Lehrstuhl für
Wirtschaftsinformatik
Information Systems
Management

No. 47
December 2009

Bayreuther Arbeitspapiere zur Wirtschaftsinformatik

Tina Balke, Serena Villata, Daniel Villatoro (Eds.)

Proceedings of the 11th European Agent Systems Summer School Student Session

Bayreuth Reports on Information Systems Management

UNIVERSITÄT
BAYREUTH

ISSN 1864-9300

Proceedings of the

11th European Agent Systems Summer School

Student Session

Preface

This volume contains the papers presented at the Student Session of the *11th European Agent Systems Summer School* (EASSS) held on 2nd of September 2009 at Educatorio della Provvidenza, Turin, Italy.

The Student Session, organised by students, is designed to encourage student interaction and feedback from the tutors. By providing the students with a conference-like setup, both in the presentation and in the review process, students have the opportunity to prepare their own submission, go through the selection process and present their work to each other and their interests to their fellow students as well as internationally leading experts in the agent field, both from the theoretical and the practical sector.

As the goal of the Student Session is to provide the speakers with constructive feedback and a means to be introduced to the community, the competitive elements often found in conferences (best paper award, best presentation award) are intentionally omitted. Preparing a good scientific paper is a difficult task, practising it is the benefit of this session.

All submissions were peer-reviewed and accepted paper submissions were assigned a 25 minute slot for presentation at the Summer School. Typically a presentation either detailed the intended approach to a problem or asked a specific question, directed at the audience.

The review process itself was extremely selective and many good papers could not be accepted for the final presentation. Each submission was reviewed by 4 programme committee members on the average, which decided to accept the 4 full and 4 short papers that are presented in these proceedings.

Overall, the EASSS'09 Student Session as well as the Summer School in general were a great success that could not have been achieved without the support of the numerous reviewers as well as the local EASSS organizers. We want to thank all of these people and are looking forward to seeing you again next year.

October 2009

Tina Balke
Serena Villata
Daniel Villatoro

Student Session Organization

Programme Chairs

Tina Balke
Serena Villata
Daniel Villatoro

Local Organization

Serena Villata

Programme Committee

Stephane Airiau
Giulia Andrighetto
Luis Antunes
Manuel Atencia
Patrizio Barbini
Guido Boella
Marco Campenni
Jordi Campos Miralles
Dan Cartwright
António Castro
George Christelis
Marina De Vos
Irina Diana Coman
Massimo Cossentino
Petar Curkovic
Sylvain Dekoker
Maria del Carmen Delgado
Juergen Dix
Ulle Endriss
Marc Esteva
Torsten Eymann
Berndt Farwer
Maria Fasli
Francesco Figari
José M. Gascueña
Nicola Gatti
Carlos Grilo
Davide Grossi
Hanno Hildmann
Benjamin Hirsch
Sebastian Hudert
Joris Hulstijn
Manoela Ilic
Wojciech Jamroga
Fredrik Johansson
Jean Christoph Jung
Rosine Kitio
Franziska Klügl
Lena Kurzen
Tobias Küster
João Leite
Shuangyan Liu
Brian Logan
Marin Lujak
Alessandro Maccagnan
Mircea Moca
Sanjay Modgil
Ambra Molesini
Pablo Noriega
Ingrid Nunes
Heather S. Packer
Julian Padget
Mario Paolucci
Damien Pellier
Antonio Pereira
Adrian Perreau de Pinninck
Gabriella Pigozzi
Isaac Pinyol
Michele Piunti
Eric Platon
Evangelos Pournaras
Abdur Rakib
Alessandro Ricci
Mikheil Rukhaia
Jordi Sabater-Mir
Norman Salazar
Ahmad Sardouk
Julien Siebert
Marija Slavkovik
Jackeline Spinola de Freitas
Eugen Staab
Tomislav Stipancic
Gaia Trecarichi
Leon van der Torre
Wamberto Vasconcelos
Laurent Vercouter
Meritxell Vinyals
Cees Witteveen
Yining Wu
İlker Yıldırım

Table of Contents

Towards an Inductive Algorithm for Learning Trust Alignment

Andrew Koster
Artificial Intelligence Research Institute, CSIC
Bellaterra, Spain
andrew@iiia.csic.es

Jordi Sabater-Mir
Artificial Intelligence Research Institute, CSIC
Bellaterra, Spain
jsabater@iiia.csic.es

Marco Schorlemmer
Artificial Intelligence Research Institute, CSIC
Bellaterra, Spain
marco@iiia.csic.es

ABSTRACT

Knowing which agents to trust is an important problem in open multi-agent systems. A way to help solve this problem is by allowing agents to relay information about trust to each other. We argue trust is a subjective phenomenon and therefore needs aligning. We present a mathematical framework for communicating about trust in terms of interactions. Based on this framework we present an algorithm based on clustering and inductive logic programming techniques to align agents' trust models.

Keywords

inductive logic programming, trust, alignment, learning

1. INTRODUCTION

In complex, distributed systems, such as multi-agent systems, the artificial entities have to cooperate, negotiate, compete, etc. amongst themselves. Thus the social aspect of these systems plays a crucial role in their functioning. One of the issues in such a social system is the question of whom to trust and how to find this out. There are several systems already in development that model trust and reputation [16], ranging from a straightforward listing of evaluations (such as eBay's [13] reputation system), to complex cognitive models (such as Repage [18]). We anticipate that in an open multi-agent system, there will be a large diversity of models in concurrent use by different agents, depending on the wishes of the programmer and the user. However, even if there is consensus on some model, this is still only a consensus on the computational representation. In a heterogeneous environment it is inevitable that, if the trust model an agent uses is based on cognitive principles, the way different agents interpret their environment will still lead to differences in trust. We will show how, despite agreeing on the ontological underpinnings of the concepts, there is the need to align trust so as to enable reliable *gossip*. With gossip we refer to all communication about trust.

We will emphasize the need to align trust further by considering a simple example of a multi-agent system with three agents.

Alice wants to know if Dave would be a good keynote speaker for the conference she is organizing. However, she does not know enough about him. She asks Bob. Bob has never collaborated with Dave directly, but they work at the same institute and play squash together. Through these interactions, Bob has trust in Dave and tells this to Alice.

Lets analyse Bob's model. He does not know Dave professionally and bases his trust in Dave on personal interactions. This is a perfectly valid model, but lets assume Alice's model works differently: she only takes academic accomplishments into account. She should therefore disregard Bob's gossip, because it is based on, what she considers, *unreliable* information. We emphasize that we differentiate between the trust she has in Bob and the reliability of the information he sends her. Her trust in Bob is grounded in her trust and reputation model. However, what we want to find out is whether the gossip Bob sends can be interpreted reliably in Alice's model.

This short example shows that even in simple situations the concepts related to trust are highly personal and communication about them is no straightforward matter. In the case that two agents wish to exchange information about trust it is therefore important to clarify what trust means to each of them. This can be done in an alignment process, based on similar protocols in ontology alignment, concept formation and other related fields. Some work has been done in defining common ontologies for trust [14, 7], however in practice these ontologies do not have the support of many of the different trust methodologies in development. Even if support were added for all systems and a common ontology emerged, we could still not use it to communicate effectively. Trust is an inherently personal phenomenon and has subjective components which cannot be captured in a shared ontology. An adaptable approach that takes the different agents' points of view into account is needed.

Abdul-Rahman and Hailes' reputation model [1] approaches the problem from another direction, by defining the trust evaluations based on the actual communications. The interpretation of gossip is based on previous interactions with the same sender. The problem with this, however, is that it is incomplete: firstly it assumes all other agents in the system use the same model, which in a heterogeneous environment will hardly ever be the case. Secondly, it uses a heuristic based on prior experiences, called the semantic distance, to "bias" received messages. The semantic distance is an average of all previous experiences. They do not differentiate between recommendations about different agents, which are based on different types of interactions.

We propose to enrich the model of communication by con-

Cite as: Towards an Inductive Algorithm for Learning Trust Alignment, Andrew Koster, Jordi Sabater-Mir and Marco Schorlemmer, *European Agents System Summer School 2009)*

sidering it separate from the actual trust model. By doing this, we can allow for different trust models. We note, however, that while trust is modeled in disparate ways, all definitions do agree on the fact that trust is a social phenomenon. Just as any social phenomenon, it arises from the complex relationships between the agents in the environment and, without losing generality, we say these relationships are based on any number of interactions between the agents. These interactions can have many different forms, such as playing squash with someone, buying a bicycle on eBay or telling Alice that Dave is a trustworthy keynote speaker. Note that not all interactions are perceived equally by all participants. Due to having different goals, agents may observe different things, or even more obviously: by having a different vantage point. Simply by having more (or different) information available, agents may perceive the interaction itself differently. In addition, interactions may be accompanied by some kind of social evaluation of the interaction. These can range from an emotional response, such as outrage at being cheated in a trade, to a rational analysis. Thus, we see that how an agent experiences an interaction is unique and personal. This only adds to the problem we are considering. To be able to align, there needs to be some common ground from which to start the alignment, but any agent's experience of an interaction is subjective, and thus not shared. We call this personal interpretation of the interaction an *observation*. We say an agent's observations allow it to evaluate trust.

Now that we have discussed what interactions mean to a single agent, we will return to the focus of communicating about trust. One interaction may be observed by any number of agents, each making different observations, which support different trust evaluations of different targets performing different roles. However, to communicate about trust evaluations, the agents need to have a starting point: some basic building blocks they implicitly agree they share. We note that the interactions provide precisely such a starting point. While all the agents' observations are different, they do share one specific thing: *the interaction itself*. We therefore argue that to find a reliable alignment between two agents they can align based on these interactions.

Our approach uses these shared interactions as building blocks to align the agents' trust models, based on the gossip they send each other. The gossip specifies certain interactions, which each agent observes differently. These observations form the support for an agent's trust evaluation. If another agent communicates this trust evaluation, the interpretation should be based on the underlying interactions. An alignment of the trust models gives a way of doing this by gossiping about the agents' trust evaluations and the observations (and thus interactions) they base these on.

Semantic alignment based on interactions has been studied in [2]. This approach to semantic alignment is based on the general framework of Channel Theory [3, 19]. We use this same mathematical theory as a framework for aligning trust and introduce it in the next section before discussing the technical details of the algorithm.

2. THE ALGORITHM

Before we consider possible solutions we need a clear definition of the problem we are considering. We follow the formalization we described in [10] and will summarize it briefly in the following sections. Firstly we consider agents with heterogeneous trust models, but we have no clear description of what a trust model is in the first place. We explain this in Section 2.1. Furthermore, to align, the agents need to communicate. For this we will need to define a language in Section 2.1.1. And finally, the agents need to have some method of forming an alignment based on the statements in this language. This we describe in Section 2.1.2. In Section 2.2 we describe the computational approach we take. The whole process is summarized in Figure 1.

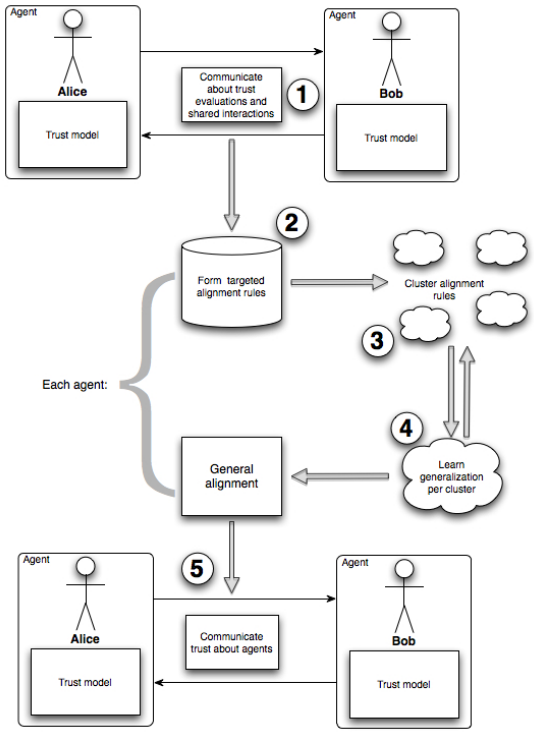


Figure 1: Schematic diagram of the steps in the alignment process

2.1 A Formal Representation

As argued in Section 1, interactions form the building blocks for talking about trust. An interaction is observed by different agents and represented internally by them. These observations then lead to trust evaluations of the various agents involved. Any trust model can therefore be described as a binary relation between an agent's observations and its trust evaluations. In addition, trust always has a target: any form of representing trust will have a trusting agent and a target agent, which is the agent the trust evaluation is about. It is assumed that any agent's trust evaluations can be represented in some formal language $\mathcal{L}_{Trust}$. Note that because trust is a subjective phenomenon, the semantics of this language aren't shared, but by sharing the syntax the agents can communicate about it. A trust model is therefore a binary relation $\models$, such that $X \models \varphi$ means that there is a set of observations X which support trust evaluation $\varphi \in \mathcal{L}_{Trust}$. The observations X are unknown as they are an internal representation of the agent. However, we know these are based on some set of interactions. If $\mathfrak{O}$ is the set of an agent's possible observations and $\mathfrak{I}$ is the

set of all interactions in the environment, then each agent A has a function $observe_A : \mathfrak{I} \to \mathfrak{O}_A$ which associates interactions with observations. The observations X in the trust model are therefore generated (with the *observe*-function) from some set of interactions $I \subseteq \mathfrak{I}$. These interactions are facts in the environment all agents may know about and can be used as the basis of an alignment.

2.1.1 Formalizing gossip

In addition to $\mathcal{L}_{Trust}$ a second language is needed for effective trust alignment: a language in which to talk about the interactions. Knowing which information about the interactions is relevant depends on the domain. Thus a language for discussing interactions comes from the domain the agents operate in. Usually such a language already exists or is defined together with the MAS. We call this language $\mathcal{L}_{Domain}$ and note that it is a shared language: both the syntax and the semantics are known by all agents in the system, as opposed to the semantics of $\mathcal{L}_{Trust}$, which is interpreted differently by the agents. With this shared language it is possible to define exactly what it means for two agents to share an interaction. A set of interactions I is shared by agents A and B if there is some $\varphi \in \mathcal{L}_{Domain}$ such that φ is in both A and B's sets of observations of interaction I, or, in other words, φ is the information shared between the agents about I. Formally neither agent can know that φ is observed by the other agent, however if we limit $\mathcal{L}_{Domain}$ to objective and easily observable properties of the domain, we assume such φ exist.

