Biologically Inspired Cognitive Architectures II:
Papers from the AAAI Fall Symposium (FS-09-01)

The GLAIR Cognitive Architecture

Stuart C. Shapiro and **Jonathan P. Bona**
Department of Computer Science and Engineering and Center for Cognitive Science
State University of New York at Buffalo
{shapiro|jpbona}@buffalo.edu

Abstract

GLAIR (Grounded Layered Architecture with Integrated Reasoning) is a multi-layered cognitive architecture for embodied agents operating in real, virtual, or simulated environments containing other agents. The highest layer of the GLAIR Architecture, the Knowledge Layer (KL), contains the beliefs of the agent, and is the layer in which conscious reasoning, planning, and act selection is performed. The lowest layer of the GLAIR Architecture, the Sensori-Actuator Layer (SAL), contains the controllers of the sensors and effectors of the hardware or software robot. Between the KL and the SAL is the Perceptuo-Motor Layer (PML), which grounds the KL symbols in perceptual structures and subconscious actions, contains various registers for providing the agent's sense of situatedness in the environment, and handles translation and communication between the KL and the SAL. The motivation for the development of GLAIR has been "Computational Philosophy", the computational understanding and implementation of human-level intelligent behavior without necessarily being bound by the actual implementation of the human mind. Nevertheless, the approach has been inspired by human psychology and biology.

1. Introduction

GLAIR (Grounded Layered Architecture with Integrated Reasoning) is a multi-layered cognitive architecture for embodied agents operating in real, virtual, or simulated environments containing other agents (Hexmoor, Lammens, and Shapiro 1993; Lammens, Hexmoor, and Shapiro 1995; Shapiro and Ismail 2003). It was an outgrowth of the SNePS Actor (Kumar and Shapiro 1991). Our motivating goal has been what is called "Computational Philosophy" in (Shapiro 1992), that is, the computational understanding and implementation of human-level intelligent behavior without necessarily being bound by the actual implementation of the human mind. Nevertheless, our approach has been inspired by human psychology and biology.

Although GLAIR is a cognitive architecture appropriate for implementing various cognitive agents, we tend to name all our cognitive agents "Cassie." So whenever in this paper we refer to Cassie, we mean one or another of our implemented GLAIR agents.

2. GLAIR as a Layered Architecture

2.1 The Layers

The highest layer of the GLAIR Architecture, the Knowledge Layer (KL), contains the beliefs of the agent, and is the layer in which conscious reasoning, planning, and act selection is performed.

The lowest layer of the GLAIR Architecture, the Sensori-Actuator Layer (SAL), contains the controllers of the sensors and effectors of the hardware or software robot.

Between the KL and the SAL is the Perceptuo-Motor Layer (PML), which, itself is divided into three sublayers. The highest, the PMLa, grounds the KL symbols in perceptual structures and subconscious actions, and contains various registers for providing the agent's sense of situatedness in the environment. The lowest of these, the PMLc, directly abstracts the sensors and effectors into the basic behavioral repertoire of the robot body. The middle PML layer, the PMLb, handles translation and communication between the PMLa and the PMLc.

2.2 Mind-Body Modularity

The KL constitutes the mind of the agent; the PML and SAL, its body. However, the KL and PMLa layers are independent of the implementation of the agent's body, and can be connected, without modification, to a hardware robot or to a variety of software-simulated robots or avatars. Frequently, the KL, PMLa, and PMLb have run on one computer; the PMLc and SAL on another. The PMLb and PMLc handle communication over I/P sockets.[1]

3. The KL: Memory and Reasoning

The KL contains the beliefs of the agent, including: short-term and long-term memory; semantic and episodic memory; quantified and conditional beliefs used for reasoning; plans for carrying out complex acts and for achieving goals; beliefs about the preconditions and effects of acts; policies about when, and under what circumstances, acts should be performed; self-knowledge; and metaknowledge.

The KL is the layer in which conscious reasoning, planning, and act selection is performed. The KL is implemented

[1] Other interprocess communication methods might be used in the future.

in SNePS (Shapiro and Rapaport 1992; Shapiro 2000b; Shapiro and The SNePS Implementation Group 2008), which is simultaneously a logic-based, frame-based, and network-based knowledge representation and reasoning system, that employs various styles of inference as well as belief revision.

As a logic-based KR system, SNePS implements a predicate logic with variables, quantifiers, and function symbols. Although equivalent to First-Order Logic, its most unusual feature is that every well-formed expression is a term, even those that denote propositons (Shapiro 1993). This allows for metapropositions, propositions about propositions, without restriction and without the need for an explicit `Holds` predicate (Morgado and Shapiro 1985; Shapiro et al. 2007). For example the asserted term, `Believe(B8,Rich(B8))` in the context of the asserted term, `Propername(B8,Oscar)`, denotes the proposition that Oscar believes himself to be rich (Rapaport, Shapiro, and Wiebe 1997). SNePS supports forward- backward- and bidirectional-reasoning (Shapiro 1987; Shapiro, Martins, and McKay 1982) using a natural-deduction proof theory, and belief revision (Martins and Shapiro 1988).

Every functional term in SNePS is represented as an assertional frame in which the argument positions are slots and the arguments are fillers. This allows for sets of arguments to be used to represent combinatorially many assertions. For example, `instanceOf({Fido, Lassie, Rover}, {dog, pet})` might be used to represent the assertion that Fido, Lassie, and Rover are dogs and pets. It also allows sets to be used for symmetric relationships, for example `adjacent({US, Canada})` can represent the assertion that the US and Canada are adjacent to each other (Shapiro 1986). The frame view of SNePS supports "slot-based inference", whereby an asserted frame logically implies one with a subset or superset of fillers in given slots (Shapiro 2000a).

By treating the terms as nodes and the slots as labeled directed arcs, SNePS can be used as a propositional network (Shapiro and Rapaport 1987). This supports a style of inference driven by following paths in the network (Shapiro 1978; 1991).

3.1 The Active Connection Graph

Reasoning is performed by an active connection graph (ACS) (McKay and Shapiro 1980; 1981). Viewing the SNePS knowledge base as a propositional graph, every proposition-denoting term can be considered to be a node with arcs pointing to its arguments. This includes non-atomic propositions such as implications, each of which has one set of arcs pointing to its antecedents and another pointing to its consequents. Each proposition has a process charged with collecting and transmitting inferred instances of its propositions along the arcs to interested other processes in a multiprocessing, producer-consumer, message-passing system (Shubin 1981). This allows recursive rules to be used without getting into an infinite loop, and prevents the same inference from being worked on multiple times even if it is a subgoal in multiple ways (McKay and Shapiro 1981), and has not yet been satisfied.

