re-representation) to find a suitable solution. In contrast, E–generalization allows us to perform abstraction while modeling equivalent representations by appropriate equations between terms. All equivalent representations are considered *simultaneously* in the abstraction process. Therefore, abstraction becomes insensitive to representation changes.

Up to now we have looked at *first-order* AU, which corresponds to the SME (Falkenhainer et al., 1989) where relations must be preserved. First-order AU can be generalized to second-order, allowing that functions (or operators or relations) are generalized to function variables. Again we give a simple example:

$t_3 = 10 + (15 \cdot 10)$ and $t_4 = 3.5 - (20.2 \cdot 3.5)$.

In the first-order case, t_3 and t_4 generalize to the object variable x because the expressions are built over different operators ($(+)$ in t_3 and $(-)$ in t_4). In the second-order case we obtain $t_{3,4} = x\ F\ (y \cdot x)$ where x and y are object variables and F is a function variable with $\sigma_3 = \{F \mapsto (+), x \mapsto 10, y \mapsto 15\}$ and $\sigma_4 = \{F \mapsto (-), x \mapsto 3.5, y \mapsto 20.2\}$. In general, second-order AU allows for deletion, insertion and permutation of arguments of function variables.

AU is founded on the mathematics of term algebras. Therefore, in contrast to the typical cognitive models, it allows for precise statements and reasoning about the quality of solutions. Algorithms for efficiently realizing AU can be derived easily from this general framework. Moreover, we are able to clearly separate this kernel method from experimental application–dependent heuristics based upon it.

AU can be combined with *subsumption* to allow for a qualitative criterion of structural similarity (Plaza, 1995; Schmid, Sinha, & Wysotzki, 2001). A target term might be anti-unified with different candidate terms, resulting in different anti-instances. The best fitting term of the base can be considered to be that term which results in the most specific anti-instance. This most specific anti-instance can be found by calculating a subsumption relation between the anti-instances: An anti-instance (term) a_1 subsumes a term a_2 iff there exists a substitution σ such that $a_1\sigma = a_2$. Subsumption can also be defined relative to an equational theory. Obviously, terms and their anti-instance are in a subsumption relation: In Fig. 3.a t_1 and t_2 are subsumed by t (and t is subsumed by t'). Analogously, the E-subsumption relation holds for the example in Fig. 3.b.

Solving proportional analogies

In the following, we sketch how to solve an analogical reasoning task $(A : B) :: (C : X)$ by E–generalization. For sake of brevity, we omit a detailed description of the algorithms and the theoretical results they are based on; both can be found in Burghardt (2002). We take the analogical reasoning terminology and our running example from Dastani, Indurkhya, and Scha (1997). Our goal is to induce a transformation rule $(P : Q)$ that meets the *projectibility criterion* which can be completely formalized in our setting:

1. the rule transforms A into B, i.e. $(P : Q)\sigma_1 \stackrel{E}{=} (A : B)$ for some substitution σ_1;
2. it is applicable also to C, i.e. $P\sigma_2 \stackrel{E}{=} C$ for some σ_2;
3. whenever applicable, the rule yields a well–defined result, i.e. each variable used in Q is "defined" in P.

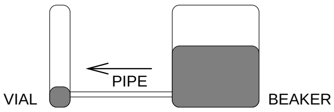
We use $(abc : abd) :: (kji : X)$ as a running example. Let the equational theory E consist of the associativity law, equations for the letter successor s and predecessor p functions,

[2] The usual definition of substitution is that variables can be replaced by arbitrary terms, that is, by other variables ("renaming"), by constants, or by more complex expressions.

Base

entities:

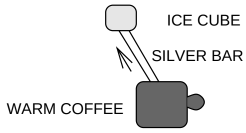
vial: *inanimate*
beaker: *inanimate*
water: *inanimate*
pipe: *inanimate*



expressions:

liquid(water)
flat_top(water)
greater(diameter(beaker),diameter(vial))
greater(pressure(beaker),pressure(vial))
flow(beaker,vial,water,pipe)
cause(greater(pressure(vial),pressure(beaker)), flow(vial,beaker,water,pipe))

Target



entities:

coffee: *inanimate*
ice_cube: *inanimate*
bar: *inanimate*
heat: *inanimate*

expressions:

liquid(coffee)
flat_top(coffee)
greater(temperature(coffee),temperature(ice_cube))
flow(coffee,ice_cube,heat,bar)

