Embodied Enumeration: Appealing to Activities for Mathematical Explanation*

Albert Goldfain
Department of Computer Science and Engineering
Center for Cognitive Science
University at Buffalo
Buffalo, NY 14260
ag33@cse.buffalo.edu

Abstract

A computational cognitive agent capable of mathematical reasoning should be able to: (1) perform mental operations over abstract internal representations (e.g., counting numbers) and (2) perform physical operations over concrete external objects (e.g., counting apples). In a context of justification, both of these types of activity can contribute to mathematical explanations. In this paper, we focus on an agent's ability to enumerate external objects, the knowledge-level representational requirements (and side-effects) for such an ability, and the applicability of the enumeration result during an explanation. The SNePS knowledge-representation, reasoning, and acting system is used for a preliminary implementation of Cassie, our enumeration agent. The GLAIR architecture, which is used to model Cassie's embodiment, is also discussed.

Introduction: Computational Math Cognition

This paper reports on some results from a larger research project whose ultimate goal is a computational characterization of (developmentally) early mathematical cognition. Mathematical cognition is an interdisciplinary investigation into the cognitive foundations of mathematical reasoning. For the most part, this investigation has been concerned with *human* mathematical cognition (Butterworth 1999; Dehaene 1997; Gelman & Gallistel 1978; Lakoff & Núñez 2000; Piaget 1965; Wiese 2003), with fewer (but nevertheless important) contributions involving computational models of human mathematical reasoning (Bobrow 1968; Davis & Lenat 1982; Dellarosa 1985; Fletcher 1985; Ohlsson & Rees 1991). Besides telling us something about the human mind, the development of AI agents with mathematical cognition can benefit the agents themselves. Even an early human ability such as ordinal numeracy gives an agent a tool which is applicable to a variety of reasoning tasks.

Our specific interest in the field of mathematical cognition concerns the nature of mathematical understanding (as opposed to learning or the acquisition of concepts) and the extent to which a computational agent can demonstrate such understanding. It will therefore be useful to state some of our general views on mathematical understanding and how these views impact the agent requirements:

- *Mathematical understanding can be demonstrated by a question-and-answer dialogue*: An agent must be capable of explaining its procedural choices and justifying its conceptual beliefs. This demands a system in which the a "questioner" can trigger agent inference after (and perhaps during) a mathematical activity. The Turing Test (Turing 1950), usually taken to be a measure of natural-language understanding, can also measure mathematical understanding.
- *Mathematical understanding is activity-driven*: An agent must be capable of acting and expanding its knowledge during action. An episodic memory of the activity should be accessible to the agent so that it can recall its experience. Simply put, mathematics involves *doing* something, not just *thinking* about things.
- *Mathematical understanding is impacted by agent embodiment*: An agent must be capable of interacting with entities external to its "mind" and aligning such entities with mental symbols.
- *Mathematical understanding requires semantics*: Mathematical concepts and procedures must be understood *in terms of* other concepts and procedures. The basis for this recursive understanding is an internalized symbolic representation of concrete external entities (Rapaport 1995).

In the hopes of meeting the challenge of these additional requirements, we take a synthetic, bottom-up approach to agent design in which the (developmentally) early abilities of counting and arithmetic are the first provided for the agent.

In this paper, we identify a situation where, in the course of an explanation, an agent may appeal to the mathematical activity of enumeration for the justification of a result. We briefly discuss the nature of abstract counting routines and the multi-modal aspect of enumeration. SNePS, the implementation platform for our enumeration agent Cassie, is described along with the GLAIR architecture for embodied SNePS agents. Taking the act of visually enumerating apples as a motivating example, we spend the rest of the paper detailing a preliminary implementation of the enumeration

*I would like to thank William Rapaport, Stuart Shapiro, Douglas Clements, and the members of the SNePS research group for their comments on earlier versions of this work.

ability in Cassie. Throughout, we will consider the impact of the implementation on Cassie's knowledge-level representations (since this is the level with which Cassie will produce mathematical explanations).

