Mathematical reasoning with higher-order anti-unifcation

Markus Guhe, Alison Pease, Alan Smaill
(m.guhe|a.pease|a.smaill@ed.ac.uk)
University of Edinburgh, School of Informatics, Informatics Forum, 10 Crichton Street, Edinburgh EH8 9AB, Scotland

Martin Schmidt, Helmar Gust, Kai-Uwe Kühnberger, Ulf Krumnack
(martisch|hgust|kkuehnbe|krumnack@uni-osnabrueck.de)
University of Osnabrück, Institute of Cognitive Science, Albrechtstr. 28, 49076 Osnabrück, Germany

Abstract

We show how *heuristic-driven theory projection* (HDTP, a method based on higher-order anti-unification) can be used to model analogical reasoning in mathematics. More precisely, HDTP provides the framework for a model of the inductive analogy-making process involved in establishing the fundamental concepts of arithmetic. This process is a crucial component for being able to generalise from the concrete experiences that humans have due to their embodied and embedded nature. Such generalisations are a cornerstone of the ability to create an abstract domain like arithmetic. In addition to generalisations, HDTP can also transfer concepts from one domain into another, which is, for example, needed to introduce the concept ZERO into arithmetic. The approach presented here is closely related to the theories of *Information Flow* and *Institutions*. The latter in particular provides a compelling way to integrate *concept blending* into the HDTP approach.

Keywords: mathematical cognition; mathematical reasoning; analogy; anti-unification; concept blending

Mathematical reasoning as a cognitive process

Although mathematics is usually presented in terms of axioms, concise proofs, theorems and so on, the actual cognitive process of mathematical reasoning is very different. For example, when a mathematician changes a definition this affects the proofs that use it, but such changes are not discussed in mathematical papers. Additionally, mathematics, at least partly, does not consist of discovering eternal, Platonic ideals but in creating mathematical concepts. For example, Lakatos's (1976) account of the history of Euler's conjecture illuminates how the concept POLYHEDRON can differ and how its definition depends on the current circumstances and needs of the mathematician. Put differently, if the Platonic ideal POLYHEDRON does exist, it is not clear how it can be identified by mathematical means – what cognitive processes mathematicians can use to find the correct definition. Thus, mathematical concepts are not necessarily the same as the ideals.

Lakoff and Núñez (2000) describe how our embodied, situated experience is the basis on which abstract mathematical concepts are developed by a process of metaphorical abstraction and transfer. In chapter 3, they describe how basic arithmetic is created from four everyday experiences, which are the source domains of the metaphors. In this way, arithmetic is grounded in situated cognition. To motivate that these four domains in particular are source domains, Lakoff and Núñez analyse linguistic expressions used in the target domain, arithmetic, which they trace back to these four domains. For example, we use the terms *add* and *take away* in arithmetic. Lakoff and Núñez argue that these terms were originally used for talking about collections of objects, such as physically placing an object into a container, e.g. *adding an onion to the soup*, or physically removing a substance or an object from a container, e.g. *take a book out of the box*.

Analogical reasoning is a central component of the process transforming knowledge of this kind into mathematical concepts. For present purposes we assume that metaphor and analogy are essentially the same cognitive process (Gentner, Bowdle, Wolff, & Boronat, 2001), and we have demonstrated how structure mapping (Gentner, 1983; Gentner & Markman, 1997) – a basic method to compute analogical relations – can account for the overall cognitive process (Guhe, Pease, & Smaill, 2009).

In this paper, we describe a formal cognitive model of this process. This has a twofold motivation: firstly, we want to specify the cognitive processes that mathematicians use, to better understand how mathematical discovery works; secondly, we want to use the model to improve automated theorem provers by incorporating cognitive mechanisms. In Guhe, Smaill, and Pease (2009a, 2009b) we presented formal representations of the four grounding metaphors (the 4Gs) and suggested how *Information Flow* theory (Barwise & Seligman, 1997) may be used to model the analogies involved. The 4Gs are: (1) arithmetic is object collection, (2) arithmetic is object construction, (3) measuring stick and (4) arithmetic is motion along a path.

