Using Information Flow for Modelling Mathematical Metaphors

Markus Guhe (m.guhe@ed.ac.uk)
Alan Smaill (A.Smaill@ed.ac.uk)
Alison Pease (A.Pease@ed.ac.uk)
School of Informatics, Informatics Forum, 10 Crichton Street
Edinburgh EH8 9AB, UK

Abstract

We argue for two points in this paper. Firstly, formal models can be a useful means for cognitive modelling, in particular for domains that traditionally already use this kind of model. Secondly, we present a formal model of how two of the grounding metaphors for arithmetic proposed by Lakoff and Núñez (2000) can be linked to basic notions of arithmetic using the infomorphisms of the Information Flow theory.

Keywords: formal model, logic, metaphor, mathematics, scientific discovery

The Cognition of Mathematics

As of yet, there is no cognitive model of the way in which people invent mathematical concepts. As part of our research on understanding the cognition of creating mathematical concepts we are working towards such a model (Guhe, Pease, & Smaill, 2009). We build on two streams of research: embodied conceptualisation, which analyses mathematical ideas as being constructed by the cognitive process of metaphor (Lakoff & Núñez, 2000), and societal conceptualisation based on Lakatos's (1976) philosophical account of the historical development of mathematical ideas. Both argue strongly against the 'romantic' (Lakoff and Núñez) or 'deductivist' (Lakatos) style in which mathematics is presented as an ever-increasing set of universal, absolute, certain truths which exist independently of humans. In contrast to this view, our main interest in the *Wheelbarrow* project is how mathematical concepts are formed and modified by the embodied and situated human mind.

While there are cognitive models of learning mathematics (eg Lebiere, 1998; Anderson, 2007), there are to our knowledge no models of how humans *create* mathematics. Collecting empirical data on how scientific concepts are created is difficult, and this is true for case studies as well as laboratory settings. Using case studies (see, for example, Nersessian, 2008) suffers from the problems that they are not reproducible (and therefore anecdotal) and that they are usually created in retrospect, which means that they are very likely to contain many rationalisations instead of an actual protocol of the thought processes. Using a laboratory setting in contrast (cf Schunn & Anderson, 1998) means that the experiment has to be designed in such a way that the participants are limited in their possible responses, ie their degree of freedom is limited and it is uncertain whether or how this is different form the unrestricted scientific process.

Lakoff and Núñez (2000) claim that the human ability for mathematics is brought about by two main factors: our embodied nature and our ability to create and use metaphors. They describe how starting from interactions with the environment we build up (more and more abstract) mathematical concepts by processes of metaphor and abstraction. More precisely, they distinguish two kinds of metaphors: grounding metaphors and linking metaphors (p 53). In *grounding* metaphors one domain is embodied and the other abstract, eg the four grounding metaphors for mathematics, which we will describe below. In *linking* metaphors, both domains are abstract, which allows the creation of more abstract mathematical concepts. For example, having established the basics of arithmetic with grounding metaphors this knowledge is used to create – among others – the concepts of points in space, spaces of any number of dimensions and functions (p 387).

We follow Gentner (1983; see also Gentner & Markman, 1997, p 48) in assuming that metaphors are similar to analogies. Gentner proposes that when comparing two concepts we can distinguish between analogies, metaphors, literal similarities or mere appearance similarities by looking at the number of relations and properties that (the representations of) the two concepts have in common. For analogies, mainly relations between concepts are matched, while for metaphors a larger amount of properties are involved. Thus, the distinction between analogy and metaphor is only a difference in degree.

According to Gentner's (1983, p 156) *structure mapping theory* the main cognitive process of analogy formation is a mapping between the (higher-order) relations of conceptual structures. Although we use this approach for creating computational cognitive models of mathematical discovery (see Guhe et al., 2009 for an ACT-R model using *path-mapping* – a realisation of structure mapping in ACT-R developed by Salvucci & Anderson, 2001), in this paper we will present a formal model that specifies the particular grounding metaphors that Lakoff and Núñez (2000) propose. This formalisation will be a basis for enhancing the ACT-R model.