Messages, containing a trust evaluation of a specific target in $\mathcal{L}_{Trust}$ and pinpointing the specific shared interactions this evaluation is based on in $\mathcal{L}_{Domain}$, form the basis of the trust alignment. We call such messages "gossip". Formally we say gossip from agent B to agent A is a message `gossip`(T, β, ψ), with T the target of the trust evaluation $\beta \in \mathcal{L}_{Trust}$ and $\psi \in \mathcal{L}_{Domain}$ describing the set of interactions I which support trust evaluation β for agent B. We cannot simply enumerate the interactions in I because agents may not be willing to do so. $\mathcal{L}_{Domain}$ serves a double purpose: firstly it may be more descriptive, giving more information than simply an enumeration of interactions. Secondly it may allow agents to describe interactions without pinpointing them exactly. This allows agents to align without divulging sensitive information. Sending gossip messages is point 1 in Figure 1.

The receiving agent A can now use its own trust model to find an $\alpha \in \mathcal{L}_{Trust}$, such that α is supported by I and the resulting rule $\alpha \leftarrow \beta, \psi$ will form the basis of our alignment. What this rule means is: the interactions which support ψ, support trust evaluation α for agent A and β for agent B. These rules are at point 2 in Figure 1. The goal is now to find a way of generalizing from such rules to a more general, predictive model, such that, for example, agent A can know what trust evaluation α' it should associate with a certain $\beta' \in \mathcal{L}_{Trust}$, given ψ, despite neither knowing the interactions which support ψ nor being able to conclude an own trust evaluation from the observation of those interactions. This would be the outcome of the algorithm, applied at point 5 in Figure 1.

2.1.2 Generalizations and coverage

Now that we have a way of describing the relationship (alignment) of two agents' trust models with regards to a specific target, we wish to expand this idea to a more predictive model: we wish to find the more general alignment between the trust models. This problem is considered as an inductive learning problem [8]. Given a number of targeted alignments with regards to different agents, is there an alignment that describes all (or most) of them?

To use inductive learning, it is necessary to define what the solution should look like. This should be a generalization of the above mentioned rules $\alpha \leftarrow \beta, \psi$. We note that both $\mathcal{L}_{Trust}$ and $\mathcal{L}_{Domain}$ are represented in a standard first-order logic. Thus it is possible to use θ-subsumption to generalize these rules. The way to do this is by structuring the search space. The solution should be the least general alignment, which covers all the rules given in the messages. A hypothetical alignment $\mathfrak{T}$ is said to cover a rule $\alpha \leftarrow \beta, \psi$ if there is a rule $\Gamma \leftarrow \Delta, \Psi \in \mathfrak{T}$ such that all sets of interactions I which support $\alpha \leftarrow \beta, \psi$ also support $\Gamma \leftarrow \Delta, \Psi$. One hypothetical alignment $\mathfrak{T}$ is more general than another $\mathfrak{T}'$ if its coverage is greater: $\mathbf{c}(\mathfrak{T}) \supset \mathbf{c}(\mathfrak{T}')$. We write this $\mathfrak{T} \succ \mathfrak{T}'$. The overall trust alignment between two agents can now be found by finding a minimally general generalization, which covers all the communicated rules.

2.2 An Inductive Algorithm

As described in the preceding section, our algorithm must generalize the specific targeted alignments to a predictive ruleset. This is very similar to the problem in concept formation. The approach taken in these problems is by clustering the data together and finding a description of each cluster. However, the fact that we have descriptions in first-order logics invalidates the use of propositional clustering algorithms for this purpose [9]. Some more modern approaches combine clustering and ILP methods [12, 5] to allow for clustering of first-order formulas. This is exactly the problem we are trying to solve and we therefore propose a modification of these algorithms, using the distance function from [17] and a conventional agglomerative clustering algorithm. The found clusters can then be used as the input for an ILP algorithm to learn the generalizations. Furthermore, we have an additional wish: our partitioning may be too strict, which will not allow for enough positive examples and too many negative examples to learn anything useful. In these cases we will want to relax our partitioning criteria to amplify the base of positive examples, in the hope that this will allow for a better generalization. This obviously comes at the cost of accuracy of the predictive ruleset found, but this can be taken into account.

2.2.1 A short overview

The input of the algorithm will be any number of rules $\mathcal{R}$ generated from `gossip` statements. These rules, the same as described in Section 2.1.1, will serve as the initial input and have the form shown below, where $T_1, \ldots, T_m$ are target agents, $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n \in \mathcal{L}_{Trust}$ and $\psi_1, \ldots, \psi_n \in \mathcal{L}_{Domain}$ describe the interactions.

$$\alpha_1[T_1] \leftarrow \beta_1[T_1], \psi_1$$
$$\vdots$$
$$\alpha_i[T_j] \leftarrow \beta_i[T_j], \psi_i$$
$$\vdots$$
$$\alpha_n[T_m] \leftarrow \beta_n[T_m], \psi_n$$

Algorithm 1 Generalize rules $\mathcal{R}$

```
1: INPUT: set of SRAs to be generalized R
2: INPUT: distance measure on L_Trust D(x, y).
3: INPUT: set of increasing distances for clustering S
4: General_rules := ∅
5: Clusters := {{r}|r ∈ R}
6: Covered := ∅
7: for all Stopcriteria s in S do
8:    Clusters := agglomerative_clustering(Clusters, s, D)
9:    if |Clusters| = 1 then
10:      break
11:   end if
12:   for all C ∈ Clusters do
13:      H := generalize_head(C, R\C)
14:      if H ≠ null then
15:         G := generalize_body(C, R\C)
16:         if G ≠ null then
17:            General_rules := General_rules ∪ {⟨H ← G, s⟩}
18:            Covered := Covered ∪ C
19:         end if
20:      end if
21:   end for
22:   if Covered = R then
23:      break
24:   end if
25: end forGeneral_rules
```

This says there are n different rules about m different agents. To learn the underlying structure we will use Algorithm 1.

We use three important procedures, which we will explain in more detail: the distance metric D on targeted alignment rules, the clustering algorithm in line 1 and the generalization algorithm we use on the clusters in lines 1 and 1. This last one takes as input the rules in the cluster as positive examples and the rules outside clusters as negative examples and uses an ILP algorithm to learn a generalization. Furthermore we use the flag "terminate" to end the algorithm if at a certain clustering resolution we have rules covering all targeted alignments. In this case there is no reason to continue, because we have a maximum coverage of the examples.

We are interested in finding generalizations which allow us to predict what the receiving agent's trust evaluation α would be, given that the sending agent's trust evaluation is β, based on interactions which support ψ. We therefore need to be able to cluster the rules above according to the relative distance between the receiving agent's trust evaluations. The rest of the information in the rules is used to learn the generalization.

2.2.2 A distance metric

An agent's trust evaluation is in the $\mathcal{L}_{Trust}$ language, which in general could be any first-order logic. Distances on first-order logic objects have received a lot of attention lately [17]. Such distance measures work on arbitrary clauses, however, they do require them to be rewritten in clausal normal form (CNF). We note that for any closed formula in a first-order logic its CNF can be found in polynomial time [15]. The distance measure is then split up into two different parts:

- A context-dependent part, defining the distance between the disjunctions in the CNF in $\mathcal{L}_{Trust}$
- A generic part, defining the distance between phrases, based on the distance between the clauses in each phrase.

We stipulate, however, that the distance metric can be agent-specific and may be as complicated as the programmer wishes. To further illustrate this description of a distance metric, we give an example of $\mathcal{L}_{Trust}$ and a metric on it. Our example of $\mathcal{L}_{Trust}$ has the following predicates: $image(A, V)$ and $reputation(A, V)$, where A is an agent and $V \in [1, 10] \subset \mathbb{N}$. For the context-dependent part of the metric we use the closure under symmetry of the following recursive definition:

1. $d(\varphi_1 \vee \varphi_2, \psi_1 \vee \psi_2) = \dfrac{min\big[\big(d(\varphi_1,\psi_1)+d(\varphi_2,\psi_2)\big),\big(d(\varphi_1,\psi_2)+d(\varphi_2,\psi_1)\big)\big]}{2}$
2. $d(\varphi_1 \vee \varphi_2, \psi) = \dfrac{min\big[d(\varphi_1,\psi), d(\varphi_2,\psi)\big]+1}{2}$
3. $d(\neg\varphi, \neg\psi) = d(\varphi, \psi)$
4. $d(\neg\varphi, \psi) = 1$
5. $d(image(A_1, V_1), image(A_2, V_2)) = \dfrac{|V_1 - V_2|}{10}$
6. $d(reputation(A_1, V_1), reputation(A_2, V_2)) = \dfrac{|V_1 - V_2|}{10}$
7. $d(\varphi, \psi) = 1$ otherwise

As mentioned above, this distance measure is dependent on the language and the agent. All we require in the continuation is that it is defined for all simple clauses in $\mathcal{L}_{Trust}$ and that it is a metric. For that it must satisfy the following properties:

1. *non-negativeness:* $\forall\varphi, \psi : d(\varphi, \psi) \geq 0$
2. *reflexivity:* $\forall\varphi : d(\varphi, \varphi) = 0$
3. *symmetry:* $\forall\varphi, \psi : d(\varphi, \psi) = d(\psi, \varphi)$
4. *strictness:* $\forall\varphi, \psi : d(\varphi, \psi) = 0$ iff $\varphi \equiv \psi$
5. *triangle inequality:* $\forall\varphi, \psi, \theta : d(\varphi, \psi)+d(\psi, \theta) \geq d(\varphi, \theta)$

It is easy to prove that the measure we provided above is a metric, disregarding inequalities between agents.

A generic metric.

Now we can define a generic metric, which uses the context-dependent metric described above. A clausal form can be represented as a set of disjunctions, which allows us to use distance metrics on sets. There are several such metrics available in the literature, but one has been developed for defining distances between first-order logic objects. This metric, designed by Ramon and Bruynooghe [17] uses a matching between two clausal forms to calculate the distance. We use this metric, because it allows a direct syntactic comparison between different formulas. It is once again free to the designer to choose a different metric. All that is really required for the algorithm is for there to be a distance measure on sentences in $\mathcal{L}_{Trust}$. Clustering algorithms work better with metrics, because the triangle inequality can be used to prune the choices.

2.2.3 Clustering

Because we wish to learn generalizations which predict the receiving agent's trust evaluations, based on the gossip sent, we want to consider those rules where the receiving agent's trust evaluations are "near each other". That means we wish to cluster based on the heads of the rules. It is immediately obvious why an agglomerative hierarchical is the best fit:

- We want to work our way from small precise clusters to large clusters covering a broad spectrum of trust evaluations.
- We want to be able to stop the algorithm when we have found general rules covering all examples.

Bottom-up incremental clustering algorithms fit these criteria best, which leads us to the family of agglomerative clustering algorithms [21]. In this family, complete-link clustering creates more balanced clusters than single-link algorithms, yet has less overhead than average-link algorithms. All other clustering algorithms we explored require the computation of some form of centroid or medioid of the cluster, which speeds up the agglomeration process at the cost of calculating this centroid. Because it is hard to find a centroid for phrases in a first-order logic and we do not expect to have more than a few thousand data points, our choice fell on complete-link clustering. A drawback of complete-link clustering is that it deals badly with outliers. However, we are clustering on the agent's *own* trust evaluations. If there are outliers, they will not be in these evaluations, but rather the alignment rule itself will be an outlier. We will need to deal with the outliers in the learning of the body, but we should not encounter them when clustering.

Complete-link clustering algorithm.

To start, the complete-link agglomerative clustering algorithm places each element in a separate cluster. It then iteratively merges the two clusters that are nearest together, according to a distance measure between clusters. This distance measure is the maximum distance between two single elements in each cluster, using the distance measure as in Section 2.2.2. This process of agglomeration is continued until there is either only one cluster left, which contains all examples, or some stop criterion has been reached. This stop criterion is defined in line 1 of Algorithm 1. We stop the agglomeration when the distance between two clusters is greater than `s`.

A naive implementation of the complete-link agglomerative algorithm would take $O(n^3)$ time, where n is the number of elements to be clustered. The reason is fairly obvious: we start with each element in its own cluster. For each cluster we need to find the distance to each other cluster. This needs to be repeated any time a cluster is merged. Because we start with n clusters, this naive algorithm takes $O(n^3)$ time. This is fairly prohibitive, even for the relatively small datasets we expect to cluster. Luckily there are improvements. Because the distance measure is symmetric, it stands to reason we can skip some calculations. Furthermore, if we merge two clusters then the distance from that cluster to any other cluster is the maximum distance of either of those clusters to the other cluster. This allows us to reduce the algorithm to $O(n^2)$ time in a fairly straightforward manner: for each cluster we need to calculate the distance to each other cluster for which this hasn't been calculated. There are computational methods, some of which only work for metrics, for optimizing it even further. This makes the computation of clusters quite doable. Clustering is the process at point 3 in Figure 1.

2.2.4 Learning rules

For each distance `s` we will have a set of clusters. For each of these clusters we shall attempt to generalize the rules. This is point 4 of Figure 1. Although we clustered on clausal normal forms of only the heads of the rules, for this part we revert back to the full rule written in the original form. Within the cluster are two or more rules of the form: $\alpha_i[T_j] \leftarrow \beta_i[T_j], \psi_i$.

Learning the head.

All the α_i within a cluster are within distance `s` of each other. We therefore start with finding the "centre" of all α_i. Firstly we note that each α_i has a target agent T_j. We will immediately replace all these agents with a variable, because we do not wish to be dependent on the agent. In the future we may not wish to do this, but rather abstract to some subset of all the agents which fulfill a certain role, are within a subgraph of a social network or use other background information about the agents to refine the algorithm. For now, however, we do not distinguish between individual agents and assume trust is global and based only on the interactions. The "centre" of the cluster will be the least general generalization of the α_i under θ-subsumption. It is relatively easy to compute using an algorithm such as Aleph [20]. This is an inductive learning algorithm which uses the "learn from example" setting [8]. We wish to learn some phrase α^* in $\mathcal{L}_{Trust}$ such that if α^* holds then all α_i hold. As parameters for learning we therefore use the definitions of $\mathcal{L}_{Trust}$ and as the set of positive examples the α_i. Because we're learning the least general generalization (lgg), we can use only positive examples and assume everything that is not a positive example is a negative one. In actual fact this is not quite the case. For example in our example of $\mathcal{L}_{Trust}$ above, if we have the formulas $image(X, 5)$ and $image(X, 7)$ in the same cluster, we will wish to learn that the cluster holds for all phrases such that $image(X, Y) \wedge Y \in [5, 7]$, while this will not be the lgg considering only the given examples as positive: $image(X, 6)$ will necessarily be considered a negative example, leading to the generalization: $image(X, 6) \vee image(X, 7)$. Therefore depending on $\mathcal{L}_{Trust}$ we will want to define some background knowledge in the learner to rectify cases like these.