Here, we present a proof-of-concept of how performing anti-unification (Plotkin, 1970) on such representations can account for aspects of the analogical reasoning involved in the 4Gs. This inductive kind of reasoning provides us with a procedural version of the otherwise static Information Flow models and enables us to computationally determine the relationships between the domains (classifications in the case of Information Flow). More precisely, we will use *Heuristic-Driven Theory Projection* (HDTP; Schwering, Krumnack, Kühnberger, & Gust, 2009), a general framework for making analogies. HDTP provides us with the means to generalise over two of Lakoff and Núñez's domains to establish a basis for arithmetic as well as the means to generalise over one of the domains as source domain and arithmetic as the target domain to add concepts to arithmetic that are only present in one of the grounding domains. We will also outline how this conception of mathematical reasoning is linked to Goguen's (2006) notion of *concept blending* (which is based on notions by Gärdenfors, 2000 and Fauconnier & Turner, 2002), a further cognitive process for creating mathematical concepts.

Metaphors for arithmetic

Arithmetic is object collection

The arithmetic is object collection metaphor (Table 1) is based on the notion that the repeated manipulation of (small, countable, physical) collections of objects lets us notice certain regularities. For example, we can determine which one of two collections is bigger by aligning the objects in the two collections one-to-one, and the collection that has at least one unpaired object left over is the bigger collection. (Smaller and equal are, of course, determined correspondingly.) This corresponds to the (abstract) arithmetic notion GREATER.

By comparing collections of objects in this way we can also group such collections into groups of collections of equal size, i.e. where after the aligning procedure no object is left unpaired. Each of these groups corresponds to a number in arithmetic.

There are two things to note about this basic metaphor. Firstly, it does not produce a concept of ZERO, because the empty collection is a collection that does not exist physically. (Even calling one object a *collection with one object* is an abstraction of the term *collection*.) Lakoff and Núñez (2000, p. 64) propose that an *entity-creating metaphor* is required to create a concept that is not part of the basic metaphor (like ZERO). This corresponds well with the fact that, historically, ZERO was a rather late invention. Secondly, the subtraction operation requires that a smaller collection be taken from a bigger one, because physically, negative objects do not exist.[1]

Table 1: Arithmetic is object collection metaphor (Lakoff & Núñez, 2000, p. 55)

object collection	arithmetic
collections of objects of the same size	numbers
size of collection	number
bigger	greater
smaller	less
smallest collection	the unit (one)
putting collections together	addition
taking a smaller collection from a larger collection	subtraction

Arithmetic is object construction

The arithmetic is object construction metaphor (Table 2) runs along the same lines, except that it is not based on collections of objects, but on objects that are constructed from smaller objects. In this way, fractions are added to arithmetic, although they are not part of the basic metaphor. Consider, for example, an object that is constructed out of seven smaller objects. If now a smaller object that consists of three of the seven overall objects is removed from the original object, the two resulting objects have a size of $\frac{3}{7}$ and $\frac{4}{7}$ of the original.

[1]One is reminded of the old joke: If on one floor 5 people leave an elevator containing 3 people, 2 people have to enter the elevator on the next floor in order for it to be empty.

Table 2: Arithmetic is object construction metaphor (Lakoff & Núñez, 2000, pp. 65–66)

object construction	arithmetic
objects	numbers
smallest whole object	the unit (one)
size of object	size of number
bigger	greater
smaller	less
constructed object	result of arith. operation
whole object	a whole number
putting objects together to form larger objects	addition
taking smaller objects from larger objects to form other objects	subtraction

Arithmetic is motion along a path

The *motion along a path* metaphor (Table 3) adds concepts to arithmetic that we experience by moving along straight paths. Numbers are point locations on paths. Addition and subtraction correspond to a movements from point one point on the path to another point on the path. An important new concept that is added to arithmetic by this metaphor is ZERO, which is based on the concept of a path's origin and which provides a direction for the movements along paths, namely towards the origin or away from it.

Table 3: Arithmetic is motion along a path metaphor (Lakoff & Núñez, 2000, p. 72)

motion along a path	arithmetic
acts of moving along the path	arith. operations
a point location on the path	result of an operation; number
origin; beginning of the path	zero
unit location, a point location distinct from the origin	one
further from the origin than	greater
closer to the origin than	less
moving away from the origin a distance	addition
moving toward the origin a distance	subtraction

Heuristic-Driven Theory Projection

Overview

This section provides a short overview of the basic ideas of heuristic-driven theory projection (HDTP), a formal framework to model analogical mapping and reasoning. A more detailed description can be found in Schwering et al. (2009).

HDTP establishes analogies between two domains, the source and the target, by detecting common structures. In the mapping phase, source and target are compared for structural commonalities and a generalised description is created, which subsumes the matching parts of both domains. In the transfer phase, unmatched knowledge in one domain can be mapped to the other to establish new hypotheses.