Mathematical Explanations

In the following dialogue, a questioner tries to probe the addressee's understanding of the fact the greatest common divisor (GCD) of 8 and 6 is 2:

Q1: Why is 2 the greatest common divisor of 8 and 6?
A1: 2 is the greatest of the common divisors of 8 and 6.
Q2: What makes 2 the greatest of the common divisors?
A2: 2 is said later in the counting sequence than any of the other common divisors.
Q3: Why is 2 a common divisor of 8 and 6?
A3: 2 is a divisor of 8 and 2 is a divisor of 6.
Q4: Why is 2 a divisor of 6?
A4: There is a number that, when multiplied by 2, gives 6, and that number is 3.
Q5: Why is 2 times 3 = 6?
A5: Multiplication is repeated addition; 2 plus 2 is 4 and 4 plus 2 is 6
Q6: Why is 2 plus 2 = 4?
A6: When I count from 2 for two numbers I end up at 4
Q7: How do you know that you will end up at 4?
A7: I counted two groups of apples, both of which had 2 apples in them, and ended up with 4 total apples.
Q8: What is 2?
A8: It is a number and the greatest common divisor of 8 and 6.
Q9: What is a number?
A9: Some examples are 2,4,6 and 8 . . . It is something that can be counted, added, multiplied . . . and something that can be the result of finding a greatest common divisor

Although such a dialogue is rather long-winded (i.e., a human might answer a question like *Q6* with "It just is") it does show several characteristics of mathematical understanding. After such an explanation, it would seem quite unfair to claim that the addressee does not *understand* that the GCD of 8 and 6 is 2. We are interested in developing a computational cognitive agent that can produce such explanatory answers.

The addressee shows the ability to explain a complex procedure in terms of a simpler procedure (*A4,A5,A6*), the ability to manipulate the phrasing of a question to form a logical answer (*Q1-A1,Q3-A3*), and the ability to "define" a concept in terms of other known concepts (*A8,A9*). We wish to focus on the question and answer pair *Q7-A7*. All of the pairs except this one involve the performance of strictly "mental" operations by the addressee.[1] Thus, the entire dialogue except *Q7-A7* could, in theory, be carried out by a disembodied AI agent with no access to external input.

The answer *A7* is an appeal to an enumeration of concrete entities (apples). This is an empirical result that justifies the result of an abstract counting routine stated in *A6*. An agent that takes for granted that $2 + 2 = 4$ may "fake" the answer *A7* without actually performing the enumeration of apples. Yet it seems reasonable that there will be some situations where an agent will want to point to an experience (that could turn out one way or another) whose result has some mathematical significance. Furthermore, we shall see that having gone through with the activity will have positive experiential side effects which will make future performances of the activity easier.

Counting and Enumeration

We take counting to be the ability to generate a correct sequence of number names (or number symbols) starting from the number 1. We take enumeration to be the ability to count a set of (internal or external) entities (i.e., to make a cardinal assignment to this set which will stand for the "size" or "cardinality" of the set).

For standard counting of the natural numbers, a counting routine must include: (1) a representation of the single digit numbers and their standard ordering, and (2) syntactic rules for forming the successor of a multidigit number. By performing the activity of counting, an agent establishes the predecessor-successor relationship between each newly encountered number and its predecessor. This ordering of the natural numbers can then be used for enumeration. Once an agent has the ordering, each number in the ordering can be assigned to a member of the set being enumerated. For our agent, counting is a prerequisite for enumerating external objects (rather than one of its subroutines); counting generates the natural numbers for our agent and places them in a usable ordering.

As a base case, counting can be considered as an enumeration with the target entity being the number symbols (or names). Each number reached is assigned to itself. This is an internal enumeration because it does not require a model of embodiment. Other internal entities can be enumerated. An agent should be able to count its own actions (e.g., how many dinners have I made in the last hour) or its own thoughts (e.g., how many times have I thought about dinner in the last hour). An agent may also have to rely on its memory to generate mental images for enumeration when sensory information is not available. A common example in the mental imagery literature (e.g., (Kosslyn 1978)) is asking "How many windows are in your home?" when the addressee is away from their home. Intuitively, this procedure seems to require a mental image of the house to which enumeration can be applied.

Enumeration of external entities can be performed across multiple sensory modalities. A single abstract counting procedure underlies the visual enumeration of apples, the auditory enumeration of telephone rings, and the tactile enumeration of one's own teeth (using the tongue alone). What varies in each of these enumeration actions is the set of properties that distinguish an entity for a particular modality. These distinguishing properties will separate the entity from other entities in that sensory modality. Knowledge of shape and color will help in visual enumeration. Knowledge of duration and pitch will help in auditory enumeration. Knowledge of shape and hardness will help in tactile

[1] Assuming the addressee does not need an extended medium such as scrap paper.

enumeration. A combination of these distinguishing properties can be used when an agent has access to more than one sensory modality. Also, this allows for a single enumeration of entities with varying modalities (i.e., counting the number of apples AND rings of a telephone during a given time frame).

To support enumeration across modalities, an agent must: (1) be conscious of the modality/ies in use during enumeration, (2) be able to generate a set of distinguishing properties for the given modality/ies, (3) be able to retrieve the distinguishing properties from its knowledge base, and (4) be able to apply each distinguishing property within a sensory field (serially or in parallel) to determine if each candidate fits the criteria for being one of the enumeration targets.