Learning the body.

The real work comes in when we wish to learn the body. We rewrite our rules with α^* in the head, such that we have a list of rules: $\alpha^*[X] \leftarrow \beta_i[X], \psi_i$, which count as positive examples of the concept α^*. All rules that fall outside the cluster count as negative examples for α^*. Thus giving us the basis required for applying an inductive learning algorithm. Furthermore we note that we have more information available than when we learn the generalization of the head, namely we have a list of situations β_i, ψ_i in which the example holds. This coincides with the "learning from interpretation" setting of ILP [8] and we can use TILDE [4] to learn these generalizations.

3. DISCUSSION AND FUTURE WORK

We are currently in the process of implementing the algorithm as described above. While we do not have any computational results yet, we will discuss our expectations. In [11] we discuss a preliminary proof of concept we implemented using Aleph to learn the rules. This small scenario taught us that the approach is viable, however using that implementation, the computational limitations were inhibitive to scaling the example up. For this reason we have taken great

caution in this approach to keep the computational complexity of each step into account. Firstly we must note that we are dealing with several NP-complete problems: finding the θ-subsumption of a set of clauses has been shown to be NP-complete, as has calculating the coverage of a given clause [8]. It was therefore very important to search for approaches which reduce this complexity. Firstly by clustering our examples and then considering them as positive and negative examples for some concept allows us to use established algorithms for learning. The clustering and learning of the head is a typical example of concept formation, which has an established body of research and is applied in various data mining problems. We feel confident that these approaches, tested in various datamining scenarios will tackle this initial problem well. The second part of the problem uses "learning from interpretations". While this is still a computationally hard problem, it is easier to learn than the approach using Aleph. TILDE has been tested on some very large data sets and performs efficiently. It is implemented with many optimizations in the ACE package [6].

We are currently implementing the overall system and testing the various components. This is the work for the immediate future. In addition it will be important to assess the quality of the aligned trust models, by comparing the performance of agents using the system to agents using the simpler model of Abdul-Rahman and Hailes [1] as well as agents not aligning at all. We will also extend the algorithm to allow for background knowledge, which can give the system extra information about the agents involved or background knowledge about the interactions and the environment. Furthermore, this model assumes agents always give truthful information. If this is not the case, the learning algorithm will need to be able to cope with "lies". The mathematical framework we have designed allows for all of this and the combination of different algorithms we use in practice looks promising.

Acknowledgements

This work is supported by the Generalitat de Catalunya under the grant *2009-SGR-1434*, the Agreement Technologies Project *CONSOLIDER CSD2007-0022, INGENIO 2010* and the LiquidPub Project *CIT5-028575-STP*. M. Schorlemmer is supported by a *Ramón y Cajal* research fellowship from Spain's Ministry of Science and Innovation, which is partially funded by the European Social Fund.

4. REFERENCES

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 6, 2000.

[2] M. Atencia and M. Schorlemmer. A formal model for situated semantic alignment. In *Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007)*, volume 6, pages 1270–1277, Honolulu, Hawaii, USA, 2007.

[3] J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, 1997.

[4] H. Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[5] H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In J. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann, 1998.

[6] H. Blockeel, L. Dehaspe, B. Demoen, G. Janssens, J. Ramon, and H. Vandecasteele. Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, 16:135–166, 2002.

[7] S. Casare and J. Sichman. Towards a functional ontology of reputation. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 505–511, New York, NY, USA, 2005. ACM.

[8] L. De Raedt. *Logical and Relational Learning*. Springer Verlag, 2008.

[9] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[10] A. Koster, J. Sabater-Mir, and M. Schorlemmer. Formalization of the trust and reputation alignment problem. Technical Report TR-2009-03, CSIC-IIIA, 2009. http://www2.iiia.csic.es/∼andrew/files/techreport.pdf.

[11] A. Koster, J. Sabater-Mir, and M. Schorlemmer. An interaction-oriented model of trust alignment. Technical Report TR-2009-05, CSIC-IIIA, 2009. http://www2.iiia.csic.es/∼andrew/files/techreport2.pdf.

[12] F. A. Lisi. Building rules on top of ontologies for the semantic web with inductive logic programming. *Theory and Practice of Logic Programming*, 8(3):271–300, 2008.

[13] P. Omidyar. Ebay. http://www.ebay.com, retrieved September 26, 2008, 1995.

[14] I. Pinyol and J. Sabater-Mir. Arguing about reputation. the lrep language. In *Proceedings of the 8th Annual International Workshop "Engineering Societies in the Agents World" (ESAW'07)*, volume 4995, pages 284–299. Springer LNCS, 2007.

[15] D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2:293–304, 1986.

[16] S. D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.

[17] J. Ramon and M. Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37:765–780, 2001.

[18] J. Sabater-Mir, M. Paolucci, and R. Conte. Repage: REPutation and imAGE among limited autonomous partners. *JASSS - Journal of Artificial Societies and Social Simulation*, 9(2), 2006.

[19] M. Schorlemmer, Y. Kalfoglou, and M. Atencia. A formal foundation for ontology-alignment interaction models. *International Journal on Semantic Web and Information Systems*, 3(2):50–68, 2007.

[20] A. Srinivasan. The aleph manual. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/, retrieved February 9, 2009.

[21] R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

Towards an inductive algorithm for learning trust alignment

Andrew Koster, Jordi Sabater-Mir and Marco Schorlemmer
IIIA-CSIC, Campus UAB, 08193 Bellaterra, Catalonia, Spain
{andrew, jsabater, marco}@iiia.csic.es

Motivation

- In open multi-agent systems, the agents have different models of trust and reputation.
- Agents benefit from receiving trust and reputation information from other agents.
- How can an agent handle trust and reputation information if the other agent's trust model is dissimilar to its own model?
- **We present a framework for aligning trust models. Based on this framework, we present an algorithm which uses clustering and inductive logic programming to form the alignment.**

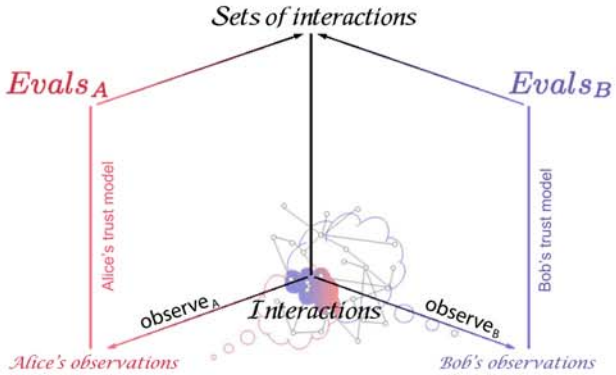


Figure 1. Basing trust on shared interactions

Interactions

- Observed interactions between agents are what all trust models base their evaluations on.
- Every trust evaluation is *based on* some set of interactions

Alignment

- By talking about the underlying interactions agents can *understand* each others' trust evaluations.
- Align trust models based on this type of gossip.

Algorithm

1. **INPUT:** Messages from the other agent, relating trust evaluations to underlying interactions
2. For each message, the agent finds his own trust evaluation of the same interactions
3. Cluster the messages based on the own trust evaluations
4. For each cluster, use an inductive learning algorithm to learn:
   a. a generalization of the own trust evaluations in the cluster (learning the head)
   b. a generalization of the other agent's trust evaluation and interactions (learning the body)
5. Interpret new messages from the other agent by finding the corresponding own trust evaluation in the alignment: this is what the other agent's gossip *means* to the receiving agent.

- Implemented using Prolog, Java and OWL.

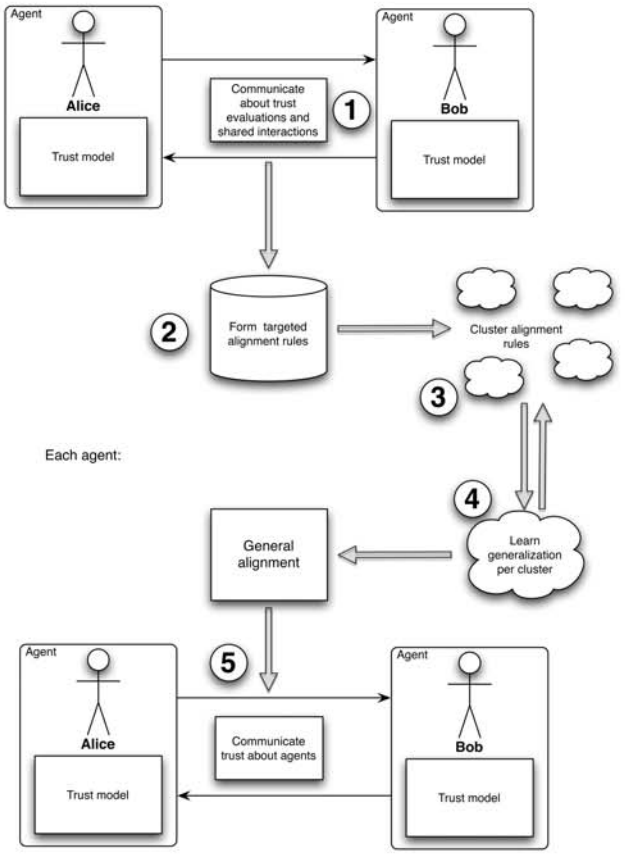


Figure 2. Schematic overview of the alignment process

Conclusions

- Trust alignment based on shared interactions
- We use clustering and inductive learning in an alignment algorithm

A Preliminary Proposal for Model Checking Command Dialogues

Rolando Medellin, Katie Atkinson and Peter McBurney
Department of Computer Science
University of Liverpool, UK
{medellin,katie,mcburney}@liverpool.ac.uk

ABSTRACT

Verification that agent communication protocols have desirable properties or do not have undesirable properties is an important issue in agent systems where agents intend to communicate using such protocols. In this paper we explore the use of model checkers to verify properties of agent communication protocols, with these properties expressed as formulae in temporal logic. We illustrate our approach using a recently-proposed protocol for agent dialogues over commands, a protocol that permits the agents to present questions, challenges and arguments for or against compliance with a command.

Keywords: agent communication, command dialogues, CDP, interaction protocols, model checking, NuSMV.

1. INTRODUCTION

The last two decades have seen considerable research on agent communication languages and agent interaction protocols. In the typical formulation, such as the generic agent language *FIPA ACL*, developed by FIPA [12], agent utterances are represented as two-layers: an inner layer of material directly related to the topic of the discussion, and an outer- (or wrapper-) layer comprising a speech act. An example of such a wrapper is the FIPA ACL locution, *inform(.)*, which allows the agent uttering it to tell another agent some statement which the first agent believes to be true. With such a structure, the same set of locutions may be used for dialogues on many different topics, on each occasion wrapping different content. Such generic languages create state-space explosion problems for intending agent dialogue participants, however, and so research attention has also been given to the design of agent interaction protocols. These may be viewed as agent communication languages designed for more specific purposes than is a generic language, in the same way, say, that a standard (human) ascending-price auction protocol is more specific than is a natural human language, such as English. For a recent review of research in agent interaction protocols, see [14].

As with any software, verification that agent interaction protocols have desired properties (or do not have undesired properties) is important. In this paper, we explore the use of model checking technologies for verification of properties of agent interaction protocols. In order for model checking approaches to be applied, we need to express the properties in a logical formalism, and we use a branching-time temporal logic for this. We illustrate the approach on an agent protocol designed for arguments over commands, called **CDP** [4]. This protocol was selected because it allows for argument between the participants, and because it is sufficiently complex that an automated approach to verification of protocol properties should prove of value to human software engineers.

The structure of the paper is as follows. The next section summarizes the command dialogue protocol, **CDP**. This is followed by a brief discussion of model checking and of **NuSMV**, the model checker we have used for this work. After that, we present the results of model-checking CDP, showing the graphical representation of the protocol, and the temporal logical representation of the properties we desire to verify. We finish with some concluding remarks and indications of areas for future work.

2. COMMAND DIALOGUE PROTOCOL

Commands are instructions issued by one agent to one or more other agents to execute some action (or not), or to bring about some state. Not all commands are issued legitimately, and even those which are legitimate may require subsequent elaboration or explanation before they can be executed. Thus, it is possible for agents to engage in an argumentative interaction over a command. As explained in [4], the rise of distributed computer systems and rival centres of control of the elements of such systems make commands and agent dialogues over commands increasingly common. Indeed, Hyper-Text Transfer Protocol (HTTP) [17] may be viewed as a protocol for two-agent dialogues over commands, although it is rather impoverished in terms of the commands enabled to be represented and the arguments permitted over them. In recent work [4], a formalism for the representation of commands and a dialogue protocol for argument over commands was presented, making use of an argument scheme for action proposals. In this formalism, the agent issuing the command was called *Commander*, while the intended recipient was called *Receiver*. The Command Dialogue Protocol (**CDP**) allowed Commander to issue a command to Receiver, and allowed Receiver to question, challenge, refuse or accept this command. If questioned or challenged, Commander could respond with additional information or arguments in support of the original command, and/or re-iterate it, modify it, or retract it.

Command dialogues are not explicitly mentioned in the Walton and Krabbe typology of human dialogues [21]. In a dialogue where a command has been issued, but not yet refused or accepted, the participants may enter into interactions which resemble those in the Walton and Krabbe typology, for example, Information-seeking, Inquiry, Persuasion Negotiation, Deliberation or Eristic dialogues. Not all command dialogues will have all such interactions, how-

Cite as: A Preliminary Proposal for Model Checking Command Dialogues, Rolando Medellin, Katie Atkinson and Peter McBurney, *European Agents System Summer School 2009)*

ever, and accordingly we believe it appropriate to consider Command dialogues as a type of dialogue distinct from those in the Walton and Krabbe list.

We now present an outline of the Command Dialogue Protocol (CDP) of [4], which uses an argument scheme for action proposals to specify commands. In an argument scheme, arguments are presented as general inference rules where, under a given set of premises, a conclusion can be presumptively drawn [20]. The argument scheme presented in CDP states that : *given the social context X, on the current circumstances R, action A should be performed to achieve new circumstances S, which will realise some goal G and will promote some value V*. This scheme allows commands to be justified through the promotion or demotion of some social value or interest, where a certain state or circumstance is achieved. Justification is based on current circumstances and elements of the social context. The CDP specifies the rules to formally represent imperatives in a multi-agent dialogue and provides means by which the participants may question, challenge, justify, accept or reject a command. Commands are represented as action proposals to the Receiver similar to the representation in [2]. In contrast with proposals or promises, commands require a set of preconditions in a regulatory environment to be executed validly. A command represents a presumptive argument attacked by a set of critical questions whose answers may defeat the initial argument or command. Critical questions represent questions the Receiver could pose to the Commander either to question or to challenge the command such that more evidence will be needed to justify it. Questions about the appropriateness, suitability, feasibility and normative rightness could be posed to the Commander.