HDTP is a formal framework that computes analogical relations and inferences for domains represented in first-order logic. Both, source and target domain, are given by axiomatisations, i.e. finite sets of first-order formulae. The basic idea is to associate pairs of formulae from the domains in a systematic way. HDTP uses anti-unification (Plotkin, 1970) to identify common patterns in formulae. In anti-unification, two formulae are compared and the least general generalisation that subsumes both formulae is identified.

Figure 1 provides some examples of anti-unification of terms. Terms are generalised to an anti-instance where different constants or function symbols are replaced by a variable. In (i), first-order anti-unification is sufficient. However, the terms in (ii) differ in the function symbols, i.e. first-order anti-unification fails to detect structural commonalities. Here, higher-order anti-unification generalises function symbols to a variable and retains the structural commonality. It is even possible to generalise terms in which common parts are embedded structurally in a different way, as shown in (iii).[2] Substitutions accompanying the generalised terms are created, which can be used to reconstruct the original terms.

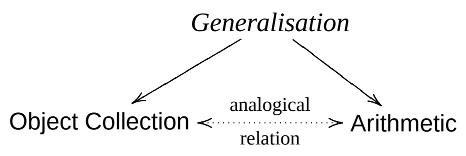
$$\begin{tikzcd}
f(X) \arrow[d, "X \to a"'] \arrow[dr, "X \to b"] & & F(a) \arrow[d, "F \to f"'] \arrow[dr, "F \to g"] & & F(a,b) \arrow[d, "F \to f"'] \arrow[dr, "F(x,y) \to h(x,g(y))"] & \\
f(a) & f(b) & f(a) & g(a) & f(a,b) & h(a,g(b)) \\
\text{(i)} & & \text{(ii)} & & \text{(iii)} &
\end{tikzcd}$$

Figure 1: Anti-unification of terms

HDTP extends this classical anti-unification of terms to formulae and logical theories by iteratively picking pairs of formulae to be generalised from the domains. This process is driven by heuristics. Coherent mappings are preferred, i.e. mappings in which substitutions can be reused. The generalised theory together with its substitutions specifies the analogical relation between source and target. Additional information about the source domain, i.e. formulae with no correspondence in the target domain, can be transferred by replacing symbols using the established substitutions.

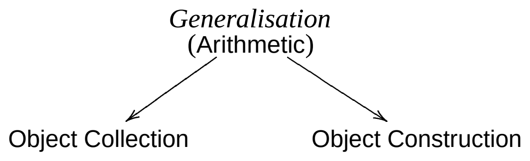
Modelling the arithmetical metaphors

HDTP provides two different ways in which Lakoff and Núñez's (2000) grounded domains (Object Collection, Object Construction etc.) can be related to the abstract domain of Arithmetic. Following Lakoff and Núñez, the grounded domains constitute the source, while Arithmetic is the target domain. To establish an analogical relation between Object Collection and Arithmetic, HDTP can construct a generalisation of these domains:

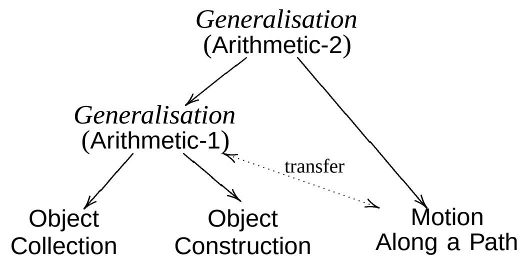


In this model, both domains are already given. The analogy explains abstract concepts like numbers by linking them to familiar entities from the grounded domains. Thus, the generalisation provides a description of the commonalities of the grounded and the abstract domains.

However, from the cognitive perspective, Arithmetic does not initially exist – it has to be created by an act of abstraction as well. This idea can be modelled by analogically relating two grounded domains, e.g. Object Collection and Object Construction. Arithmetic then emerges as a generalisation of these domains.



In our view, a combination of both approaches is needed to model the cognitive bootstrapping process. By generalising over two grounded domains, an abstract domain is established, which serves as a 'proto-domain' of Arithmetic, i.e. a domain that already contains some arithmetical concepts. This is then refined subsequently, by relating it analogically to other grounded domains, removing peculiarities of the two original domains and/or adding new concepts by analogical transfer.



