AI Magazine Volume 28 Number 1 (2007) (© AAAI)

Metacognition in SNePS

Stuart C. Shapiro, William J. Rapaport, Michael Kandefer, Frances L. Johnson, and Albert Goldfain

■ The SNePS knowledge representation, reasoning, and acting system has several features that facilitate metacognition in SNePS-based agents. The most prominent is the fact that propositions are represented in SNePS as terms rather than as sentences, so that propositions can occur as arguments of propositions and other expressions without leaving first-order logic. The SNePS acting subsystem is integrated with the SNePS reasoning subsystem in such a way that: there are acts that affect what an agent believes; there are acts that specify knowledge-contingent acts and lack-of-knowledge acts; there are policies that serve as "daemons," triggering acts when certain propositions are believed or wondered about. The GLAIR agent architecture supports metacognition by specifying a location for the source of self-awareness and of a sense of situatedness in the world. Several SNePS-based agents have taken advantage of these facilities to engage in self-awareness and metacognition.

The terms *metacognition*, *metarepresentation*, and *metareasoning* have been defined in various ways, varying from general to specific: "Broadly defined *metacognition* is any knowledge or cognitive process that refers to, monitors, or controls any aspect of cognition" (Moses and Baird 1999, italics in the original). "Cognitive systems are characterized by their ability to construct and process representations of objects and states of affairs. Mental representations and public representations such as linguistic utterances are themselves objects in the world, and therefore potential objects of second-order representations, or "metarepresentations" (Sperber 1999). "Metareasoning is reasoning about reasoning—in its broadest sense, any computational process concerned with the operation of some other computational process within the same entity" (Russell 1999). "A metacognitive reasoner is a system that *reasons* specifically about itself (its knowledge, beliefs, and its reasoning process), not one that simply *uses* such knowledge" (Cox 2005, italics in original).

The SNePS[1] knowledge representation, reasoning, and acting system and its predecessor systems were designed from their beginnings to support metarepresentations (Shapiro 1971, 1979) (see Shapiro and Rapaport [1992]). More recently, SNePS-based cognitive agents, usually called "Cassie," have taken advantage of these facilities to have models of themselves and to have explicit acting plans that control their own beliefs (Shapiro and Rapaport 1987; Shapiro 1989, 1998; Santore and Shapiro 2003; Shapiro and Kandefer 2005). This is further facilitated by the grounded layered architecture with integrated reasoning (GLAIR) agent architecture that specifies how an agent acquires beliefs about its own actions and percepts and how it has a sense of its own situatedness in the world.

In this article, we will report on various aspects of SNePS, GLAIR, and Cassie that facilitate metacognition and give examples of metacognition in SNePS-based systems.

The GLAIR Architecture

GLAIR (Hexmoor et al. 1993, Hexmoor and Shapiro 1997, Shapiro and Ismail 2003), illustrated in figure 1, is a cognitive agent architecture with five layers.

The knowledge layer (KL) is the layer at which "conscious" reasoning takes place. The KL is implemented in SNePS and its subsystem

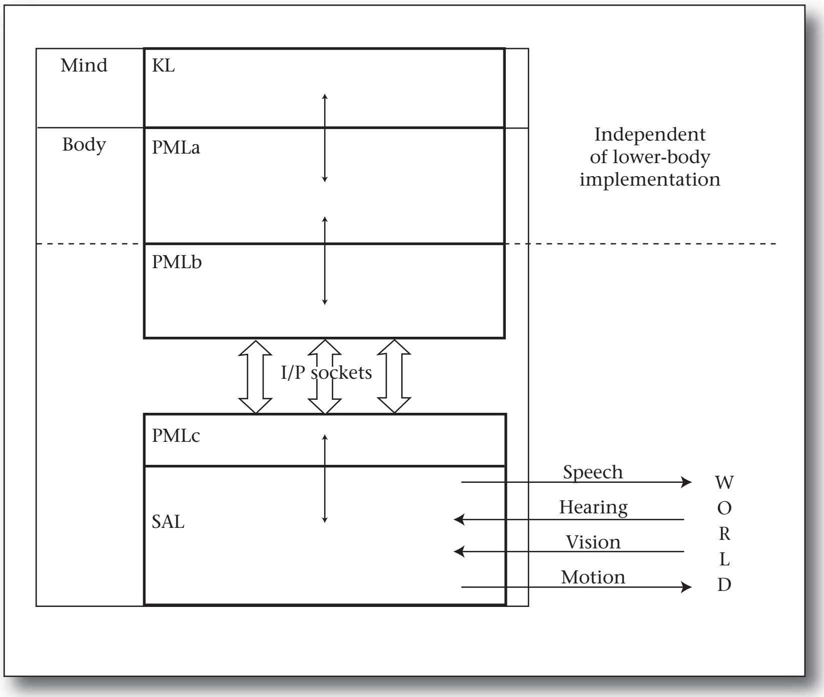

Figure 1. The GLAIR Architecture.

The KL layer is the agent's mind, the PML and SAL layers are its brain or body. The KL and PMLa layers are independent of whether the agent's body is implemented in software, virtual reality, or hardware. If the PMLc and SAL run on a different computer from the KL, PMLa, and PMLb, then the PMLb and PMLc communicate over IP sockets, one for each modality. The SAL controls the sensors and effectors.

SNeRE (the SNePS rational engine) (Shapiro and Rapaport 1992; Kumar 1996; Kumar and Shapiro 1994a, 1994b; Shapiro 2000b; Shapiro and The SNePS Implementation Group 2004). SNePS, in turn, is implemented in Common Lisp. In the KL, terms of the SNePS logical language represent the mental entities conceived of and reasoned about by the agent.

The perceptuo-motor layer, sublayer *a* (PMLa), contains the Common Lisp implementation of the actions that are primitive at the KL, that is, the routine behaviors that can be carried out without thinking about each step. PMLa is implemented in Common Lisp in a way that is independent of the implementation of the lower layers.

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.

The perceptuo-motor layer, sublayer *c* (PMLc), contains the implementations of the PMLb functions for the particular hardware or software agent body being used. PMLc has been implemented variously in C, Java, and the Ygdrasil (Pape et al. 2003) virtual reality authoring system.

The sensori-actuator layer (SAL) contains the sensor and effector controllers of the agent body.

SNePS

We view the contents of the SNePS knowledge base (KB), appropriately for its use as the agent's KL, to be the contents of the agent's mind. SNePS terms denote the mental entities of the agent, entities the agent has conceived of (Maida and Shapiro 1982, Shapiro and Rapaport 1987). Mental entities must include everything an agent can possibly conceive of, including particular individuals, classes, relations, acts, and, most significantly for this article, propositions (Shapiro 1993). The root of the SNePS ontology is thus entity. The next level, developed by considering what a SNePS-based agent can do with each entity, is currently proposition, act, policy, and thing. A *proposition* is an entity such that the agent can believe it or its negation. An *act* is an entity that the agent can perform. A *policy* is an entity, connecting propositions and acts, which the agent can adopt. A *thing* is an entity that is neither a proposition, act, nor policy.[2] Note that propositions, acts, policies, and things are exhaustive and mutually exclusive subclasses of entity.

A subset of the propositions represented in the KB are "asserted" and thus believed by the agent. The implemented SNePS rules of inference specify how the assertion status can be inherited by propositions from other propositions (Shapiro 1993, Chalupsky and Shapiro 1994). Open terms (terms containing unbound variables) do not exist in the latest version of SNePS, SNePS 3 (Shapiro 2000a), which is based on the logic L_A (Shapiro 2004), and is still being implemented. They also did not exist in the variant of SNePS discussed by Ali and Shapiro (1993).

An act may be performed by an agent and is composed of an *action* and zero or more arguments. For example, for the Fevahr[3] version of Cassie (Shapiro 1998), the term find(Bill) denotes the act of finding Bill (by looking around in a room for him), composed of the action find and the object Bill.[4] That is, an action is a represented by an act-valued function symbol.

