A COMPUTATIONAL FRAMEWORK FOR IMPLEMENTING BAARS' GLOBAL WORKSPACE THEORY OF CONSCIOUSNESS

Ivan Moura, HEC-Inforge, BFSH1, University of Lausanne, 1015 Lausanne, Switzerland
ivan.moura@unil.ch, tel. +41 21 692 34 10, fax +41 21 692 34 05

Pierre Bonzon, HEC-Inforge, BFSH1, University of Lausanne, 1015 Lausanne, Switzerland
pierre.bonzon@unil.ch, tel. +41 21 692 34 12, fax +41 21 692 34 05

ABSTRACT

We consider Baars' "Global Workspace" theory of consciousness and discuss its possible representation within a model of intelligent agents. We first review a particular agent implementation that is given by an abstract machine, and then identify the extensions that are required in order to accommodate the main aspects of consciousness. According to Baars' theory, this amounts to unconscious process coalitions that result in the creation of contexts. These extensions can be formulated within a reified virtual machine encompassing a representation of the original machine as well as an additional introspective component. This computational framework is illustrating throughout using a simple working example.

1 INTRODUCTION

The study of consciousness can follow quite a few different paths. At one end of the spectrum lies the so-called "search for the neural correlates of consciousness" [2]. Briefly, this approach involves "isolating the neural processes that correlate with various states of consciousness". Following the increased availability of brain imaging techniques, the experimental research conducted under this banner is burgeoning. At the other end of the spectrum, we find theories that are formulated merely in terms of information processing presumably conducted by the brain [7]. This line of research has often been described as constituting a complementary "search for the computational correlates of consciousness". This second approach however has yet to mature to the point of producing concrete definitions leading to experimental platforms. The aim of our own research is to contribute to such an effort. Towards this end, we decided first to delineate a functional aspect of consciousness that could easily be amenable to a computational process. To allow then for practical experiments that could be replicated by others, we decided to situate our developments within a model of intelligent agents enjoying both formal and executable specifications. We are thus able to present a generic computational framework for implementing Baars' *Global Workspace* theory of consciousness [3]. More precisely, we reproduce in concrete terms the rather abstract notions of *unconscious processor coalitions* and the subsequent creation of unconscious *contexts* that lie at the heart of this theory.

The rest of this paper is organized as follows: in section 2, we briefly review existing models of consciousness, and explain why we did retain Baars' theory; in section 3 we introduce Baars' global workspace model; in section 4, we identify its various computational correlates; in section 5 we introduce a particular model of a deliberative agent given as a sequential abstract machine; section 6 presents the corresponding reified virtual machine given under the form of concurrent threads with an introspective extension; finally section 7 allows for the creation of contexts. We conclude by confronting our resulting computational framework with Aleksander and Dunmall's [1] axioms for consciousness.

2 VARIETIES OF CONSCIOUSNESS

It has become customary to map the functional aspect of consciousness into four distinct roles i.e., *access*, *phenomenal*, *monitoring* and *self*-consciousness [4]. Briefly, *access* consciousness (A-consciousness) refers to the human capacity to use language and explicit planning in order to act towards a specific goal. *Phenomenal* consciousness (P-consciousness) allows one to actually feel (emotional experiences, sensations) and thus to get qualitative inputs (appropriately named *qualia*) that amount to differentiating perceptions. *Monitoring* consciousness (M-consciousness) refers to the state or process of awareness that leads to one's sensations and percepts, as opposed to the contents of those sensations and percepts themselves. Finally, *self*-consciousness (S-consciousness) is the reflective capability we enjoy when we think about ourselves. Few information processing theories (if any) make a clear distinction between these different types of consciousness, and it is still unclear whether or not these functions can actually be dissociated.

Existing computational models of consciousness can be further distinguished according to the following

orthogonal properties [2]: *process* vs. *representation* theories, on one hand, and *specialized* vs. *non specialized* ones, on the other. We quote from [2]: "The process vs. representation dimension opposes models that explain consciousness in terms of specific processes operating over mental representations, with models that explain consciousness in terms of intrinsic properties of mental representations". The specialized versus non-specialized distinction simply refers to the existence or non existence of a dedicated machinery for consciousness. In the absence of such dedicated machinery, consciousness could be said to emerge from the "collective activity of many components distributed both spatially and functionally across the brain, none of them responsible for consciousness on its own".