Lakoff and Núñez's Four Basic Metaphors of Arithmetic

Lakoff and Núñez (2000, chapter 3) propose that humans create the conceptual space of arithmetic with four different grounding metaphors that create an abstract conceptual space from embodied experiences, ie interactions with the real world. Since many details are required for describing these metaphors adequately, we can only provide the general idea here.

Object Collection The first metaphor, *arithmetic is object collection*, describes how by interacting with objects we experience that objects can be grouped and that there are certain regularities when creating collections of objects, eg by removing objects from collections, by combining collections, etc. By the process of metaphor (analogy) these regularities are mapped into the domain of arithmetic, for example, collections of the same size are mapped to the concept of number and putting two collections together is mapped to the arithmetic operation of addition.

Object Construction Similarly, in the *arithmetic is object construction* metaphor we experience that we can combine objects to form new objects, for example by using toy building blocks to build towers. Again, the number of objects that are used for the object construction are mapped to number and constructing an object is mapped to addition.

Measuring Stick The *measuring stick* metaphor captures the regularities of using measuring sticks for the purposes of establishing the size of physical objects, eg for constructing buildings. Here numbers correspond to the physical segments on the measuring stick and addition to putting together segments to form longer segments.

Motion Along A Path The *motion along a path* metaphor, finally, adds concepts to arithmetic that we experience by moving along straight paths. For example, numbers are point locations on paths and addition is moving from point to point.

Note that these metaphors are not interchangeable. All are used to create the basic concepts of arithmetic. For this initial proposal we will only consider the first two metaphors.

Formal Models

The field of cognitive modelling makes only little use of formal methods.[1] A reason for this may be the recognition that traditional claims that logic describes the way humans reason do not stand up to scrutiny – at least not in this generality.[2] Consequently, logic is hardly used for modelling human cognition. However, this is throwing out the baby with the bath water, because the rigour of logical models is a great methodological advantage. Moreover, for the domain that we are interested in (the cognition of mathematics) the results of the cognitive processes (the mathematical structures and processes) are usually already modelled with logic, which makes them easy to use. Having said this, it is also clear that such models are on a high level of abstraction, one comparable to differential equations or statistics. An advantage of this high level is the models' conciseness, which makes it easy to have models with a broad coverage.

Artificial intelligence, mathematics and automated theory formation, which all mainly use formal models, usually do not consider the work carried out in cognitive modelling. A major aim of our project is to bring the research in these disciplines closer to the research in cognitive modelling. Cognitive modelling will also profit from our approach, because most of the work on the cognitive abilities we are investigating (linguistics, mathematics) is not done as cognitive modelling approach but formally. Instead of recreating this research in cognitive modelling it is advantageous to transfer or link the existing research in a principled manner to cognitive modelling.

Finally, cognitive modelling is only concerned with creating models of the mind. Only rarely is there a computational or formal characterisation of the properties of the model itself. For example, a cognitive reasoning model is not usually specified with respect to completeness (is the model able to make all valid deductions?) or soundness (are all inferences drawn by the model correct given the used premises?) Determining such properties of a model (theory/system) is a strong point of formal systems.

To illustrate that logic is still a useful way to describe cognition we would like to draw attention to the Wason selection task.[3] The apparent failures of humans in this task can convincingly be explained as being effects of the participants having problems 'with interpreting how the experimenter intends the task and materials to be understood' (Stenning et al., 2006, p 63). In the case of the Wason selection task Schooler (2001) and Stenning et al. (2006) demonstrate that the apparent shortcomings of the participants are due to their understanding the task as being an inductive information gathering task rather than a deductive one, where they are supposed to reason from a set of premises to a conclusion. It has been observed that participants have no problems drawing the conclusions desired by the experimenter in a task with the identical logical structure but framed as a task of, for example, reasoning about the drinking age of youngsters.[4] The reason, however, that the inferences are 'correct' in this case is not, as is often suggested, that the problem is set in a different domain (a social situation instead of an abstract logic task) but that the participants understand the goal of the task in the way the experimenter intends, namely as being a deductive task – for which logic is a good model.