The ACS is key to SNePS' bidirectional inference (Shapiro, Martins, and McKay 1982; Shapiro 1987). Inference processes are created both by backward inference and by forward inference. If such a process is needed and already exists, a forward-chaining process (producer) adds its results to the process's collection, and a backward-chaining process (consumer) is added to the producer-process's consumers to be notified. If a query is asked that can't be answered, the processes established for it remain, and can be found be subsequent forward inferences. When new beliefs are added to the KL with forward inference, and existing consumer-processes are found for them, new consumer-processes are not established. The result of this is that after a query, additional new information is considered in light of this concern. In other words, a GLAIR agent working on a problem considers relevant new data only as it relates to that problem, focussing its attention on it.

The ACS can be deleted. It is then reestablished the next time a forward- or backward- inference begins. In this way the GLAIR agent changes its attention from one problem to another. When this change of attention happens is, however, currently rather *ad hoc*. A better theory of when it should happen is a subject of future research.

3.2 Contexts

Propositions may be asserted in the KL because they entered from the environment. Either they were told to the agent by some other agent, possibly a human, or they are the result of some perception. Alternatively, a proposition might be asserted in the KL because it was derived by reasoning from some other asserted propositions. We call the former *hypotheses* and the latter *derived propositions.* When a proposition is derived, an *origin set,* consisting of the set of hypotheses used to derive it is stored with it (Martins and Shapiro 1988) à la an ATMS (de Kleer 1986). At each moment, some particular *context,* consisting of a set of hypotheses, is current. The asserted propositions, the propositions the GLAIR agent believes, are the hypotheses of the current context and those derived propositions whose origin sets are subsets of that set of hypotheses. If some hypothesis is removed from the current context (*i.e.,* is disbelieved), the derived propositions that depended on it remain in the KL, but are no longer believed. If all the hypotheses in the origin set of a derived proposition return to the current context, the derived proposition is automatically believed again, without having to be rederived (Martins and Shapiro 1983; Shapiro 2000b).

4. The PMLa

The PMLa, contains: the subconscious implementation of the cognitively primitive actions of the KL; the structures used for the perception of objects and properties in the environment; various registers for providing the agent's sense of situatedness in the environment, such as its sense of "I", "You", "Now", and the actions it is currently engaged in; and procedures for natural language comprehension and generation. Further discussion of the PMLa, and its connections to the KL may be found in §9. and (Shapiro and Ismail 2003).

5. The Behavior Cycle

Several cognitive architectures, such as ACT-R (Anderson and Lebiere 1998), Soar (Laird, Newell, and Rosenbloom 1987), Icarus (Langley, Cummings, and Shapiro 2004), and PRODIGY (Carbonell, Knoblock, and Minton 1990) are based on problem-solving or goal-achievement as their basic driver. GLAIR, on the contrary, is based on reasoning: either thinking about some percept (often linguistic input), or answering some question. The acting component is a more recent addition, allowing an GLAIR agent also to obey a command, either to perform an act or to achieve a goal. However, the focus of the design remains on reasoning. Problem solving *vs.* reasoning, however, are not incompatible tasks, but alternative approaches to the ultimate goal of achieving an AI-complete (Shapiro 1992) system.

GLAIR agents execute a sense-reason-act cycle, but not necessarily in a strict cyclical order. GLAIR was developed around implementations of SNePS as an interactive natural language comprehension, knowledge representation, and reasoning system. The basic behavior cycle is:

1. input a natural language utterance.
2. analyze the utterance in the context of the current beliefs
   - the analysis may require and trigger reasoning
   - the analysis may cause new beliefs to be added to the KL
3. if the utterance is a statement
   (a) add the main proposition of the statement as a belief
   (b) that proposition will be output

   if the utterance is a question
   (a) perform backward reasoning to find the answer to the question
   (b) the answer will be output

   if the utterance is a command
   (a) perform the indicated act
   (b) the proposition that the agent performed the act will be output
4. generate a natural language utterance expressing the output proposition
   - reasoning may be performed to formulate the utterance

The categorization of input into either statement (informative), question (interrogative), or command (imperative) assumes that there are no indirect speech acts (Searle 1975) or that the real speech act has already been uncovered. An alternative would be to represent each input as *"X said S,"* and reason about what the agent should do about it. Natural language analysis and generation is an optional part of the GLAIR architecture. If it is omitted, the utterance is expressed in a formal language, such as SNePSLOG (Shapiro and The SNePS Implementation Group 2008) (the formal language used in this paper) and only step (3) is performed.

If this input-reason-output behavior cycle seems too restricted for a cognitive agent, note that the input might be "Perform a", where a is an act, or "Achieve g", where g is a goal, and that might start an arbitrarily long sequence of behaviors. In fact, any of the reasoning episodes might trigger afferent or efferent acts, and any act might trigger reasoning (Kumar 1993; Kumar and Shapiro 1994).

There can be both passive and active sensing. Passive sensing, such as seeing the environment as the agent navigates through it, may result in percepts that, in a data-driven fashion, motivate the agent to perform some act. Active sensing, such as attending to some specific aspect of the environment, may be used in a goal-directed fashion to gain particular information that can be used to decide among alternative acts. For example, we have implemented a GLAIR delivery agent that navigates the hallways of one floor of a simulated building, and may be told to get a package from one room, and deliver it to another. A primitive act of this agent is `goForward()`: " move one unit in the direction it is facing." As a result of such a move, and without another act on its part, it believes either that it is facing a room, a blank wall, or more corridor. Adding the appropriate belief to the KL is built into the PMLa implementation of `goForward()`, and is an example of passive sensing. On the other hand, if the agent needs to know where it is, and it is facing a room, it can deliberately read the room number by performing the primitive act, `readRoomNumber()`. This is an example of active sensing.

6. The Acting Model[2]

GLAIR's acting model consists of: actions and acts; propositions about acts; and policies.

6.1 Policies

Policies specify circumstances under which reasoning leads to action. An example of a policy is, "when the walk light comes on, cross the street." Policies are neither acts nor propositions. We say that an agent *performs* an act, *believes* a proposition, and `adopts` a policy. To see that policies are not acts, note that one cannot perform "when the walk light comes on, cross the street." A good test for an expression ϕ being a proposition is its ability to be put in the frame, "I believe that it is not the case that ϕ." It does not make sense to say, "I believe that it is not the case that when the walk light comes on, cross the street." Note that this is different than saying, "I believe that it is not the case that when the walk light comes on, I *should* cross the street." An agent might explicitly believe "I *should* cross the street" without actually doing it. However, if an GLAIR agent has adopted the policy, "when the walk light comes on, cross the street," and it comes to believe that the walk light is on, it *will* cross the street (or at least try to).