Figure 4: The *Water Flow*/*Heat Flow* analogy

and equations for the iteration function $It(\cdot,\cdot,\cdot)$:

$$It(x,\mathsf{s},1) \stackrel{E}{=} x \qquad It(x,\mathsf{s},y+1) \stackrel{E}{=} x \cdot It(\mathsf{s}(x),\mathsf{s},y)$$
$$It(x,\mathsf{p},1) \stackrel{E}{=} x \qquad It(x,\mathsf{p},y+1) \stackrel{E}{=} x \cdot It(\mathsf{p}(x),\mathsf{p},y)$$
$$(x \cdot y) \cdot z \stackrel{E}{=} x \cdot (y \cdot z) \qquad \mathsf{p}(\mathsf{s}(x)) \stackrel{E}{=} x \qquad b \stackrel{E}{=} \mathsf{s}(a) \qquad \ldots$$

We start from the set of all terms equal to abc, which contains e.g. $It(a,\mathsf{s},3)$ and $\mathsf{p}(\mathsf{s}(a)) \cdot (b \cdot c)$. It is infinite, but can be finitely described by a regular tree grammar.[3] Similarly, we describe the set of all terms equal to abd and kji by a grammar each. From these, we can compute a grammar $\mathcal{G}_{AC}$ describing the set $\mathcal{L}(\mathcal{G}_{AC})$ of E–generalizations of abc and kji. Each $P \in \mathcal{L}(\mathcal{G}_{AC})$ is an explicit representation of a *Gestalt* common to abc and kji, satisfying $P\sigma_1 \stackrel{E}{=} abc$ and $P\sigma_2 \stackrel{E}{=} kji$ for certain substitutions σ_1 and σ_2. The elements of $\mathcal{L}(\mathcal{G}_{AC})$ can be enumerated in order of increasing complexity, where the complexity measure can be user–defined to cover e.g. *information load*. For example, one such element is $It(x,y,3)$, which maps to $It(a,\mathsf{s},3)$ and $It(k,\mathsf{p},3)$ under σ_1 and σ_2, respectively. Next, we compute a grammar $\mathcal{G}_B$ describing the set of all Q with $Q\sigma_1 \stackrel{E}{=} abd$.

Any transformation rule $(P : Q)$ built from some $P \in \mathcal{L}(\mathcal{G}_{AC})$ and some $Q \in \mathcal{L}(\mathcal{G}_B)$ satisfies the projectibility criteria 1. and 2. from above. There are two ways to achieve 3. also. The first way needs few computation time, while the second allows to enumerate rules in order of increasing overall complexity. The least complex rule is considered to be the one that reflects the *mutual contextualization* of A, B, and C; the result of applying it to C is considered to be the solution for X, which equals $Q\sigma_2$. In our example, we may obtain e.g. the transformation rule $It(x,y,3) : It(x,y,2) \cdot \mathsf{s}(\mathsf{s}(\mathsf{s}(x)))$, which covers the similarity between abc and kji as well as that between abc and abd. In Fig. 1, e.g. $d(abc)$ corresponds to $It(a,\mathsf{s},3)$, while both f and f' is realized by the rule $It(x,y,3) : It(x,y,2) \cdot \mathsf{s}(\mathsf{s}(\mathsf{s}(x)))$, and μ by σ_1 and σ_2.

Like Dastani et al. (1997), we use an algebraic approach, which is, however, based on standard grammar algorithms, allowing us to include further grammar computation modules as needed, e.g. to build in additional filters that might be application–required. Moreover, its computational complexity can easily been estimated: the required run–time grows quadratically with the size of the input grammars, if the fast method to achieve criterion 3. is used. Since grammar computation is independent of the employed complexity measure, result grammars can be retained after a subsequent revision of the *information load* measure.

[3] A typical grammar rule is $N_b ::= b \mid \mathsf{s}(N_a) \mid \mathsf{p}(N_c)$, which intuitively reads: "A term equal to b may consist either of the constant b, or the successor of a term equal to a, or the predecessor of a term equal to c".

Solving predictive analogies

Besides the Rutherford analogy discussed above, there are some other examples for predictive analogy (Falkenhainer et al., 1989; Indurkhya, 1992) such as *water flow*/*electrical current* or *water flow*/*heat flow*. In the following, we use the *water flow*/*heat flow* analogy, where behavior in the domain of thermodynamics is predicted by knowledge from the domain of fluid dynamics. First, we discuss the SME model (Falkenhainer et al., 1989) and afterwards we show how analogical reasoning and generalization can be realized with AU. More details of our approach can be found in Gust, Kühnberger, and Schmid (2003).