General Reasoning with Local Processing

A major difference between formal and cognitive modelling is that formal models usually consider all the knowledge in the

[1] 'Formal' in the sense of *logic* or *mathematics*. Computational models are formal as well, of course, and as they are usually realised on digital computers, they are also logical models.

[2] To be fair, it should be noted that most logicians today would say that logic describes how humans *ought* to reason. However, we propose that formal theories can contribute to understanding cognition.

[3] In the Wason selection task, the participants are presented with four cards, showing letters and numbers, for example: A, B, 3, 8. They are given a rule like *If one side has a vowel, then the other has an even number*. The participants now have to decide, which cards must be turned to see whether this rule is correct for these four cards. They should turn as few cards as possible. (Stenning, Lascarides, & Calder, 2006, p 28) The failure consists in the fact that participants usually turn over more cards than is necessary to draw the requested conclusion. This is often considered to be an example of a *confirmation bias* (Ross & Anderson, 1982, p 149), ie the preference to seek information that confirms held beliefs instead of trying to disconfirm such beliefs.

[4] The cards show the name of a drink on one side and the age of the drinker on the other, eg whiskey, orange, 19, 16. The rule is *If you drink alcohol, then you must be over 18 years old*. The participants are instructed to check whether all drinkers follow the rule.

system in each step. However, it is clear that this is not how cognition works. Instead only a small subset of the available knowledge is used for each computational step such as the firing of a production rule. This reduction of the considered knowledge to what we call a *local context* (Guhe, 2007) is the main reason why cognitive processes require much less computational power than artificial systems. Furthermore, the reason that (natural) cognitive systems can cope with the complexities of the real world while an artificial system is either prone to fall off a cliff (not enough knowledge considered) or being caught by a predator (computations are too slow), is that current artificial systems are very bad at establishing suitable local contexts – if they do it at all.

A reason for this is that the idea of a localised processing is a big challenge for formal models, because not taking all available knowledge into account can introduce inconsistencies, which will almost inevitably cause the system to fail. However, the *Information Flow* theory by Barwise and Seligman (1997) provides just what is needed to define distributed, localised formal systems. This means, the system consists of multiple subsystems (classifications, theories, local logics) that are connected by *infomorphisms* in a formally sound way. This makes it almost ideally suited for describing cognition in a formal manner, because humans are not only good at establishing local contexts but also at connecting the local contexts.

Coming back to the reasons for using formal methods the advantage of using logic for cognitive modelling is that it provides a general-purpose mechanism for reasoning – which is a main motivation for inventing and using logics in the first place. Although it is clear that human reasoning is strongly influenced by the current task and the current task demands, there is also a general ability to reason from premises to conclusions. It seems wasteful to have, for example, a different version of modus ponens in each task model. This does not mean that we propose a 'logic module', just that a general reasoning ability exists somewhere in the system. It can be implemented with means provided by existing cognitive architectures.

We have three main cases in mind where such a general reasoning mechanism is useful. Firstly, it can be used as a general model of distributed reasoning: if the system knows something within a local context and also knows how this knowledge is connected to another local context, then there is a principled way to use this connection to reason about the distal local context. Secondly, on a local level the reasoning on the chosen local context retains all the desirable properties of the chosen logic (soundness, completeness). Thirdly, such a mechanism is a good way to approach cognitive mathematics, because the results of the cognitive process (the mathematical structures) are already represented formally.