Policies are represented as functional terms in the KL, along with other conscious memory structures. Three policy-forming function symbols are built into GLAIR, each of which take as arguments a proposition ϕ and an act α:

- `ifdo(`ϕ, α`)` is the policy, "to decide whether or not ϕ, perform α";
- `whendo(`ϕ, α`)` is the policy, "when ϕ holds, perform α";

[2]Parts of this section were taken from (Shapiro et al. 2007).

- `whenever do(`ϕ, α`)` is the policy, "whenever ϕ holds, perform α".

A blocks-world example of `ifdo` is "*To decide whether block A is red, look at it*": `ifdo(ColorOf(A,red), lookAt(A))` (Kumar and Shapiro 1994).[3]

The policies `whendo` and `wheneverdo` are similar to the production rules of production systems in that they are condition-action rules triggered when forward-chaining matches the condition. In the case of both `whendo` and `wheneverdo`, if the policy has been adopted, the agent performs α when forward inference causes ϕ to be believed. Also, α is performed if ϕ is already believed when the policy is adopted. The difference is that a `whendo` policy is unadopted after firing once, but a `wheneverdo` remains adopted until explicitly unadopted.

6.2 Categories of Acts

An act may be performed by an agent, and is composed of an *action* and zero or more arguments. For example, for the Fevahr[4] version of Cassie (Shapiro 1998) (henceforth Cassie$_F$), 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`.[5]

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 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 the architecture; they must be supplied by each agent designer.

Mental acts affect the agent's beliefs and policies. There are four:

1. `believe(`ϕ`)` is the act of asserting the proposition ϕ and doing forward inference on it;
2. `disbelieve(`ϕ`)` is the act of unasserting the proposition ϕ, so that it is not believed, but its negation is not necessarily believed;
3. `adopt(`π`)` is the act of adopting the policy π;
4. `unadopt(`π`)` is the act of unadopting the policy π.

Before `believe` changes the belief status of a proposition ϕ, it performs a limited form of prioritized belief revision (Alchourrón, Gärdenfors, and Makinson 1985). If `andor(0,0)`$\{\ldots, \phi, \ldots\}$ is believed,[6] it is `disbelieved`. If `andor(`i`,1)`$\{\phi_1, \phi_2, \ldots\}$ is believed, for $i = 0$ or $i = 1$, and ϕ_2 is believed, ϕ_2 is `disbelieved`.

Control acts are the control structures of the GLAIR acting system. The predefined control actions are:

- `achieve(`ϕ`)`: If the proposition ϕ is not already believed, infer plans for bringing it about, and then perform `do-one` on them.
- `snsequence(`α_1, α_2`)`: Perform the act α_1, and then the act α_2.
- `prdo-one(`$\{$`pract(`x_1, α_1`), ..., pract(`x_n, α_n`)`$\}$`)`: Perform one of the acts α_j, with probability $x_j / \sum_i x_i$.
- `do-one(`$\{\alpha_1, \ldots, \alpha_n\}$`)`: Nondeterministically choose one of the acts $\alpha_1, \ldots, \alpha_n$, and perform it.
- `do-all(`$\{\alpha_1, \ldots, \alpha_n\}$`)`: Perform all the acts $\alpha_1, \ldots, \alpha_n$ in a nondeterministic order.
- `snif(`$\{$`if(`ϕ_1, α_1`), ..., if(`ϕ_n, α_n`), [else(`δ`)]`$\}$`)`: Use backward inference to determine which of the propositions ϕ_i hold, and, if any do, nondeterministically choose one of them, say ϕ_j, and perform the act α_j. If none of the ϕ_i can be inferred, and if `else(`δ`)` is included, perform δ. Otherwise, do nothing.
- `sniterate(`$\{$`if(`ϕ_1, α_1`), ..., if(`ϕ_n, α_n`), [else(`δ`)]`$\}$`)`: Use backward inference to determine which of the propositions ϕ_i hold, and, if any do, nondeterministically choose one of them, say ϕ_j, and perform the act `snsequence(`α_j`, sniterate(`$\{$`if(`ϕ_1, α_1`), ..., if(`ϕ_n, α_n`), [else(`δ`)]`$\}$`))`. If none of the ϕ_i can be inferred, and if `else(`δ`)` is included, perform δ. Otherwise, do nothing.
- `withsome(`$x, \phi(x), \alpha(x), [\delta]$`)`: Perform backward inference to find entities e such that $\phi(e)$ is believed, and, if such entities are found, choose one of them nondeterministically, and perform the act α on it. If no such e is found, and the optional act δ is present, perform δ.
- `withall(`$x, \phi(x), \alpha(x), [\delta]$`)`: Perform backward inference to find entities e such that $\phi(e)$ is believed, and, if such entities are found, perform the act α on them all in a nondeterministic order. If no such e is found, and the optional act δ is present, perform δ.

The acts `snif`, `sniterate`, `withsome`, and `withall` all trigger reasoning. The default implementation of `do-one` uses a pseudorandom number generator to choose the act to perform, and the default implementation of `do-all` uses a pseudorandom number generator to choose the order of the acts. However, an agent implementer may replace either pseudorandom number generator with reasoning rules to make the choice, in which case these acts will also trigger reasoning.

[3] `ifdo` was called `DoWhen` in (Kumar and Shapiro 1994).

[4] "Fevahr" is an acronym standing for "Foveal Extra-Vehicular Activity Helper-Retriever".

[5] 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 paper.

[6] `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`)`$\{\phi_1, \ldots, \phi_n\}$ denotes the proposition that at least i and at most j of the ϕ_k's are true.

Primitive, Defined, and Composite Acts GLAIR actions and acts may also be classified as either primitive, defined, or composite. Primitive acts constitute the basic repertoire of an GLAIR agent. They are either provided by the architecture itself, or are implemented at the PMLa. An example predefined action is `believe`; an example primitive action defined at the PMLa is the Fevahr Cassie's `find` (Shapiro 1998). Because primitive actions are implemented below the KL, GLAIR 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 (Shapiro and Kandefer 2005), 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 GLAIR agent is to perform a defined act α, it deduces plans p for which it believes the proposition `ActPlan(`α, 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)`, which is defined by `ActPlan(turn(around), snsequence(go(right), go(right)))`.

6.3 Propositions About Acts

Four propositions about acts are predefined parts of the GLAIR architecture:

1. `Precondition(`α, ϕ`)`: In order for the agent to perform the act α, the proposition ϕ must hold.
2. `Effect(`α, ϕ`)` An effect of an agent's performing the act α is that the proposition ϕ will hold. The proposition ϕ could be a negation, to express the effect that some proposition no longer holds, such as `Effect(putOn(`x, y`), clear(`y`))`.
3. `ActPlan(`α, p`)`: One way to perform the act α is to perform the plan p.
4. `GoalPlan(`ϕ, p`)`: One way to achieve the goal that the proposition ϕ holds is to perform the plan p.