Base and target problem are illustrated in Fig. 4. SME infers that heat flows from the coffee via the bar to the ice cube, because the temperature of the coffee is greater than the temperature of the ice cube. This inference seems to be somewhat trivial, since the facts *flow(beaker, vial, water, pipe)* and *flow(coffee, ice_cube, heat, bar)* are explicitely given and guide the match *pressure*/*temperature*. This leads to a transfer of the causal relation from base to target.

The fact that mapping of *pressure* to *temperature* is preferred over mapping *diameter* to *temperature* by SME is determined by the predefined structures of the base and target representations only. Using just this systematicity principle and no knowledge of the behavior of the physical system this purely syntactical approach can easily result in erroneous transfer: Adding *cause(greater(diameter(beaker), diameter(vial)), greater(capacity(beaker), capacity(vial)))* and *greater(capacity(beaker), capacity(vial))* to the base and *greater(capacity(coffee), capacity(ice_cube))* to the target, an application of SME may result in the inference *cause(greater(temperature(coffee), temperature(ice_cube), greater(capacity(coffee),capacity(ice_cube)))* which is inconclusive.

Table 1: Alternative Representation of *Water Flow*/*Heat Flow*

Base

types:
real, massterm, object, time

entities:

vial:	*object*
beaker:	*object*
water:	*massterm*
pipe:	*object*

functions:

observable	*height:*	$object \times time \rightarrow real \times \{cm\}$
observable	*footprint:*	$object \times time \rightarrow real \times \{cm^2\}$
operator	*in:*	$massterm \times object \rightarrow object$
theoretical	*volume:*	$object \times time \rightarrow real \times \{cm^3\}$

function definitions:
$volume(x,t) := footprint(x,t) \cdot height(x,t)$

facts:
connected(beaker,vial,pipe)
liquid(water)
$\forall\, t_1{:}time,\ t_2{:}time:$
$footprint(beaker,t_1) > footprint(vial,t_1) \wedge$
$footprint(beaker,t_1) = footprint(beaker,t_2) \wedge$
$footprint(vial,t_1) = footprint(vial,t_2)$

laws:
$\forall\, t_1{:}time,\ t_2{:}time : t_2 > t_1 \wedge$
$height(water\ in\ beaker,\ t_1) > height(water\ in\ vial,\ t_1)$
$\rightarrow$
$height(water\ in\ beaker,\ t_2) > height(water\ in\ beaker,\ t_1) \wedge$
$volume(water\ in\ beaker,\ t_2) - volume(water\ in\ beaker,\ t_1)$
$= volume(water\ in\ vial,\ t_2) - volume(water\ in\ vial,\ t_1)$

Target

types:
real, massterm, object, time

entities:

coffee:	*massterm*
cup:	*object*
b_cube:	*object*
bar:	*object*

functions:

observable	*temperature:*	$object \times time \rightarrow real \times \{C\}$
operator	*in:*	$massterm \times object \rightarrow object$

facts:
connected(coffee in cup, b_cube,bar)
liquid(coffee)

experiment:
$t_1{:}time,\ t_2{:}time:$
$t_2 > t_1 \wedge$
$temperature(coffee\ in\ cup,\ t_1)$
$> temperature(b\_cube,t_1)$
$\rightarrow$
$temperature(coffee\ in\ cup,\ t_2)$
$< temperature(coffee\ in\ cup,t_1) \wedge$
$temperature(b\_cube,t_2)$
$> temperature(b\_cube,t_1)$

Before we present how the *water flow*/*heat flow* analogy can be solved with AU, we introduce some modifications of the domain representations given in Fig. 4: First, we distinguish between massterms and individuals, that is, *water* and *coffee* as massterms can *flow* and *water* and *coffee* as individuals can have specific *pressure* resp. *temperature*. Second, the ice cube is replaced by a metal cube.[4] While the ice cube keeps its temperature at 0° Celsius until it is melted away, in a metal cube, temperature will increase gradually. We use *footprint* instead of *diameter* to allow not only for cylindrical but also for arbitrarily shaped containers. The modified representations for base and target are given in Tab. 1.

As in the SME representations a domain consists of entities and facts (called expressions in SME). Besides this structural information, knowledge about the physical system is given in form of typed functions and laws (for the base) resp. experimental observations (for the target). In our opinion, doing experiments is crucial for obtaining additional information about the domain and, more important, for checking the validity of inferences. Since not all characteristics of a physical system are observable, functions are divided into observable and theoretical terms. Furthermore, we use an operator *in* to construct individuals from massterms.