Information Flow

This section provides a short introduction to Information Flow theory. We will focus on the aspects that we need for our formalisation; a detailed discussion of Information Flow can be found in Barwise and Seligman (1997). We only need three of the main notions for our purposes here: classification, infomorphism and channel.

Classification A *classification* $\boldsymbol{A}$ consists of a set of tokens tok($\boldsymbol{A}$), a set of types typ($\boldsymbol{A}$) and a binary classification relation $\models_{\boldsymbol{A}}$ between tokens and types. In this way, the classification relation classifies the tokens, for example, for a token $a \in \mathrm{tok}(\boldsymbol{A})$ and a type $\alpha \in \mathrm{typ}(\boldsymbol{A})$ the relation can establish $a \models_{\boldsymbol{A}} \alpha$.

Graphically, a classification is usually depicted as in left part of figure 1, ie with the types on top and the token on the bottom.

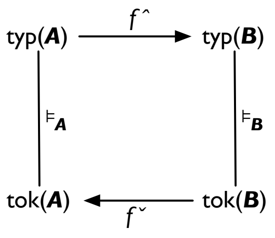
$$\begin{tikzcd}
\mathrm{typ}(\boldsymbol{A}) \arrow[r, "f^{\wedge}"] \arrow[d, dash, "\models_{\boldsymbol{A}}"] & \mathrm{typ}(\boldsymbol{B}) \arrow[d, dash, "\models_{\boldsymbol{B}}"] \\
\mathrm{tok}(\boldsymbol{A}) & \mathrm{tok}(\boldsymbol{B}) \arrow[l, "f^{\vee}"]
\end{tikzcd}$$

Figure 1: Two classifications ($\boldsymbol{A}$ and $\boldsymbol{B}$) and an infomorphism (f) in Information Flow

Infomorphism An *infomorphism* $f : \boldsymbol{A} \rightleftarrows \boldsymbol{B}$ from a classification $\boldsymbol{A}$ to a classification $\boldsymbol{B}$ is a (contravariant) pair of functions $f = \langle f^{\wedge}, f^{\vee} \rangle$ that satisfies the following condition:

$$f^{\vee}(b) \models_{\boldsymbol{A}} \alpha \quad \text{iff} \quad b \models_{\boldsymbol{B}} f^{\wedge}(\alpha)$$

for each token $b \in \mathrm{tok}(\boldsymbol{B})$ and each type $\alpha \in \mathrm{typ}(\boldsymbol{A})$, cf figure 1.

Note that the 'type relation' $f^{\wedge}$ and the 'token relation' $f^{\vee}$ point in opposite directions. (They are contravariant.) As a mnemonic the $^{\wedge}$ of $f^{\wedge}$ points upwards, where the types of classifications are usually written.

Channel A *channel* is a set of infomorphisms that have a common codomain. For example, the channel C depicted in figure 2 consists of a family of four infomorphisms f_1 to f_4 that connect the four classifications $\boldsymbol{A}_1$ to $\boldsymbol{A}_4$ to the common codomain $\boldsymbol{C}$. The common codomain is the *core* of the channel. Note that the infomorphisms of defining a channel are all pairs of functions, ie $f_1 = \langle f_1^{\wedge}, f_1^{\vee} \rangle$, etc.

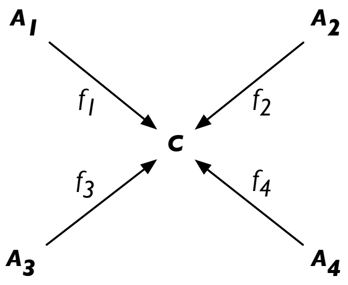
$$\begin{tikzcd}
\boldsymbol{A_1} \arrow[dr, "f_1"'] & & \boldsymbol{A_2} \arrow[dl, "f_2"] \\
& \boldsymbol{C} & \\
\boldsymbol{A_3} \arrow[ur, "f_3"] & & \boldsymbol{A_4} \arrow[ul, "f_4"']
\end{tikzcd}$$

Figure 2: Channel $C = \{f_1, f_2, f_3, f_4\}$ and its core $\boldsymbol{C}$

The core is the classification that contains the information connecting the tokens of the classifications $\boldsymbol{A}_1$ to $\boldsymbol{A}_4$. The tokens of $\boldsymbol{C}$ are called *connections*, because they connect the tokens of the other classifications. In our application to arithmetic the core is the arithmetic knowledge that represents what is common to the different source domains – the common arithmetic properties of object collections, object constructions, etc.