Interestingly, modality information is usually dropped during a dialogue. The question "How many apples did you count?" is more naturally answered by "I counted three apples" than "I counted three apples visually (or with my vision)". The speaker abstracts away from how the apples were counted. Obtaining this information may require the further question "How did you count them?"

Reasoning with Quantities

It is important to realize that the task of enumeration involves quantities, numbers attached to units, and not just numbers alone. The unit a number attaches to is characterized in the philosophical literature as a sortal predicate, i.e., a predicate that "gives the criterion for counting items of that kind" (Grandy 2006). The sortal predicate specifies the distinguishing properties of the enumeration target.

So what counts as a sortal predicate? Consider the different units at play in the following example requests:

- Count the apples on the table.
- Count the pieces of fruit on the table.
- Count the interesting items on the table.

The units "apples on the table" and "pieces of fruit on the table" yield very different criteria for counting and "interesting items on the table" only yields a criterion for counting for some particular agent (in some sense, a "subjective" sortal predicate). Thus, for pragmatic reasons, we would like units whose distinguishing features can be algorithmically generated from an agent's knowledge base. Note, however, that for some particular agent, this may not exclude "interesting items on the table".

The attachment of a number to a unit is manifested linguistically as either a cardinal assignment (e.g., the three apples) or an ordinal assignment (e.g., the third apple) (Wiese 2003). Our implementation does not currently make the cardinal/ordinal assignment distinction for quantities because the enumeration task is one which determines the unknown cardinality (i.e., makes the cardinal assignment).

Models of Enumeration

The responsibilities assigned to low-level (subconscious) and high-level (conscious) processing give rise to two different models of enumeration. An early production-system implementation of both models was given by Klahr (1973). Both models are illustrated using the case of a single die in Figure 1. Under the model shown in (1a), low-level processing uses pattern recognition to automatically make a cardinal assignment to the number of dots on the face of the die. This ability is known as subitization in the literature (Clements 1999) and, in human beings, is considered an innate ability for small numerosities. A low-level pattern recognition routine must be coupled with a high-level addition routine that accumulates the subitized results.

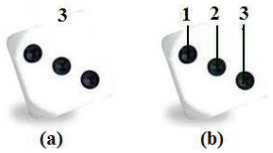

Figure 1: Two models of enumeration

Under the model shown in (1b), low-level processing uses object recognition to isolate each individual dot on the die. The agent uses a high-level routine which assigns a number to each dot as it is detected and marked in the visual field. The cardinal assignment for the set of dots is complete once the final dot has been assigned a number; this number is the cardinality for the entire set. This is the model we will use for our enumeration agent. An implementation of this model requires the following from the agent:

1. A system that coordinates the low-level object-recognition routine and the high-level cardinal assignment routine.
2. The ability to mark the counted entities in the sensory (e.g., visual) field.
3. The ability to detect when all target entities in the sensory field have been assigned a cardinal number.
4. The ability to direct attention to a new entity.
5. The ability to tag each counted entity with an ordinal number. This will enable a question like "Which did you count fourth?" to be answered during a dialogue.
6. The ability to tag the entire set with a cardinal number. This will enable a question like "How many did you count?" to be answered during a dialogue.

We now turn to a description of SNePS, our platform for implementing our enumeration agent.

SNePS and GLAIR

The SNePS knowledge-representation, reasoning, and acting system (Shapiro & Rapaport 1987; 1995) is used to implement Cassie (the SNePS cognitive agent). The underlying data structure of SNePS is a propositional semantic network representing Cassie's beliefs. SNePS is implemented as a package of Lisp and provides two user interface languages: SNePSUL (The SNePS User Language: a Lisp-like language) and SNePSLOG (The SNePS Logical Language: a Prolog-like language). SNePS creates a molecular node (i.e., a node that points to other nodes) for each proposition

in Cassie's belief space. The arcs of the semantic network are labeled with user-defined relations. A node with no arcs emanating from it is called a base node. Base nodes represent the concepts Cassie can reason with. Both numeric and linguistic concepts reside in the same semantic network in the form of base nodes. The SNePS representation of the proposition "3 is the successor of 2" is shown in Figure 2 (the '!' next to the molecular node name means that the proposition is believed by Cassie).