The only difference between a "plan" and an "act" is that a plan is an act that appears in the second argument position of an `ActPlan` or a `GoalPlan` proposition. However, in a proposition of the form `ActPlan(`α, p`)`, it is assumed that p is "closer" to primitive acts than α is.

6.4 Conditional Plans

Consider a defined act for which there are different plans depending on circumstances. For example, to get the mail, if I'm at home, I go to the mailbox, but if I'm in the office, I go to the mailroom. Such conditional plans may be represented by implications:

```
at(home) => ActPlan(get(mail),
                    go(mailbox))
at(office) => ActPlan(get(mail),
                      go(mailroom))
```

```
perform(act):
  pre := {p | Precondition(act,p)};
  notyet := pre - {p | p ∈ pre & ⊢ p};
  if notyet ≠ nil
    then
       perform(
         snsequence(
           do-all({a | p ∈ notyet
                       & a = achieve(p)}),
           act))
    else
      {effects := {p | Effect(act,p)};
       if act is primitive
         then apply(
                primitive-function(act),
                objects(act));
         else perform(
                do-one({p
                        | ActPlan(act,p)}))
       believe(effects)}
```

Figure 1: The acting executive

In a context in which `at(home)` is derivable, the plan for getting the mail will be `go(mailbox)`. When the context changes so that `at(home)` is no longer derivable, `ActPlan(get(mail), go(mailbox))` will no longer be asserted, nor derivable. However, when the context is reentered, `ActPlan(get(mail), go(mailbox))` will again be asserted without the need to rederive it.

7. The Acting Executive

The procedure for performing an act is shown in Fig. 1. Notice that:

- Backward inference is triggered to find:
  - the preconditions of `act`;
  - whether each precondition currently holds;
  - the effects of `act`;
  - plans that can be used to perform `act`, if `act` is not primitive.
- After the attempt is made to achieve the preconditions of `act`, `perform(act)` is called again, which will again check the preconditions, in case achieving some of them undid the achievement of others.
- Effects of `act` are derived before `act` is performed in case the effects depend on the current state of the world.
- If `act` is a defined act, only one way of performing it is tried, and that is assumed to be successful. This will be changed in future versions of GLAIR.
- After `act` is performed, all its effects are believed to hold. This is naive, and will be changed in future versions of GLAIR. We have already implemented GLAIR agents that only believe the effects of their acts that they sense holding in the world, but this has been done by giving them no `Effect` assertions.

8. Modalities

Especially on hardware robots, the sensors and effectors can operate simultaneously. To take advantage of this, GLAIR supports a set of modalities. A modality represents a limited resource—a PMLc-level behavior that is limited in what it can do at once (for example, a robot cannot go forward and backward at the same time), but is independent of the behaviors of other modalities (a robot can navigate and speak at the same time). Each modality runs in a separate thread, and uses its own communication channel between the PMLb and PMLc layers. Each KL primitive action is assigned, at the PMLa layer, to one or more modalities. Modalities that have been implemented in various GLAIR agents include speach, hearing, navigation, and vision. We intend to make the organization into modalities a more thoroughgoing and pervasive principle of the architecture. That version of the architecture will be called MGLAIR.

9. Symbol Anchoring[7]

9.1 Alignment

There are KL terms for every mental entity Cassie has conceived of, including individual entities, categories of entities, colors, shapes, and other properties of entities. There are PML structures (at the PMLb and PMLc sub-levels) for features of the perceivable world that Cassie's perceptual apparatus can detect and distinguish. Each particular perceived object is represented at this level by an n-tuple of such structures, $\langle v_1, \ldots, v_n \rangle$, where each component, v_i, is a possible value of some perceptual feature domain, D_i. What domains are used and what values exist in each domain depend on the perceptual apparatus of the robot. We call the n-tuples of feature values "PML-descriptions".

Each KL term for a perceivable entity, category, or property is grounded by aligning it with a PML-description, possibly with unfilled (null) components. For example, Cassie$_F$ used two-component PML-descriptions in which the domains were color and shape. The KL term denoting Cassie$_F$'s idea of blue was aligned with a PML-description whose color component was the PML structure the vision system used when it detected blue in the visual field, but whose shape component was null. The KL term denoting people was aligned with a PML-description whose shape component was the PML structure the vision system used when it detected a people in the visual field, but whose color component was null.

Call a PML-description with some null components an "incomplete PML-description", and one with no null components a "complete PML-description". KL terms denoting perceivable properties and KL terms denoting recognizable categories of entities are aligned with incomplete PML-descriptions. Examples include the terms for blue and for people mentioned above, and may also include terms for the properties tall, fat, and bearded, and the categories man and woman. The words for these terms may be combined into verbal descriptions, such as "a tall, fat, bearded man," whose incomplete PML-descriptions may be used to perceptually recognize the object corresponding to the entity so described.

A complete PML-description may be assembled for an entity by unifying the incomplete PML-descriptions of its known (conceived-of) properties and categories. Once a PML-description is assembled for an entity, it is cached by aligning the term denoting the entity directly with it. Afterwards, Cassie can recognize the entity without thinking about its description. On the other hand, Cassie may have a complete PML-description for some object without knowing any perceivable properties for it. In that case, Cassie would be able to recognize the object, even though she could not describe it verbally.

If Cassie is looking at some object, she can recognize it if its PML-description is the PML-description of some entity she has already conceived of. If there is no such entity, Cassie can create a new KL term to denote this new entity, align it with the PML-description, and believe of it that it has those properties and is a member of those categories whose incomplete PML-descriptions unify with the PML-description of the new entity. If there are multiple entities whose PML-descriptions match the object's PML-description, disambiguation is needed, or Cassie might simply not know which one of the entities she is looking at.

9.2 Deictic Registers

An important aspect of being embodied is being situated in the world and having direct access to components of that situatedness. This is modeled in GLAIR via a set of PML registers (variables), each of which can hold one or more KL terms or PML structures. Some of these 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.

9.3 Modality Registers

GLAIR agents know what they are doing via direct access to a set of PML registers termed "modality registers". For example, if one of Cassie's modalities were speech, and she were currently talking to Stu, her `SPEECH` register would contain the KL term denoting the state of Cassie's talking to Stu (and the term denoting Stu would be in the `YOU` register). In many cases, a single modality of an agent can be occupied by only one activity at a time. In that case the register for that modality would be constrained to contain only one term at a time.

One of the modality registers we have used is for keeping track of what Cassie is looking at. When she recognizes an object in her visual field, the KL term denoting the state of looking at the recognized entity is placed in the register, and is removed when the object is no longer in the visual field. If one assumed that Cassie could be looking at several objects at once, this register would be allowed to contain several terms. If asked to look at or find something that is already

[7] This section is taken from (Shapiro and Ismail 2003).

in her visual field, Cassie recognizes that fact, and doesn't need to do anything.