Three policy-forming function symbols are built into SNePS, each of which take as arguments a proposition *p* and an act *a:* ifdo(*p, a*) is the policy that if the agent wants to know whether to believe p, it should perform *a*; whendo(*p, a*) is the policy that when the agent believes *p,* it should perform *a;* wheneverdo(*p, a*) is the policy that whenever the agent believes *p,* it should perform *a.*

A blocks-world example of ifdo is "If the agent wants to know whether block *A* is red, it should look at it": ifdo(ColorOf(A, red), lookAt(A)) (Kumar and Shapiro 1994a).[5]

In the case of both whendo and wheneverdo, if the policy has been adopted, the agent performs *a* when forward inference causes *p* to be believed. Also, *a* is performed if *p* is already believed when the policy is adopted with forward inference. The difference is that a whendo policy is unadopted after firing once, but a wheneverdo remains adopted until explicitly unadopted.

Things are all entities other than propositions, acts, and policies. Things mentioned so far in this section are the person denoted by Bill, the block denoted by A, the color denoted by red, the actions denoted by find and lookAt, and the relation denoted by ColorOf. Fevahr Cassie's things also include the category of robots, and the sentence "Neither a borrower nor a lender be," denoted by a functional term.

Bidirectional Inference

SNePS performs inference in a bidirectional fashion (Shapiro, Martins, and McKay 1982). A rule of the form all(x)(P(x) => Q(x)) can be used for backward chaining if an instance of Q(x) is a goal or subgoal, or for forward chaining if an instance of P(x) is asserted into the KB in an appropriate way. (When a belief is asserted into the KB, it may be done in a way that triggers forward chaining or not.) If both all(x)(P(x) => Q(x)) and all(x)(P(x) => R(x)) are asserted, and P(a) is not asserted, and Q(a) is issued as a goal or subgoal, and then P(a) is asserted with forward chaining, bidirectional inference focuses the forward chaining so that Q(a) is inferred, but R(a) is not.

Metaknowledge

Metaknowledge is knowledge about knowledge. More accurately, we are concerned with metabeliefs—beliefs about beliefs. Since propositions are represented by terms in the SNePS logic (Shapiro 1993; Shapiro 2000b), metabeliefs—which are propositions about propositions—are easily represented without leaving first-order logic. One example of a proposition represented by a term is from the Wumpus-World-agent version of Cassie (Shapiro and Kandefer 2005). There, Have is a function symbol, and Have(gold) is a proposition-valued functional term denoting the proposition that "I have the gold."

A SNePS proposition of the form P=>Q is not a sentence denoting true when P is false or Q is true; rather it is a functional term denoting the proposition that, from the agent's point of view, if I believe P, then I am justified in believing Q (Shapiro 1993, Chalupsky and Shapiro 1994). Notice that it is not "if I believe P, then I believe Q," because SNePS agents are not logically omniscient. If an agent believes P and P=>Q, then it will believe Q only when the "rule" P=>Q "fires" (to borrow terminology from production systems). Rather than being a function from truth values to truth values, => is a function from propositions to propositions, a proposition-valued function. Moreover, ~(P=>Q) simply denotes the proposition that belief in P is not justification for believing Q. So from ~(P=>Q) it does not follow that P, much to the consternation of some traditionalists. Similarly, P=>(Q=>R) denotes the proposition that if I believe P, then I am justified in believing Q=>R. Although P=>(Q=>R) and {P, Q}&=>R are logically equivalent,[6] in the sense that they both say that belief in R is justified when both P and Q are believed, they are treated differently by the SNePS inference system. If SNePS back chains into the latter rule, P and Q will be established as parallel subgoals, but if SNePS back chains into the former rule, just P will be a subgoal. Only if P is then inferred will Q=>R be believed and Q established as another subgoal. In either case, if SNePS forward chains into either P or Q, it will then back chain on the other, and R will be asserted if that back chaining is successful. From this point of view, P=>Q, ~(P=>Q), P=>(Q=>R), and {P, Q}&=>R are all examples of metabeliefs just as are the more obvious cases such as Believes(a, p), where a denotes some agent, p denotes some proposition, and Believes(a, p) is a proposition-valued functional term denoting the proposition that a believes p (see Wiebe and Rapaport [1986]; Shapiro and Rapaport [1991]; Rapaport, Shapiro, and Wiebe [1997]). The policies ifdo(p, a), whendo(p, a), and wheneverdo(p, a), although not beliefs about beliefs (because policies are not entities to be believed), are metapropositions in the sense that they are SNePS terms whose arguments are propositions and they are used to monitor beliefs.

Categories of Acts

SNePS acts may be categorized on two independent dimensions: an act may be either an external, a mental, or a control act; and an act may be either a primitive, a defined, or a composite act.

External, Mental, and Control Acts

SNePS actions and, by extension, acts, may be subclassified as either external, mental, or control. External acts either sense or affect the real, virtual, or simulated outside world. An example mentioned above from the Fevahr version of Cassie is find(Bill). No external acts are predefined in SNePS; they must be supplied by each agent designer.

Mental acts affect the agent's beliefs and adopted policies. There are four: believe(*p*) is the act of asserting the proposition *p* and doing forward inference on it; disbelieve(*p*) is the act of unasserting the proposition *p*, so that it is not believed, but its negation is not necessarily believed; adopt(*p*) is the act of adopting the policy *p*; unadopt(*p*) is the act of unadopting the policy *p*.

Before believe changes the belief status of a proposition *p*, it performs a limited form of belief revision (Alchourrón, Gärdenfors, and Makinson 1985). If andor(0, 0){..., *p*, ...} is believed,[7] it is disbelieved. If andor(*i*, 1){*p*, *q*, ...} is believed, for $i = 0$ or $i = 1$, and *q* is believed, *q* is disbelieved. Mental acts are metacognitive in that they are agent acts that explicitly affect what the agent believes.

Control acts are the control structures of the SNePS acting system. The SNePS control actions are achieve, do-all, do-one, prdo-one, snif, sniterate, snsequence, withall, and withsome.

If the proposition *p* is asserted, performing the act achieve(*p*) does nothing. Otherwise, performing the act achieve(*p*) consists of inferring plans for bringing about the proposition *p* (by inferring instances of the proposition GoalPlan(*p, x*)), and then performing do-one on them.

Performing the act do-all({a_1, ..., a_n}) consists of performing all the acts a_1, ..., a_n in a nondeterministic order.

Performing the act do-one({a_1, ..., a_n}) consists of nondeterministically choosing one of the acts a_1, ..., a_n, and performing it.

Performing the act prdo-one({pract(x_1, a_1), ..., pract(x_n, a_n)}) consists of performing one of the acts a_j, with probability $x_j / \Sigma_i x_i$.

Performing the act snif({if(p_1, a_1), ..., if(*pn*, *an*)[, else(*d*)]}) consists of using backward inference to determine which of the p_i hold, and, if any do, nondeterministically choosing one of them, say p_j, and performing a_j. If none of the p_i can be inferred, and if else(*d*) is included, *d* is performed. Otherwise, nothing is done.

Performing the act sniterate({if(p_1, a_1), ..., if(*pn, an*)[, else(*d*)]}) consists of using backward inference to determine which of the p_i hold, and, if any do, nondeterministically choosing one of them, say p_j, performing a_j, and then performing the entire sniterate again. If none of the p_i can be inferred, and if else(*d*) is included, *d* is performed. Otherwise, nothing is done.

Performing the act snsequence(a_1, a_2) consists of performing the act a_1 and then the act a_2.

Performing the act withall(*x*, *p*(*x*), *a*(*x*), [*d*]) consists of performing backward inference to find entities *e* such that *p*(*e*) is believed, and, if such entities are found, performing *a* on them all in a nondeterministic order. If no such *e* is found, and the optional act *d* is present, *d* is performed.

Performing the act withsome(*x*, *p*(*x*), *a*(*x*), (*d*)) is like performing withall, but if any entities *e* such that *p*(*e*) are found, *a* is performed on one of them, nondeterministically chosen.

Primitive, Defined, and Composite Acts

SNePS actions and acts may also be classified as either primitive, defined, or composite. Primitive acts constitute the basic repertoire of a SNePS agent. They are either provided by the SNePS system itself or are implemented at the PML. An example predefined action is believe; an example primitive action defined at the PML is the Fevahr Cassie's find. Because primitive actions are implemented below the KL, SNePS agents have no cognitive insight into how they perform them.