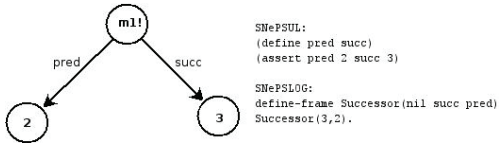

Figure 2: SNePS representation of the proposition `m1`: 3 is the successor of 2

SNeRE, the SNePS Rational Engine (Kumar 1993), is Cassie's model of acting. Part of Cassie's semantic network will represent actions that she can perform. These include primitive actions, the atomic actions from which all complex behaviors are built, and complex acts, which are sequences, iterations, and conditionals structuring a set of primitive and complex actions. Cassie performs a complex act `a` by attempting to find a plan `p` for its performance. This is represented in SNePSLOG as `ActPlan(a,p)`, the semantics being "`p` is a plan for performing `a`". Cassie then attempts to perform `p`. Usually `p` is structured by control acts and is composed of simpler actions. Every plan can eventually be decomposed into a sequence of primitive actions. Cassie's semantic network grows *while* she is acting, forming an episodic memory of the activity, which can be accessed for question answering after the activity (Goldfain 2006). For the larger task of explanation, it is also significant that procedural information (acts and plans) resides in the same semantic network as conceptual information (words and numbers).

SNIP, the SNePS Inference Package, provides for both node-based inference, which allow Cassie to infer new propositions using rules, and path-based inference, which allow Cassie to infer new relations from paths in her network (Shapiro 1978). Currently, we do not have a natural language interface for our enumeration agent implementation, so SNIP is our mechanism for asking Cassie questions.

GLAIR, the Grounded Layered Architecture with Integrated Reasoning (Hexmoor & Shapiro 1997; Shapiro & Ismail 2003), is used to simulate[2] Cassie's embodiment. A three-part distinction is made between an agent's knowledge layer (KL), perceptuo-motor layer (PML), and sensori-actuator layer (SAL). Shapiro *et al.* (forthcoming) defines the responsibilities of each layer. The following is a brief summary of these responsibilities:

1. **The Knowledge Layer (KL)** is the layer at which "conscious" reasoning takes place. The KL is implemented in SNePS and its acting subsystem SNeRE. In the KL, terms of the SNePS logical language represent the mental entities conceived of and reasoned about by the agent.
2. **The Perceptuo-Motor Layer, Sublayer a (PMLa)** contains the Common Lisp implementation of the actions that are primitive at the KL, *i.e.*, the routine behaviors that can be carried out without thinking about each step. PMLa is implemented in a way that is independent of the implementation of the lower layers.
3. **The Perceptuo-Motor Layer, Sublayer b (PMLb)** implements the functions of PMLa taking into account the particular implementation of the agent's body. PMLb is implemented in Common Lisp using, when necessary, its facilities for interfacing with programs written in other languages.
4. **The Perceptuo-Motor Layer, Sublayer c (PMLc)** contains the implementations of the PMLb functions for the particular hardware or software body being used.
5. **The Sensori-Actuator Layer (SAL)** contains the sensor and effector controllers of the agent body. For robotic embodiments, the SAL acts as a low-level interface to the hardware.

Implementation

In this section, we describe a preliminary SNePS implementation of the enumeration agent Cassie. Cassie is given the task of enumerating the apples in a 2D image which contains several types of fruit. Our implementation uses PML object-recognition with a KL counting routine (rather than a PML subitization with a KL accumulation routine). Since we are most interested in the KL level impact of the simulated embodiment, we can make several simplifying assumptions about the agent embodiment.[3] We limit Cassie to the modality of vision and the distinguishing properties color and shape. Each of the test images contains fruit displayed in a characteristic pose (e.g., a banana is presented lengthwise) with no occlusion (no overlapping fruit). The SAL layer of GLAIR is not required since the embodiment is only simulated. We also do not introduce a model of attention for our agent.

Symbolic representations at each level of GLAIR must be associated across layers to be useful during enumeration, this is known as symbol anchoring:

> Anchoring is achieved by associating (we use the term "aligning") a KL term with a PML structure, thereby allowing Cassie to recognize entities and perform actions, but not to discuss or reason about the low-level recognition or performance (Shapiro & Ismail 2001)

Such an alignment will have the effect of linking the type, subtype, and token representations of the entity being enumerated. Consider the various representations of apple:

- KL apple: The type apple. A lexical entry representing a class of objects whose extension includes real-world apples.

[2]GLAIR has been also been used as a physical embodiment for a robotic version of Cassie called FEVHAR (Shapiro 1998)

[3]A more robust agent embodiment for enumeration is given by Vitay (2005).

- KL apples-in-my-sensory-field: A symbol representing the external objects which Cassie is consciously aware of and can talk about.
- PMLb apple: The subtype apple. Any entity satisfying the criteria for being an apple as determined by the distinguishing properties retrieved from the KL.
- PMLc apple: The token apple. A symbolic presentation (set of pixels on a 2D image) of a specific instance of the type apple.