9.4 Actions

Each KL action term that denotes a primitive action is aligned with a procedure in the PMLa. The procedure takes as arguments the KL terms for the arguments of the act to be performed. For example, when Cassie is asked to perform the act of going to Bill, the PMLa going-procedure is called on the KL Bill-term. It then finds the PML-description of Bill, and (via the SAL) causes the robot hardware to go to an object in the world that satisfies that description (or causes the robot simulation to simulate that behavior). The PMLa going-procedure also inserts the KL term denoting the state of Cassie's going to Bill into the relevant modality register(s), which, when NOW moves , causes an appropriate proposition to be inserted into Cassie's KL.

9.5 Time

As mentioned above, the NOW register always contains the KL term denoting the current time (Shapiro 1998; Ismail 2001; Ismail and Shapiro 2000; 2001). Actually, since "now" is vague (it could mean this minute, this day, this year, this century, etc.), NOW is considered to include the entire semi-lattice of times that include the smallest current now-interval Cassie has conceived of, as well as all other times containing that interval.

NOW moves whenever Cassie becomes aware of a new state. Some of the circumstances that cause her to become aware of a new state are: she acts, she observes a state holding, she is informed of a state that holds. NOW moves by Cassie's conceiving of a new smallest current now-interval (a new KL term is introduced with that denotation), and NOW is changed to contain that time. The other times in the old NOW are defeasibly extended into the new one by adding propositions asserting that the new NOW is a subinterval of them.

Whenever Cassie acts, the modality registers change (see above), and NOW moves. The times of the state(s) newly added to the modality registers are included in the new NOW semi-lattice, and the times of the state(s) deleted from the modality registers are placed into the past by adding propositions that assert that they precede the new NOW.

To give GLAIR agents a "feel" for the amount of time that has passed, the PML has a COUNT register acting as an internal pacemaker (Ismail 2001; Ismail and Shapiro 2001). The value of COUNT is a non-negative integer, incremented at regular intervals. Whenever NOW moves, the following happens:

1. the value of COUNT is quantized into a value δ which is the nearest half-order of magnitude (Hobbs 2000) to COUNT, providing an equivalence class of PML-measures that are not noticeably different;
2. a KL term d, aligned with δ, is found or created, providing a mental entity denoting each class of durations;
3. a belief is introduced into the KL that the duration of t_1, the current falue of NOW, is d, so that the agent can have beliefs that two different states occurred for about the same length of time;
4. a new KL term, t_2 is created and a belief is introduced into the KL that t_1 is before t_2;
5. NOW is reset to t_2;
6. COUNT is reset to 0, to prepare for measuring the new now-interval.

9.6 Language

Cassie interacts with humans in a fragment of English. Although it is possible to represent the linguistic knowledge of GLAIR agents in the KL, use reasoning to analyze input utterances (Neal and Shapiro 1985; 1987b; 1987a; Shapiro and Neal 1982), and use the acting system to generate utterances (Haller 1996; 1999), we do not currently do this. Instead, the parsing and generation grammars, as well as the lexicon, are at the PML. (See, e.g. (Rapaport, Shapiro, and Wiebe 1997; Shapiro 1982; Shapiro and Rapaport 1995).) There are KL terms for lexemes, and these are aligned with lexemes in the PML lexicon. We most frequently use a KL unary functional term to denote the concept expressed by a given lexeme, but this does not allow for polysemy, so we have occasionally used binary propositions that assert that some concept may be expressed by some lexeme. There may also be KL terms for inflected words, strings of words, and sentences. This allows one to discuss sentences and other language constructs with GLAIR agents.

10. Bodily Feedback

The control acts snsequence, do-all, sniterate, and withall each cause a sequence of acts to be performed before it is completed. In a normal, single-processor, procedural/functional architecture this would not cause a problem as each act in the sequence would be performed only after the previous one returns control to the control act. However, in GLAIR, primitive acts are performed in modalities operating concurrently with reasoning, so it is important for the control act to get feedback from the body that an act has completed before it proceeds to the next act in the sequence. Think of the problem deaf people have speaking at a "normal" rate without being able to hear themselves. In previous agents (Shapiro et al. 2005b; 2005c), bodily feedback for the speech modality was provided for via the hearing modality, but this was included explicitly at the KL and using a special pacedSequence act. We intend to build bodily feedback directly into the GLAIR architecture in the future.

11. Properties of cognitive architectures

In this section, we discuss GLAIR using propertes listed in (Langley, Laird, and Rogers 2009).

11.1 Representation of knowledge

Knowledge (more properly, beliefs) is represented in the GLAIR Knowledge Layer in SNePS, which is simultaneously a logic-based, assertional frame-based, and

graph(network)-based knowledge representation and reasoning system. Noteworthy features of the SNePS representation are: every well-formed expression is a term, even those denoting propositions; all beliefs and conceived-of entities are represented in the same formalism, including reasoning rules (such as conditionals) and acting plans. SNePS is more fully discussed above and in the cited papers.

Single notation vs. Mixture of formalisms Although all knowledge is represented in a single formalism, namely SNePS, SNePS, itself, is simultaneously three different formalisms: logic-based, which supports a natural-deduction-style inference mechanism; assertional frame-based, which supports inference from one frame to another with a subset or superset of fillers in some of the slots; and graph/network-based, which supports inference of labeled arcs from the presence of paths of labeled arcs.

Support for metaknowledge Since every SNePS expression is a term, including those that denote propositions, propositions about propositions may be represented without restriction and without the need for an explicit `Holds` predicate. The default acts included as options in `snif`, `sniterate`, `withsome`, and `withall` provide for lack-of-knowledge acting. The use of conditional plans, as discussed in § 6.4 , has allowed a GLAIR agent to use contextual information to select among alternative mathematical procedures to perform (Shapiro et al. 2007).

By including in the Knowledge Layer a term that refers to the agent itself, GLAIR agents are able to represent and reason about themselves. As mentioned in § 9.2, a deictic register in the PML is a pointer to the self-concept. PMLa implementations of primitive acts can insert beliefs into the KL about what the agent is currently doing, and the movement of time, as discussed in § 9.5, gives the agent an episodic memory.

Giving GLAIR agents knowledge of the actions they are currently performing above the level of primitive actions is a subject of further work.

Declarative vs. Procedural representations The Knowledge Layer contains declarative representations of knowledge, even of procedures for carrying out defined acts (*see* § 6.2). The PMLa contains implementations of primitive acts in a way that is not cognitively penetrable. We have not yet experimented with GLAIR agents that learn such procedural representations of primitive acts.