A composite act is one structured by one of the control acts. For example, the Wumpus-World Cassie, whose only primitive turning acts are go(right) and go(left), can turn around by performing the composite act, snsequence(go(right), go(right)).

A defined act is one that, unlike composite acts, is syntactically atomic, and unlike primitive acts, is not implemented at the PML. If a SNePS agent is to perform a defined act *a*, it deduces plans *p* for which it believes the proposition ActPlan(*a, p*), and performs a do-one of them. Such a plan is an act which, itself, can be either primitive, composite, or defined. For example, the Wumpus-World Cassie has a defined act turn(around) defined by ActPlan(turn(around), snsequence(go(right), go(right))).

Examples of Metacognition

In this section, we discuss five example SNePS projects that incorporate metacognition: self-awareness; lack-of-knowledge acting; consistency-maintenance and optimization; contextual vocabulary acquisition; and mathematical problem-solving.

```
all(p)(Person(p)
       => ActPlan(call(p), withsome(n,
                                    Has(p,PhoneNumber,n),
                                    dial(n),
                                    snsequence(lookup(p), call(p)))))
```

Figure 2. A Plan for Calling a Person.

Self-Awareness

If "a metacognitive reasoner is a system that reasons specifically about itself" (Cox 2005), it needs a model of itself. There are two aspects to our approach to self-modeling: representation techniques in SNePS at the KL; and PML-KL interaction. At the KL, the agent, itself, is represented by a SNePS term that is in no way different from the SNePS terms representing other agents. Beliefs about itself are represented as propositions containing the self-term as an argument. Beliefs about what the agent is doing and has done are represented using an interval model of time (Allen 1983), with the times principally connected by before and during relations (which each represent a disjunction of several of Allen's temporal relations). These beliefs form the agent's episodic memory.

The PML is involved in self-awareness in order to model an agent's sense of embodiedness and of being situated in the world. There are two aspects to this: a source of beliefs; and a locus of indexical and similar information. Beliefs about what the agent is doing and beliefs derived from the agent's sensory organs are put into the KL directly from the implementation of primitive sensory and efferent actions in the PMLa. For example, as mentioned previously, find is a primitive action for the Fevahr Cassie (Shapiro and Ismail 2003). The implementation of find in the PMLa inserts into the KL the belief that Cassie has found whatever she did. So after being told to find(Bill), Cassie believes, and then says (through her natural language [NL] generation component) "I found Bill." Having the agent's beliefs of what it senses and does inserted directly by the PML models these beliefs as "first-person privileged knowledge."

Another aspect of being situated in the world is continuous awareness of one's self and of one's general orientation in the world. We accomplish this through a set of PML deictic registers and modality registers, each of which is a variable whose value is one or a set of KL terms.

The deictic registers derive from the theory of the deictic center (Duchan, Bruder, and Hewitt 1995), and include: I, the register that holds the KL term denoting the agent itself; YOU, the register that holds the KL term denoting the individual the agent is talking with; and NOW, the register that holds the KL term denoting the current time. It is by using these registers that the NL generator generates "I found Bill" in first-person, in past tense, and using the third-person referring expression, "Bill." If she had been talking to Bill at the time, she would have said, "I found you" or "I found you, Bill."

Each modality register contains the KL term or terms representing the act(s) the agent is currently performing, or the percept(s) the agent is currently having in each effector or affector modality. The mechanisms for advancing NOW use these registers, as they change their contents, to place past acts and percepts in the past (Ismail 2001). For example, Fevahr Cassie starts out talking with Stu. If Stu then tells her to "Talk to Bill," and Bill asks her, "Who have you talked to?" she says,"I talked to Stu and I am talking to you."

Lack of Knowledge Acting

The optional default act in the SNePS control acts withall and withsome provide for acting on the basis of lack of knowledge (see Moore [1988]). We are using this facility for a SNePS solution to what we call McCarthy's second telephone number problem (McCarthy 1977). In this problem, the agent is asked to call someone. If the agent knows the person's phone number, it should make the call. However, if it doesn't, it must obtain the number from some directory or other agent, and then dial it. Obtaining the number is to result in the agent's knowing the person's phone number.

Our agent's plan for calling someone is depicted in figure 2, where Has(p, PhoneNumber, n) means that "*p*'s phone number is *n*." Figure 3 demonstrates a SNePS agent solving this problem. The agent engaged in an information-acquiring act because it lacked the information needed to perform another act. After acquiring the information, it performed the latter act, and then knew the information it acquired.

```
:  Person(?x)?
Person(Stu)
Person(Mike)

:  Has(?x,PhoneNumber,?y)?
Has(Mike,PhoneNumber,N(5,N(5,N(5,N(5,N(6,N(1,2)))))))

:  perform call(Mike)
I am pressing 5.  I am pressing 5.  I am pressing 5.  I am pressing 5.  I
am pressing 6.  I am pressing 1.  I am pressing 2.

:  perform call(Stu)
I could not find Stu?s phone number in any external information source
available to me.
Do you know Stu?s number?  yes
What is Stu?s number (e.g.  555-5555)?  555-7890
I am pressing 5.  I am pressing 5.  I am pressing 5.  I am pressing 7.  I
am pressing 8.  I am pressing 9.  I am pressing 0.

:  Has(?x,PhoneNumber,?y)?
Has(Stu,PhoneNumber,N(5,N(5,N(5,N(7,N(8,N(9,0)))))))
Has(Mike,PhoneNumber,N(5,N(5,N(5,N(5,N(6,N(1,2)))))))
```

Figure 3. The Agent Knows Mike's Phone Number, but Must Get Stu's Number from the User.

Material output by the agent is shown in bold. The colon is the system prompt. User input is in roman.

Metacognition in Consistency Maintenance and Optimization

Any computational system that stores and reasons with information must have some means for adding new information, changing existing information, and removing information. These belief change operations are especially needed to resolve contradictions (also called "consistency maintenance" or "truth maintenance").

SNePS uses a monotonic logic, so contradiction resolution requires that at least one base (input) belief that underlies a contradiction must be retracted to eliminate that contradiction.[8] The retraction process unasserts the belief, effectively removing it from the current base (also called the current context); both the contradiction and any derived beliefs that were dependent on that base belief are no longer derivable. The retracted belief is retained in the KB as an unasserted belief about which the system can still reason.[9]

Belief change is managed in SNePS by SNeBR, the SNePS belief revision system (Martins and Shapiro 1988, Shapiro 2000b). As implemented in SNePS 2.6.1, SNeBR detects contradictions, presents the user with possible culprits, provides the interface for base belief retraction, and removes from the active KB derived beliefs whose assertion status depended on any removed base beliefs. Automated retraction of base beliefs (Johnson and Shapiro 1999, Shapiro and Johnson 2000) and the belief base reoptimizing operation of reconsideration (Johnson and Shapiro 2005a, Johnson 2006) are implemented in other, developmental versions of SNePS.

Key guidelines for consistency maintenance are[10] when some beliefs are considered more important (or credible) than others, and consistency maintenance forces the removal of some beliefs, then the least important beliefs should be selected for removal; and any beliefs removed during consistency maintenance must be responsible for the inconsistency.

Because each belief is a term in the SNePS logic, metabeliefs can be used to indicate various levels of credibility. For example, we can order the beliefs Truck(Obj1) (object 1 is a truck) and Helicopter(Obj1) (object 1 is a helicopter)

```
Source(UAV, Truck(Obj1))
```
(An unmanned aerial vehicle (UAV) is the source of the information that Object 1 is a truck.)

```
Source(Satellite, Helicopter(Obj1))
```
(Satellite imagery is the source of the information that Object 1 is a helicopter.)

```
all(s1,s2,b1,b2)({Source(s1,b1),Source(s2,b2)}
    &=> {(sourceMoreCredible(s1,s2) => moreCredible(b1,b2)),
         (sourceEquallyCredible({s1,s2}) => equallyCredible({b1,b2}))})
```
(Information from a more credible source is more credible than information from a less credible source, but information from two equally credible sources are equally credible.)

Figure 4. Assigning Sources to Information and Ordering the Sources.

to indicate that the first is more credible by asserting moreCredible(Truck(Obj1), Helicopter(Obj1)). This belief credibility ordering enables the automation of contradiction resolution. Reasoning with the rule that all(x)(andor(0, 1){Truck(x), Helicopter(x)}) (an object cannot be both a helicopter and a truck) the system detects a contradiction and, following the guidelines listed above, removes the least credible belief[11] and reports,

I will remove the following node:
Helicopter(Obj1).