Based on elements from the argumentation scheme the critical questions associated with the scheme can be grouped into four categories. The first category concerns questions about the selection of the action similar to the work presented in [3]. These questions are aimed at finding evidence regarding the current circumstances, the new circumstances to be achieved, the desired goal and the value to be promoted. In the second category, the questions posed to the Commander concern the choice of the agent or agents being tasked with execution of the command, and the expected consequences for the Receiver in performing the command. The third category of questions concerns the social roles of the agents involved, including issues such as the authority of the Commander to issue the command to the Receiver at this time, in this manner. The final category is questions to clarify the precise details of the task to be executed or state to be achieved. Questions in this category may consider issues regarding time, duration and specific instructions related to the performance and delegation of the action. Whether or not the Receiver agent questions or challenges the command, and whether or not the Commander responds to such questions or challenges with further arguments or evidence, the CDP protocol allows the Receiver to accept or refuse the command at any time. Likewise, the Commander may re-state or revise or retract the command at any time. The protocol allows such responses to be made by the agent concerned, regardless of the extent of evidence or justification presented in the dialogue up to that point.

The CDP syntax enables agents to interact using seven locutions: issue, accept, reject, question, challenge, justify and retract [4]. Locutions to issue or retract a command are inherent to the Commander and are comprised of options to state propositions defined in the initial argumentation scheme. As for the Receiver, the protocol defines locutions to respond to a command by accepting, refusing, questioning or challenging it. Expanding the 'question' locution CDP grows to 76 locutions[1] available to Receiver when questioning or challenging a command. Locutions to challenge and provide information can be used by both agents participating in the dialogue.

3. MODEL CHECKING

The verification of multi-agent systems showing that a system is correct with respect to stated requirements is an increasingly important issue [7]. Currently, the most successful approach to the verification of computer systems against formally expressed requirements is that of Model Checking [9]. Model checking is an automatic technique for verifying finite-state reactive systems, such as communication protocols. Given a model of a system M and a formula φ (representing a specification), model checking is the problem of verifying whether or not φ is true in M ($M \models \varphi$). In model checking, the design to be verified is modeled as a finite state machine, and the specification is formalized by writing temporal logic properties. An efficient search procedure is used to determine whether or not the state-transition graph satisfies the specifications [9]. The power of model checking is that it is exhaustive, no regions of the operating space are unexplored. Although model checking techniques have been most widely applied to the verification of hardware systems, they also have been used in the verification of software systems, protocols, [19], agent dialogues [10, 11] and multi-agent-systems [22, 7].

3.1 NuSMV

The possibility of verifying systems with realistic complexity changed dramatically in the late 1980s with the discovery of how to represent transition relations using ordered binary decision diagrams (BDD) [9]. A BDD is a data structure that is used to represent a Boolean function. The original model checking algorithm, with the new representation for transition relations, is called symbolic model checking. The symbolic model verifier (SMV) system is a tool for checking finite state systems against specifications in the temporal logic CTL (Computation Tree Logic) [15]. The input language of SMV is designed to allow the description of finite state systems and allows a rich class of temporal properties, including safety, fairness, liveness and deadlock freedom. NuSMV [2] is a reimplementation and extension of SMV and has been designed as an open architecture for model checking. This new version is aimed at reliable verification of industrially sized designs, for use as a back-end for other verification tools and as a research tool for formal verification techniques [8]. NuSMV2 uses a technique called Bounded Model Checking (BMC), which uses a propositional SAT solver rather than BDD manipulation techniques. SAT or propositional satisfiability is the problem of determining if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to TRUE [6].

[1] In case this number of locutions is thought prolix, note that CDP is intended for machine-to-machine communications; for comparison, the machine interaction protocol, Hypertext Transfer Protocol (HTTP), defines 41 standard status-code responses to a GET command, and allows for several hundred additional non-standard codes [17].

[2] NuSMV is a symbolic model checker developed as a joint project between the Formal Methods group in the Automated Reasoning System division at ITC- IRST, the Model Checking group at Carnegie Mellon University, the Mechanized Reasoning Group at University of Genova and the Mechanized Reasoning Group at University of Trento [8].

4. MODEL CHECKING CDP

Rather than propose a new model checking algorithmic approach to verify agent-communication protocols as in [5] our aim is to use existing model checkers to validate properties on a dialogue protocol.In [19] a Multi-Agent Dialogue Protocol (MAP) is used to define the communicative process between agents considering complex, concurrent and asynchronous patterns. To verify the MAP protocols Walton uses the SPIN Model checker [13] translating the MAP representation into the PROMELA language that SPIN uses as input language and then construct LTL formulas to validate against the PROMELA representation. This is probably the most similar approach to what we intended here. The main difference is that the MAP is a generic language to define communicative processes and we are focusing on a single protocol.

Agent dialogue protocols exhibit behaviour characterized in terms of execution traces which can be represented as branching trees. Trees can be represented in terms of a state-transition system and then translated into the NuSMV input language. The NuSMV model checker uses an exhaustive search procedure to determine whether or not a specification or property satisfies the modeled system. We aim to take the advantages of the NuSMV model checker to validate properties of the protocol. We focus on the CDP [4] and its desirable properties. The protocol is represented with the NuSMV input language, and properties we want to validate in the model are temporal CTL formulae. CTL formulae can be evaluated in transition systems [9] where the states are dialogue states and the transitions are the protocol valid locutions. In case the property is not valid, a counterexample is generated in the form of a sequence of states. In general, properties are classified to "safety" and "liveness" properties. Safety properties express what should not happen (equivalently, what should always happen), and liveness properties declare what should eventually happen.

Among the properties we want to verify for the protocol are:

1. Does any infinite loop or deadlock [3] situation exist in the protocol? If a deadlock or loop does exist, which dialogue sequence leads to that loop or deadlock?

2. Can we reach every outcome state? The motivation behind this property is to ensure the protocol has valid paths in all the possible combinations of the dialogue.

3. Is it possible to utter a particular locution in a particular state? This approach suggests a way to validate locutions in a dialogue.

4. Given a particular state (either an end-state or not), is there a valid dialogue sequence to reach that state?

5. Given a particular state, is there a dialogue sequence which avoids that state? An agent may wish to know if it can enter into a dialogue while avoiding particular states, e,g. concessions to other participants.

6. If the dialogue has reached a particular state, is a particular outcome state still reachable?. It could be the case, for example, that certain intermediate states in a dialogue preclude some outcome states.

[3]A deadlock is a situation wherein two or more competing actions are waiting for the other to finish, and thus neither ever does.

4.1 State-transition diagrams

The CDP can be modeled as a high level state-transition diagram where states represent dialogue states and transitions represent valid locutions. The diagrams presented in this section represent a command dialogue in an abstract way, leaving out explicit details about the content of messages, concurrency and the environment. Dialogue states are represented as circles and locutions as directed arrows labelled with valid locutions. Diagrams capture the protocol rules for agents engaged in a command dialogue specifying the path to reach any outcome state.

The dialogue states for the CDP are: *Initial, ReceiverCommanded, CommanderQuestioned, CommanderChallenged, ReceiverwithEvidence1, ReceiverwithEvidence2, CommandRetracted, CommandAccepted and CommandRefused* (we number the ReceiverwithEvidence status because we want to distinguish the state where evidence comes from a question from that where it comes from a challenge). The locutions for the CDP are: *command, question, challenge, provide, refuse, retract and accept*. We are excluding from the model for now the mental states of the agents and the environment state. We also have not yet considered the critical questions from [4] within our model.

The diagram in Figure 1 represents dialogue states numbered from s0 to s8 and the valid transitions for each state. The diagram shows how locutions are constrained depending upon the dialogue state, for example, we can only access the state where the command has been accepted (s7) from the states { s1, s5, s6 }, where the Receiver has been commanded or has been provided with evidence. From the moment an Agent *C* (the Commander) issues a command a range of valid locutions is available for each agent. Valid Commander locutions are represented with doted arrow-lines and Receiver locutions are represented with normal arrow-lines.The CDP assumes a strict-turn-taking only for the Receiver that needs to wait for the Commander's locution. Assuming the agent is rational and because of a change in the environment the commander could retract or reissue the command at any time. If we assume a strict-turn-taking for the commander arrows, 2a, 2d and 2e would be left out the diagram.

As we have discussed, the finite state transition diagram can be expressed as a tree. We do this transforming outcome-states in final nodes of a tree repeating states as necessary. The tree-diagram representation is presented in Figure 2.

With this second diagram we can visualize all the possible computation paths for the protocol. Instead of representing a state just once, we repeat the state to avoid locutions returning to the same state. Loops are now represented as infinite paths and the paths to reach an outcome state are clearer. Since we are using a branching time temporal logic (CTL) this model is useful to construct temporal formulae to validate. To represent how the dialogue advances we associate a propositional value with each state and specify where the expression is true in each state. For the initial state, for example, we assign propositional variable 'a' and make it true only in that state. In this way we can construct temporal formulae with propositional variables representing each state. We also assign a variable related to the 'turn' of each agent in the dialogue, represented by 'tc' in the case where the Commander is allowed to issue a locution, and 'tr' in the case of the Receiver. These variables allow us to construct temporal formulae related to the turn of an agent to issue a locution.

The properties we want to validate for the protocol could be

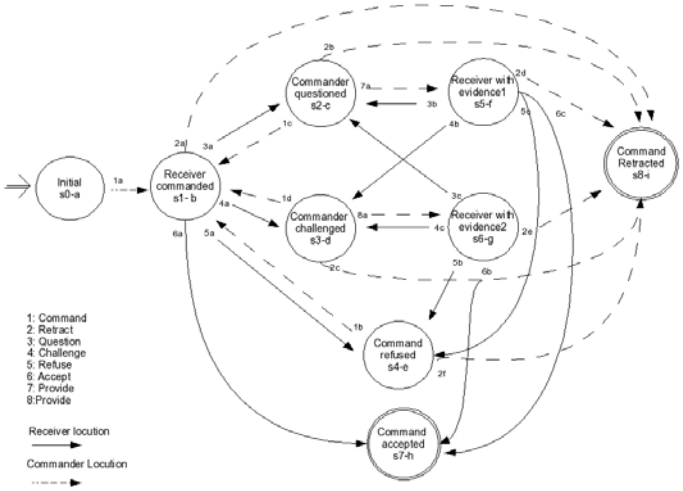


Figure 1: State-Transition Diagram for CDP

rephrased as temporal properties related to the tree-model in Figure 2. In Table 1 the properties presented earlier are now rephrased and a temporal formula is associated for each one. CTL is built from path quantifiers and temporal operators. There are two path quantifiers, A and E, where A means "for every path" and E means "there exists a path" in the tree. A path is an infinite sequence of states such that each state and its successor are related by the transition relation. CTL has four temporal operators presented as follows: $\bigcirc\phi$ meaning "ϕ holds at the next time step" (where ϕ is a propositional formula), $\diamond\phi$, "ϕ holds at some time step in the future", $\Box\phi$, "ϕ holds at every time step in the future" and $\phi U\psi$, "ϕ holds until ψ holds" [6].

Tree-oriented property	Temporal property
1. Do infinite paths exist in the tree-diagram?	$A\diamond(A\bigcirc h \vee A\bigcirc g)$
1a. Which is the path?	Counterexample from 1
2. Is there a valid path to reach every outcome state?	$E\diamond h$
3. Is a transition valid from a specific node?	$A\Box(c \rightarrow E\bigcirc d)$
4. Given a node, is there a path which leads to that node?	$E\diamond g$
5. Given a node, is there a path which avoids that node?	$A\Box(\neg c \rightarrow E\diamond i)$
6. If a command has been issued and questioned can the dialogue still reach a state where the command is accepted?	$A\Box(c \rightarrow E\diamond i)$

Table 1. Properties and Temporal formulae.

4.2 NUSMV implementation

We use NuSMV for model checking because the input language allows us to represent the dialogue as a finite-state diagram and we can verify temporal properties in it. If the property specified does not satisfy the NuSMV model, the model checker offers a counterexample specifying the path where the formula fails to be true. Input to NuSMV is via a file which describes the state transition system in terms of variables and value assignments. Dialogue states are represented with a variable *state* that can obtain the value of any of the 9 states defined in the CDP, plus an error state to specify non-valid moves. Transitions are represented using the NuSMV *case* expression. For each state we define a set of possible next states that represent the valid transition relations. Expressions are evaluated sequentially and the first one that is true determines the resulting value.

We use the keyword *SPEC* in NuSMV to specify CTL properties to be checked. For example, to express if it is true that at some path there is a case where the command is retracted, we use the CTL formula $E\diamond i$. Variable i represents the retracted state. In the NuSMV input language it is represented as *SPEC EF (i)*. As for the variables related to the turn-taking we can specify a property to check if there is an option to issue a locution at every state (except on final states) $A\Box((tc|tr)|(\neg tc\&\neg tr\&h)|(\neg tc\&\neg tr\&i))$.

4.3 Preliminary Results

All the properties presented were translated to the NuSMV language and validated against the model. For the first property we are trying to check if there exists any infinite path on the model. The idea is to construct a formula that represents that eventually on all paths the final nodes could be reached. The property constructed in the NuSMV input language is $SPECAG(AXh \mid AXi)$. The formula is false for the model since the protocol allows the participants to engage in an infinite loop in several situations. Another way to construct this property without making reference to a particular state is $AGEX\top$ which states that there exist a path such that every node on that path still has some immediate successor.

Property number two ("Is there a valid path to reach every outcome state?") is True. The protocol allows to reach an outcome state in all paths. Property number three ("Is a transition valid from a specific state?") depends on the state we are choosing. In the example we are validating if issuing a locution from state "c" (commander questioned) is valid to a state "d" (commander challenged), in this case is False. This seems obvious if we analyse the diagram, but human visual inspection will not scale to larger and more complex protocols, nor operate at runtime. Property four ("Given a

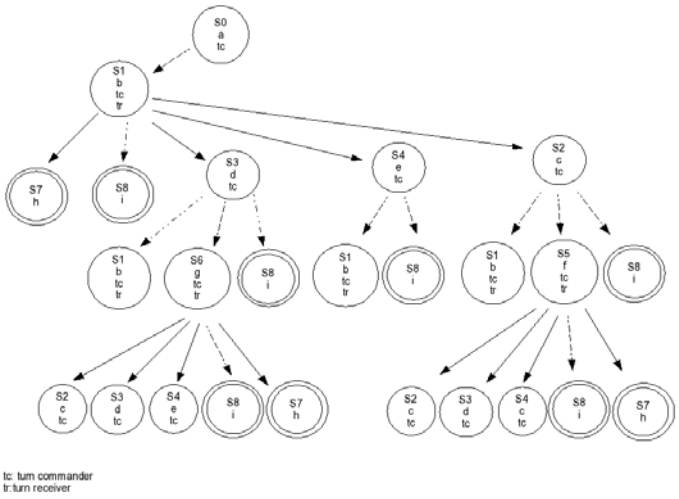


Figure 2: Tree diagram for CDP

node is there a path which leads to that node?") tries to confirm if a valid path exists to reach a specific state. In the example the formula is true for state "g". Property number five ("Given a state is there a path which avoids that state?") is true for state "i" avoiding state "c". Finally property six ("If a command has been issued and questioned can the dialogue still reach a state where the command is accepted?") is True for the specified states.