It should be noted that in pursuing this approach the results may vary depending on which grounded domains are chosen for generalisation and on the order in which other grounded domains are added for refinement. This is due to the heuristics that HDTP applies when building up the generalisation. The more similar the grounded domains are, the richer the generalisation will be, while dissimilar domains give coarser results. Nevertheless, we expect that this effect can be compensated by further mapping the initial generalisation to other domains. A detailed examination of this will be a focus of our future work.

Formalisation of domains

We demonstrate the feasibility of the outlined approach by applying it to simple formalisations of Lakoff and Núñez's

[2]HDTP uses a restricted form of higher-order substitutions, that allows to expand terms by introducing arguments and nested structures as described in Krumnack, Schwering, Gust, and Kühnberger (2007).

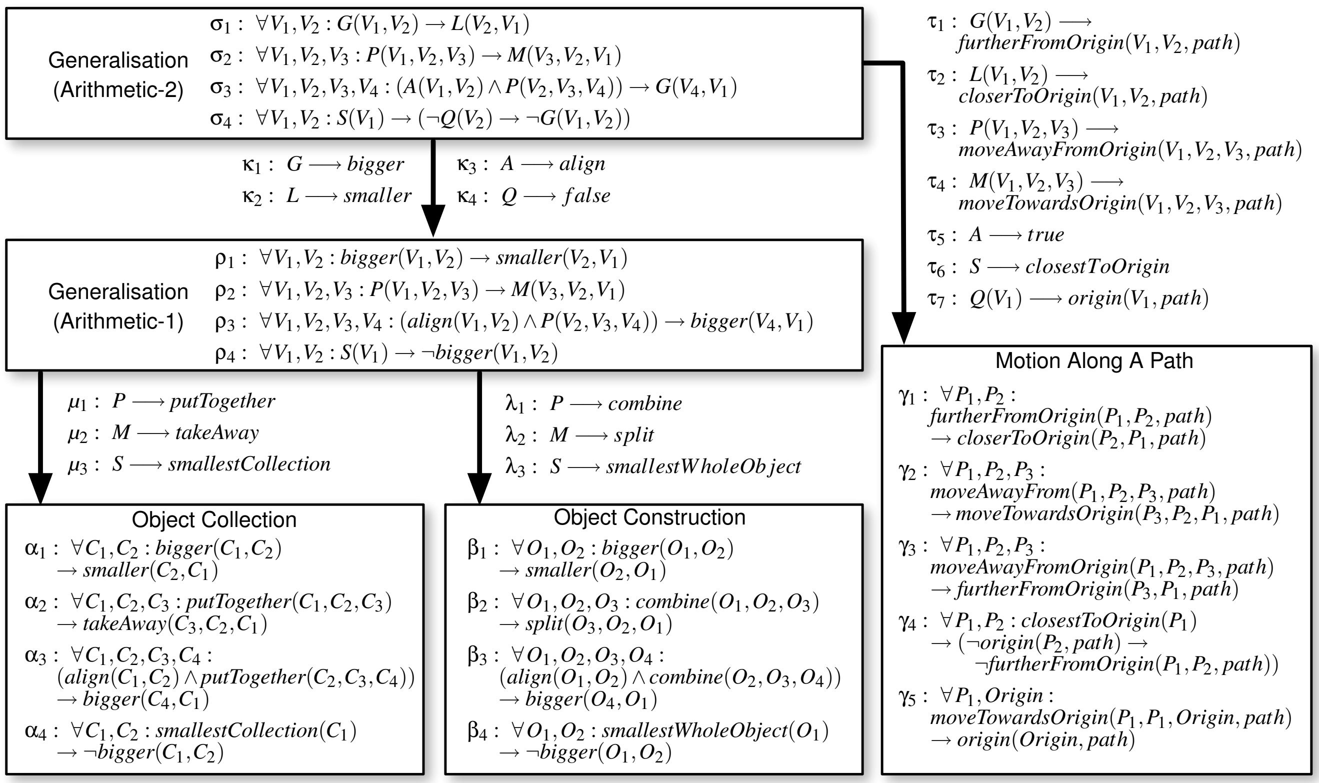

Generalisation (Arithmetic-2)
$\sigma_1: \forall V_1,V_2: G(V_1,V_2) \rightarrow L(V_2,V_1)$
$\sigma_2: \forall V_1,V_2,V_3: P(V_1,V_2,V_3) \rightarrow M(V_3,V_2,V_1)$
$\sigma_3: \forall V_1,V_2,V_3,V_4: (A(V_1,V_2) \wedge P(V_2,V_3,V_4)) \rightarrow G(V_4,V_1)$
$\sigma_4: \forall V_1,V_2: S(V_1) \rightarrow (\neg Q(V_2) \rightarrow \neg G(V_1,V_2))$
$\kappa_1: G \longrightarrow bigger$
$\kappa_2: L \longrightarrow smaller$
$\kappa_3: A \longrightarrow align$
$\kappa_4: Q \longrightarrow false$
Generalisation (Arithmetic-1)
$\rho_1: \forall V_1,V_2: bigger(V_1,V_2) \rightarrow smaller(V_2,V_1)$
$\rho_2: \forall V_1,V_2,V_3: P(V_1,V_2,V_3) \rightarrow M(V_3,V_2,V_1)$
$\rho_3: \forall V_1,V_2,V_3,V_4: (align(V_1,V_2) \wedge P(V_2,V_3,V_4)) \rightarrow bigger(V_4,V_1)$
$\rho_4: \forall V_1,V_2: S(V_1) \rightarrow \neg bigger(V_1,V_2)$
$\mu_1: P \longrightarrow putTogether$
$\mu_2: M \longrightarrow takeAway$