Clearly, any candidate theory for an effective computational simulation should qualify as a *specialized process* model. Furthermore, if we take into account the results of decades of research done in the field of Artificial Intelligence, A-consciousness appears to be the function most likely to be ever replicated on a computer. Among all existing A-consciousness specialized process models, Baars' *global workspace* theory [3] is undoubtedly the most elaborate one, and thus became our natural choice.

As a prerequisite towards reproducing A-consciousness, we need an effective model of coordinated action and planning. Then only ultimately, when this gets implemented as an abstract machine, models of embodied cognition (such as A-consciousness) will possibly emerge from complex systems that are built upon them. Our recent proposal, that integrates formal communication primitives within a multi-agent system with plans [5] and defines a language for agent dialogues [6], is an example of a basic abstract machine that could be used for that purpose. We may add that S-consciousness, and perhaps also forms of P- and M-consciousness could only emerge from complex systems that are interpreted by a declarative self-representation of such an abstract machine (we are presently working towards that goal).

3 THE GLOBAL WORKSPACE MODEL

In this model, consciousness is seen as a place where unconscious elements interact with the system in order to access the *global workspace*.

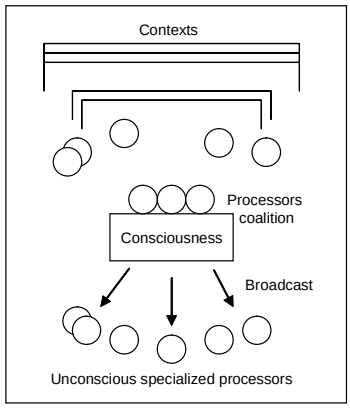

Figure 1: Baars' global workspace model

Figure 1 offers a view of the essential components of the model. The *unconscious specialized processors* represent the basic components in an entity. They are autonomous and work in parallel with other processors. Some processors may have to achieve a more complex task and thus do need to collaborate with other processors. Towards this end, a processor has to access consciousness. Consciousness is seen as a *global workspace* whose role is to *broadcast* the needs of the conscious processor. The processors fitting these needs form a *coalition*. A coalition can in turn access the global workspace if it is not able to achieve the required task, and so on until all the necessary processors are in the coalition. A same coalition formed several times is fixed into a *context*. A context involves then several unconscious processors pursuing a common goal. A context is unconscious as it has been *learned* by the consciousness while forming coalitions. The model does not specify how much "several times" means, therefore we consider in our implementation that once a coalition was able to achieve a task it is fixed into a context.

Hence contexts represent the framework of the system and influence directly the processors behaviour. For example the perception of the horizon is different from the hard ground or from a pitching boat. Several contexts can be active at the same time, they are then organised in a hierarchy. A context can be disrupted, its processors must then gain access to the global workspace to form it again. Moreover if the disrupting context embeds other contexts, all of them disrupt in series. These disruptions actually constitute feelings like *surprise*.

4 IDENTIFYING THE COMPUTATIONAL CORRELATES OF CONSCIOUSNESS

Observing the global workspace effects leads to

notice that it has actually at least two functions. First consciousness gives unconscious processors the opportunity to form a coalition thus allowing to achieve a given task. This is related to *deliberation.* Indeed the entity embedding consciousness has to *think about* how to achieve a given task. Secondly the global workspace allows to *learn* repetitive tasks creating *automatic reflexes*, called *contexts*. From a computational point of view, creating such contexts amounts to *compile* the deliberation process. We propose then to distinguish these two levels of consciousness as they pursue two different goals in parallel.

To implement our model, we identify two types of processors (in the sense of Baars):

- Processors that *depict the environment*, providing either a representation of external elements or a situation information. They are typically autonomous and do not require any coalitions as they represent a *fact*.
- Processors that *activate an action*. The action itself is not represented, even if an action can be made of several smaller actions (e.g. moving a step forward implies a vast number of movements). We assume that such an action is typically automated, i.e. represented within a context. Calling an action amounts then to call a context and action activation results in *action selection*. Some actions can be autonomous and can be self activated without appealing to consciousness (e.g. breathing, heart beating). More interesting are the actions that can be activated only in some situations, depending on the environment change. Such processors are not autonomous and need to access consciousness in order to build a coalition with one or many processors depicting environment.

We then define *deliberation* as the formation of *coalitions* leading to activate an action. A process needed by the action to be activated constitutes a *condition* for the action selection. Deliberation is recursive in the sense that a coalition may need to access the global workspace to associate new processors to the coalition.