Ordering beliefs by their source or category (Johnson and Shapiro 1999, Shapiro and Johnson 2000) is more reasonable than ordering all beliefs individually. This results in a preorder over the beliefs; some sources may be equivalent (or incomparable) in strength, and beliefs from the same source (or equivalent sources) would have equivalent strength (unless there is an additional subordering of those beliefs).

Again, because each belief is a term in the language, assigning and ordering sources is easily represented in SNePS (figure 4). When we then input sourceMoreCredible(UAV, Satellite) (the UAV is more credible than satellite imagery), the system infers moreCredible (Truck(Obj1), Helicopter(Obj1)), which is used during automated contradiction resolution, as before.

Ordering beliefs is also helpful if a system needs to reoptimize the base. This can happen when new input and/or inferences result in changes being made to the base or its ordering. For example, if the new information forces contradiction resolution to remove some belief *p* from the base, then any belief *q* that is weaker than *p* and was removed because it conflicted with *p* should be considered for possible return to the base. Alternatively, if the new information reorders the sources or beliefs (or further refines the preordering), the base may need to be reoptimized with respect to this new ordering. This optimization process is called "reconsideration" (Johnson and Shapiro 2005a, Johnson 2006) and is implemented using the any-time algorithm of dependency-directed reconsideration (DDR) (Johnson and Shapiro 2005b, Johnson 2006).

DDR processes unasserted base beliefs that *might be* able to return to the current base due to a stronger conflicting belief being recently retracted or a reordering of the base beliefs. A belief can be reasserted if its reassertion either (1) does not raise an inconsistency or (2) each inconsistency its return raises can be resolved by retracting one or more strictly weaker beliefs (called belief swapping). Beliefs are processed in a nonincreasing order of credibility and are examined only if they are directly linked to some change in the belief base.

This scales well in the case where new information forces a retraction. Each inconsistency resolution typically involves a small subset of base beliefs, and reconsideration would involve a similarly small subset.[12] If a belief is returned with swapping, reconsideration continues but is progressing through ever decreasing credibility weights; if there is no swap—whether or not the belief is reasserted—reconsideration along that branch of the KB is pruned. The computational complexity of the DDR algorithm is minimal when a belief cannot return (no change to the base) and has a higher cost when the base is improved.

Reordering base beliefs (both asserted and unasserted) might occur when a source is discovered to be unreliable—forcing information from that source to be considered less credible than once thought. Reconsideration reoptimizes the base to reflect the new ordering by processing retracted base beliefs that are now more credible than conflicting base beliefs that are currently asserted. Unasserted base beliefs that have not changed their order relative to stronger conflicting beliefs need not be processed.

Any implemented system is restricted by the real-world limitations of the size of its memory and the time it takes to reason. There may be times when DDR must be interrupted so that further inferences can be made or actions taken. Because the logic used in SNePS is a paraconsistent logic,[13] the SNePS system can even reason in an inconsistent space—though, typically, the inconsistency needs to be unrelated to the propositions being used in the reasoning. When time allows, DDR can be recalled to reoptimize the KB and provide a more preferred belief base from which to act and reason.

Whether for contradiction resolution or base optimization, the system uses metabeliefs to determine what it should believe—ensuring that it maintains and reasons from the most credible knowledge base possible given the current constraints and needs of the user.

Metacognition and Contextual Vocabulary Acquisition

Reading is a cognitive activity that begins by requiring a great deal of conscious processing (in order to assign sounds to letters and meanings to words and sentences). Readers must think about which sounds go with which letters, and then whether the resulting combined sound is a word that they know. For good readers, this process evolves into an automatic process requiring little or no conscious activity (for example, when looking at a text in their native language, good readers cannot not read the passage, since reading has become so automatic). But, for excellent readers, it evolves once again into a conscious activity during which the reader monitors his or her reading (for example, in order to draw inferences about the passage being read). (See, for example, McNamara and Kintsch [1996], van Oostendorp and Goldman [1999], and Ruddell and Unrau [2004].) In short, for excellent readers, reading involves metacognition.

But even for readers who are not that excellent, reading contains metacognitive components. Moreover, we believe that practice with at least one of these components (vocabulary acquisition) can improve reading and comprehension skills. In our contextual vocabulary acquisition project,[14] we are developing a computational theory of how readers can "figure out" (that is, compute) a meaning for an unknown word from the textual context augmented by the reader's prior knowledge (Ehrlich 1995, Ehrlich and Rapaport 1997, Rapaport and Ehrlich 2000, Rapaport 2003, Rapaport 2005, Rapaport and Kibby 2007). This is a metacognitive task requiring readers to monitor their prior knowledge while reading, that is, to think about their thoughts in order to improve their vocabulary. We are adapting our theory (implemented in SNePS) to a school curriculum that, we hope, will help improve students' reading and comprehension abilities by showing them a precise series of steps they can follow (that is, by providing them with an algorithm) for figuring out meaning from textual context plus prior knowledge.

In this section, we present one example of how our algorithm works. In *Le Morte D'Arthur* (Malory 1470)—one of the earliest versions of the King Arthur legends—there occurs the following passage (here minimally paraphrased into modern English):

> Therewith two of them (that is, two evil knights) *dressed* their spears and Sir Ulfyus and Sir Brastias (Arthur's knights) *dressed* their spears; they ran together with great force. (Malory 1470, p. 15)

Even a very excellent reader who can manage to penetrate the archaic grammar (itself probably a metacognitive task) is likely to be puzzled by this use of "dress" given that for modern readers, "dress" means—without question—"to put clothes on." Did these knights really put clothes on their spears?

Thus, puzzled readers compare what they are reading with what they antecedently believed about dressing and spears in order to discover an inconsistency. This is therefore a metacognitive activity; metacognition tells the reader that this makes no sense. So perhaps we should take the passage metaphorically; perhaps the "clothes" they put on their spears were some sort of sheaths. But that, too, makes no sense: One would expect them to *remove* such sheaths before battle, not put them on. Such (conscious) thinking about a cognitive process is also a metacognitive task.

A bit later, Merlin advises King Arthur on the disposition of troops:

> When it is daylight, dress your battle right before the Northern host and the strong passage-way, so that they may see all your host. (Malory 1470, p. 19)

Here, the context strongly suggests that "battle" means "troops." Does this mean that the great King Arthur puts clothes on his troops before battle? Again, even a metaphorical reading according to which he helps his troops don their armor makes little sense; surely, King Arthur has better things to do with his time.

We have conflicts among what the text says, the reader's knowledge of the meaning of "dress," and the reader's prior beliefs that spears don't wear clothing and that kings don't put clothing on their troops.

Our implemented algorithm handles this situation as follows (as adapted from Rapaport and Ehrlich [2000]: First, the system's prior knowledge base for this example includes the following two beliefs: all(x, y)(dresses(x, y) => {wears(y, clothesOf(y)), Isa(clothesOf(y), clothing)}) (if *x* dresses *y*, then *y* wears *y'*s clothes); all(x, y)({Instance(x, spear), Isa(y, clothing)} &=> ~wears(x, y)) (spears don't wear clothing.)

Next, the system reads a sequence of passages containing "dress," adapted from Malory (1470), interspersed with questions and requests for definitions. First the system reads,

Passage D1. King Arthur dressed himself.

Then, asked to define "dress," the system produces,

Definition D1. A person can dress itself; result: it wears clothing.

Then it reads,

Passage D2. King Claudas dressed his spear ....

At this point, the system infers that King Claudas's spear wears clothing. However, challenged with

Question D2. What wears clothing?

A contradiction is detected, and SNeBR is invoked, automatically replacing the prior belief that if *x* dresses *y*, then *y* wears *y'*s clothes (rather than the prior belief that spears don't wear clothing, because of the occurrence of a verb in the antecedent, since people tend to revise beliefs about verbs rather than beliefs about nouns [Gentner 1981]). There follow several passages in the text in which dressing spears precedes fighting. Rather than *rejecting* the prior definition, it is modified. The system decides that to dress is either to put clothes on or to prepare for battle:

Definition D3. A person can dress a spear or a person; result: the person wears clothing or the person is enabled to fight.