Semantic memory vs. Episodic memory The Knowledge Layer is the locus of both semantic and episodic memory. Most of the beliefs of GLAIR agents we have developed so far are parts of semantic memory. As mentioned above, PMLa implementations of primitive acts can insert beliefs into the KL about what the agent is currently doing, and the movement of time, as discussed in § 9.5, gives the agent an episodic memory.

11.2 Organization of knowledge

Flat vs. Structured/Hierarchical organization of knowledge SNePS uses an inherently structured organization of knowledge. Its term-based predicate logic representation allows for nested functional terms, including proposition-valued terms, the act-valued terms that constitute composite acts, and reasonng rules. SNePS has often been used to represent hierarchical information, including subsumption hierarchies, parthood and other mereological relations, and similar information used in ontological reasoning.

Short-term vs. long-term memory GLAIR currently has no short-term memory from which some memories migrate into long-term memory. The closest thing to a short-term or working memory is the active connection graph (*see* § refsec:acg), which contains the demons currently working on one problem, which are discarded when the agent changes to another problem.

12. Evaluation criteria for cognitive architectures

In this section, we evaluate GLAIR according to criteria listed in (Langley, Laird, and Rogers 2009).

12.1 Generality, versatility, and taskability

Generality The KL and PMLa layers are independent of the implementation of the lower body and the environment as long as there is some way for the primitive sensory and effector acts at the PMLa layer to be implemented in the SAL layer. The agent designer designs the PMLb and PMLc layers to effect the connection. GLAIR agents have been active in: a real world laboratory setting (Shapiro 1998); a virtual reality world (Shapiro et al. 2005c); a world simulated by ASCII input/output (Kandefer and Shapiro 2007); and graphically simulated worlds (Shapiro and Kandefer 2005; Anstey et al. in press).

Versatility The GLAIR architecture lends itself to modular design for new environments and tasks. If the designers have a specific agent body and environment in mind, they must identify the afferent and efferent behavior repertoire of the agent. They can then specify the actions to be implemented at the PMLa layer. These become the primitive actions at the KL layer, and high-level actions can be programmed using the acting model described in § 6.
Since the control actions, which include `snsequence`, `snif`, and `sniterate`, form a Turing-complete set, a GLAIR agent can perform any task that can be composed computationally from its primitive acts (Böhm and Jacopini 1966).

Once the KL primitive actions have been designed, it is common to test and further develop the agent in a simulated environment before moving it to a hardware robot in the real world, or to a more detailed simulation in a graphical or virtual world.

Taskability One benefit of representing acts in the same formalism as other declarative knowledge is that agents that

communicate with a GLAIR agent can command it to perform tasks using the same communication language. The formal language commonly used is SNePSLOG, the language in which the acting model was explained in § 6., but GLAIR agents have been built that use fragments of English (Shapiro 1989; Shapiro, Ismail, and Santore 2000; Shapiro and Ismail 2003; Kandefer and Shapiro 2007). The meaning of verb phrases are represented in the act structures of the acting model. If English versions of contol acts are included in the fragment of English, GLAIR agents may be given composite acts to perform. For example, Cassie$_F$ (Shapiro 1998) can be told to *"Go to the green robot and then come here and help me."* With appropriate grammatical support, natural language may be used to teach GLAIR agents new tasks. For example, an early GLAIR agent was told, *"IF a block is on a support then a plan to achieve that the support is clear is to pick up the block and then put the block on the table"* (Shapiro 1989; Shapiro, Ismail, and Santore 2000).

12.2 Rationality and optimality

Rationality When attempting to achieve a goal ϕ, a GLAIR agent chooses an act α to perform based on its belief that the act will achieve the goal, as expressed by `GoalPlan(`ϕ, α`)`. However, we have not yet experimented with GLAIR agents that formulate such beliefs by reasoning about the effects of various acts. That is, we have not yet developed GLAIR agents that do traditional AI planning. Nor have we experimented with GLAIR agents that formulate `GoalPlan` beliefs after acting and noting the effects of its acts. So a GLAIR agent is rational in the sense that it selects an act that it *believes* will lead to the goal. However, it doesn't *know* that the act will lead to the goal.

Optimality The GLAIR architecture allows for, but does not explicitly implement as part of the architecture, agents that choose optimal actions based on preferences they form. This has been used to implement agents that can prefer certain shortcut ways of performing arithmetical operations (Goldfain 2008), and by metacognitive agents such as those described in (Shapiro et al. 2007), which are able to observe themselves at work and prefer efficient (requiring fewer steps than the alternatives) ways of accomplishing a goal.

12.3 Efficiency and scalability

SNePS does not place any formal restriction on the number of terms that can be represented and stored, nor on the number of relations between them. There is a naturally-occurring limit that depends on the computational resources available to the system and will vary from one machine to the next. The upper limit for any instance of SNePS depends on the heap size of the Lisp image in which it is running. We have not evaluated SNePS in terms of formal computational complexity, A recent technical report on SNePS' efficiency (Seyed, Kandefer, and Shapiro 2008) shows that the system can reason over knowledge bases that include tens of thousands of terms/propositions, though some reasoning tasks take many seconds to complete in this situation. The same report outlines steps to increase the number of supported memory elements and the speed with which they are processed by the system. Some of these planned modifications have already been implemented in the latest releases. Other proposed changes include introducing a sophisticated scheme for moving to long-term memory information in the KB that is not being used in the service of reasoning at that time and is not likely to be used soon. SNePS3 (currently under development) will introduce even more efficiency gains.

12.4 Reactivity and persistence

GLAIR's use of separate buffers for separate perceptual modalities facilitates reactivity by ensuring that sensory data from one modality does not block and demand all of an agent's "attention." In some of our work with GLAIR-based agent-actors for virtual drama(Shapiro et al. 2005a), agents interact with human audience participants in a 3D virtual world that allows the human a great deal of freedom to move around in, and effect changes within, the world. In one case, the agent's task is to guide the participant on a quest and complete a series of activities. The participant's actions cannot be fully anticipated, and may include verbally addressing the agent, losing interest and wandering off, making unrestricted movements unrelated to the task, etc. The agent's task then consists of following and keeping up with the participant and reacting as appropriately as is possible to her actions while simultaneously trying to convince her to participate in the assigned task. This requires an implementation of persistence in which the agent keeps track of goals for the current task (and the greater quest), while simultaneously dealing with unpredictable changes in the environment due to the participant's activities.

12.5 Improvability

Improvability GLAIR includes several forms of learning:

Learning by being told: Propositions and policies added to the KL, whether from a human, another agent, or via perception are immediately available for use. For example, a GLAIR agent is instructable. If it is unable, due to lack of knowledge, to perform some act, a human may instruct it so that the agent will be able to perform that act in the future.