A *context* is defined as an unconscious and permanent processor coalition. A context discharges consciousness from forming coalitions as they are *compiled*. Hence it allows to select directly an action without calling on the global workspace.

An example of deliberation is provided in section 5 when introducing plan deduction. An example of context creation is provided in section 6 after having introduced multithreading.

5 A SEQUENTIAL ABSTRACT MACHINE

As already indicated in section 2, our implementation of the computational correlates we just identified will rely on an abstract machine. This abstract machine consists of a set of procedures implementing abstract functions defining a general model of reactive agent with sensing [8]. These procedures generate runs for individual non-deterministic agents with plans.

The agents' behaviour is defined with *plans*. A plan is given under the form of universally quantified implications *"conditions"* $\Rightarrow$ *do(p,a)* for action selection or *"conditions"* $\Rightarrow$ *switch(p,p')* for changing plan where *a* is an action and *p* a plan name. A plan describes a situation where one can *execute* an action - if available in a given plan - or *switch* plan if the related conditions are met.

As an example consider the following plans description for an agent whose goal is to wander in a grid and to suck dirt when he finds some.

true	$\Rightarrow$ ***switch****(initial, start)*
dirt(X,Y)	$\Rightarrow$ ***switch****(start, work)*
dirt(X,Y) $\wedge$ *in(X,Y)*	$\Rightarrow$ ***switch****(work, suck(X,Y))*
true	$\Rightarrow$ ***do****(suck(X,Y), suck(X,Y))*
$\neg$*dirt(X,Y)* $\wedge$ *in(X,Y)*	$\Rightarrow$ ***switch****(work, move(X,Y))*
true	$\Rightarrow$ ***do****(move(X,Y), move(X,Y))*
$\neg$*dirt(X,Y)*	$\Rightarrow$ ***switch****(start, home)*
in(X,Y)	$\Rightarrow$ ***switch****(home, back(X,Y))*
true	$\Rightarrow$ ***do****(back(X,Y),back(X,Y))*

The *switch/2* predicate makes the agent change plan when the conditions are verified and *do/2* predicate executes an action in a given plan. Note that in this example the plan and the action do not have to bear the same name, the only requirements being that the arguments have to be passed to the action as it depends on the localisation.

These plans actually implement the agent's deliberation as previously defined. Conditions in the plans correspond to the processors needed to depict a precise aspect of the environment in the deliberation process. Plan names mark the deliberation progress, i.e. they set a coalition in a certain state. As long as an action can not be selected, the coalition needs to check a new condition. Once a coalition is complete, i.e. all the

conditions are met, an action is selected.

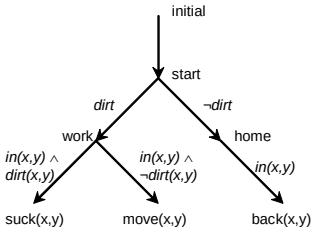

Figure 2: Tree view of our example's plans

Deliberation can be represented by a tree (see figure 2). The leaves represent the action selection processors, the *italic* branches represent the processors depicting environment and the nodes are the plan names, i.e. the deliberation progress.

The abstract machine is defined with the following procedures:

procedure *react(e,l,p)*
if $l \vdash do(p,a)$
then $e \leftarrow \tau_e(e,a)$
else *if* $l \vdash switch(p,p')$
then react(e,l,p')

procedure *sense(l,e)*
if *"the agent perceives s"*
then $l \leftarrow \tau_a(l,s)$

procedure *run(e,l)*
loop *sense(l,e);*
if $l \vdash plan(p_0)$
then *react(e,l,*p_0*)*

where *l* is the agent's local state, *e* the environment, *p* a plan (p_0 is the initial plan), *a* an action. $e \leftarrow \tau_e(e,a)$ and $l \leftarrow \tau_a(l,s)$ are transition functions on the environment when an action *a* is executed respectively on the local state when *s* is perceived. Thus *react* tries to deduce a predicate *do/2*; if it succeeds it executes the action *a* and updates the environment otherwise it switches plan. Procedure *sense* updates the agents local states when it perceives a percept *s* from the environment. Finally the main procedure *run* is a loop on the *sense* and *react* procedures. Note that *run* calls *react* with an initial plan that is deduced at each cycle, thus representing the proactive capability of an agent.