Such disjunctive definitions are consistent with dictionarylike definitions of polysemous words.[15] We plan to investigate a method to induce a more general definition: In the present case, further experience with such phrases as "salad dressing," "turkey dressing," and so on, should lead the reader to decide that "dress" more generally means something like "prepare" (for the day, by putting on clothes; for battle, by preparing one's spear; for eating, by preparing one's salad; for cooking, by preparing one's turkey; and so on).

Metacognition arises in this situation in a variety of ways. First, there is metacognitive activity when the reader breaks out of the predominantly subconscious process of reading when faced with an inconsistency; the reader believes that what she or he is reading is inconsistent with other things that she or he believes. This belief about a belief is metacognitive. Second, when the reader decides to reinterpret what she or he reads in order to eliminate the inconsistency, there is metacognitive activity, because this is a decision about what to believe. It is a cognitive process (deciding) about another cognitive process (what to believe). Finally, the search for a more general meaning that subsumes the apparently inconsistent ones is metacognitive, because it involves reasoning *about* not merely *with* one's other beliefs.

Metacognition in a Math-Capable Computational Agent

Mathematical problem solving often requires some degree of metacognitive reasoning. The importance of metacognition in mathematics education is well known (Cohors-Fresenborg and Kaune 2001, Gartmann and Freiberg 1995, Schoenfeld 1992)) and should not be ignored when developing math-capable AI agents.

The applicability of multiple solution methods to any given problem is a driving force for metacognition in mathematics. The differences between solution methods may involve different mathematical contexts (for example, a student may know how to solve a problem both algebraically and geometrically) or subtle variations (for example, a child may count from either the smaller or larger addend when count-adding two numbers). Furthermore, a student may use (or improvise) a "nonstandard" solution method in certain problem contexts (for example, checking to see whether 4152342 is even to find the remainder of 4152342/2, rather than performing the long division as with the other numbers). These abilities suggest two metacognitive requirements for a math-capable AI agent: the ability to represent multiple procedures for the same task; the ability to select between procedures using contextual information.

In this section, we will describe how these abilities can be supported in SNePS agents. The focus will be on mathematical procedures at the level of counting and arithmetic.

```
all(x,y)({Number(x),Number(y)}
        &=> ActPlan(Add(x,y),
                    {CountAdd(x,y), ArithAdd(x,y), CalcAdd(x,y)})).
```

Figure 5. Specifying That There Are Three Procedures for Adding Two Numbers.

Multiple Plans for a Single Act

Even in the early mathematical domain of arithmetic, a plurality of procedures arises for each of the arithmetic operations. There are at least three ways two natural numbers, *x* and *y*, can be added: count-addition; arithmetic addition; and calculator addition.

In *count-addition,* $x + y$ is computed by counting from *x* for *y* numbers, and seeing what number has been reached. The result of the counting procedure is the result of the addition procedure.

In *arithmetic addition,* single-digit sums for all pairs of single-digit numbers are given. The numbers *x* and *y* are treated as strings of decimal digits, and the notion of a "carry" is used when a single-digit sum exceeds 9. The sum $x + y$ is computed by performing columnwise sums for each increasing place-value in *x* and *y,* and concatenating these results in the appropriate way.

In *calculator addition,* $x + y$ is computed by performing a sequence of button presses on a calculator (for example, entering *x,* pressing +, entering *y,* pressing =).

Each of these procedures specifies a potentially infinite set of ActPlans, one for each *x* and *y* pair.[16] These are specified in SNePS as shown in figure 5.

The propositions representing the result for each specific plan should be different, so that Cassie will remember which way of addition was used. The three ways of adding two numbers require three propositions: CountSum(x, y, z), ArithSum(x, y, z), and CalcSum(x, y, z). Whenever Cassie performs a series of arithmetic operations, she will have a trail of result propositions that indicate which plan was chosen at each step. This memory of how a plan was performed is essential for metacognitive reasoning.

In addition to specific result propositions, there should also be a general result proposition so the sum may be used in other operations without regard to how it was computed (figure 6).

When Cassie is asked to perform an operation, say perform CountAdd(2, 3), she will believe CountSum(2, 3, 5) and, through forward inference, Sum(2, 3, 5). The question "What is the sum of 2 and 3?" can be asked in two ways: Sum(2, 3, ?x)?, which is plan-independent; and CountSum(2, 3, ?x)?, which depends on Cassie having used the CountAdd method to obtain her sum.

Without any additional machinery, Cassie will nondeterministically select an addition plan whenever she is asked to perform Add(x, y). Metacognition requires that Cassie make a more informed decision. A plan should be chosen when it is an appropriate way of doing addition in the current context. To determine the contextual relevance of a plan, it will be useful to categorize the different kinds of arithmetic procedures.

Categorization of Mathematical Procedures

Clearly, there are many more procedures for addition than those given in the previous section. The three listed above are representative of three different procedural categories. Count-addition is a procedure in which an operation is performed by appealing to a more basic operation. In fact, we can build up all of the fundamental arithmetic operations in this way: addition is achieved by counting, subtraction is inverted addition, multiplication is repeated addition, and division is inverted multiplication. These are categorized as *semantic* procedures because they involve a semantic interpretation of the given procedure in terms of another (simpler) procedure. Arithmetic addition is a procedure characterized by syntactic operations in a positional number system and uses columnwise operations. This is done *without* a semantic interpretation for each of the single-digit sums (that is, it doesn't tell us what addition is in terms of another concept). Such procedures are categorized as syntactic procedures. Calculator addition requires nonmathematical operations (that is, button presses) and an external (trusted) source. Such procedures are categorized as extended procedures.

The borders between these procedural categories are not sharp ones. Arithmetic addition, for example, usually requires a student to mark

```
all(x,y,z)({CountSum(x,y,z), ArithSum(x,y,z), CalcSum(x,y,z)}
            v=> Sum(x,y,z)).
```

Figure 6. Regardless of How the Sum of x and y Was Computed, It Is the Sum of x *and* y*.*

```
all(x,y)({Number(x),Number(y)}
         &=> {(InContext(Explanation) => ActPlan(Add(x,y),CountAdd(x,y))),
              (InContext(ShowingWork) => ActPlan(Add(x,y),ArithAdd(x,y))),
              (InContext(Discovery) => ActPlan(Add(x,y),CalcAdd(x,y)))}).
```

Figure 7. Specifying Different Contexts for the Three Procedures for Adding x *and* y*.*

single-digit sums on a piece of paper; using an extended medium is a characteristic of an extended procedure. However, even with an imperfect categorization, it is important to assign some category to each of the agent's procedures. A procedure's category will be indicative of the situations in which a procedure will be most useful.

Semantic routines will be most useful for an agent in a context of explanation. This might include anything from a tutoring system for a human user to an agent taking a Turing test (in which mathematical understanding is tested alongside natural language understanding). Syntactic routines will be most useful when an agent must "show its work" or leave some indication of its activities. The utility of extended routines is very much dependent on the extended medium being used. In general, extended routines may be most useful in various contexts of discovery, or when a syntactic or semantic routine is simply unavailable. An agent that must obtain mathematical results quickly (perhaps reacting in a time-sensitive setting) and use those results without explanation could apply an extended routine that uses a calculator. Extended routines may also be applied when real-world objects (external to the agent) must be manipulated to find a solution. The problem context will ultimately determine which category is most appropriate.

Enforcing a problem context in SNePS can be done in two ways: (1) by extending the antecedent constraints for ActPlans and (2) by redefining the SNeRE do-one primitive act to guide Cassie's decision making.

Antecedent Constraints

An act such as Add(x, y) can be meaningfully performed by Cassie only if x and y are restricted to the domain of numbers. This is achieved through the Number(x) proposition. Given a contextual proposition InContext(x) indicating that the agent is in current context x, we might have figure 7. Unless each antecedent is satisfied, including the contextual proposition, Cassie will not consider acting upon an ActPlan. This is an a priori and "conscious" consideration (that is, it will explicitly appear in Cassie's chain of reasoning before any act is attempted). Further refinement of antecedent constraints will lead to more specific contextual behavior. For example, if an agent is told to add 2 and 3, CountAdd (that is, counting from 2 for 3 numbers) will not be an expensive operation. On the other hand, if an agent is adding 123 and 728123, CountAdd is very costly (that is, counting from 123 for 728000 numbers!). We might add antecedent constraints forcing an agent to perform CountAdd when the numbers are single digits (through a Digit(x) proposition) and ArithAdd for multidigit numbers.