$\mu_3: S \longrightarrow smallestCollection$
$\lambda_1: P \longrightarrow combine$
$\lambda_2: M \longrightarrow split$
$\lambda_3: S \longrightarrow smallestWholeObject$
Object Collection
$\alpha_1: \forall C_1,C_2: bigger(C_1,C_2) \rightarrow smaller(C_2,C_1)$
$\alpha_2: \forall C_1,C_2,C_3: putTogether(C_1,C_2,C_3) \rightarrow takeAway(C_3,C_2,C_1)$
$\alpha_3: \forall C_1,C_2,C_3,C_4: (align(C_1,C_2) \wedge putTogether(C_2,C_3,C_4)) \rightarrow bigger(C_4,C_1)$
$\alpha_4: \forall C_1,C_2: smallestCollection(C_1) \rightarrow \neg bigger(C_1,C_2)$
Object Construction
$\beta_1: \forall O_1,O_2: bigger(O_1,O_2) \rightarrow smaller(O_2,O_1)$
$\beta_2: \forall O_1,O_2,O_3: combine(O_1,O_2,O_3) \rightarrow split(O_3,O_2,O_1)$
$\beta_3: \forall O_1,O_2,O_3,O_4: (align(O_1,O_2) \wedge combine(O_2,O_3,O_4)) \rightarrow bigger(O_4,O_1)$
$\beta_4: \forall O_1,O_2: smallestWholeObject(O_1) \rightarrow \neg bigger(O_1,O_2)$
$\tau_1: G(V_1,V_2) \longrightarrow furtherFromOrigin(V_1,V_2,path)$
$\tau_2: L(V_1,V_2) \longrightarrow closerToOrigin(V_1,V_2,path)$
$\tau_3: P(V_1,V_2,V_3) \longrightarrow moveAwayFromOrigin(V_1,V_2,V_3,path)$
$\tau_4: M(V_1,V_2,V_3) \longrightarrow moveTowardsOrigin(V_1,V_2,V_3,path)$
$\tau_5: A \longrightarrow true$
$\tau_6: S \longrightarrow closestToOrigin$
$\tau_7: Q(V_1) \longrightarrow origin(V_1,path)$
Motion Along A Path
$\gamma_1: \forall P_1,P_2: furtherFromOrigin(P_1,P_2,path) \rightarrow closerToOrigin(P_2,P_1,path)$
$\gamma_2: \forall P_1,P_2,P_3: moveAwayFrom(P_1,P_2,P_3,path) \rightarrow moveTowardsOrigin(P_3,P_2,P_1,path)$
$\gamma_3: \forall P_1,P_2,P_3: moveAwayFromOrigin(P_1,P_2,P_3,path) \rightarrow furtherFromOrigin(P_3,P_1,path)$
$\gamma_4: \forall P_1,P_2: closestToOrigin(P_1) \rightarrow (\neg origin(P_2,path) \rightarrow \neg furtherFromOrigin(P_1,P_2,path))$
$\gamma_5: \forall P_1,Origin: moveTowardsOrigin(P_1,P_1,Origin,path) \rightarrow origin(Origin,path)$