This abstract machine implements the deliberation process via sequential procedures. Recalling that consciousness is made of parallel processors, that several coalitions can access consciousness and that consciousness itself is made of two concurrent levels, i.e. deliberation and context formation, calls for a parallel abstract machine.

6 A MULTITHREADED VIRTUAL MACHINE

Using a concurrent agent language [6], we can translate the sequential abstract machine into concurrent dialogues, each dialogue being executed in separated thread. These dialogues are compiled into plans executed on an abstract machine similar to the previous one (actually extended to support multithreading). Hence this multithreaded virtual machine encompasses the sequential abstract machine. The following dialogues implement the multithreaded virtual machine:

dialog(*run(*P_0*), []*
*[****concurrent****(sense),*
concurrent*(react(*P_0*))])*,

dialog(*sense, [S]*,
*[****ask****(native_sense, S),*
execute*(store(S)),*
resume*(sense)])*,

dialog(*react(P), []*,
*[((do(P,A) | [****concurrent****(do(P,A))]);*
*(switch(P,Q) | [****concurrent****(switch(P,Q))]))])*,

dialog(*do(P,A), []*,
*[****execute****(A),*
concurrent*(react(*P_0*))])*,

dialog(*switch(P,Q), []*,
*[****concurrent****(react(Q))]).*

The behaviour of an agent is described in the same manner as previously, i.e. the plans presented in the previous section apply on this machine. Figure 3 offers a graphical representation of the machine. Note that the arrows representing interactions with the environment are doted as they are taken over from the lower abstract machine (e.g. *ask(native_sense, S)* in the *sense* dialogue

refers to the *sense* procedure of the latter).

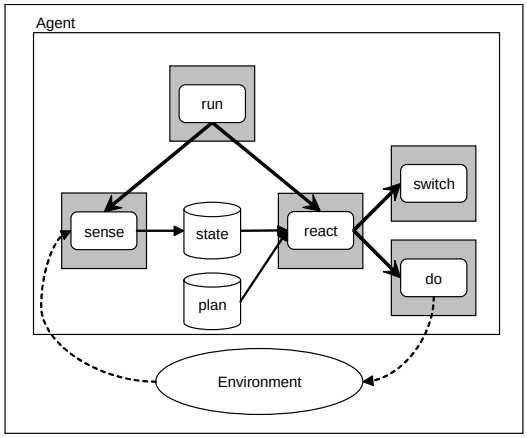

Figure 3: A representation of the virtual machine

This multithreaded virtual machine is made of five concurrent dialogues but actually of only two parallel independent (i.e. not synchronized) loops: *sensing* and *plan/action* selection. These loops correspond to the main elements of deliberation fitting the needs for an action to be selected on the basis of the environment depiction.

At this level, the multithreaded machine behaves similarly to the sequential machine (see section 5). In order to initiate consciousness we *reflect* the agent's deliberation to a meta level: the *consciousness*. Reflection achieves the agent's ability to become conscious of its deliberation steps. Towards this end we create a new dialogue *introspect* that starts a new thread attached (i.e. synchronized) on both basic deliberation threads: *switch* and *do* (see figure 4).

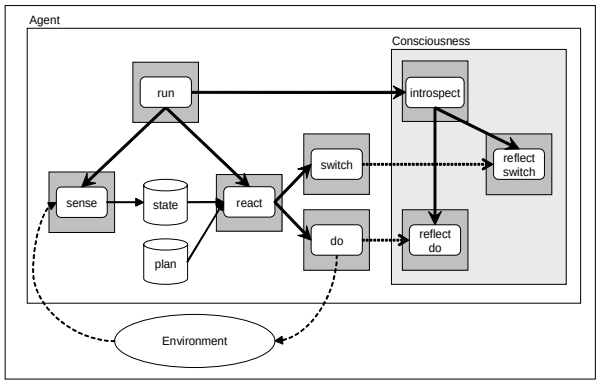

Figure 4: Reflection is achieved by synchronizing the basic deliberation threads with their correspondent reflect thread, represented with the bold doted arrows.