As we shall see, the same multi-layer association must be made for the distinguishing properties. We present a top-to-bottom description of our implementation in the KL and PML.

KL

Cassie's knowledge of apples will be represented in the KL as a set of molecular nodes expressing universal rules. These rules serve to link apples with modality-specific distinguishing properties. A subset of these might be:

- `m1`: Apples are round.
- `m2`: Round is a shape.
- `m3`: Apples are red.
- `m4`: Red is a color.
- `m5`: Apples are edible.

The semantic network representation of these propositions is given in Figure 3. Realistically, we should expect Cassie to

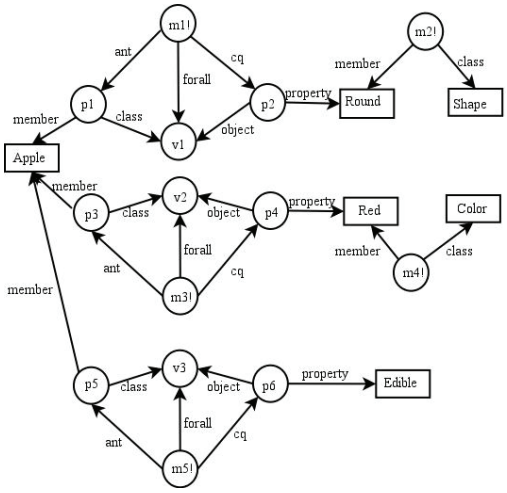

Figure 3: A part of Cassie's KL

also have knowledge involving apples that is completely irrelevant to enumeration. For example, she may believe that, in the context of the story of *Genesis*, Eve gave Adam an apple. This will not interfere in any way because the "apple" node at the KL represents the generic class of apples. This "apple" will remain static throughout the act of enumeration. We only use it to assert that this type of thing is the enumeration target.

Once Cassie is told to perform an enumeration, she must create a KL representation of the apples in her sensory field (call it *apples in my sensory field*). Unlike the type "apple", the base node representing this concept will be involved in a changing set of propositions during the act of enumeration. Before the enumeration, the concept is underspecified: "the arbitrary number of apples I am about to count". During the enumeration, this concept becomes "Apples I have counted so far". After the enumeration, this concept becomes a fully-specified quantity "The three apples I counted". The PML must "update" propositions involving *apples-in-my-sensory-field* during the enumeration. One PML interface to the KL is the `tell` function, which adds a SNePS proposition to Cassie's belief space from Lisp. This PML update has not yet been implemented, but a PML update mechanism for *apples-in-my-sensory-field* would involve a series of appropriately timed invocations of `tell` (see below).

The distinguishing properties for enumerating apples are represented as subtypes of more general concepts. The propositions represented by `m2` and `m4` will be used by the PMLa to extract the distinguishing properties of shape and color for the vision modality. As we shall see, the property of "being red" and the property of "being round" will be represented quite differently in the PML.

PML

The PML simulates Cassie's embodiment by providing low-level routines for perception and motor control. This is her "window" on the world; an interface between the simulated external objects and the KL. We consider each sublayer of the PML in turn:

PMLa The PMLa is the layer in which we implement Cassie's primitive actions. The PMLa is written in Lisp and utilizes the SNePS `define-primaction` function to create primitive actions and the `attach-primaction` function to associate each primitive action with a KL base node that names the action. All information flowing from the Cassie's "body" to her "mind" passes through the PMLa. The PMLa is responsible for coordinating the low-level routine of object recognition and the high-level routine of counting.

The crux of the PMLa is the `BuildEnumFrame(entity,modality)` primitive action. Given the name of the enumeration target (i.e., KL symbolic representation as a base node in SNePS) and the target modality for the enumeration (e.g., vision), this action isolates the relevant distinguishing properties for the entity. It puts these results in a frame (this is described as a feature vector in (Shapiro & Ismail 2001)).

From the PML, a call is made to the KL to determine how the slots of the enumeration frame will be filled for the modality of vision. This is done in via the path-based inference `find` command provided by SNIP. Once the modality of vision is passed in, the PML finds the relevant Shape and Color with:

```
(setf vis-mod-color
      #!((find (member- class) Color
         (property- ! object) ~entity)))
```

```
(setf vis-mod-shape
      #!((find (member- class) Shape
         (property- ! object) ~entity)))
```

The `vis-mod-color` slot is a node which has a path consisting of an inverse `member` arc followed by a class arc ending at the base node `Color` and has a path consisting of an inverse `property` arc going through an asserted molecular node to an object arc pointing to the entity (in our case, the base node `Apple`). The `vis-mod-shape` slot is filled in a similar way. Once constructed, the enumeration frame is passed down to the PMLb.