Figure 2: Developing Arithmetic from Object Collection, Object Construction and Motion Along a Path

grounding metaphors. The original descriptions of the domains are only given informally, but we tried to stay as closely to original as possible. One possible axiomatisation in HDTP of the Object Collection domain is given in Table 4. Such a formalisation specifies the vocabulary that is used in the form of *sorts*, *entities* and *predicates* and then provides *facts* and *laws* to describe the structure of the domain. For example, axiom α_3 states that if two collections C_1 and C_2 can be aligned, i.e. all their objects can be paired up, and C_4 is created by putting C_2 and C_3 together, then C_4 will be bigger than C_1. Note that further formulae need to be added to get a complete axiomatisation, but such formulae can easily be introduced into the system as long as some elementary consistency constraints are satisfied. While adding more formulae to this formalisation might strengthen the support for a specific alignment, it does not necessarily introduce new mappings of concepts to other domains. An example for this is α_6, which states the transitivity of *bigger*. This formula embeds *bigger* further in the structure of the domain but does not introduce new concepts. *putTogether*, *takeAway*, *bigger* and *smaller* are considered core concepts of the Object Collection domain. In what follows, we will restrict our axiomatisations to such simple versions in which just the cores of the domains are represented and connected to each other. Furthermore, we will omit technical details as well as the specification of sorts and signatures for a more concise presentation.

Generalising two domains

We tested various alternative formalisations, which all resulted in HDTP being able to establish appropriate analogies. Here we present axiomatisations of the grounded domains that are compact and consistent and that import integral parts of the domains. Furthermore, we demonstrate how the transfer of knowledge from one domain to another one works, because this is a hallmark of ‘interesting’ analogies.

In a first step, we generalise the domains of Object Collection and Object Construction. (We only use the basic version of the Object Construction domain here, which largely resembles Object Collection. This version is not sufficient to introduce the concept of fractions.) The axiomatisation of the two domains can be found in the two boxes in the bottom left of figure 2. The grounded domains are restricted to the operations that in arithmetic correspond to *greater*, *less*, *addition* and *subtraction*. The axioms α_i and β_i (for $i \in \{1,\ldots,4\}$) correspond to each other and are generalised in the obvious way by introducing individual variables and variables for operations. For example, the predicate *putTogether* of the Object Collection domain is identified with *combine* of Object Construction and generalised to a variable P. The substitutions μ_1 and λ_1 can be used to reconstruct the original expressions. Note that aligning corresponding clauses in formalisations is only done for the convenience of the reader; HDTP does not rely on such an ordering to find the best possible analogical mapping.

Table 4: Formalisation of the Object Collection domain

Sorts
coll
Entities
$singleton : coll$
Predicates
$smallestCollection : coll$
$bigger : coll \times coll$
$smaller : coll \times coll$
$equal : coll \times coll$
$putTogether : coll \times coll \times coll$
$takeAway : coll \times coll \times coll$
Laws
$\alpha_1 : \forall C_1 : coll, C_2 : coll :$
$bigger(C_1, C_2) \to smaller(C_2, C_1)$
$\alpha_2 : \forall C_1 : coll, C_2 : coll, C_3 : coll :$
$putTogether(C_1, C_2, C_3) \to takeAway(C_3, C_2, C_1)$
$\alpha_3 : \forall C_1 : coll, C_2 : coll, C_3 : coll, C_4 : coll :$
$align(C_1, C_2) \wedge putTogether(C_2, C_3, C_4) \to bigger(C_4, C_1)$
$\alpha_4 : \forall C_1 : coll, C_2 : coll :$
$smallestCollection(C_1) \to not(bigger(C_1, C_2))$
$\alpha_5 : \forall C_1 : coll, C_2 : coll :$
$equal(C_1, C_2) \to (\neg bigger(C_1, C_2) \wedge \neg smaller(C_1, C_2))$
$\alpha_6 : \forall C_1 : coll, C_2 : coll, C_3 : coll :$
$(bigger(C_1, C_2) \wedge bigger(C_2, C_3)) \to bigger(C_1, C_3)$
...
Facts
$\alpha_7 : smallestCollection(singleton)$
...

Refining the generalisation

The formulae computed above by generalising Object Collection and Object Construction serve as a first formalisation of elementary arithmetic (labelled Arithmetic-1 in figure 2). The variables introduced by anti-unification are regarded as entities and predicates of this new domain. Because the grounded domains chosen were very similar, and in particular because the grounded domains neither have the concept EMPTY COLLECTION nor EMPTY CONSTRUCTION, the system computes only a subtheory of arithmetic that lacks a neutral element with respect to the operation P (representing ADDITION). A second step of creating analogical mappings is needed to transfer the concept ZERO from a differently structured domain into our Arithmetic-1. This is achieved by the second generalisation between the formalisation of Motion Along a Path and Arithmetic-1 resulting in Arithmetic-2 depicted in figure 2.