These threads are given by

dialog*(reflect(do), [P,A],*
*[**ask**(do(P,A), do(P,A)),*
** some handling *,*
resume*(reflect(do))]),*

dialog*(reflect(switch), [P,Q],*
*[**ask**(switch(P,Q), switch(P,Q)),*
** some handling *,*
resume*(reflect(switch))]),*

All these dialogues are very similar: they first "listen" to their corresponding thread and after having handled the information they resume. The corresponding threads have to be modified as follows thus achieving synchronisation:

dialog*(do(P,A), [],*
*[**execute**(A),*
tell*(reflect(do), do(P,A)),* ← sync.
concurrent*(react(P_0))]),*

dialog*(switch(P,Q), [],*
*[**tell**(reflect(switch), switch(P,Q)),* ← sync.
concurrent*(react(Q))]),*

7 IMPLEMENTING THE COMPUTATIONAL CORRELATES OF CONSCIOUSNESS

We can now extend the machine to implement context learning, by adding new concurrent threads. Towards this end, we first handle the reflected deliberation progress recording each plan switching with the related condition. A reflected action starts the compilation process. These dialogues are given hereafter:

dialog*(reflect(switch), [P,Q],*
*[**ask**(switch(P,Q), switch(P,Q)),*
call*(switch(P,Q), C ⇒ switch(P,Q)),*
execute*(gw.store(switched(C,P,Q))),*
resume*(reflect(switch))]),*

dialog*(reflect(do), [P,A],*
*[**ask**(do(P,A), do(P,A)),*
concurrent*(trace(done(P,A))),*
resume*(reflect(do))]),*

Reflect(switch) stores a *switched/3* predicate containing the condition *C* that triggered the plan switching from *P* to *Q* in a dedicated object *gw* (standing for *global workspace)*. The condition is obtained through a request communication act synchronizing the *switch* dialogue with its reflect dialogue. The former has then to be modified as follows:

dialog*(switch(P,Q), [Request],*
*[**tell**(reflect(switch), switch(P,Q)),*
__return__(reflect(switch,Request)), ←
__concurrent__(react(Q))])

Reflect(do) starts the compilation process initiated by the dialogue *trace*. Applying reflection to our example we may obtain the following predicates in the *gw* object:

switched(true, initial,start),
switched(dirt(0,1), start, work),
switched((in(0,0) ∧ not dirt(0,0)), work, move(0,0))

Compilation consists in grouping all conditions having led to select an action and generating a new plan of type *"context"* ⇒ *do(initial,a)* where *context* is the context formed with all conditions and generalizing their arguments. These plans are similar to those used in deliberation but they select directly an action once the context is met. Compilation is implemented with the following dialogues:

dialog*(trace(done(P,A)), [Q],*
*[**execute**(gw.retrieve(switched(_,Q,P))),*
__enter__(get_context(A,Q,P,[]))]),

dialog*(get_context(A,P,Q,Trace), [C,O],*
*[**execute**(gw.retrieve(switched(C,P,Q))),*
((not (P=initial) |
*[**execute**(gw.retrieve(switched(_,O,P))),*
__resume__(get_context(A,O,P,[C|Trace]))]);
(P=initial |
*[**execute**(gw.generate(A,[C|Trace]),Ctx),*
__concurrent__(teach(Ctx))]))]),

Applying it to our example, the following plans are generated:

(in(X,Y) ∧ not dirt(_,_) ∧ not alarm)
⇒ ***do****(initial, back(X,Y))*

((in(X,Y) ∧ not dirt(X,Y)) ∧ dirt(_,_) ∧ not alarm)
⇒ ***do****(initial, move(X,Y))*

((in(X,Y) ∧ dirt(X,Y)) ∧ dirt(_,_) ∧ not alarm)
⇒ ***do****(initial, suck(X,Y))*

(in(X,Y) ∧ not dirt(_,_) ∧ not alarm)
⇒ ***do****(initial, back(X,Y))*

Again, these plans are very similar to the previous plans. However an important distinction has to be pointed out: they are only action selection plans as they comprise no *switch* instructions.

A new plan is set in the agent's state through the dialogue *teach* that run at the termination of *get_context*. The dialogue *teach* simply transmits the plan to a *learn* dialogue that inserts it in the agent's local state. These dialogues are defined as follows:

reflexive*(teach(P), [],*
*[**tell**(learn,P)]),*

reflexive*(learn, [P],*
*[**ask**(teach(P),P),*
__execute__(store(P)),
__resume__(learn)])

Figure 5 provides a complete representation of the virtual machine just presented. To sum up, consciousness reflects deliberations and actions to new components that generate new specialized contexts under the form of plans and feed them back to the agent.