To perform experiments we can supply qualitatively specified parameters like $t_2 > t_1$ and $temperature(coffee\ in\ cup,\ t_1) > temperature(b\_cube,t_1)$ (see lh-side of the implication in the experiment for the target in Tab. 1). These parameters might be instantiated with $t_1 = 0s$, $t_2 = 20s$, $temperature(coffee\ in\ cup,\ t_1) = 80°\ Celsius$ and $temperature(b\_cube,t_1) = -10°\ Celsius$. For the second observation time t_2, the results may be $temperature(coffee\ in\ cup,\ t_2) = 70°\ Celsius$ and $temperature(b\_cube,t_2) = 0°\ Celsius$,

Table 2: Anti-Unification of *Heat*/*Water Flow*

	Base	Target	G
(1)	*connected(beaker, vial, pipe)*	*connected(coffee in cup, b_cube, bar),*	*conn. A, B, C),*
(2)	*liquid(water)*	*liquid(coffee)*	*liquid(D)*
(3)	*height(water in beaker, t_1) > height(water in vial, t_1)*	*temperature(coffee in cup, t_1) > temperature(b_cube, t_1)*	*T(A, t_1) > T(B t_1)*
(4)	*height(water in beaker, t_1) > height(water in beaker, t_2)*	*temperature(coffee in cup, t_1) > temperature(coffee in cup, t_2)*	*T(A, t_1) > T(A, t_2)*
(5)	*height(water in vial, t_2) > height(water in vial, t_1)*	*temperature(b_cube, t_2) > temperature(b_cube, t_1)*	*T(B, t_2) > T(B, t_1)*

Mappings:

$A \mapsto$ *beaker / coffee in cup*	*from (1)*
$B \mapsto$ *vial / b_cube*	
$C \mapsto$ *pipe / bar*	
$D \mapsto$ *water / coffee*	*from (2)*
$T \mapsto \lambda\, x,t$*: height(water in x, t) / temperature*	*from (3).*

which corresponds to the qualitative result $temperature(coffee\ in\ cup,\ t_2) < temperature(coffee\ in\ cup,t_1)$ and $temperature(b\_cube,t_2) > temperature(b\_cube,t_1)$ (see rh-side of the implication in the experiment for the target in Tab. 1).

Our target representation is less structured than SME's target representation (Fig. 4), because we only represent the temperature differences between coffee and cube at two observation times. In order to compare the experimental findings of the target with the facts and laws of fluid dynamics given for the base the universal quantification has to be instantiated by the two concrete time points used for the target.

Now we have introduced all preliminaries necessary to perform analogical transfer by AU. For analogical transfer all facts, the instantiated laws and experimental observations are candidates for anti-unification. The slightly simplified result of AU is given in Tab. 2.

AUs (1) and (2) are first order syntactical, AU (3) is second

[4] Beryllium, specific heat 3.37 J/qcm*K, would be suitable.

order syntactical and AUs (4) and (5) are first order E-AUs with $x<y \Leftrightarrow y>x$. For the anti-instance G holds *Base* = $G\sigma_1$ and *Target* = $G\sigma_2$ with the mappings

Abstraction G represents the concept of *energy flow*, generalizing over fluid dynamics and thermodynamics. The general laws known for the base domain are transferred to the target and can be checked by further experiments. An interesting side-effect of the calculated abstraction is, that the unobservable term *heat* can be inferred: Function *height* with *water in vial* as first argument is mapped onto function *temperature* with *b_cube* as first argument using second-order AU. This corresponds to the projection *height'(vial,* t_1*)* $\approx$ *temperature(b_cube,* t_1*)*. Working with the original expression *height(water in vial,* t_1*)*, given the object mappings from above, this corresponds to a new representation *temperature'(? in b_cube,* t_1*)* where *?* represents *heat* and consequently the analogy of *water flow* and *heat flow* can be completed by inferring a new theoretical concept. That is, using AU in predictive analogies models how scientific discovery can be guided my abstraction.

Conclusion

We introduced anti-unification as a formally sound and powerful approach to analogy. Anti-unification captures the notion that analogy is based on an *explicit* detection of the structural commonalities between a base and a target problem, represented by the most specific generalization of base and target structure. Instead of a direct mapping of objects (and functions or relations), mapping is performed via the generalized structure, thereby constraining mapping by the *role* of the entities which are mapped. There is no need to model learning as a seperate process, since generalization over the common structure of base and target occurs as a side-effect during analogical inference. In anti-unification, additional knowledge about a domain can be used in form of equational theories to rewrite representations. The formal framework covers first-order and higher-order mappings and there exist efficient implementations for both cases. Since for the higher-order case substitutions of function variables allow for insertion, deletion, and permutations of their arguments, adaptation of a base solution to the target problem can be realized in a natural way. In our opinion, it is plausible to assume that human reasoners use the functional roles of objects in the given problem structures as a guideline to mapping, which is exactly how AU algorithms work. Empirical investigations are necessary to show that our approach is not only theoretically and algorithmically sound but also models some aspects of human analogical problem solving.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, *7*(1), 39–59.