Redefining the do-one Act

Another technique for enabling context-driven action is to modify the do-one primitive act. When Cassie is able to derive multiple plans for an act, the SNeRE executive cycle schedules a do-one on the set of those plans. In its unmodified form, do-one selects one of the plans at random. It can be modified to make a more deliberative decision by incorporating path-based reasoning (Shapiro 1991) to determine the current context (using an InContext(x) proposition as in the previous section) and the procedural category of each plan being considered. The latter piece of metaknowledge can be stored in a proposition ActCategory(x, y) (figure 8).

```
ActCategory(CountAdd(x,y), Semantic).
ActCategory(ArithAdd(x,y), Syntactic).
ActCategory(CalcAdd(x,y), Extended).
```

Figure 8. Giving Each Adding Procedure a Different Category.

A	B
$x - 2 = 0$	$x - 2 = 0$
$+2 \quad +2$	$-2 \quad -2$
$x = 2$	$x - 4 = -2$
	$+4 \quad +4$
	$x = 2$

Figure 9. Two Procedures for Solving $x - 2 = 0$ *for* x.

With an appropriately redefined do-one, making an assertion about context guarantees that the appropriate procedure is chosen (for example, selecting CountAdd(x, y) when InCategory(Semantic) is asserted).

In contrast to the method of adding antecedent constraints, this method is online and "subconscious" for Cassie (that is, it happens after Cassie has decided to perform the general act Add(x, y) and uses path-based inference to decide upon a plan in the PML layer).

Sensitivity to context (as provided by either of the above techniques) is a useful application of metacognition. It will also lead to a more humanlike behavior of the arithmetic task (that is, we tend to count-add on our fingers only for small numbers and sum up larger numbers in columns and rows).

Self Regulation

The capacity for control and self-regulation is an important feature of mathematical metacognition: "How well do you keep track of what you're doing when (for example) you're solving problems, and how well (if at all) do you use the input from those observations to guide your problem solving action" (Schoenfeld 1987, p. 190).

Given a standard algebraic problem such as $x - 2 = 0$, consider the two solution procedures (solving for x) in figure 9. Clearly procedure *A* is a more direct solution. Procedure *B* seems to be a roundabout way towards the solution. The "directness" or "roundaboutness" of a procedure is a metacognitive judgment. This judgment is based on the fact that procedure *B* uses more operations (of comparable complexity) than procedure *A*.

Goldfain (2005) describes a technique with which SNePS agents can enumerate their own actions *while they are acting*. By counting up the arithmetic routines in a mathematical task (for example, an algebraic task), an agent can get a rough sense of a plan's efficiency. An agent must be provided with enough logic to establish which operations are comparable in complexity (this is even more metacognitive information).

A math-capable agent should have some sense of whether each step in its reasoning is bringing it any closer to a solution. For the example above, adding y to both sides to produce $x - 2 + y = y$ leaves us with something more complex than the original problem. An agent who performs such a step should stop in its tracks; detecting that it is moving "farther" away from an answer. We would never teach a student a rule to *always* make every line of work "simpler" than the previous one. However, there are several mathematical procedures where this would at least be a good rule-of-thumb. A SNePS implementation of this ability is beyond the scope of this article, but it is worth further investigation.

Conclusions

The SNePS knowledge representation, reasoning, and acting system has several features that facilitate metacognition in SNePS-based agents. The most prominent is the fact that propositions are represented in SNePS as terms rather than as logical sentences. The effect is that propositions can occur as arguments of propositions, acts, and policies without limit, and without leaving first-order logic. The SNePS acting subsystem is integrated with the SNePS reasoning subsystem in such a way that there are acts (believe and disbelieve) that affect what an agent believes, there are acts (snif, sniterate, withall, withsome) that specify both knowledge-contingent acts and lack-of-knowledge default acts, and there are policies (ifdo, whendo, wheneverdo) that serve as "daemons," triggering acts when certain propositions are believed or wondered about.

The GLAIR agent architecture supports metacognition by specifying the PML as the source of self-awareness. Affective and effective actions implemented in the PML are the source of first-person privileged knowledge about what the agent is sensing and doing. Deictic and modality registers located in the PML provide the agent a sense of situatedness in its world.

Several SNePS-based agents have taken advantage of these facilities to engage in self-awareness and metacognition: Fevahr Cassie has a model of herself and episodic memory of what she has done; a SNePS agent uses lack-of-knowledge acting to acquire information it needs to make a telephone call; an enhanced SNePS belief revision system can use belief credibility and source credibility information, represented as metaknowledge, to do automatic belief revision on knowledge from multiple sources; a SNePS agent modifies its prior beliefs of word meaning to accommodate unusual usages in a text; a version of Cassie uses knowledge of the context to choose an appropriate method of arithmetic. The ongoing implementation and development of SNePS may allow for further applications of metacognition.

Acknowledgments

This work was supported, in part, by the U.S. Army Communications and Electronics Command (CECOM), Ft. Monmouth, NJ, through contracts with CACI Technologies and Booze·Allen & Hamilton. The authors appreciate the comments of other current members of the SNePS Research Group, and the contributions of the many past members.

Notes

1. SNePS (semantic network processing system) is available for download without charge from www.cse.buffalo.edu/sneps/Downloads.

2. In previous papers,"things" were called "individuals," but that was probably confusing, since this kind of entity includes classes and relations.

3. *Fevahr* is an acronym standing for "Foveal extra-vehicular activity helper-retriever."

4. Actually, since the Fevahr Cassie uses a natural language interface, the act of finding Bill is represented by the term act(lex(find), b6), where find is a term aligned with the English verb "find"; lex(find) is the action expressed in English as "find"; and b6 is a term denoting Bill. However, we will ignore these complications in this article.

5. *ifdo* was called *DoWhen* in Kumar and Shapiro (1994a).

6. The term $\{P_1, \ldots, P_n\}$ &=> $\{Q_1, \ldots, Q_m\}$ denotes the proposition that if all the P_i are believed then belief is justified in any or all of the Q_j (Shapiro 1979). If either *n* or *m* is 1, the set brackets can be omitted.

7. andor (Shapiro 1979) is a parameterized connective that takes a set of argument-propositions, and generalizes *and, inclusive or, exclusive or, nand, nor,* and *exactly n of.* A formula of the form andor(*i, j*)$\{P_1, \ldots, P_n\}$ denotes the proposition that at least *i* and at most *j* of the P_ks are true.

8. In this section, "propositions" are referred to as "beliefs," whether they are asserted or not.

9. Note the difference between the *current base,* which is the context composed of the input propositions that are *currently* believed, and the entire *knowledge base* (KB), which contains *all* base propositions (both asserted and unasserted) as well as derived beliefs. The active KB contains only asserted base beliefs and the beliefs that have been derived from them. It is important to retain noncurrently believed propositions when reasoning in multiple contexts (Martins and Shapiro 1983, Shapiro 2000b), and when allowing for reconsideration, as will be discussed later on.

10. These guidelines are paraphrased from Gärdenfors and Rott (1995) and Hansson (1997).

11. In this example, rules are preferred over observations. In the implementation of automated contradiction resolution (Johnson and Shapiro 1999, Shapiro and Johnson 2000), which is not included in the SNePS 2.6.1 release, if the system cannot identify a single belief for removal, it advises the user, who must make the final selection.

12. This assumes inconsistency maintenance uses dependency-directed, truth maintenance system (TMS) techniques (de Kleer 1986; Forbus and de Kleer 1993) as opposed to some "entire base" technique using SAT testing, where scalability is an issue.

13. It is *not* the case that everything is derivable from an inconsistency.

14. www.cse.buffalo.edu/~rapaport/CVA/.

15. Their resemblance to "weakening" in belief revision is worthy of further investigation; see, for example, Benferhat et al. (2004).