The formalisation we chose for Motion Along a Path is different from the other domains in that the predicates take an extra argument, *path*, to indicate the path along which the motion occurs. As a consequence, higher-order anti-unification is applied which leads to the slightly more complex substitutions τ_1 to τ_7 and κ_1 to κ_4. As before, these substitutions can be used to reconstruct the source and target domains from the generalisation. A further point to note is that γ_4 contains an additional condition in comparison to ρ_4. This mismatch is handled by introducing a generalised predicate Q, which is mapped to *false* by κ_4 and therefore can be neglected in Arithmetic-1. However, this dummy predicate is used as a hint for refinement. It indicates that an elaborated version of ρ_4 might be used to describe Arithmetic-1, namely

$$\rho'_4 : \forall V_1, V_2 : S(V_1) \to (\neg Q(V_2) \to \neg bigger(V_1, V_2))$$

which mainly states that if V_1 is the smallest number, then either V_2 is ZERO or V_1 is not bigger than V_2. This new predicate Q can also be used for the transfer of additional formulae, e.g. γ_5 can be introduced into Arithmetic-1 resulting in

$$\rho_5 : \forall V_1, O : M(V_1, V_1, O) \to Q(O)$$

basically saying V_1 minus V_1 equals ZERO. Thereby, the basic ideas on ZERO are incorporated into Arithmetic-1 by refinement and transfer from the Motion Along a Path domain.

Goguen's notion of concept blending

Another important means to create new mathematical concepts is *concept blending*, in particular in the form presented by Goguen (2006). His figure 3, reproduced below, gives an overview. Each node in this graph corresponds to a *conceptual space* in the sense of Gärdenfors (2000), which, roughly speaking, is a subset of the system's knowledge. The arrows preserve the inferential structure from space to space, and the diagram commutes. Goguen does not discuss examples from arithmetic, but how from the concepts HOUSE and BOAT the concepts HOUSEBOAT and BOATHOUSE are created by concept blending. The G space contains generic elements, such as PERSON or OBJECT; the I spaces represent more specific conceptual spaces, in his example I_1 represents that a HOUSE is on LAND and that a PERSON LIVES in it and I_2 that a PASSENGER RIDES in a BOAT and that the BOAT is on WATER. Concept blending takes these conceptual spaces and maps them into another space (the B space) in a way that, for example, that the BOAT is mapped onto the PERSON LIVING in a HOUSE, resulting in the concept of a house in which the boat 'lives' – a BOATHOUSE.

$$\begin{tikzcd} & G \arrow[dl] \arrow[dr] \arrow[dd] & \\ I_1 \arrow[dr] & & I_2 \arrow[dl] \\ & B & \end{tikzcd}$$

Fauconnier and Turner (2002, pp. 242–245) discuss blends in arithmetic. Their presentation can be formulated in the form suggested by Goguen (they are a major influence on Goguens conception in the first place), thus giving an extension to the work described in this paper. For example, Lakoff and Núñez's extended version of the motion along a path metaphor supports an analogue of the rational numbers. Taking this as I_2 and object collection as I_1, a generalisation G can be found as above, which ignores the division operation of I_2. Forming the blend B then allows the extra operation to be incorporated into a conceptualisation which respects the generalisation. The blend can be seen as an updated view of I_1:

> Once we have the blend, and reify it, we can adopt the view that the previous conception of number was 'miss-

ing' several numbers that were 'there' but not yet 'discovered'. (Fauconnier & Turner, 2002, p. 244)

Conclusions and future work

We examined to which extent the cognitive processes underlying mathematical thinking can be made formally precise and algorithmically operationalised. For this purpose we took the mathematical metaphors of Lakoff and Núñez (2000) and used the analogy engine HDTP to compute generalisations from the basic source domains of arithmetic based on higher-order anti-unification.

For this, we used formalisations similar to the ones in our earlier approaches using Information Flow theory and created a first generalisation that contained the fundamental concepts of arithmetic. We extended the first generalisation produced by HDTP by incorporating a transfer of concepts, which added new concepts to the 'growing' domain of arithmetic (in our case the idea of a neutral element). Thus, anti-unification cannot only serve to find abstractions of two source domains but also to transfer concepts.