Properties are closely related to the CDP protocol and the states that emerge from it; a more generic set of formulae may be desirable to develop. Nevertheless, we need to take into account that for dynamic verification, on-the-fly models need to be constructed and validated.

5. CONCLUSIONS

In this paper we have explored the possibility of using model-checking methods to automatically verify that a complex agent interaction protocol using argumentation has desired properties (or does not have undesired properties). Our key contribution has been to show by example that this is possible, using the model checker **NuSMV** to verify specific properties of the command dialogue protocol, **CDP**. Because this protocol supports multi-agent argumentation, it is reasonably complex and thus the value of automated verification approaches is likely to be considerable. Such verification could take place well prior to implementation, for example, as part of the human-led protocol design process. Or it could take place at run-time just prior to invocation of the protocol, if agents were enabled to select and verify protocols on-the-fly at the moment before they enter into dialogue, as in [16]. For agents having dynamic goals, on-the-fly verification of protocols will be important to ensure that protocols they use to engage in dialogue are able to achieve states currently desired or avoid states currently not desired.

In future work we intend to extend our model to account for the critical questions associated with the argument scheme as given in **CDP** since we have not considered them here. Our approach would be much more complex if we add rules and states considering the critical questions where more states and variables need to be added to the model. We also hope to investigate how our model can be extended to handle different types of dialogue in addition to CDP. For example, in [2] a protocol is given for persuasion dialogues based on a similar argument scheme that is used for CDP, so this would be a good candidate protocol to model next. Additionally, some recent work [1] has looked at how the argument scheme for practical reasoning discussed here can be formalised in terms of action-state semantics [18]. It would be also interesting to see how our approach to model checking dialogues could be applied to this representation.

5.1 Acknowledgements

Rolando Medellin is grateful for financial assistance from CONACYT of Mexico. We thank Clare Dixon and the anonymous reviewers for their comments. A later version of this work was presented at the AAAI Fall Symposium 2009 in the Workshop "The Uses of Computational Argumentation".

6. REFERENCES

[1] K. Atkinson and T. Bench-Capon. Action-state semantics for practical reasoning. Proceedings of 2009 Fall Symposium on The Uses of Computational Argumentation. Arlington, VA, USA, 2009.

[2] K. Atkinson, T. Bench-Capon, and P. McBurney. A dialogue game protocol for multi-agent argument over proposals for action. *Autonomous Agents and Multi-Agent Systems*, 11(2):153–171, 2004.

[3] K. Atkinson, T. Bench-Capon, and P. McBurney. Computational representation of practical argument. *Synthese*, 152(2):157–206, 2006.

[4] K. Atkinson, R. Girle, P. McBurney, and S. Parsons. Command Dialogues. In I. Rahwan and P. Moraitis, editors, *Argumentation in Multi-Agent Systems*, Fifth International Workshop, pages 93–106, Berlin, Heidelberg, 2009. Springer-Verlag.

[5] J. Bentahar, B. Moulin, and J.-J. Ch. Meyer. A new model checking approach for verifying agent communication protocols. *Canadian Conference on Electrical and Computer Engineering, CCECE '06*, pages 1586–1590, May 2006.

[6] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without bdds. In *TACAS '99: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 193–207, London, UK, 1999. Springer-Verlag.

[7] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying multi-agent programs by model checking. *Autonomous Agents and Multi-Agent Systems*, 12(2):239–256, 2006.

[8] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2:2000, 2000.

[9] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. Springer, 1999.

[10] U. Endriss. Temporal logics for representing agent communication protocols. *Agent Communication II: International Workshops on Agent Communication*, 3859/2006:15–29, 2006.

[11] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Logic-based agent communication protocols. In *Advances in Agent Communication Languages*, volume 2922, pages 91–107. Springer-Verlag, 2004.

[12] FIPA. Communicative Act Library Specification. Standard SC00037J, Foundation for Intelligent Physical Agents, 3 December 2002.

[13] G. Holzmann. *The SPIN Model Checker. Primer and Reference Manual*. Addison-Wesley, 2004.

[14] P. McBurney and S. Parsons. Dialogue games for agent argumentation. In I. Rahwan and G. Simari, editors, *Argumentation in Artificial Intelligence*, chapter 13, pages 261–280. Springer, Berlin, Germany, 2009.

[15] K. L. McMillan. The SMV system. *Cadence Berkeley Labs*, 1999.

[16] T. Miller and P. McBurney. Annotation and matching of first-class agent interaction protocols. In L. Padgham, D. Parkes, J. P. Mueller, and S. Parsons, editors, *Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, New York, NY, USA, 2008. ACM Press.

[17] Network Working Group. Hypertext Transfer Protocol — HTTP/1.1. Technical Report RFC 2616, Internet Engineering Task Force, June 1999.

[18] C. Reed and T. J. Norman. A formal characterisation of Hamblin's action-state semantics. *Journal of Philosophical Logic*, 36:415–448, 2007.

[19] C. Walton. Model checking agent dialogues. In *Declarative Agent Languages and Technologies II*, Lecture Notes in Computer Science, pages 132–147. Springer, July 2004.

[20] D. N. Walton. *Argumentation Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 1996.

[21] D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. SUNY Series in Logic and Language. State University of New York Press, Albany, NY, USA, 1995.

[22] M. Wooldridge, M. Fisher, M.-P. Huget, and S. Parsons. Model checking multi-agent systems with MABLE. In *AAMAS '02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 952–959, New York, NY, USA, 2002. ACM.

Model Checking Command Dialogues

Department of Computer Science
Agent ART Group

Angel Rolando Medellin

Objective

Verify that agent communication protocols have desirable properties or do not have undesirable properties is an important issue for agents intending to communicate with such protocols. In this work we explore the use of a model checker to verify properties of an agent communication protocol, with these properties expressed as formulae in temporal logic. We illustrate our approach using a protocol dialogue over commands **CDP** and the model checker **NuSMV** with properties expressed as **CTL** formulae.

1.- Command Dialogue Protocol (CDP)

Formalism for the **representation of commands** and a **dialogue protocol** for argument over commands , making use of an **argument scheme** for action proposals.

Argumentation scheme for action proposals:

- Given the social context X
- In current circumstances R
- **Action X should be performed** <— **Command**
- To achieve new circumstances S
- Which will realise some goal G
- Which will promote some value V

2.- NuSMV

Tool for checking finite state systems against specifications in the temporal logic CTL. If the property specified is not true in the NuSMV model, the model checker offers a counterexample specifying the path where the formula fails to be true.

CTL (Computation Tree Logic). - Branching-time logic in which model of time is a tree-like structure with different paths in the future, anyone of which might be an actual path that is realised.

Quantifiers over paths

A φ - : φ has to hold on all paths from the current state.

E φ - : there exists at least one path starting from the current state where φ holds.

Path-specific quantifiers

○ φ - : φ has to hold at the next state

□ φ - : φ has to hold on the entire subsequent path.

◊ φ - : φ eventually has to hold .

3.- State Transition Diagram

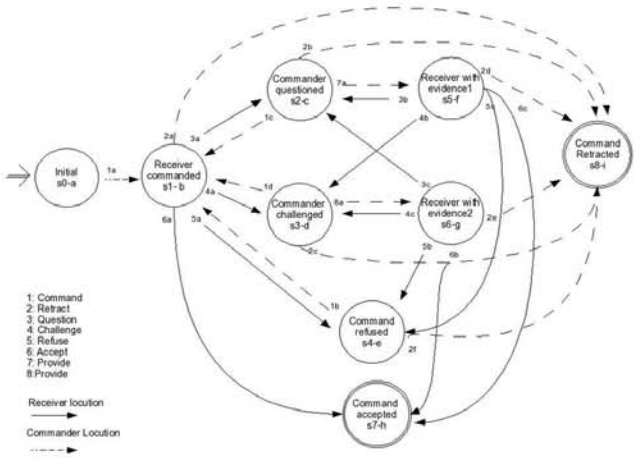
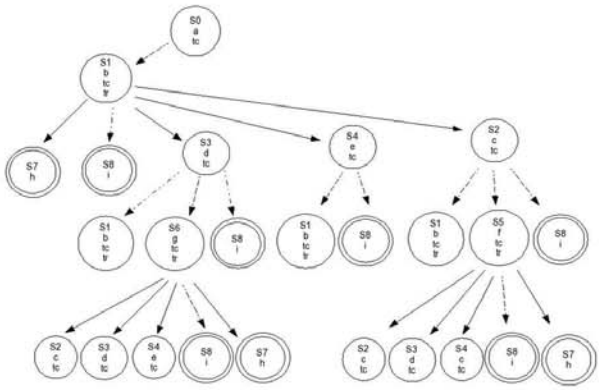


Diagram can be represented as a Tree specifying loops as infinite paths. CTL formulas can be validated against this model.



4.- Properties

– Does infinite paths exist?	A◊ (A ○h ∨ A ○i) or E□ E○ T
– Does a valid path exist to reach every final node?	E◊ h
– Is a transition valid from a specific node?	A□ (c → E ○d)
– Given a node, is there a path which avoids that node?	A□ (¬c → E ○ h)
– If a command has been questioned. Is the accepted node still reachable?	A□ (c → E ◊ i)

5.- Results

- Properties were validated against the model.
- Limitations on the approach. (Model Checker, Rule Specification)

6.- Future Work

-Use the SPIN model checker (Explicit Model checker for asynchronous systems).

- Critical questions approach.
- Add new variables to the model- environment variables, variables related to reputation and trust.
- Consider LTL , CTL*, ATL.

http://www.csc.liv.ac.uk/~medellin/.

September 2009

Norm Convergence in Populations of Dynamically Interacting Agents

Declan Mungovan
Department Of Information Technology
National University of Ireland, Galway
Galway, Ireland
declan.mungovan@nuigalway.ie

Enda Howley
Department Of Information Technology
National University of Ireland, Galway
Galway, Ireland
enda.howley@nuigalway.ie

Jim Duggan
Department Of Information Technology
National University of Ireland, Galway
Galway, Ireland
james.duggan@nuigalway.ie

ABSTRACT

Agent Based Modelling (ABM) is a methodology used to study the behaviour of norms in complex systems. Agent based simulations are capable of generating populations of heterogeneous, self-interested agents that interact with one another. Emergent norm behaviour in the system may then be understood as a result of these individual interactions. Agents observe the behaviour of their group and update their belief based on those of others. Social networks have been shown to play an important role in norm convergence. In this model[1] agents interact on a small world network with members of their own social group plus a second random network that is composed of a subset of the remaining population. Random interactions are based on a weighted selection algorithm that uses an individual's path distance on the network. This means that friends-of-friends are more likely to randomly interact with one another than agents with a higher degree of separation. Using this method we investigate the effect that random interactions have on the dissemination of social norms when agents are primarily influenced by their social network. We discover that increasing the frequency and quality of random interactions results in an increase in the rate of norm convergence.

1. INTRODUCTION

Normative behaviour, or norms, can be defined as a set of conventions or behavioural expectations that people in a population abide by. They help maintain one's popularity within a group and ensure that individuals can productively cooperate with one another. Ignoring social norms, or conventions, can lead to negative repercussions for individuals including being ostracised from a group. Social norms present a balance between individual freedom on the one hand and the goals of the society on the other [18]. Conventions play an important role in creating a framework in which agents can structure their actions to help reduce social friction. There are two types of social norm conventions: top-down and bottom up. Top-down norms represent laws that are enforced on the population [7]. Bottom up conventions, such as shaking hands when introducing oneself, represents emergent behaviour from within the group. In this scenario agents, acting in their own self interest, choose which action to take based upon their interactions with others in the population. This is the type of social norm conversion that we investigate in this paper. Agents use locally available information to determine their selection of social norms.

ABM specifies a population as a collection of interacting, self-interested agents, where macroscopic behaviour is explained by the interaction of different individuals over time. Using this approach we can explain system-wide characteristics as emergent behaviour emanating from individual interactions of the agents.

The proposal we present in this paper is that an agent is unlikely to change its immediate social network of acquaintances very much. An individual in the population will, generally speaking, have the same wife, boss, friend etc. from one day to the next. They will, however, have a series of random ad hoc interactions with members of the general public. We recognise, however, that not all random interactions are the same. One is more likely to randomly meet one's next door neighbours best friend than a complete stranger. To account for this we bias random interactions based on the social distance that separates agents in the network. We then run a number of experiments that test the importance of the frequency and quality of these random interactions. We aim to discover at what point random interactions will influence the emergence of a global conventions. Specifically, we we aim to:

1. Design an algorithm that selects a random individual based on their social distance in the network.
2. Test the random interaction conditions that are most important in determining the emergence of a global convention on a population of agents.

The rest of the paper is structured as follows; Section 2 presents an introduction to previous work in the area of norm convergence and social networks. Section 3 gives a description of the formal model used to define the agent based simulator and an explanation of how the simulator was designed and implemented. Section 4 presents the experimen-

[1]The support of Science Foundation Ireland is gratefully acknowledged.

Cite as: Norm Convergence in Populations of Dynamically Interacting Agents, Declan Mungovan, Enda Howley and Jim Duggan, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

tal results. Finally, in Section 5 we outline our conclusions and possible future work.