A second primitive action in our implementation is `Perceive(update-entity)`. In the case of enumerating apples, it will take a KL node to be updated *during* enumeration representing apples in the sensory field. This primitive action does the following:

1. Makes a request of the PMLb to obtain an image from the PMLc.
2. Initiates an object recognition task in the PMLb.
3. Awaits a result of the object recognition task. If a new object is recognized, it increments the count by finding the successor of the current total and updates the number assigned to *update-entity* in the KL, then it initiates another object recognition task. If no further objects are found it asserts the belief that the enumeration is complete.

The argument *update-entity* functions as an ordinal marker in the sequence of counted items. As discussed above, the unit must yield the distinguishing features of "entities of this kind" algorithmically (i.e., by using `BuildEnumFrame`).

PMLb The current PMLb is an Lisp implementation of Cassie's visual system. Each distinguishing property in the enumeration frame passed from the PMLa determines a test for detecting the property in a perceived object. These tests are applied serially. Each passed test is a vote of "yes" for the recognized object being the enumeration target (in terms of the property corresponding to that test).

The tests are performed in conjunction with a feature vector of prototypical values for the enumeration target. These represent Cassie's previous experience (if any) with the enumeration target. As a side-effect of enumeration, these values will be updated to reflect the agent's experience with recognizing the target.

The 2D image obtained from PMLc is stored as 3 two-dimensional array of pixel brightness values: a red array, a green array, and a blue array. The color value at pixel (i, j) is given by the triple $(red(i, j), green(i, j), blue(i, j))$.

The prototypical feature for "an apple's redness" is also a red-green-blue (RGB) triple, which we shall write as $(red_{apple}, green_{apple}, blue_{apple})$. The first test is to find the absolute difference between each pixel component-color and the corresponding prototypical component value. A summation of these values gives the component error for pixel (i, j):

$$\begin{aligned} Err(i,j) \quad = \quad & (red_{apple} - red(i,j)) + \\ & (green_{apple} - green(i,j)) + \\ & (blue_{apple} - blue(i,j)) \end{aligned}$$

A binary 2D array $Binary(i, j)$ is then created based on a predefined error threshold for color $Thresh_{color,apple}$.

$$Binary(i,j) = \begin{cases} 1, & \text{if } Thresh_{color,apple} \leq Err(i,j) \\ 0, & \text{otherwise} \end{cases}$$

The binary array represents the candidate object pixels after the first test. An idealized example of such an image is given in Figure 4(b). Realistically, the binary array will be far more noisy, with stray candidate pixels needing to be rejected by future tests.

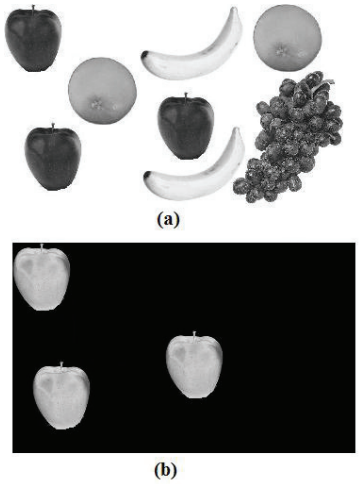

Figure 4: (a) Original 2D image (b) Ideal extracted object pixels

Shape is represented using an 8-direction chain-code representation (Sonka, Hlavac, & Boyle 1999). The chain code for a contiguous set of object pixels is a list of cardinal directions (i.e., N,NE,E,SE,S,SW,W,NW) on a path around the border of the object. We begin our chain code from the top left object pixel. A chain-code representation is illustrated in Figure 5 The utility of the chain code representation is that it

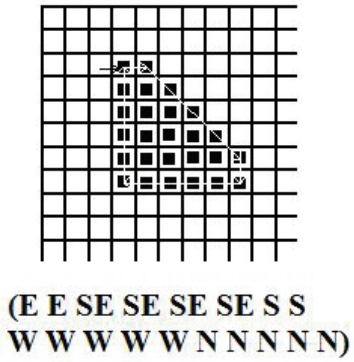

Figure 5: Chain-code representation of shape

provides a syntactic representation of shape that can be used

in further operations. Given a chain code $(d_1, d_2, \ldots d_n)$ for an object x, we have the perimeter $P(x)$ is:

$$P(x) = \sum_{i=1}^{n} l_i$$

where l_i is given by:

$$l_i = \begin{cases} \sqrt{2}, & \text{when } d_i = NE, SE, NW, SW \\ 1, & \text{when } d_i = N, E, S, W \end{cases}$$

It is also straightforward to compute the area of an object given a chain-code representation. We determine a bounding box around the object and sum up the number of object pixels inside this region.