We also briefly described, how Goguen's concept blending is a direct extension of the HDTP approach. A paper detailing the role of concept blending for arithmetic and a treatment within the HDTP framework is currently submitted.

The demonstrated generalisation examples are still quite simple. Enriching the domains to get more interesting transfer candidates is therefore the next step. This notion of 'interestingness' is central to a comprehensive treatment of mathematical discovery, because there is an unlimited number of possible theorems and theories, but only a fraction of these are deemed interesting and useful enough for mathematicians to consider. For automated theorem provers, this is a hard problem; one on which we expect the heuristic nature of the HDTP engine will shed more light.

The grounded domains as we used them here are already generalisations of concrete situations, e.g. for the object collection domain the person/system must already have abstracted over concrete instances of the acts of putting collections together and realised that this is a general law holding in this domain. HDTP should be suited to create these abstractions as well. A more pressing and fundamental case seems to be, however, to create an abstract, generalised number concept that extends beyond the subitising range, i.e. those cardinalities (ranging from one to three or four) for which humans don't need to count but immediately perceive the number of objects and which seem to be innate.

Some other directions in which to extend our work are: (1) How are the results influenced by the order of generalisation? (2) How can the object construction domain be extended such that fractions (rational numbers) can be introduced into the domain of arithmetic? (This is Lakoff and Núñez's fraction extension of the basic metaphor.) (3) How can our approach be extended to include Lakoff and Núñez's *linking metaphors*, which are used for creating more abstract mathematical concepts.

Acknowledgments

The research reported here was supported by EPSRC grant EP/F035594/1 for the *Wheelbarrow* project.

References

Barwise, J., & Seligman, J. (1997). *Information flow: The logic of distributed systems*. Cambridge: Cambridge University Press.

Fauconnier, G., & Turner, M. (2002). *The way we think: Conceptual blending and the mind's hidden complexities*. New York: Basic Books.

Gärdenfors, P. (2000). *Conceptual spaces: The geometry of thought*. Cambridge, MA: MIT Press.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, *7*(2), 155–170.

Gentner, D., Bowdle, B. F., Wolff, P., & Boronat, C. (2001). Metaphor is like analogy. In D. Gentner, K. J. Holyoak, & B. N. Kokinov (Eds.), *The analogical mind: Perspectives from cognitive science* (p. 199-253). Cambridge, MA: MIT Press.

Gentner, D., & Markman, A. B. (1997). Structure mapping in analogy and similarity. *American Psychologist*, *52*(1), 45–56.

Goguen, J. (2006). Mathematical models of cognitive space and time. In D. Andler, Y. Ogawa, M. Okada, & S.Watanabe (Eds.), *Reasoning and Cognition: Proceedings of the Interdisciplinary Conference on Reasoning and Cognition* (pp. 125–128). Keio University Press.

Guhe, M., Pease, A., & Smaill, A. (2009). A cognitive model of discovering commutativity. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society.* Austin, TX: Cognitive Science Society.

Guhe, M., Smaill, A., & Pease, A. (2009a). A formal cognitive model of mathematical metaphors. In B. Mertsching, M. Hund, & Z. Aziz (Eds.), *KI 2009: Advances in Artificial Intelligence* (pp. 323–330). Berlin: Springer.

Guhe, M., Smaill, A., & Pease, A. (2009b). Using Information Flow for modelling mathematical metaphors. In *Proceedings of the 9th International Conference on Cognitive Modeling.*

Krumnack, U., Schwering, A., Gust, H., & Kühnberger, K.-U. (2007). Restricted higher-order anti-unification for analogy making. In *AI 2007: Advances in Artificial Intelligence* (pp. 273–282). Berlin: Springer.

Lakatos, I. (1976). *Proofs and refutations: The logic of mathematical discovery*. Cambridge: Cambridge University Press.

Lakoff, G., & Núñez, R. E. (2000). *Where mathematics comes from: How the embodied mind brings mathematics into being.* New York: Basic Books.

Plotkin, G. D. (1970). A note on inductive generalization. *Machine Intelligence*, *5*, 153–163.

Schwering, A., Krumnack, U., Kühnberger, K.-U., & Gust, H. (2009). Syntactic principles of Heuristic-Driven Theory Projection. *Journal of Cognitive Systems Research*, *10*(3), 251–269.