2. RELATED RESEARCH

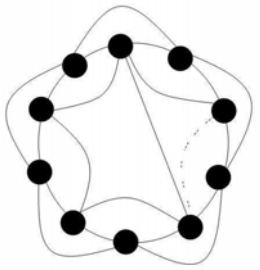
2.1 Norm Emergence

Agent-based Modelling (ABM) has been used in recent years as a method of studying social norms [13] [4] [2]. Savarimuthu et al. [14] generated a network of agents whose topology changes dynamically. Agents initially randomly collide on a 2D grid and then proceed to form social networks. Villatoro et al. [17] investigate the effect that network topology has on the emergence of norms. They simulate agents interacting on a lattice and scale free network. They found that highly clustered networks resulted in norm convergence in a shorter time. Conte et al. [2] and Walker et al. [18] describe a framework on integrating concepts of Multi Agent Systems with normative behaviour and how both disciplines interact. A considerable amount of the literature has studied the effects of norm emergence in populations that are fully connected and interact in a random fashion [15] [18]. The network that agents interact on, however, has been shown to play a significant role on the dynamics of diffusion [10][17][14]. Most of this work has dealt with static networks that are generated at initialisation time and do not change for the duration of the simulation. There have, however, been some attempts to frame research within the bounds of dynamic networks [14].

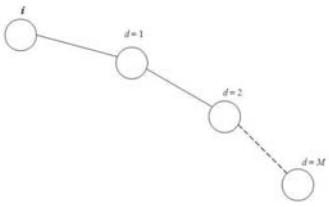
2.2 Small World Social Networks

The idea of *Small World Networks* first gained popularity with Stanley Milgram's small-world study of large and sparse networks [12]. Watts et al. later describe these networks as being formed by rewiring the edges of regular lattices with probability p_w [20] . Small World Networks are highly clustered, yet have length scaling properties equivalent to the expectations of randomly assembled graphs [19]. Notice in Fig. 1(a) that the link with the dashed line has been re-wired to another part of the network. This creates an instant shortcut to distant nodes. Small world graphs span the gap between ordered lattices and random graphs. Note that when $p_w = 1$, then all links are randomly assigned and the network becomes a random network. Lee et al. [11] investigated the effect that changing the value of p_w has on the emergence of a winner take all outcome in product adoption. They discovered that as p_w is increased the chance of a winner take all outcome becomes more likely. This is because as the value of p_w gets closer to one the network starts to become more like a random network. This prevents localized cliques of products from existing. An analysis of a number of real world human networks [21] [16] [3] [1] have shown that they form small world networks. The Maximum Path Length (MPL), Fig. 1(b), is the maximum number of steps required to get to the furthest node, or nodes, on the network. Dijkstra's algorithm [22] uses a breadth first search to traverse the network and discover the shortest path to each agent.

Fenner et al. [5] describe a stochastic model for a social network. Individuals may join the network, existing actors may become inactive and, at a later stage, reactivate themselves. The model captures the dynamic nature of the network as it evolves over time. Actors attain new relations according to a preferential attachment rule that weights different agents according to their degree[2].



(a) Small World Network



(b) Path Distance from Node

Figure 1:

2.3 Randomness in Agent Based Modeling

Agent based modelling (ABM) has a number of advantages over classical game theory approaches. Firstly, ABMs are capable of implementing Monte Carlo[3] type stochastic iterations of a complex system. Izquierdo et al. [8] highlight the fact that any computer model is in fact, due to its very nature, deterministic. However, we can use pseudo-random number generators to simulate random variables within the model and generate an artificial Monte Carlo generator. The pseudo-random number generator is an algorithm that takes in a random input seed value and generates a result that approximates a random number. This property allows us to simulate randomness that is present in real world systems. In this fashion, an agent based simulation that provides the same input variables but implements a level of randomness can produce, sometimes, significantly different outcomes. A key challenge of analysing an ABM is in identifying an appropriate set of state variables.

We can see from the section that norm emergence is heavily influenced by the individuals that an agent meets in the network. Real world interactions are dynamic, this is a feature we aim to capture in this paper.

3. MODEL DESIGN

The following section describes formally the decision making rules agents use to choose random interactions and the actions they take based on their observations. Agents receive a utility from observing the norms that have been adopted by the other individuals it encounters. Agents interact with

[2]The degree of an agent is the number of acquaintances it has in its social network.

[3]A Monte Carlo algorithm relies on repeated random sampling to compute their results.

s members of their social network and r randomly selected agents. Initially nodes are set to having adopted either social convention j or k. Nodes interact with a period drawn randomly from an exponential distribution with mean duration $\epsilon_i = 3$. This models the fact that all agents don't update their norm selection simultaneously. An agent, i, will chose to adopt norm j if the utility it observes from adopting this norm is greater than the utility it would receive from adopting convention k as defined in 1.

$$u^j_{i,t} > u^k_{i,t} \tag{1}$$

The utility that agents receive from each norm is defined in 2. This is divided into the utility communicated from its direct neighbours, $D^j_{i(t-1)}$, plus the utility it receives from the random interactions it makes, $R^j_{i(t-1)}$.

$$u^j_{i,t} = \alpha D^j_{i(t-1)} + \beta R^j_{i(t-1)} \tag{2}$$

Where α is the weighting placed on an agent interacting with the members of its own social network and β is the weighting of interactions taking place with random members of the agents network. The higher the β value the more importance agents place on random interactions. The direct network effects are defined in 3 where n is the total number of nodes on the network and $\theta^j_{h(t-1)} = 1$ if agent h has adopted social convention j.

$$D^j_{i(t-1)} = \sum_{h=1}^{n} \mu_{ih}\theta^j_{h(t-1)} \qquad \mu_{ih} \begin{cases} 1 \text{ if } i \text{ is an} \\ \text{acquaintance of } h \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

Similarly we define the random network effects in 4 where n is the total number of nodes on the network and $\omega^j_{h(t-1)} = 1$ if agent h has adopted social convention j.

$$R^j_{i(t-1)} = \sum_{h=1}^{n} \phi_{ih}\omega^j_{h(t-1)} \qquad \phi_{ih} \begin{cases} 1 \text{ if } i \text{ has a random} \\ \text{interaction with } h \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

Agents interact with random members of the population using a Weighted Random Interaction (WRI) algorithm based on their distance from others on the network. We use a modified version of Zipf's law 5 to calculate a nodes weight. The probability of agent i, with Maximum Path Distance (MPD) of M, randomly interacting with agent h having a path distance of d from i is equal to:

$$p_{ih}(d) = \frac{\frac{1}{(d-1)^\lambda}}{\sum_{m=1}^{M-1}\left(\frac{1}{m^\lambda}\right)} \qquad d \geq 2 \tag{5}$$

Where λ is the exponent that characterises the distribution. For the experiment carried out in this paper we set $\lambda = 1$. Note the condition that d ≥ 2 as a node is assumed to interact with members of its social network ($d = 1$). It can be seen from 5 [6] that the distribution is normalised and the frequencies sum to 1 as expressed in 6.

$$\sum_{d=2}^{M-1} p_{ih}(d) = 1 \tag{6}$$

The graph shown in Fig. 2 shows the distance probability distribution of three different nodes with Maximum Path Distance (MPD) ranging from 5 to 10. We can see from the diagram that agents with a lower path distance are more likely to interact than ones with a higher path distance.

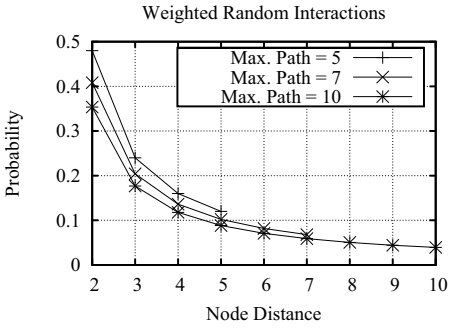


Figure 2: Weighted Random Interactions

4. RESULTS

In all the simulations conducted, a population of 1000 agents in a small world configuration with average degree of 10 was generated. Initially all nodes are randomly given one of two norms. All the results shown are the average of 1000 different simulations. A new Small World network was generated for each simulation. Initially each agent on the network maps its social distance from every other agent. We used Dijkstra's algorithm [9] to calculate the MPL for each node. Every time an agent interacts it generates a new set of ad hoc random interactions based on the WRI algorithm described above. We conducted four different experiments using the model described in Section 3. In Experiment 1 we simply vary the rewiring probability of the network and investigate the effect of norm convergence. Experiment 2 introduces random interactions taking place over the core small world network. We study the effect on norm convergence by varying both the value for the strength of random interactions, β, and the number of random interactions that an agent has, r. Experiment 3 and Experiment 4 both explore the level of norm convergence over time.

4.1 Experiment 1: Varying Rewiring Probability

Fig. 3 shows the effect of norm convergence when the rewiring probability is changed. We observe that the probability of all agents converging on a common norm is increased when the value of p_w is increased. This is similar to the finding of Lee et al. [11] in the domain of product adoption mentioned earlier. While increasing the value of p_w results in an increase in norm convergence, it reduces the level of clustering in the network. Real world human networks have high levels of clustering so this means that increasing p_w is unrealistic. In the next three experiments we maintain a core, highly clustered, small world network but introduce ad hoc random interactions that the agents have with others in the population.

4.2 Experiment 2: Adding Random Interactions

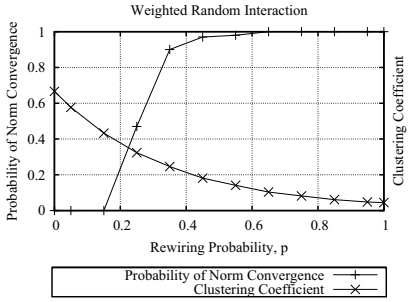


Figure 3: Probability of Norm Convergence varying *p*

In this experiment a small world network is created with a rewiring probability of $p_w = 0.05$ and $\alpha = 1$. As we have seen in Fig. 3 norm convergence will not happen when p_w is at this level. In Fig. 4 we observe the effect of norm convergence when agents are allowed to interact with randomly selected individuals on the network who are not part of their social network. We vary both the value agents place on random interactions, β, and the number of random interactions that they have, r. The number of random interactions starts at 0 and is increased by a value of 2 until it reaches 16. The strength of random interactions, β, is increased from 0 to 1. When β equals 1 then agents place the same strength on interactions with random members of the populations as on their own social network. We can see from Fig. 4 that norm

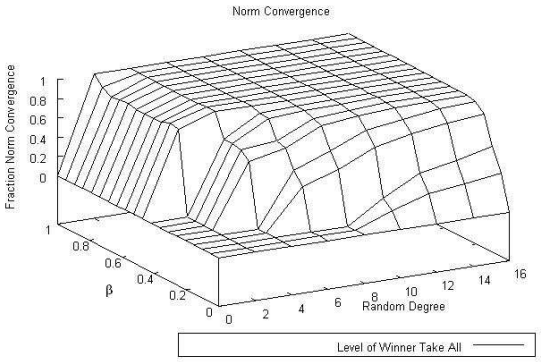


Figure 4: Probability of Norm Convergence

convergence fails to occur when both the strength and quantity of random interactions is too low. Indeed, when agents are having up to 8 random interactions but those occurrences only carry a weight of 0.1 of random interactions then norm emergence will not occur. Increasing the number of random interaction or increasing the strength of these interactions results in norm emergence. From Fig. 4 we can see that if $\beta > 0.5$ and the number of random interactions $r > 6$ then norm convergence is guaranteed. If agents have the same number of random interactions as members of their social network, or $s = r = 10$, then β only needs to be 0.2 to almost guarantee norm convergence. This experiment shows that random interactions with members of a nodes social network plays an important role in norm convergence.

4.3 Experiment 3: Varying Random Interactions

In Fig. 5 we set the level of $\beta = 0.1$ and increment the number of random interactions. We can see that there is no norm convergence when the time is less than approximately 50 or the number of random interactions is less than 10. We can see that once the population overcomes this threshold level of random interactions then there is a steady increase in the number of simulations resulting in convergence. The graph appears to be a series of terraces because the random degree is increased in steps of 2.

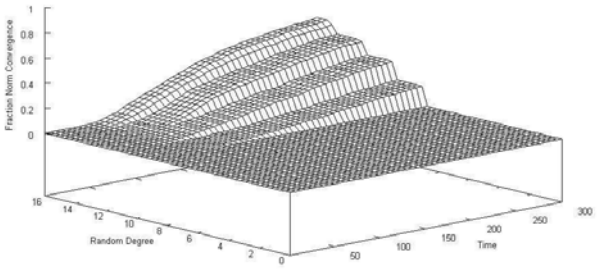


Figure 5: Level of Norm Convergence over Time Varying Random Degree

4.4 Experiment 4: Varying the level of β

In this final experiment we set the level of random interactions to 4, $\alpha = 1$ and incremented the level of β. We can see from Fig. 6 that there are several jumps in norm convergence when we increase the level of β. Specifically, when $\beta \leq 0.2$ then none of the simulations converge to a common norm. When $\beta \geq 0.5$ then all the simulations converge to a common norm. We can also see that that when $0.2 \leq \beta \leq 0.3$ then some norm emergence does occur but at a much slower rate.

5. CONCLUSIONS

The aim of this paper was to construct a more realistic network of agent interactions that might help explain the emergence of norms in society. We defined an algorithm that uses a nodes social distance on the network to calculate its chance of interacting with a random member of the population. We have demonstrated the importance that random agents can have on the emergence of social norms. Particularly our research demonstrates how norms can rapidly take hold in environments were agents interact heavily with random individuals outside their social network. This would perhaps be analogous to people in a large city interacting with lots of random individuals versus residents of a rural area that mostly meet members of their own social network. Our results from Section 4 highlight that norm convergence is dependant on both the frequency and quality agents place on random interactions.

6. REFERENCES

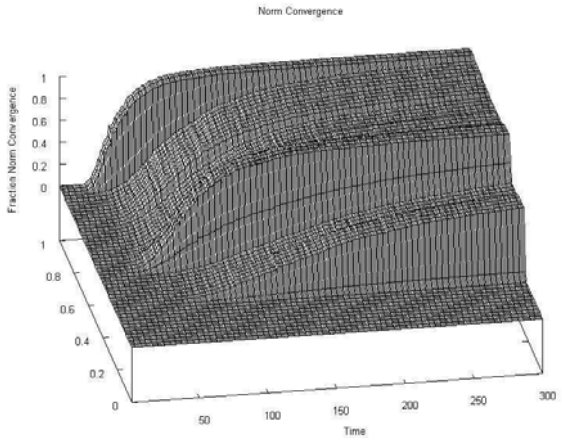


Figure 6: Level of Norm Convergence over Time Varying Beta

[1] Joel A. C. Baum, Andrew V. Shipilov, and Tim J. Rowley. Where do small worlds come from? *Ind Corp Change*, 12(4):697–725, 2003.

[2] Rosaria Conte, Rino Falcone, and Giovanni Sartor. Introduction: Agents and norms: How to fill the gap? *Artificial Intelligence and Law*, 7(1):1–15, March 1999.

[3] Gerald F. Davis, Mina Yoo, and Wayne E. Baker. The Small World of the American Corporate Elite, 1982-2001. *Strategic Organization*, 1(3):301–326, 2003.