Channels and cores are the main way in which Information Flow achieves a distributed, localised kind of representing knowledge. In other words, this is the property of the Information Flow approach the fits to the localised representation and processing found in cognition. At the same time, infomorphisms provide a principled way of representing the connection between the different local contexts. This is not the place to go into the details about this aspect of Information Flow, but Barwise and Seligman (1997) give a comprehensive account of the properties that are or are not preserved when following an infomorphism from one classification to another one.

Formalisation of the Arithmetic Metaphors

The basic idea of how to apply Information Flow theory to the four basic metaphors of Lakoff and Núñez (2000) is that each domain (object collection, object construction, measuring stick, motion along a path and arithmetic) is represented as a classification and the metaphors/analogies between the domains are infomorphisms.

Information flow (which give the theory its name) captures regularities in the distributed system (see the First Principle of Information Flow, Barwise & Seligman, 1997, p 8). So, the infomorphisms between the four source domains and the core (arithmetic) capture the regularities that link these domains to arithmetic, and the arithmetic classification represents the knowledge of what these domains have in common. (A full arithmetic classification contains more than these commonalities – think of arithmetic concepts arising by linking metaphors like the concept of zero –, but for our current purposes it suffices to think of it this way.)

Object Collection

Classification We define a classification ***CL*** for the domain of object collections, cf table 1. The tokens of the object collection domain are actual physical instances of collections of objects that are or have been encountered by the cognitive agent. Formally, we represent them as sets of objects named $coll_A, coll_B, \ldots$

The tokens are classified by the size (cardinality) of the collection, ie types are sets with a number of distinct elements. Following Lakoff and Núñez (2000, p 55) we assume an innate or early developed subitising ability, ie the ability to determine the cardinality of small object collections of up to three or four objects. As a convention we write oc_1 for the type set with one object, oc_2 for the one with two objects, etc.

The classification relation $\vDash_{\boldsymbol{CL}}$ for object collections relates those sets for which each object of the token set can be mapped to exactly one object of the type set, ie no object of the token set and no object of the type set is left over and each object is mapped to exactly one element of the other set.

Table 1: The arithmetic is object collection metaphor.

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

Given this classification, we can now assign a type to each token, eg $coll_A \vDash_{\boldsymbol{CL}} oc_2, coll_B \vDash_{\boldsymbol{CL}} oc_1$. Figure 3 shows an example for an object collection with three objects.

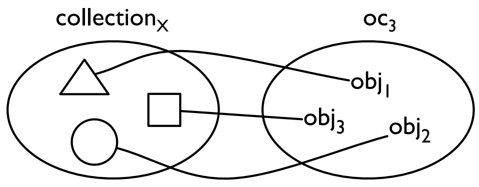

Figure 3: Example of the token–type relation for an object collection with cardinality 3

By proposing this classification we do not want to suggest that this is the only suitable classification for object collections; it is simply one that is suited for our goal of linking this source domain to arithmetic. A classification suitable for other purposes may be a classification by kind like blocks or balls.

Size The size of an object collection is the cardinality of the type set. Thus, given a token set $coll_A$, a type set oc_A with $coll_A \vDash_{\boldsymbol{CL}} oc_A$

$$size_{\boldsymbol{CL}}(col_A) = |oc_A|.$$

Smallest collection The smallest collection (the unit collection) is the type set with a single object, ie oc_1.