Contextual learning: As discussed in §3.2, when a proposition ϕ is derived in a context $\mathcal{C}$, its origin set o, a set of hypotheses, is stored with it. As the agent performs, the context will probably change and some of the hypotheses in o be removed from the context. When a new context arises that again contains all the hypotheses in o, ϕ will again be asserted without having to be rederived. Consider a conditional plan such as ϕ `=> GoalPlan(`α, p`)`. The first time the plan is considered in an appropriate context, ϕ and then `GoalPlan(`α, p`)` will have to be derived. If another situation arises in which the hypotheses in o are asserted, `GoalPlan(`α, p`)` will be asserted without the need for rederivation.

Experience-Based Deductive Learning Consider the general definition of transitivity, expressible in SNePSLOG

as

```
all(r)(Transitive(r)
       => all(x,y,z)({r(x,y), r(y,z)}
                     &=> r(x,z)))
```

along with the belief that `Transitive(ancestor)`. The first time an `ancestor` question is posed to Cassie, she will use the transitivity definition to derive

```
all(x,y,z)({ancestor(x,y),
            ancestor(y,z)}
           &=> ancestor(x,z)))
```

and then answer the question. The specific `ancestor` rule will be stored in the Knowledge Layer. The next time an `ancestor` question is posed, Cassie will use the specific `ancestor` rule, but not the general transitivity definition. Even though the knowledge base is now larger (two rules are stored instead of one), the second question will be answered more quickly than if the first question hadn't been asked. Cassie has developed a kind of expertise in ancestor-reasoning (Choi and Shapiro 1991; Choi 1993).

Several other forms of improvability have not yet been added to GLAIR. For example, we have not yet experimented with agents that use the observed effects of its acts to modify or extend its plans. Nor have we yet experimented with agents that "compile" defined acts into primitive acts.

12.6 Autonomy and extended operation

The GLAIR acting system allows for agents that act autonomously for long periods of time, though we have not made any formal measure of agents' degrees of autonomy. Many of our agents can act indefinitely independent of any explicit instructions from, or interactions with, an operator by pursuing goals, following plans, and responding to changes in the environment as they are perceived.

13. Future Concerns

Several issues that are certainly important for cognitive architectures have not yet been addressed in the development of GLAIR. These include uncertainty and considerations of real-time operation to limit the amount of reasoning.

14. Current Status

SNePS has been under development, with numerous uses, modifications, additions, and reimplementations since before 1979 (Shapiro 1979). Likewise, GLAIR has been under development since before 1993 (Hexmoor, Lammens, and Shapiro 1993), and has been used for a variety of agents, for some examples see (Shapiro and Kandefer 2005; Kandefer and Shapiro 2007; Anstey et al. in press). MGLAIR is still being defined, although prototype versions have been used to build a variety of agents (Shapiro et al. 2005a).

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.

Anderson, J. R., and Lebiere, C. 1998. *The Atomic Components of Thought*. Mahwah, NJ: Lawrence Erlbaum.

Anstey, J.; Seyed, A. P.; Bay-Cheng, S.; Bona, J.; Hibit, S.; Pape, D.; Shapiro, S. C.; and Sena, V. in press. The agent takes the stage. *International Journal of Arts and Technology,*.

Böhm, C., and Jacopini, G. 1966. Flow diagrams, turing machines, and languages with only two formation rules. *Communications of the ACM* 9(5):366–371.

Carbonell, J. G.; Knoblock, C. A.; and Minton, S. 1990. PRODIGY: An integrated architecture for planning and learning. In VanLehn, K., ed., *Architectures for Intelligence*. Hillsdale, NJ: Lawrence Erlbaum. 241–278.

Choi, J., and Shapiro, S. C. 1991. Experience-based deductive learning. In *Third International Conference on Tools for Artificial Intelligence TAI '91*. Los Alamitos, CA: IEEE Computer Society Press. 502–503.

Choi, J. 1993. *Experience-Based Learning in Deductive Reasoning Systems*. PhD dissertation, Technical Report 93-20, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY.

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, Inc.

Goldfain, A. 2008. *A Computational Theory of Early Mathematical Cognition*. Ph.D. Dissertation, State University of New York at Buffalo, Buffalo, NY.

Haller, S. 1996. Planning text about plans interactively. *International Journal of Expert Systems* 9(1):85–112.

Haller, S. 1999. An introduction to interactive discourse processing from the perspective of plan recognition and text planning. *Artificial Intelligence Review* 13(4):259–333.

Hexmoor, H.; Lammens, J.; and Shapiro, S. C. 1993. Embodiment in GLAIR: a grounded layered architecture with integrated reasoning for autonomous agents. In Dankel II, D. D., and Stewman, J., eds., *Proceedings of The Sixth Florida AI Research Symposium (FLAIRS 93)*. The Florida AI Research Society. 325–329.

Hobbs, J. R. 2000. Half orders of magnitude. In Obrst, L., and Mani, I., eds., *Papers from the Workshop on Semantic Approximation, Granularity, and Vagueness*, 28–38. A Workshop of the Seventh International Conference on Principles of Knowldege Representation and Reasoning, Breckenridge, CO.

Ismail, H. O., and Shapiro, S. C. 2000. Two problems with reasoning and acting in time. In Cohn, A. G.; Giunchiglia, F.; and Selman, B., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR 2000)*, 355–365. San Francisco: Morgan Kaufmann.

Ismail, H. O., and Shapiro, S. C. 2001. The cognitive clock: A formal investigation of the epistemology of time. Technical Report 2001-08, Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY.

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

Kandefer, M., and Shapiro, S. C. 2007. Knowledge acquisition by an intelligent acting agent. In Amir, E.; Lifschitz, V.; and Miller, R., eds., *Logical Formalizations of Commonsense Reasoning, Papers from the AAAI Spring Symposium, Technical Report SS-07-05*, 77–82. Menlo Park, CA: AAAI Press.

Kumar, D., and Shapiro, S. C. 1991. Architecture of an intelligent agent in SNePS. *SIGART Bulletin* 2(4):89–92.

Kumar, D., and Shapiro, S. C. 1994. Acting in service of inference (and *vice versa*). In Dankel II, D. D., ed., *Proceedings of The Seventh Florida AI Research Symposium (FLAIRS 94)*. The Florida AI Research Society. 207–211.

Kumar, D. 1993. A unified model of acting and inference. In *Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences*. Los Alamitos, CA: IEEE Computer Society Press.

Laird, J. E.; Newell, A.; and Rosenbloom, P. S. 1987. SOAR: an architecture for general intelligence. *Artificial Intelligence* 33(1):1–64.

Lammens, J. M.; Hexmoor, H. H.; and Shapiro, S. C. 1995. Of elephants and men. In Steels, L., ed., *The Biology and Technology of Intelligent Autonomous Agents*. Berlin: Springer-Verlag, Berlin. 312–344.