Anderson, J., & Thompson, R. (1989). Use of analogy in a production system architecture. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning* (p. 267-297). Cambridge University Press.

Burghardt, J. (2002). An algebraic approach to abstraction and representation change. *Artificial Intelligence*. (forthcoming)

Burstein, M. (1986). A model of learning by incremental analogical reasoning and debugging. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine learning - an artificial intelligence approach* (Vol. 2, p. 351-370). San Mateo, CA: Morgan Kaufmann.

Dastani, M., Indurkhya, B., & Scha, R. (1997). An algebraic method for solving proportional analogy problems. In *Proceedings of Mind II.* Dublin, Ireland.

Evans, T. G. (1968). A program for the solution of a class of geometric-analogy intelligence-questions. In M. Minsky (Ed.), *Semantic information processing* (p. 271-353). MIT Press.

Falkenhainer, B., Forbus, K., & Gentner, D. (1989). The structure mapping engine: Algorithm and example. *Artificial Intelligence*, *41*, 1-63.

Forbus, K., Ferguson, R., & Gentner, D. (1994). Incremental structure-mapping. *16th CogSci* (p. 313-318). Hillsdale, NJ: Erlbaum.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, *7*, 155-170.

Gentner, D. (1989). The mechanisms of analogical learning. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning* (p. 199-241). Cambridge, UK: Cambridge University Press.

Gust, H., Kühnberger, K.-U., & Schmid, U. (2003). *Anti-unification of axiomatic systems.* (`http://www.cogsci.uni-osnabrueck.de/~helmar/analogy1.ps`)

Hofstadter, D., & The Fluid Analogies Research Group. (1995). *Fluid concepts and creative analogies.* New York: Basic Books.

Hummel, J., & Holyoak, K. (1997). Distributed representation of structure: A theory of analogical access and mapping. *Psychological Review*, *104*(3), 427-466.

Indurkhya, B. (1992). *Metaphor and cognition.* Dordrecht, The Netherlands: Kluwer.

Jain, B. J., & Wysotzki, F. (2002). Self-organizing recognition and classification of relational structures. In *24th CogSci* (p. 488-493). Mahwah, NJ: Erlbaum.

Keane, M., Ledgeway, T., & Duff, S. (1994). Constraints on analogical mapping: A comparison of three models. *Cognitive Science*, *18*, 387–438.

Klix, F. (1993). Analytische Betrachtungen über Struktur und Funktion von Inferenzen. *Zeitschrift für Psychologie*, *201*, 393-414.

Kuehne, S., Forbus, K., Gentner, D., & Quinn, B. (2000). SEQL: Category learning as progressive abstraction using structure mapping. *22nd CogSci* (p. 770-775). Mahwah, NJ: Erlbaum.

O'Hara, S. (1992). A model of the redescription process in the context of geometric proportional analogy problems. In *Int. Workshop on Analogical and Inductive Inference (AII '92)* (Vol. 642, p. 268-293). Springer.

Plaza, E. (1995). Cases as terms: A feature term approach to the structured representation of cases. In *Proc. ICCBR-95* (Vol. 1010, pp. 265–276). Springer.

Plotkin, G. D. (1969). A note on inductive generalization. In *Machine intelligence* (Vol. 5, pp. 153–163). Edinburgh University Press.

Reynolds, J. C. (1970). Transformational systems and the algebraic structure of atomic formulas. In *Machine intelligence* (Vol. 5, pp. 135–151). Edinburgh University Press.

Schmid, U., Sinha, U., & Wysotzki, F. (2001). Program reuse and abstraction by anti-unification. In *Professionelles Wissensmanagement – Erfahrungen und Visionen* (p. 183-185). Shaker.

Schmid, U., Wirth, J., & Polkehn, K. (2003). A closer look on structural similarity in analogical transfer. *Cognitive Science Quarterly*, *3*(1), 57-89.

Vosniadou, S., & Ortony, A. (1989). Similarity and analogical reasoning: A synthesis. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning* (p. 1-17). Cambridge: Cambridge University Press.