Bigger and smaller For comparing two object collections the type sets are aligned and a one-to-one mapping is established between the two type sets for as many elements as possible. The bigger collection is the one with at least one unmapped object. The smaller object collection is the other collection.

Formally, we define it as follows. Given two type sets oc_A and oc_B:

$$bigger_{\boldsymbol{CL}}(oc_A, oc_B) = \begin{cases} true, & \text{if } |oc_A| > |oc_B|, \\ false, & \text{if } |oc_A| < |oc_B|. \end{cases}$$

$smaller_{\boldsymbol{CL}}$ is the inverse.

Although not mentioned by Lakoff and Núñez (2000) in this context, two object collections are equal (in size) if they have the same type (or if they have the same cardinality):

$$equal_{\boldsymbol{CL}}(oc_A, oc_B) \text{ iff } oc_A = oc_B.$$

Putting collections together Putting separate and disjoint collections together is the union of token sets:

$$putTogether_{\boldsymbol{CL}}(coll_A, coll_B) = coll_A \cup coll_B.$$

The corresponding type is the type set that has a one-to-one-mapping between all objects in both sets. We write this as $oc_A \bullet_{\boldsymbol{CL}} oc_B$ (where $coll_A \vDash_{\boldsymbol{CL}} oc_A$ and $coll_B \vDash_{\boldsymbol{CL}} oc_B$), which is another possible type. Note that because the sets are disjoint $|oc_A \cup oc_B| = |oc_A| + |oc_B|$.

Taking a smaller collection from a larger collection Taking a smaller set from a larger set[5] is the set difference. Thus, for two object collections $coll_A$ and $coll_B$ with $coll_A \vDash_{\boldsymbol{CL}} oc_A$, $coll_B \vDash_{\boldsymbol{CL}} oc_B$ and $bigger_{\boldsymbol{CL}}(oc_B, oc_A)$:

$$takeSmaller_{\boldsymbol{CL}}(coll_A, coll_B) = coll_A \setminus coll_B$$

The corresponding type is $oc_A \circ_{\boldsymbol{CL}} oc_B$ (with $coll_A \vDash_{\boldsymbol{CL}} oc_A$ and $coll_B \vDash_{\boldsymbol{CL}} oc_B$), which is another possible type.

Object Construction

For the object construction domain we define a classification ***CN***. In contrast to ***CL*** the tokens and types of this classification are not defined as simple sets but as sets of sets. We name token sets as $cons_A, cons_B$, etc and type sets as ocn_A, ocn_B, etc. Note that in contrast of ***CL*** there is no implicit (and incidental) coding of size in the types of ***CN*** as the types can have just as complex substructures as the tokens, see the examples below.

Table 2: The arithmetic is object construction metaphor.

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 arithmetic 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

Smallest Whole Object A smallest whole object is represented as a singleton set for token sets, ie a set that contains one physical object that cannot be deconstructed. The corresponding type set is the set containing the empty set, ie $\{\varnothing\}$.

[5] Note that attempting to take a larger set from a smaller set has no physical correlate. This operation can only be performed in the abstract arithmetic domain, where it leads to the invention (or discovery) of negative numbers.

Size of Object The size of an object is the cardinality of the flattened type set. Thus, given a token set $cons_A$ and a type set ocn_A with $cons_A \vDash_{\boldsymbol{CN}} ocn_A$

$$size_{\boldsymbol{CN}}(cons_A) = |flat(ocn_A)|.$$

A flattened set is obtained by applying the function $flat$:

$$flat(\{A_1, A_2, \dots\}) = \begin{cases} \{A_1, flat(A_2, \dots)\}, & \text{if } |A_1| = 1, \\ \{flat(A_1), flat(A_2, \dots)\}, & \text{otherwise.} \end{cases}$$

$flat$ is defined on sets in general and is not restricted to ***CN***.