[4] Michael Luck Fabiola López y López and Mark dŠInverno. A normative framework for agent-based systems. *Computational & Mathematical Organization Theory*, 12(2):227–250, 2006.

[5] Trevor Fenner, Mark Levene, George Loizou, and George Roussos. A stochastic evolutionary growth model for social networks. *Comput. Netw.*, 51(16):4586–4595, 2007.

[6] Xavier Gabaix. Zipf's law for cities: An explanation*. *Quarterly Journal of Economics*, 114(3):739–767, 1999.

[7] Brent Goldfarb and Magnus Henrekson. Bottom-up versus top-down policies towards the commercialization of university intellectual property. *Research Policy*, 32(4):639 – 658, 2003.

[8] Luis R. Izquierdo, Segismundo S. Izquierdo, José Manuel Galán, and José Ignacio Santos. Techniques to understand computer simulations: Markov chain analysis. *Journal of Artificial Societies and Social Simulation*, 12(1):6, 2009.

[9] Donald B. Johnson. A note on dijkstra's shortest path algorithm. *J. ACM*, 20(3):385–388, 1973.

[10] James Kittock. Emergent conventions and the structure of multi–agent systems. In *Lectures in Complex systems: the proceedings of the 1993 Complex systems summer school, Santa Fe Institute Studies in the Sciences of Complexity Lecture Volume VI, Santa Fe Institute*, pages 507–521. Addison-Wesley, 1995.

[11] Eocman Lee, Jeho Lee, and Jongseok Lee. Reconsideration of the Winner-Take-All Hypothesis: Complex Networks and Local Bias. *MANAGEMENT SCIENCE*, 52(12):1838–1848, 2006.

[12] Stanley Milgram. The small world. *Psychology Today*, 2:60Ű67, 1967.

[13] Partha Mukherjee, Sandip Sen, and Stéphane Airiau. Norm emergence under constrained interactions in diverse societies. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 779–786, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[14] Bastin Tony Roy Savarimuthu, Stephen Cranefield, Martin Purvis, and Maryam Purvis. Norm emergence in agent societies formed by dynamically changing networks. In *IAT '07: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 464–470, Washington, DC, USA, 2007. IEEE Computer Society.

[15] Yoav Shoham and Moshe Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73:231–252, 1995.

[16] Bart Verspagen and Geert Duysters. The small worlds of strategic technology alliances. *Technovation*, 24(7):563 – 571, 2004.

[17] D. Villatoro, N. Malone, and Sandip Sen. Effects of interaction history and network topology on rate of convention emergence. *Proceedings of 3rd International Workshop on Emergent Intelligence on Networked Agents*, 2009.

[18] A. Walker and M. Woolridge. Understanding the emergence of convensions in multi agent systems. *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS '95)*, (1):384–389, 1995.

[19] D.J. Watts. Small worlds: The dynamics of networks between order and randomness. In *Princeton University Press*, 1999.

[20] D.J. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, pages 440–442, 1998.

[21] Duncan J. Watts. Networks, dynamics, and the small-world phenomenon. *American Journal of Sociology*, 105(2):493–527, 1999.

[22] F. Benjamin Zhan and Charles E. Noon. Shortest Path Algorithms: An Evaluation Using Real Road Networks. *TRANSPORTATION SCIENCE*, 32(1):65–73, 1998.

Norm Convergence in Populations of Dynamically Interacting Agents

Declan Mungovan, Enda Howley and Jim Duggan
firstname.secondname@nuigalway.ie
System Dynamics Research Group, Department of Information Technology.

Introduction

Agent Based Modelling (ABM) is a methodology used to study the behaviour of norms in complex systems (Conte 1999). Agent based simulations are capable of generating populations of heterogeneous, self-interested agents that interact with one another. Emergent norm behaviour in the system may then be understood as a result of these individual interactions (Walker 1995). Agents observe the behaviour of their group and update their belief based on those of others. Social networks have been shown to play an important role in norm convergence. In this model agents interact on a small world network with members of their own social group plus a second random network that is composed of a subset of the remaining population. Random interactions are based on a weighted selection algorithm that uses an individual's path distance on the network. This means that friends-of-friends are more likely to randomly interact with one another than agents with a higher degree of separation. Using this method we aim to investigate the effect that random interactions have on the dissemination of social norms when agents are primarily influenced

Small World Networks

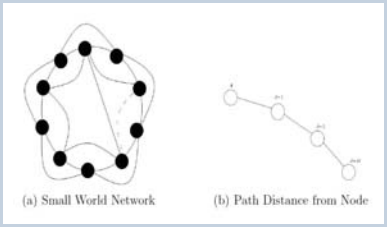


Figure 1

Small World Networks are highly clustered, yet have length scaling properties equivalent to the expectations of randomly assembled graphs (Watts 1999). Notice in Fig. 1(a) that the link with the dashed line has been re-wired to another part of the network. This creates an instant shortcut to distant nodes. Small world graphs span the gap between

ordered lattices and random graphs.

Randomness

ABMs are capable of implementing monte carlo type stochastic iterations of a complex system. Izquierdo et al. (Izquierdo 2009) highlight the fact that any computer model is in fact, due to its very nature, deterministic. However, we can use pseudo-random number generators to simulate random variables within the model and generate an artificial monte carlo generator. The pseudo-random number generator is an algorithm that takes in a random input seed value and generates a result that approximates a random number. This property allows us to simulate randomness that is present in real world systems.

Model Design

Agents receive a utility from observing the norms that have been adopted by the other individuals it encounters. This includes members of its social network and random members of the population.

$$u^j_{i,t} = \alpha D^j_{i(t-1)} + \beta R^j_{i(t-1)}$$

Agents interact with random members of the population using a Weighted Random Interaction (WRI) algorithm based on their distance from others on the network. We use a modified version of Zipf's law to calculate a nodes weight. The probability of agent ***i***, with Maximum Path Distance (MPD) of ***M***, randomly interacting with agent ***h*** having a path distance of ***d*** from ***i*** is equal to:

$$p_{ih}(d) = \frac{\frac{1}{(d-1)^\lambda}}{\sum_{m=1}^{M-1}\left(\frac{1}{m^\lambda}\right)} \qquad d \geq 2$$

Simulator Design

In all the simulations conducted, a population of 1000 agents in a small world configuration with average degree of 10 was generated. Initially all nodes are randomly given one of two norms. All the results shown are the average of 1000 different simulations. A new Small World network was generated for each simulation. Initially each agent on the network maps its social distance from every other agent.

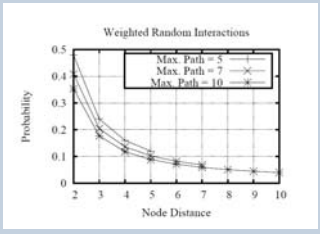


Figure 2

Results

Experiment 1

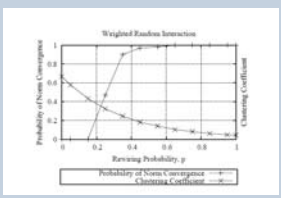


Figure 3

Results Cont'd

Experiment 2

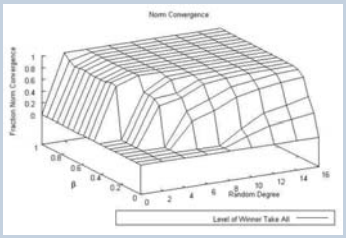


Figure 4

Experiment 3

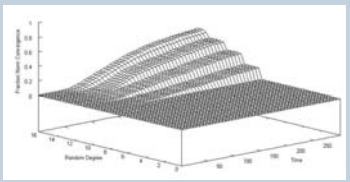


Figure 5

Experiment 4

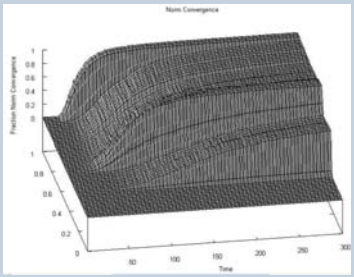


Figure 6

References

D.J. Watts. Small worlds: The dynamics of networks between order and randomness. In *Princeton University Press, 1999.*

Luis R. Izquierdo, Segismundo S. Izquierdo, Jose Manuel Galan, and Jose Ignacio Santos. Techniques to understand computer simulations: Markov chain analysis. *Journal of Artificial Societies and Social Simulation, 12(1):6, 2009.*

Rosaria Conte, Rino Falcone, and Giovanni Sartor. Introduction: Agents and norms: How to fill the gap? *Artificial Intelligence and Law, 7(1):1{15, March* 1999.

A.Walker and M.Woolridge. Understanding the emergence of convensions in multi agent systems. *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS '95), (1):384{389, 1995.*

Acknowledgements

This research is funded by the Science Foundation of Ireland and their support is acknowledged.

sfi science foundation ireland

NUI Galway OÉ Gaillimh

Argumentation on Bayesian Networks for Distributed Decision Making*

Akın Günay†
Department of Computer Engineering
Boğaziçi University
Bebek, 34342, Istanbul,Turkey
akin.gunay@boun.edu.tr

ABSTRACT

Argumentation is a daily reasoning mechanism used by human beings and it is studied under many different disciplines of science. Computational argumentation concentrates on the modeling and analyzing issues of argumentation processes. Besides, computational argumentation also provides a powerful basis to develop methodologies for multi-agent interactions. In this paper we propose an argumentation framework with the aim of distributed decision making. In our framework we represent beliefs of agents through Bayesian networks. Accordingly, we specify what an argument is, how agents interpret arguments and how agents update their beliefs according to the exchanged arguments. Through conducting a case study we also investigate effects of initial beliefs of agents and parameters of our framework on the results of distributed decision making process.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Design

Keywords

Argumentation, Bayesian Networks

1. INTRODUCTION

Computational argumentation systems are used by the artificial intelligence (AI) community to model and analyze the act of human argumentation [2]. Argumentation is mainly used as a non-monotonic reasoning mechanism in cases such as medicine and law, where classical formal logics are inadequate. More recently argumentation is used by the multi-agent systems (MAS) community to model interaction between agents as a negotiation mechanism for decision making [1, 5]. Early argumentation systems use classical logics such as propositional and predicate logics to model knowledge of agents. More recent studies use probabilistic models in order to represent the knowledge of agents better [6, 8]. Especially in [9] Vreeswijk proposes an argumentation approach using Bayesian inference. He uses the admissibility semantics of Dung [4] and proposes an algorithm to generate arguments on Bayesian networks. The approach that we propose in this paper uses a similar approach to generate arguments on Bayesian networks. However, our main concern in this paper is the use of such arguments in a multi-agent dialogue in order to achieve mutual decisions.

In this paper we develop an argumentation system for multi-agent decision making, where agents use Bayesian networks for belief representation and inference. We propose an argumentation framework that specifies the concept of an argument considering probabilistic structures. This framework also defines the belief update policies of agents on Bayesian networks based on two thresholds variables, namely, the acceptability threshold and the confidence threshold. We also provide an algorithm to guide the realization of the concepts of the framework. We implement our framework in Java language using NeticaJ API for inference. We conduct a case study, in which we investigate the effects of the initial beliefs of the agents and the threshold values proposed in the argumentation framework

This paper is organized as follows. In Section 2 we provide the background about argumentation and Bayesian networks. In Section 3 we define our framework. In Section 4 we conduct a case study on a sample Bayesian network to evaluate our argumentation framework and present our observations. Finally, in Section 5 we conclude our discussion and point out to some future directions.

2. BACKGROUND

2.1 Argumentation

Argumentation is studied in philosophy since Aristotle. It is also studied in the AI community since it is a major mechanism of daily reasoning. Computational aspects of argumentation can be divided into the following main themes [2]:

- The structure of an argument and interaction of the

*This research has been partially supported by Graduate Scholarship Program 2211 of the Scientific and Technological Research Council of Turkey and by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610.
†PhD Student

components

- The rules and protocols that define the argumentation process
- Identifying valid and invalid arguments
- The argumentation strategy(i.e. deciding which argument to propose and when future discussion is redundant

Argumentation can be used both by individual agents as an internal reasoning mechanism or by multiple agents as a medium of dialogue for information exchange, negotiation and decision making [1, 5].

2.2 Bayesian Networks

A Bayesian network [7] is a directed acyclic graph that represents a probabilistic model as a set of random variables and their conditional independence assumptions. Usually, random variables are used to model discrete states of affairs. However, it is also possible to use continuous variables. The arcs in the Bayesian network specify the independence assumptions that must hold between the variables. These independence assumptions determine the required probability information to specify the probability distribution of the random variables in the Bayesian network.

The independence assumptions have a key role for a Bayesian network since they significantly reduce the probability information required to model the underlying situation. Normally, if there are n binary random variables, without any independence assumption we need $2^n - 1$ joint probabilities to specify the complete distribution. For instance, in the Bayesian network (presented Figure 1) that we use in our case study there are five binary random variables. That is without considering the independence assumptions we need 31 joint probabilities to specify the complete distributions. On the other hand, by introducing the independence assumptions in the Bayesian network, we reduce this number to 10.

There are mainly two type of methods to evaluate a Bayesian network [3]. The first type of these methods are the exact methods. In general these methods are NP-hard. However, in the literature there are algorithms that can solve even large networks in acceptable amount of time. The other type of methods are approximate methods. These approximate methods are fast but they sacrifice some precision. However, the loss of precision is negligible in most practical cases. In general there is no single algorithm that performs the best in all cases and the performance of the algorithms is mostly dependent on the topology of the Bayesian network.

3. ARGUMENTATION FRAMEWORK

In this section we describe the details of our argumentation framework. We first discuss the role of an agent in the framework. Then we define the argument and its structure. After that we explain the belief update process. Lastly, we provide an algorithm that combines these concepts in a formal representation.

Before going into the details of our framework, we express our assumptions as follows: in our framework, we assume that agents share a common structure for their Bayesian networks, although the probability assignments of variables are different. We also assume that all variables in the Bayesian networks are discrete binary variables and are in one of the two states as true or false. When we call belief of an agent on some variable we refer to the probability assignment of this variable and if this is a conditional probability we call each possible set of state assignments of the conditional variables as a case.

3.1 Agents

DEFINITION 1. *An agent $A =< B, p >$ is a pair, where B is a Bayesian network that represents the belief base of the agent and p is a policy that defines the belief update policy of the agent.*

In our framework, an agent is a computational entity that has a belief base represented as a Bayesian network. Agents interact through arguments with each other to come up with a common conclusion on the state of some variable. Through this argumentation process each agent updates its belief base according to the information received as arguments from the other agents and its own belief update policies. The belief update policies specify the belief update behavior of the agent. Using these policies, agents do not simply change their beliefs without any question according to the beliefs of the other agents, but instead they autonomously decide about belief updates considering their own state and priorities. Therefore, agents have always the option to reject an argument of another agents without changing their beliefs. On the other hand agents can generate counter arguments to influence other agents according to their own beliefs.