Langley, P.; Cummings, K.; and Shapiro, D. 2004. Hierarchical skills and cognitive architectures. In *Proceedings of the Twenty-Sixth Annual Conference of the Cognitive Science Society*, 779–784.

Langley, P.; Laird, J. E.; and Rogers, S. 2009. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research* 10(2):141–160.

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

Martins, J. P., and Shapiro, S. C. 1983. Reasoning in multiple belief spaces. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann. 370–373.

Martins, J. P., and Shapiro, S. C. 1988. A model for belief revision. *Artificial Intelligence* 35:25–79.

McKay, D. P., and Shapiro, S. C. 1980. MULTI — a LISP based multiprocessing system. In *Proceedings of the 1980 LISP Conference*, 29–37.

McKay, D. P., and Shapiro, S. C. 1981. Using active connection graphs for reasoning with recursive rules. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann. 368–374.

Morgado, E. J. M., and Shapiro, S. C. 1985. Believing and acting: A study of meta-knowledge and meta-reasoning. In *Proceedings of EPIA-85 "Encontro Portugues de Inteligencia Artificial"*, 138–154.

Neal, J. G., and Shapiro, S. C. 1985. Parsing as a form of inference in a multiprocessing environment. In *Proceedings of the Conference on Intelligent Systems and Machines*, 19–24. Rochester, Michigan: Oakland University.

Neal, J. G., and Shapiro, S. C. 1987a. Knowledge-based parsing. In Bolc, L., ed., *Natural Language Parsing Systems*. Berlin: Springer-Verlag. 49–92.

Neal, J. G., and Shapiro, S. C. 1987b. Knowledge representation for reasoning about language. In Boudreaux, J. C.; Hamill, B. W.; and Jernigan, R., eds., *The Role of Language in Problem Solving 2*. Elsevier Science Publishers. 27–46.

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

Rapaport, W. J.; Shapiro, S. C.; and Wiebe, J. M. 1997. Quasi-indexicals and knowledge reports. *Cognitive Science* 21(1):63–107. Reprinted in (Orilia and Rapaport 1998, pp. 235–294).

Searle, J. R. 1975. Indirect speech acts. In Cole, P., and Morgan, J. L., eds., *Speech Acts: Syntax and Semantics*, volume 3. Academic Press. 59–82.

Seyed, A. P.; Kandefer, M.; and Shapiro, S. C. 2008. Sneps efficiency report. SNeRG Technical Note 43, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY.

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 Kandefer, M. 2005. A SNePS approach to the wumpus world agent or Cassie meets the wumpus. In Morgenstern, L., and Pagnucco, M., eds., *IJCAI-05 Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC'05): Working Notes*. Edinburgh, Scotland: IJCAII. 96–103.

Shapiro, S. C., and Neal, J. G. 1982. A knowledge engineering approach to natural language understanding. In *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*. Menlo Park, CA: ACL. 136–144.

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*. New York: Springer-Verlag. 263–315.

Shapiro, S. C., and Rapaport, W. J. 1992. The SNePS family. *Computers & Mathematics with Applications* 23(2–5):243–275. Reprinted in (Lehmann 1992, pp. 243–275).

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

Shapiro, S. C., and The SNePS Implementation Group. 2008. *SNePS 2.7 User's Manual*. Department of Com-

puter Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY.

Shapiro, S. C.; Anstey, J.; Pape, D. E.; Nayak, T. D.; Kandefer, M.; and Telhan, O. 2005a. MGLAIR agents in a virtual reality drama. Technical Report 2005-08, Department of Computer Science & Engineering, University at Buffalo, Buffalo, NY.

Shapiro, S. C.; Anstey, J.; Pape, D. E.; Nayak, T. D.; Kandefer, M.; and Telhan, O. 2005b. MGLAIR agents in virtual and other graphical environments. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*. Menlo Park, CA: AAAI Press. 1704–1705.

Shapiro, S. C.; Anstey, J.; Pape, D. E.; Nayak, T. D.; Kandefer, M.; and Telhan, O. 2005c. The Trial The Trail, Act 3: a virtual reality drama using intelligent agents. In Young, R. M., and Laird, J., eds., *Proceedings of the First Annual Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE-05)*, 157–158. Menlo Park, CA: AAAI Press.

Shapiro, S. C.; Rapaport, W. J.; Kandefer, M.; Johnson, F. L.; and Goldfain, A. 2007. Metacognition in SNePS. *AI Magazine* 28:17–31.

Shapiro, S. C.; Ismail, H. O.; and Santore, J. F. 2000. Our dinner with Cassie. In *Working Notes for the AAAI 2000 Spring Symposium on Natural Dialogues with Practical Robotic Devices*, 57–61. Menlo Park, CA: AAAI.

Shapiro, S. C.; Martins, J. P.; and McKay, D. P. 1982. Bidirectional inference. In *Proceedings of the Fourth Annual Meeting of the Cognitive Science Society*, 90–93.

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

Shapiro, S. C. 1979. The SNePS semantic network processing system. In Findler, N. V., ed., *Associative Networks: The Representation and Use of Knowledge by Computers*. New York: Academic Press. 179–203.

Shapiro, S. C. 1982. Generalized augmented transition network grammars for generation from semantic networks. *The American Journal of Computational Linguistics* 8(1):12–25.

Shapiro, S. C. 1986. Symmetric relations, intensional individuals, and variable binding. *Proceedings of the IEEE* 74(10):1354–1363.

Shapiro, S. C. 1987. Processing, bottom-up and top-down. In Shapiro, S. C., ed., *Encyclopedia of Artificial Intelligence*. New York: John Wiley & Sons. 779–785. Reprinted in Second Edition, 1992, pages 1229–1234.

Shapiro, S. C. 1989. The CASSIE projects: An approach to natural language competence. In Martins, J. P., and Morgado, E. M., eds., *EPIA 89: 4th Portugese Conference on Artificial Intelligence Proceedings, Lecture Notes in Artificial Intelligence 390*. Berlin: Springer-Verlag. 362–380.

Shapiro, S. C. 1991. Cables, paths and "subconscious" reasoning in propositional semantic networks. In Sowa, J., ed., *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Los Altos, CA: Morgan Kaufmann. 137–156.

Shapiro, S. C. 1992. Artificial intelligence. In Shapiro, S. C., ed., *Encyclopedia of Artificial Intelligence*. New York: John Wiley & Sons, second edition. 54–57.

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*. Menlo Park, California: AAAI Press. 136–143.

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

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*. Menlo Park, CA: AAAI Press/The MIT Press. 175–195.

Shubin, H. 1981. Inference and control in multiprocessing environments. Technical Report 186, Department of Computer Science, SUNY at Buffalo.