Bigger and smaller Analogous to the definition for object collections, bigger is defined as

$$bigger_{\boldsymbol{CN}}(ocn_A, ocn_B) = \begin{cases} true, & \text{if } size(oc_A) > size(oc_B), \\ false, & \text{if } size(oc_A) < size(oc_B). \end{cases}$$

$smaller_{\boldsymbol{CN}}$ is the inverse.

Constructed Object A constructed object is an object that consist of other objects. Thus, a constructed object is a set that has other object sets as elements. That is, given sets $cons_A$, $cons_B$, $\dots$ the constructed object set is $\{cons_A, cons_B, \dots\}$.

Whole Object Physical objects are always whole objects.

Putting Objects Together to form Larger Objects Putting objects together is defined as the union of the sets of object sets:

$$putTogether_{\boldsymbol{CN}}(cons_A, cons_B, \dots) = \{cons_A, cons_B, \dots\}.$$

The corresponding type is $ocn_A \bullet_{\boldsymbol{CN}} ocn_B$ (with $cons_A \vDash_{\boldsymbol{CN}} cns_A$ and $cons_B \vDash_{\boldsymbol{CN}} cns_B$), which is another possible type.

For example, given the object sets (o_X being a representation of a physical object) $\{\{o_1\}, \{o_2\}\}$, $\{\{o_3\}, \{\{o_4\}, \{o_5\}\}\}$ and $\{o_6\}$ the assembled object is $\{\{\{o_1\}, \{o_2\}\}, \{\{o_3\}, \{\{o_4\}, \{o_5\}\}\}, \{o_6\}\}$.

Taking Smaller Objects from Larger Objects to Form Other Objects Taking smaller objects from larger objects is defined as the set difference. More precisely, the result of this operation is a pair of sets – the difference set and the subtracted set. Thus, given two sets $cons_A$ and $cons_B$ with $cons_B \subset cons_A$

$$takeSmaller_{\boldsymbol{CN}}(\{cons_A, cons_B\}) = (\{cons_A \setminus cons_B\}, cons_B).$$

For example, given the object set $\{\{o_1\}, \{\{o_2\}, \{o_3\}\}, \{o_4\}\}$ the object $\{o_4\}$ can be removed by

$$\{\{o_1\}, \{\{o_2\}, \{o_3\}\}, \{o_4\}\} \setminus \{o_4\}$$

The smaller object taken from the larger object is then $\{o_4\}$ and the remainder of the larger object is $\{\{o_1\}, \{\{o_2\}, \{o_3\}\}\}$. The type of this operation is $ocn_A \circ_{\boldsymbol{CN}} ocn_B$ (with $cons_A \vDash_{\boldsymbol{CN}} cns_A$ and $cons_B \vDash_{\boldsymbol{CN}} cns_B$), which is another possible type.

Arithmetic

The classification $\boldsymbol{AR}$ representing arithmetic consists of the abstract numbers as tokens and concrete instances (uses) of numbers as types (for example as in *There are seven trees*).

The smallest number is 1.[6]

The arithmetic operations are defined as usual.

Metaphor Infomorphisms

With these three classifications, the infomorphisms between the two classifications representing the source domains ($\boldsymbol{CL}$ and $\boldsymbol{CN}$) and the core $\boldsymbol{AR}$ are straightforward.

Infomorphism from Object Collection to Arithmetic The infomorphism linking the object collection domain to arithmetic is defined as $f : \boldsymbol{CL} \rightleftarrows \boldsymbol{AR}$. The relation between types is then $f\hat{}(oc_A) = |oc_A|$, the relation between tokens $f\check{}(num) = coll_A$ where num is an arithmetic number and $coll_A$ is a representation of the physical object collection that the number refers to.