16. Because SNePS uses the unique variable binding rule (Shapiro 1986), which stipulates that no one term can substitute for different variables, separate ActPlans must be defined for adding a number to itself ("doubling a number").

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *The Journal of Symbolic Logic* 50(2): 510–530.

Ali, S. S., and Shapiro, S. C. 1993. Natural Language Processing Using a Propositional Semantic Network with Structured Variables. *Minds and Machines* 3(4): 421–451.

Allen, J. F. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(11): 832–843.

Benferhat, S.; Kaci, S.; Berre, D. L.; and Williams, M.-A. 2004. Weakening Conflicting Information for Iterated Revision and Knowledge Integration. *Artificial Intelligence* 153(1–2): 339–371.

Brachman, R. J., and Levesque, H. J., eds. 1985. *Readings in Knowledge Representation.* San Francisco: Morgan Kaufmann Publishers.

Chalupsky, H., and Shapiro, S. C. 1994. SL: A Subjective, Intensional Logic of Belief. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, 165–170, Hillsdale, NJ: Lawrence Erlbaum Associates.

Cohors-Fresenborg, E., and Kaune, C. 2001. Mechanisms of the Taking Effect of Metacognition in Understanding Processes in Mathematics Teaching. In *Developments in Mathematics Education in German-Speaking Countries, Selected Papers from the Annual Conference on Didactics of Mathematics,* 29–38. Ludwigsburg, Germany: Goettingen University Press.

Cox, M. T. 2005. Metacognition in Computation: A Selected Research Review. *Artificial Intelligence* 169(2): 104–141.

De Kleer, J. 1986. An Assumption-Based Truth Maintenance System. Artificial Intelligence, 28(2): 127–162.

Duchan, J. F.; Bruder, G. A.; and Hewitt, L. E., eds. 1995. *Deixis in Narrative: A Cognitive Science Perspective.* Hillsdale, NJ: Lawrence Erlbaum Associates .

Ehrlich, K. 1995. Automatic Vocabulary Expansion through Narrative Context. Ph.D. dissertation, Technical Report 95-09, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY.

Ehrlich, K., and Rapaport, W. J. 1997. A Computational Theory of Vocabulary Expansion. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society,* 205–210. Mahwah, NJ. Lawrence Erlbaum Associates.

Forbus, K. D., and de Kleer, J. 1993. *Building Problem Solvers.* Cambridge, MA: The MIT Press.

Gärdenfors, P., and Rott, H. 1995. Belief Revision. In *Handbook of Logic in Artificial*

Intelligence and Logic Programming, Volume 4, ed. D. M. Gabbay, C. J. Hogger, and J. A. Robinson, 35–132. Oxford: Oxford University Press.

Gartmann, S., and Freiberg, M. 1995. Metacognition and Mathematical Problem Solving: Helping Students to Ask the Right Questions. *The Mathematics Educator* 6(1): 9-13.

Gentner, D. 1981. Some Interesting Differences between Nouns and Verbs. *Cognition and Brain Theory* 4: 161–178.

Goldfain, A. 2005. Counting and Early Mathematical Understanding. Unpublished paper, State University of New York at Buffalo, Buffalo, NY (www.cse.buffalo.edu/~ag33/CAEMU.pdf).

Hansson, S. O. 1997. Semi-Revision. *Journal of Applied NonClassical Logic* 7(3): 151–175.

Hexmoor, H.; Lammens, J.; and Shapiro, S. C. 1993. Embodiment in GLAIR: A Grounded Layered Architecture with Integrated Reasoning for Autonomous Agents. In *Proceedings of the Sixth Florida AI Research Symposium* (Flairs 93), ed. D. D. Dankel and J. Stewman, 325–329. Gainesville, FL: The Florida AI Research Society.

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

Ismail, H. O. 2001. Reasoning and Acting in Time. Ph.D. dissertation, Technical Report 2001-11, University at Buffalo, The State University of New York, Buffalo, NY.

Iwanska, L. M., and Shapiro, S. C. 2000. *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. Cambridge, MA: AAAI Press/The MIT Press.

Johnson, F. L. 2006. Dependency-Directed Reconsideration: An Anytime Algorithm for Hindsight Knowledge-Base Optimization. Ph.D. dissertation, Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY.

Johnson, F. L., and Shapiro, S. C. 1999. Says Who? Incorporating Source Credibility Issues into Belief Revision. Technical Report 1999-08, Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY.

Johnson, F. L., and Shapiro, S. C. 2005a. Dependency-Directed Reconsideration: Belief Base Optimization for Truth Maintenance Systems. In *Proceedings of the Twentieth National Conference on Artificial Intelligence* (AAAI-05), 313–320. Menlo Park, CA: AAAI Press.

Johnson, F. L., and Shapiro, S. C. 2005b. Improving Recovery for Belief Bases. Paper presented at the IJCAI-05 Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC'05). 1 August, Edinburgh, Scotland.

Kumar, D. 1996. The SNePS BDI Architecture. *Decision Support Systems* 16(1): 3–19.

Kumar, D., and Shapiro, S. C. 1994a. Acting in Service of Inference (and Vice Versa). In *Proceedings of the Seventh Florida AI Research Symposium* (Flairs 94), ed. D. D. Dankel, 207–211. Gainesville, FL: The Florida AI Research Society.

Kumar, D., and Shapiro, S. C. 1994b. The OK BDI Architecture. *International Journal on Artificial Intelligence Tools* 3(3): 349–366.

Lehmann, F., ed. 1992. *Semantic Networks in Artificial Intelligence*. Oxford: Pergamon Press.

Maida, A. S., and Shapiro, S. C. 1982. Intensional Concepts in Propositional Semantic Networks. *Cognitive Science* 6(4): 291–330.

Malory, T. 1470, 1982. *Le Morte d'Arthur,* ed. R. M. Lumiansky. New York: Collier Books (page references are to the 1982 edition).

Martins, J. P., and Shapiro, S. C. 1983. Reasoning in Multiple Belief Spaces. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* 370–373. San Francisco: Morgan Kaufmann Publishers.

Martins, J. P., and Shapiro, S. C. 1988. A Model for Belief Revision. *Artificial Intelligence* 35(1): 25–79.

McCarthy, J. 1977. Epistemological Problems of Artificial Intelligence. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence* (IJCAI-77), volume 2, 1038–1044. Los Altos, CA: William Kaufmann, Inc.

McNamara, D. S., and Kintsch, W. 1996. Learning from Texts: Effects of Prior Knowledge and Text Coherence. *Discourse Processes* 22(3): 247–288.

Moore, R. C. 1988. Autoepistemic Logic. In *NonStandard Logics for Automated Reasoning*, ed. P. Smets, E. H. Mamdani, D. Dubois, and H. Prade, 105–127. London: Academic Press.

Moses, L. J., and Baird, J. A. 1999. Metacognition. In *The MIT Encyclopedia of the Cognitive Sciences,* ed. R. A. Wilson and F. Keil, 533–535. Cambridge, MA: The MIT Press.

Orilia, F., and Rapaport, W. J., eds. 1998. *Thought, Language, and Ontology: Essays in Memory of Hector-Neri Castañeda*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Pape, D.; Anstey, J.; Dolinsky, M.; and Dambik, E. J. 2003. Ygdrasil—A Framework for Composing Shared Virtual Worlds. *Future Generation Computer Systems* 19(6): 1041–1049.

Rapaport, W. J. 2003. What Is the "Context" for Contextual Vocabulary Acquisition? Paper presented at the 4th International Conference on Cognitive Science/7th Australasian Society for Cognitive Science Conference (ICCS/ASCS-2003), Sydney, Australia, 13–17 July.

Rapaport, W. J. 2005. In Defense of Contextual Vocabulary Acquisition: How to Do Things with Words in Context. In *Modeling and Using Context: 5th International and Interdisciplinary Conference* (Context 2005), Lecture Notes in Artificial Intelligence 3554, eds. A. Dey, B. Kokinov, D. Leake, and R. Turner, 396–409. Berlin. Springer-Verlag Publishers.

Rapaport, W. J., and Ehrlich, K. 2000. A Computational Theory of Vocabulary Acquisition. In *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language,* ed. L. Iwanska and S. Shapiro, 347–375. Cambridge, MA: AAAI Press/The MIT Press.