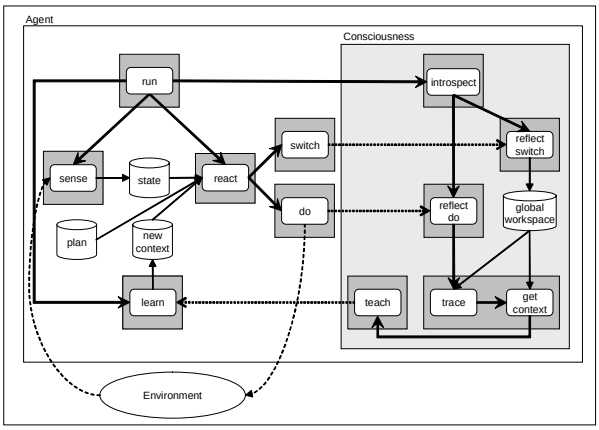

Figure 5: A complete view of the virtual machine

If all conditions of a new context are met, the action is immediately selected and the agent does not have to deliberate.

8 CONCLUSION

In order to evaluate how far our model implements consciousness from a functional point of view we confront it to the axioms proposed by Aleksander and Dunmall defining minimal consciousness [1]. We recall these axioms and confront each definition to our model:

- *(I) Depiction axiom* requires that an agent "has perceptual states" depicting part of the environment. Our *sense* procedure interacting with the environment provides perceptual states about the environment in the agent's local state represented by unconscious processors.
- *(II) Imagination axiom* says that an agent "must be

conscious of imagined as well as perceived events". The processors depicting environment may be conscious if they get involved in a coalition. On the other hand our agent is not imaginative at all as it runs predefined plans and works only on this basis. However we could consider that compiled contexts are "imagined elements" but at this level they are not conscious as they are precisely created to be unconscious.

- *(III) Attention axiom* is verified when an agent "is capable of selecting which parts of [the environment] to depict [...]". The plan/action selection function, i.e. the deliberation process coordinating processors that depict environment with processors activating actions within a coalition, achieves this axiom.
- *(IV) Planning axiom* implies that an agent "has means of control over imaginational state sequences to plan actions". Again our virtual machine doesn't implement imagination. However if we consider that self generated contexts are an imaginative process, the compilation process fits this axiom as it generates new plans leading to direct action selection.
- *(V) Emotion axiom* wants an agent to have "additional affective states that evaluate planned actions and determine the ensuing action". Following this definition we do clearly not deal with any emotion in our proposal.

Concerning the emotion axiom, recall that Baars considers that feelings like surprise - which can be seen as an emotion - is caused by a context disruption forcing the agent to use intensively its consciousness to form again the disrupted contexts. Implementing this point of view, i.e. extending the implementation of Baars' model, is the another issue of our future work.

9 REFERENCES

[1] I. Aleksander and B. Dunmall, Axioms and Tests for the Presence of Minimal Consciousness in Agents, in: O. Holland (ed.), *Machine Consciousness*, Journal of Consciousness Studies, vol 10, number 4-5, Imprint Academic, 2003.

[2] A.P. Atkinson, M.S.C. Thomas and A. Cleermans, Consciousness: mapping the theoretical landscape, *Trends in Cognitive Sciences*, Vol. 4 (10), October 2000.

[3] B.J. Baars, *A cognitive theory of consciousness*, Cambridge University Press, 1988.

[4] N. Block, On a confusion about a Function of Consciousness, Behavioral and Brain Sciences, vol. 18 (2), 227-287, 1995

[5] P. Bonzon, An Abstract Machine for Classes of Communicating Agents Based on Deduction, in: J.-J. Meyer and M. Tambe (eds), *Intelligent Agents VIII*, LNAI vol. 2333, Springer Verlag, 2002.

[6] P. Bonzon, Compiling Dynamic Agent Conversations, in: M. Jarke, J. Koehler G. and Lakemeyer (eds), *KI 2002: Advances in Artificial Intelligence, proc. of the 25th German Conference on AI*, LNAI vol. 2479, Springer Verlag, 2002.

[7] R. Sun, Computational Models of Consciousness: An Evaluation, *Journal of Intelligent Systems*, vol. 9, 1999.

[8] M. Wooldridge and A. Lomuscio, Reasoning about Visibility, Perception, and Knowledge, in: N.R. Jennings and Y. Lespérance (eds), *Intelligent Agents VI*, LNAI vol. 1757, Springer Verlag, 2000.