The smallest collection is $f\hat{}(oc_1) = 1$, and the comparison relations and operations are defined as follows:

- $f\hat{}(bigger_{\boldsymbol{CL}}) = \ >$
- $f\hat{}(smaller_{\boldsymbol{CL}}) = \ <$
- $f\hat{}(\bullet_{\boldsymbol{CL}}) = \ +$
- $f\hat{}(\circ_{\boldsymbol{CL}}) = \ -$

Infomorphism from Object Constructions to Arithmetic Similar to the definition above, the informorphism $g : \boldsymbol{CN} \rightleftarrows \boldsymbol{AR}$ relates types as $g\hat{}(obj_A) = |flat(obj_A)|$ and tokens as $g\check{}(num) = cons_A$, where $cons_A$ represents the object being referred to by the number and where $num \vDash_{\boldsymbol{AR}} \nu$, $cons_A \vDash_{\boldsymbol{CN}} cns_A$ with $size(cns_A) = \nu$. The other properties are defined as:

- $g\hat{}(\{\varnothing\}) = 1$
- $g\hat{}(bigger_{\boldsymbol{CN}}) = \ >$
- $g\hat{}(smaller_{\boldsymbol{CN}}) = \ <$
- $g\hat{}(\bullet_{\boldsymbol{CN}}) = \ +$
- $g\hat{}(\circ_{\boldsymbol{CN}}) = \ -$

Channel

Given these two infomorphisms, the channel C is the set of these two infomorphisms ($C = \{f, g\}$).

Conclusions and Future Work

We have argued that a formal approach like Information Flow can be used to great advantage for cognitive modelling. We provided a formalisation of the basic aspects of two of the grounding metaphors proposed by Lakoff and Núñez (2000) that humans use for creating arithmetic. Although this high level of modelling does not directly address human task performance, it offers important insights into the generalisations of the different source domains required to invent arithmetic.

We will extend our formalisation (1) to include the other basic metaphors and the linking metaphors within arithmetic; (2) by adding further notions from Information Flow to the formalisation, in particular regular theories and local logics. With these extensions we will be able not only to represent the required knowledge but also to model the corresponding processes.

[6]Note that there is no physical correspondence to 0. The absence of a physical object collection is not a collection. The object constructed of no object is not an object (it does not exist). Remember that historically 0 is a very late invention/discovery, a reaction to certain needs in arithmetic, cf also Lakoff and Núñez (2000).

Acknowledgments

The research reported here was carried out in the *Wheelbarrow* project, funded by the EPSRC grant EP/F035594/1.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

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

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

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

Guhe, M. (2007). *Incremental conceptualization for language production*. Mahwah, NJ: Lawrence Erlbaum.

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.*

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.

Lebiere, C. (1998). *The dynamics of cognition: An ACT-R model of cognitive arithmetic*. Unpublished doctoral dissertation, CMU Computer Science Department.

Nersessian, N. J. (2008). *Creating scientific concepts*. Cambridge: MIT Press.

Ross, L., & Anderson, C. A. (1982). Shortcomings in the attribution process. In D. Kahneman, P. Slovic, & A. Tversky (Eds.), *Judgment under uncertainty: Heuristics and biases* (pp. 129–152). Cambridge University Press.

Salvucci, D. D., & Anderson, J. R. (2001). Integrating analogical mapping and general problem solving: The path-mapping theory. *Cognitive Science*, *25*(1), 67–110.

Schooler, L. J. (2001). Rational theories of cognition in psychology. In W. Kintsch, N. J. Smelser, & P. B. Baltes (Eds.), *International encyclopedia of the social and behavioral sciences* (pp. 12771–12775). Oxford: Pergamon.

Schunn, C. D., & Anderson, J. R. (1998). Scientific discovery. In J. R. Anderson & C. Lebiere (Eds.), *The atomic components of thought* (pp. 385–427). Mahwah, NJ: Erlbaum.

Stenning, K., Lascarides, A., & Calder, J. (2006). *Introduction to cognition and communication*. Cambridge: MIT Press.