Given the perimeter $P(x)$ and the area $A(x)$, the circularity of object x is given by its compactness $\gamma(x)$ (Sonka, Hlavac, & Boyle 1999):

$$\gamma(x) = 1 - \frac{4\pi A(x)}{P(x)^2}$$

The value of $\gamma(x)$ tends towards 0 as the shape of x tends towards a perfect circle. We classify the object as "circular enough to be an apple" based on whether the distance of the target object from perfect circularity falls under a predefined threshold. Circularity in the PML corresponds with the relevant KL concept "round". However, we do not want to check for *perfect* circularity in the target object, but an "apple's circularity". This would be achieved by retrieving the prototypical compactness for an apple $compact_{apple}$ and using it instead of 1 in the measure of "apple" circularity:

$$\gamma_{apple}(x) = compact_{apple} - \frac{4\pi A(x)}{P(x)^2}$$

Like the "redness of apples", this value is learned through experience with apples (including the act of enumeration). Also like color, the candidate object can be classified based on some threshold $Thresh_{compact,apple}$.

When an object is determined to be an apple (i.e., its color and shape fall under the threshold values), the object must be "marked" as counted in the visual field. This involves several operations:

1. Return a value of `t` to the PMLb, indicating that an object has been found.
2. If no object was found on this iteration, return a value of `nil` to the PMLb.
3. (Optionally) Storing the current cardinal number for the enumeration target set along with the current top-left object pixel for the recognized apple. This will enable the agent to point to the "apple that was counted n^{th} if it is asked to explain what happened during the counting procedure.
4. Update the prototypical feature vector for the enumeration target. Average the object pixel original image color values with the existing $(red_{apple}, green_{apple}, blue_{apple})$ values. Average the compactness of the recognized object with $compact_{apple}$.
5. Compute a bounding box around the recognized object and set all of the pixels of $Binary(i, j)$ to 0. This has the effect of "erasing" the object so that it will not be counted again.

PMLc The PMLc is Cassie's image acquisition mechanism. Images in JPEG format are exported to a C Source format using the GIMP (GNU Image Manipulation Project). A small C program writes out the pixel values in the RGB array form expected by Lisp. These arrays are then pasted by hand into the Lisp PMLc. In future implementations, we plan on automating image acquisition using the foreign-function call interface provided in Lisp.

Conclusion

An agent should be capable of appealing to an external activity during a mathematical explanation. External activities require an agent embodiment and the ability to perceive the external world. We have provided a detailed example and preliminary implementation of such an activity, the enumeration of apples, which can justify an arithmetic result of addition. Enumeration can be done across modalities and requires a manner of extracting the knowledge-level distinguishing properties for a given modality. We have seen that, in the GLAIR architecture, there is a hierarchy of representations for both the enumeration target and the distinguishing properties. Each representation is based on the processing task at that layer of embodiment. Thus, the color red may be useful as a base node in a semantic network, as a slot filler in an enumeration frame, and as a permissible range of RGB values on an 2D image pixel.

One of the interesting side effects of performing the act of enumeration is that the agent acquires experience in recognizing the object in the sensory field. We model such experience as a prototype feature vector in the PML. It would be interesting to examine how a significant change in PML prototypical values might effect the KL (e.g., how an agent who has only seen red apples might learn that apples can be either red or green from experience).

After performing an enumeration task, Cassie should be able to use the empirical result as part of a mathematical explanation. Although we have not yet implemented the explanation mechanism, we know that it must rely the knowledge available in Cassie's KL after the enumeration task. Enumeration leaves a "residue of knowledge" that is available for inference during question answering.

The requirements for mathematical understanding suggest a more thorough and complete agent design that accounts for the ability to perform embodied tasks. We believe that the design and implementation of such agents is an important research goal.

References

Bobrow, D. 1968. Natural Language Input for a Computer Problem Solving System. In Minsky, M., ed., *Semantic Information Processing*. Cambridge, MA: MIT Press. 133–216.

Butterworth, B. 1999. *What Counts: How Every Brain is Hardwired for Math*. New York, NY: Free Press.

Clements, D. H. 1999. Subitizing: What is it? Why Teach it? *Teaching Children Mathematics* 5(7):400–405.

Davis, R., and Lenat, D. 1982. AM: Discovery in Mathematics as Heuristic Search. In *Knowledge-Based Systems*

in Artificial Intelligence. New York, NY: McGraw-Hill. 3–225.

Dehaene, S. 1997. *The Number Sense*. Oxford, UK: Oxford University Press.