Rapaport, W. J., and Kibby, M. W. 2007. Contextual Vocabulary Acquisition as Computational Philosophy and as Philosophical Computation. *Journal of Experimental and Theoretical Artificial Intelligence*. Forthcoming.

Rapaport, W. J., Shapiro, S. C., and Wiebe, J. M. 1997. Quasi-Indexicals and Knowledge Reports. *Cognitive Science* 21(1): 63–107.

Ruddell, R. B., and Unrau, N. J. 2004. *Theoretical Models and Processes of Reading,* 5th ed. Newark, DE: International Reading Association.

Russell, S. J. 1999. Metareasoning. In *The MIT Encyclopedia of the Cognitive Sciences,* ed. R. A. Wilson and F. Keil, 539–541. Cambridge, MA: The MIT Press.

Santore, J. F., and Shapiro, S. C. 2003. Crystal Cassie: Use of a 3-D Gaming Environment for a Cognitive Agent. Paper presented at the IJCAI 2003 Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions, 9–11 August, Acapulco, Mexico.

Schoenfeld, A. H. 1987. What's All the Fuss about Metacognition? In *Cognitive Science and Mathematics Education,* ed. A. Schoenfeld, 189–215. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schoenfeld, A. H. 1992. Learning to Think Mathematically: Problem Solving, Metacognition, and Sense-Making in Mathematics. In *Handbook for Research on Mathematics Teaching and Learning,* ed. D. Grouws, 334–370. New York: Macmillan.

Shapiro, S. C. 1971. A Net Structure for Semantic Information Storage, Deduction and Retrieval. In *Proceedings of the Second International Joint Conference on Artificial*

Intelligence, 512–523. Los Altos, CA: William Kaufmann, Inc.

Shapiro, S. C. 1979. The SNePS Semantic Network Processing System. In *Associative Networks: The Representation and Use of Knowledge by Computers,* ed. N. V. Findler, 179–203. New York: Academic Press.

Shapiro, S. C. 1986. Symmetric Relations, Intensional Individuals, and Variable Binding. *Proceedings of the IEEE* 74(10): 1354–1363.

Shapiro, S. C. 1989. The Cassie Projects: An Approach to Natural Language Competence. In *EPIA 89: 4th Portugese Conference on Artificial Intelligence Proceedings,* Lecture Notes in Artificial Intelligence 390, ed. J. P. Martins and E. M. Morgado, 362–380. Berlin: Springer-Verlag.

Shapiro, S. C. 1991. Cables, Paths and "Subconscious" Reasoning in Propositional Semantic Networks. In Principles of Semantic Networks: Explorations in the Representation of Knowledge, ed. J. Sowa, 137–156. Los Altos, CA: Morgan Kaufmann Publishers.

Shapiro, S. C. 1993. Belief Spaces as Sets of Propositions. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)* 5(2–3): 225–235.

Shapiro, S. C. 1998. Embodied Cassie. In *Cognitive Robotics: Papers from the 1998 AAAI Fall Symposium*, Technical Report FS-98-02, 136–143. Menlo Park, CA: American Association for Artificial Intelligence.

Shapiro, S. C. 2000a. An Introduction to SNePS 3. In *Conceptual Structures: Logical, Linguistic, and Computational Issues,* Volume 1867, Lecture Notes in Artificial Intelligence, ed. B. Ganter and G. W. Mineau, 510–524. Berlin: Springer-Verlag.

Shapiro, S. C. 2000b. SNePS: A Logic for Natural Language Understanding and Commonsense Reasoning. In *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language,* ed. L. Iwanska and S. Shapiro, 175–195. Cambridge, MA: AAAI Press/The MIT Press..

Shapiro, S. C. 2004. A Logic of Arbitrary and Indefinite Objects. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference* (KR2004), ed. D. Dubois, C. Welty, and M.-A. Williams, 565–575. Menlo Park, CA: AAAI Press.

Shapiro, S. C., and Ismail, H. O. 2003. Anchoring in a Grounded Layered Architecture with Integrated Reasoning. *Robotics and Autonomous Systems* 43(2–3): 97–108.

Shapiro, S. C., and Johnson, F. L. 2000. Automatic Belief Revision in SNePS. Paper presented at the 8th International Workshop on NonMonotonic Reasoning (NMR2000), Breckenridge, CO, 9–11 April (www.cs.engr.uky.edu/nmr2000/proceedings.html).

Shapiro, S. C., and Kandefer, M. 2005. A SNePS Approach to the Wumpus World Agent, or Cassie Meets the Wumpus. Paper presented at the IJCAI-05 Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC'05). 1 August, Edinburgh, Scotland.

Shapiro, S. C.; Martins, J. P.; and McKay, D. P. 1982. Bi-Directional Inference. In *Proceedings of the Fourth Annual Meeting of the Cognitive Science Society,* 90–93. Mawah, NJ: Lawrence Erlbaum Associates.

Shapiro, S. C., and Rapaport, W. J. 1987. SNePS Considered as a Fully Intensional Propositional Semantic Network. In *The Knowledge Frontier,* ed. N. Cercone and G. McCalla, 263–315. Berlin: Springer-Verlag.

Shapiro, S. C., and Rapaport, W. J. 1991. Models and Minds: Knowledge Representation for Natural-Language Competence. In *Philosophy and AI: Essays at the Interface,* ed. R. Cummins and J. Pollock, 215–259. Cambridge, MA: The MIT Press.

Shapiro, S. C., and Rapaport, W. J. 1992. The SNePS Family. *Computers and Mathematics with Applications* 23(2–5): 243–275.

Shapiro, S. C., and the SNePS Implementation Group. 2004. SNePS 2.6.1 User's Manual. Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY.

Sperber, D. 1999. Metarepresentation. In *The MIT Encyclopedia of the Cognitive Sciences,* ed. R. A. Wilson and F. Keil, 541–543. Cambridge, MA: The MIT Press.

Van Oostendorp, H., and Goldman, S. R. 1999. *The Construction of Mental Representations during Reading*. Mahwah, NJ: Lawrence Erlbaum Associates, .

Wiebe, J. M., and Rapaport, W. J. 1986. Representing de Re and de Dicto Belief Reports in Discourse and Narrative. *Proceedings of the IEEE* 74(10): 1405–1413.

Stuart C. Shapiro is professor of computer science and engineering, affiliated professor of linguistics and philosophy, and director of the Center for Cognitive Science at the University at Buffalo, The State University of New York. He received an S.B. in mathematics from the Massachusetts Institute of Technology, and an M.S. and Ph.D in computer sciences from the University of Wisconsin–Madison. His primary research interests are in knowledge representation and reasoning, especially in support of natural language competence and cognitive robotics. He is a past chair of ACM/SIGART; past president of Principles of Knowledge Representation and Reasoning, Incorporated; a senior member of the IEEE; an ACM Distinguished Scientist; and a fellow of the AAAI. He can be reached at shapiro@cse.buffalo.edu.

William J. Rapaport (Ph.D., philosophy, Indiana University, 1976; M.S., computer science, University at Buffalo, 1984) is an associate professor in the Department of Computer Science and Engineering, an affiliated faculty member in philosophy and in linguistics, and a member of the Center for Cognitive Science, all at the University at Buffalo, The State University of New York. His research interests are in semantics, cognitive science, and philosophy of computer science.

Michael W. Kandefer is a Ph.D. student in the Department of Computer Science and Engineering at University at Buffalo, The State University of New York. He holds a B.S. in computer science from Canisius College and an M.S. in computer science from the University at Buffalo. His research interests include multiagent collaboration and action recognition. He can be reached by e-mail at mwk3@cse.buffalo.edu.

Frances L. Johnson is a member of the inference team at Cycorp, with interests in belief revision, machine learning, and knowledge state optimization as applied to large, real-world reasoning systems. She holds a B.S. in chemical engineering from Princeton University and an M.S. and a Ph.D. in computer science and engineering from the University at Buffalo, The State University of New York. She can be reached at Cycorp, Inc., 3721 Executive Center Dr., Austin, Texas 78731; fjohnson@cyc.com.

Albert Goldfain (ag33@cse.buffalo.edu) is a Ph.D candidate at SUNY Buffalo. He is currently working on a computational theory of mathematical cognition using the SNePS system.