Dellarosa, D. 1985. SOLUTION: A Computer Simulation of Children's Recall of Arithmetic Word Problem Solving. Technical Report 85-148, University of Colorado at Boulder.

Fletcher, C. 1985. Understanding and Solving Arithmetic Word Problems: A Computer Simulation. *Behavioral Research Methods, Instruments and Computers* 17:365–371.

Gelman, R., and Gallistel, C. R. 1978. *The Child's Understanding of Number*. Cambridge, MA: Harvard University Press.

Goldfain, A. 2006. A Computational Theory of Inference for Arithmetic Explanation. In *Proceedings of the 5th Workshop on Inference in Computational Semantics (ICoS-5)*. 145–150.

Grandy, R. E. 2006. Sortals. In *Stanford Encyclopedia of Philosophy,* `http://plato.stanford.edu/entries/sortals/`.

Hexmoor, H., and Shapiro, S. C. 1997. Integrating Skill and Knowledge in Expert Agents. In Feltovich, P. J.; Ford, K. M.; and Hoffman, R. R., eds., *Expertise in Context*. Menlo Park, CA / Cambridge, MA: AAAI Press/MIT Press. 383–404.

Klahr, D. 1973. A Production System for Counting, Subitizing and Adding. In Chase, W. G., ed., *Visual Information Processing*. New York, NY: Academic Press. 527–546.

Kosslyn, S. M. 1978. Imagery and internal representations. In Rosch, E., and Lloyd, B., eds., *Cognition and categorization*. Hillsdale, NJ: Lawrence Erlbaum Associates. 217–257.

Kumar, D. 1993. A unified model of acting and inference. *Twenty-Sixth Hawaii International Conference on System Sciences Volume III* 483–492.

Lakoff, G., and Núñez, R. 2000. *Where Mathematics Comes From: How the Embodied Mind Brings Mathematics Into Being*. New York, NY: Basic Books.

Ohlsson, S., and Rees, E. 1991. The Function of Conceptual Understanding in the Learning of Arithmetic Procedures. *Cognition and Instruction* 8(2):103–179.

Piaget, J. 1965. *The Child's Conception of Number*. New York, NY: Norton.

Rapaport, W. J. 1995. Understanding Understanding: Syntactic Semantics and Computational Cognition. In Tomberlin, J., ed., *AI, Connectionism, and Philosophical Psychology, Philosophical Perspectives vol. 9*. Atascadero, CA: Ridgeview. 49–88.

Shapiro, S. C., and Ismail, H. O. 2001. Symbol-anchoring in cassie. In Coradeschi, S., and Saffioti, A., eds., *Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems: Papers from the 2001 AAAI Fall Symposium*. AAAI Press. 2–8.

Shapiro, S. C., and Ismail, H. O. 2003. Anchoring in a grounded layered architecture with integrated reasoning. *Robotics and Autonomous Systems* 43:97–108.

Shapiro, S. C., and Rapaport, W. J. 1987. SNePS Considered as a Fully Intensional Propositional Semantic Network. In Cercone, N., and McCalla, G., eds., *The Knowledge Frontier: Essays in the Representation of Knowledge*. New York, NY: Springer Verlag. 262–315.

Shapiro, S. C., and Rapaport, W. J. 1995. An Introduction to a Computational Reader of Narrative. In Duchan, J. F.; Bruder, G. A.; and Hewitt, L. E., eds., *Deixis in Narrative: A Cognitive Science Perspective*. Hillsdale, NJ: Lawrence Erlbaum Associates. 79–105.

Shapiro, S. C.; Rapaport, W. J.; Kandefer, M.; Johnson, F. L.; and Goldfain, A. forthcoming. Metacognition in SNePS. *AI Magazine*.

Shapiro, S. C. 1978. Path-based and node-based inference in semantic networks. In Waltz, D., ed., *Tinlap-2: Theoretical Issues in Natural Languages Processing*. New York, NY: ACM. 219–225.

Shapiro, S. C. 1998. Embodied Cassie. *Cognitive Robotics: Papers from the 1998 AAAI Fall Symposium, Technical Report FS-98-02* 136–143.

Sonka, M.; Hlavac, V.; and Boyle, R. 1999. *Image Processing, Analysis, and Machine Vision, Second Edition*. New York, NY: PWS.

Turing, A. M. 1950. Computing Machinery and Intelligence. *Mind* 59(236):433–460.

Vitay, J. 2005. Towards Teaching a Robot to Count Objects. *Fifth International Workshop on Epigenetic Robotics* 125–128.

Wiese, H. 2003. *Numbers, Language, and the Human Mind*. Cambridge, UK: Cambridge University Press.