3.2 Argument Structures

In classical logics an argument is a set of one or more propositions as premises along with another proposition as conclusion, such that the truth of the conclusion is a logical consequence of the given premises.

DEFINITION 2. *An argument $Arg =< v, s >$ is a pair where v is a variable in B and s is a conditional probability in B.*

Since we deal with probabilistic structures, in our context interpretation of an argument is different. An argument consists of two parts. The first part of an argument is the query variable. The query variable is the variable in the Bayesian network, on which the agents try to achieve a mutual decision. The second part of the argument is another variable (in the Bayesian network) with its probability (and the state of dependent variables if it is a conditional probability). To explain the idea better consider a Bayesian network that has three variables Q, W and R, and there is a conditional relation from R to W such that $P(W|R)$ and from W to Q such that $P(Q|W)$. We choose Q as our query variable and we want to create an argument using the conditional relation between R and W. In such a case, part one of an example argument contains the query variable Q and part two of the argument contains $P(W|R = true)$ (a possible state assignment of the conditional variable R). A second argument may again contain the query variable Q in part one and different than the first example $P(W|R = false)$ in the second part (another state assignment of the conditional variable R) . Note that the arguments does not contain the full conditional probability tables such as $P(W|R)$, but instead each argument involves only one assignment of the conditional variable such as $P(W|R = true)$ or $P(W|R = false)$.

3.3 Belief Update

Belief update is the major mechanism that agents use to achieve common conclusions. If two agents have different conclusions on the state of a variable in their Bayesian networks, they select it as the query variable and start to argue about this variable by proposing arguments to each other. With each argument the agent tries to influence the other agent by changing the beliefs of the other agent in the direction of its own beliefs.

An important issue in belief update in the context of multi-agent systems is autonomy. While updating its beliefs, an agent should consider its own state and priorities besides the beliefs of the other agents. Hence, in some cases, if the belief of the agent conflicts with the belief of the other agents, the agent should be able to reject the belief of the other agents and keep its belief as it is. Although this may prevent to achieve a common conclusion in some cases, it preserves the priorities of the agents.

To achieve autonomy in belief update in our framework agents use two threshold values represented as a policy of the agent.

DEFINITION 3. *An agent policy is a pair* $p =< a, C(v) >$ *where* a *is the acceptability threshold of the agent and* $C(v)$ *is a function that maps each variable* $v \in B$ *to a confidence threshold.*

The first one is *acceptability*. Acceptability defines the overall tolerance of the agent to the beliefs of the other agents. Higher acceptability means the agent is more tolerant to accept the other agents beliefs and change its own beliefs accordingly. On the other hand a low acceptability reflects the conservativeness of the agent to the beliefs of the other agents. When a belief of another agent is received through an argument, the agent compares its own belief with the received belief of the other agent. If the difference between the two beliefs is within the acceptability threshold of the agent then the agent accepts the belief of the other agent and applies belief update. On the other hand, if the difference between the beliefs is not within the acceptability threshold, the agent rejects other agents belief and does not update its own belief. Equation 1 represents the concept of acceptability threshold formally, in which accept is a binary variable that represents whether the belief of the other agent is acceptable or not, β_{agent} represents the belief of the agent, β_{arg} represents the belief of the other agent received as an argument and α is the acceptability limit.

$$acceptability = \begin{cases} 1, & |\beta_{agent} - \beta_{arg}| \leq \alpha \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The second threshold value is the *confidence*. The aim of the confidence is to represent the self-confidence of the agent on its beliefs, which effects the degree of change in beliefs while updating beliefs. A larger confidence value means that the change of the agents belief will be smaller. Instead of a general confidence threshold that effects all beliefs of the agent equally, we associate individual confidence values with each belief (each probability of a variable in the Bayesian network). In this way we can represent different confidence on different beliefs of the agent. This is indeed useful to reflect real world situations, such as the trustworthiness of the source of belief. If the belief of the agent about a fact is developed through information obtained from untrustworthy external sources, the confidence of the agent on this belief may be low. On the other hand, if the belief of the agent about another fact is developed through self observation by the agent itself, the confidence of the agent on this belief may be higher than the first case. Equation 2 represents the concept of confidence formally, in which β_{agent} represents the belief of the agent, β_{arg} represents the belief of the other agent received as an argument and $\omega_{\beta_{agent}}$ represents the confidence of the agent on the certain belief β_{agent}.

$$\beta_{agent} = \beta_{agent} \times \omega_{\beta_{agent}} + \beta_{arg} \times (1 - \omega_{\beta_{agent}}) \quad (2)$$

3.4 The Algorithm

In this section we present an algorithm for the agents that use the concepts of argumentation and belief update explained in Sections 3.2 and 3.3. Before going into details of the algorithm, let us remind our assumptions. We assume that the agents share a Bayesian network with identical structure, but with different probabilities assigned to variables, which are always binary (possible states of variables are always true and false). We also assume that all agents commonly know on which variable's state they want to achieve a mutual decision (i.e. the query variable).

An agent runs this algorithm when it receives an argument from another agent. The received argument consists of the belief of the other agent on a variable represented by β_{arg} and the state of the query variable in the belief base of the other agent represented by γ_{arg}. The agent first checks the acceptability of the received belief β_{arg} (line 1). If the received belief β_{arg} is acceptable it applies the update policy on its own belief base (line 2). Then, using the updated belief base it performs inference on the underlying Bayesian network to compute the posterior probability of the query variable and decides on the state of the query variable and assign the result to γ_{agent} (line 3). After that it compares the inferred state of the query variable (γ_{agent} and the state of the query variable as told by the other agent (γ_{arg}) (line 4). As the result of this comparison, if the agent captures that it agrees with the other agent on the state of the query variable, it sends an accept message to the other agent to conclude the argumentation process. On the other hand, if the agent captures that it disagrees with the other agent on the state of the query variable, it sends a reject message to the other agent to inform that the a mutual decision on the state of the query variable is not achieved yet (lines 5-7). If the received belief β_{arg} is not accepted (else part of the if-condition in line 1) the agent generates a counter argument and sends it as a reply to the argument of the other agent (line 10).

To generate a counter argument, the agent uses its belief base and the underlying Bayesian network as follows: the agent uses a flag for each variable in the Bayesian network in order to keep track of whether this variable is used in the argumentation process or not. If the agent decides to send a counter argument, it starts to check the flags on the variables starting from the query variable. If the query variable itself is not used (probably at the beginning of argumentation process), the agent creates the counter argument using its belief on the query variable and flags the query variable to prevent future use of this variable. Using its belief on the query variable as the first argument brings an advantage

Require: β_{arg}
Require: γ_{arg}
Require: β_{agent}
Require: α
Require: $\omega_{\beta_{agent}}$
1: **if** $|\beta_{agent} - \beta_{arg}| \leq \alpha$ **then**
2: $\beta_{agent} \times \omega_{\beta_{agent}} + \beta_{arg} \times (1 - \omega_{\beta_{agent}})$
3: γ_{agent} = inference()
4: **if** $\gamma_{agent} = \gamma_{arg}$ **then**
5: **return** *accept*
6: **else**
7: **return** *reject*
8: **end if**
9: **else**
10: return generateCounterArgument()
11: **end if**

Algorithm 1: The agent algorithm

to the agent to influence the other agent directly according to its own belief on the early stage of the argumentation process especially if the acceptability of the other agent is high. If the query variable is already used (flagged), the agent continues by the immediate parent variables of the query variable in the Bayesian network to generate a counter argument. If also the immediate parents were used before, the immediate child variables of the query variables are used by the agent to generate the counter argument. The agent repeats this process recursively until a non-flagged variable in the Bayesian network is found. If all variables are flagged the agent sends a stop message to the other agent to indicate that achieving a mutual decision on the state of the query variable is not possible, since all variables are already used for argumentation and stops the argumentation process.

Agents repeat this algorithm in a turn taking manner until either a mutual decision on the state of the query variable is achieved or both agents propose all of their possible arguments and still no mutual decision is achieved on the state of the query variable.

4. CASE STUDY

4.1 Setting

In this section we demonstrate the results of our case study. In our case study the agents use a well known Bayesian network, namely the family-out network, which we present in Figure 1 [3]. Although family-out network is simple, it is useful for observation since it makes possible to keep track of the progress of our argumentation system. In all cases we select the variable "Dog Out" as the query variable. We observe the behavior of our argumentation system on six different cases with different combinations of belief similarity and acceptability threshold of agents. For belief similarity we define two qualitative levels as close beliefs and distant beliefs. In close beliefs level, the difference between the beliefs of the agents for the same variable is in range [0, 0.25]. In distant beliefs level, the difference between the beliefs of the agents for the same variable is in range [0.2, 0.5]. For both levels we guarantee that the initial state of the query variable is always different for each agent (i.e. one agent says true and other says false). For acceptability threshold we choose two values as 0.25 (low acceptability) and 0.5 (medium acceptability). We set the confidence threshold to 0.5 for all cases,

Figure 1: Family-out Bayesian Network

which means its effect is neglected. As the result of each case study we observe the length of the argumentation process (number of arguments exchanged with maximum of 20 arguments due to the structure of the family-out network) and whether a mutual decision is achieved between agents at the end of the argumentation process. We summarize all cases in the following table.

Case	Belief Similarity	Agent-1 Accept.	Agent-2 Accept.
1	Close Beliefs	Low	Low
2	Close Beliefs	Medium	Medium
3	Close Beliefs	Low	Medium
4	Distant Beliefs	Low	Low
5	Distant Beliefs	Medium	Medium
6	Distant Beliefs	Low	Medium

4.2 Observations

- **Case 1:** In Case 1 the agents have belief bases with close probability assignments. On the other hand their acceptability thresholds are low, hence they are not tolerant to the beliefs of the other agents. In this case the argumentation process takes 7 turns and 14 arguments are proposed between the agents. The agents achieve to a mutual decision, in which the state of the query variable "Dog Out" is inferred as "false" by both agents (initial belief of Agent-2). In this case the argumentation process requires exchange of 14 arguments, which is considerably long (remember maximum possible number of argument exchange is 20). The major reason of this result is the low acceptability tolerance of both agents, which decreases the effect of close beliefs. However, the agents still achieve to a mutual decision on the state of the query variable.

- **Case 2:** In Case 2 the agents have belief bases with close probability assignments. Additionally, their acceptability thresholds are medium, hence they are more tolerant to the beliefs of the other agents and we expect them to achieve a mutual decision in a short amount of time. In this case the argumentation process takes 4 turns and 8 arguments are proposed between agents. The agents achieve to a mutual decision, in which the state of the query variable "Dog Out" is inferred as "false" by both agents (initial belief of Agent-2). In this case the argumentation process requires less time with respect to the Case 1 as expected, since the medium acceptability threshold of the agents allows them to accept each others arguments more easily and speeds up the belief update process. Hence, the agents achieve to a mutual decision in a shorter time.

- **Case 3:** In Case 3 the agents have belief bases with close probability assignments. However, they have different acceptability thresholds, such that the Agent-1 has low acceptability and Agent-2 has medium acceptability. Hence, Agent-2 is more tolerant to Agent-1's beliefs but the opposite is not true. In this case the

argumentation process takes 5 turns and 10 arguments are proposed between the agents. The agents achieve to a mutual decision, in which the state of the query variable "Dog Out" is inferred as "true" by both agents (initial belief of Agent-1). Different then Case-1 and Case-2 the mutual decision of the agents is "true" in this case. This is because Agent-1 has more influence on Agent-2 according to the acceptability thresholds and Agent-1 convince Agent-2 to its own decision.

- **Case 4:** In Case 4 the agents have belief bases with distant probability assignments. Additionally, their acceptability thresholds are low, which makes hard to achieve a mutual decision. In this case the argumentation process takes 10 turns and 20 arguments proposed by the agents. However, no mutual decision is achieved by the agents on the state of the query variable, due to distant beliefs and low acceptability.

- **Case 5:** In Case 5 the agents have belief bases with distant probability assignments. On the other hand, their acceptability thresholds are medium, hence there is still an expectation to achieve a mutual decision. However, after 10 turns and 20 argument propositions, no mutual decision is achieved between the agents on the state of the query variable. In this case, although agents are tolerant to each others beliefs, the initial difference between the beliefs prevent them to achieve a mutual decision.

- **Case 6:** In Case 6 the agents have belief bases with distant probability assignments. As in Case-3, Agent-2 has a higher acceptability threshold than Agent-1 and hence it is more tolerant to the beliefs of Agent-1. In this case the argumentation takes 10 turns and 20 arguments are proposed by the agents. No mutual decision is achieved by the agents. Although Agent-1 can not convince Agent-2 enough to accept its own decision about the query variable, by checking the probability assignments of Agent-2's Bayesian network at the end of the argumentation process, we still observe significant changes, such that the probability assignments are closer to Agent-1's probability assignments.

5. CONCLUSIONS

In this paper we develop an argumentation system for multi-agent decision making, where agents use Bayesian networks for belief representation and inference. For this purpose, we propose an argumentation framework, in which we define structure of an argument and belief update policies of the agents on Bayesian networks. Our belief update process is based on the acceptability threshold and confidence threshold variables. We provide an algorithm to realize the argumentation framework. We implement our framework in Java language and conduct a case study on family-out network, in which we investigate the effects of the initial beliefs of agents and the acceptability threshold on mutual decision achievement.

The results of the case study show that if the agents have belief bases with close probability assignments, agents can achieve to a mutual decision in all cases and the acceptability threshold effects the length of the argumentation process, such that higher acceptability shortens the argumentation process. On the other hand, if the agents have belief bases with distant probability assignments, agents can not achieve to a mutual decision. Besides, we observe that if there is a difference between the acceptability threshold values of the agents, the agent that has a lower threshold value has an influence on the beliefs of the agent that has a higher threshold value. Hence, the agent with lower threshold value can convince the agent with higher threshold value to its own decision.

This work can be extended in the following directions: our framework assumes that the agents share the structure of the Bayesian network and the variables have only two states. It will be interesting to extend our framework in such a way that these two assumptions are eliminated. The case study provides promising preliminary results. However, a detailed experimental study that involves various network structures, probability distributions and evidence information is required for validation.

6. REFERENCES

[1] L. Amgoud, Y. Dimopoulos, and P. Moraitis. A Unified and General Framework for Argumentation-based Negotiation. In *AAMAS '07: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[2] T. Bench-Capon and P. E. Dunne. Argumentation in Artificial Intelligence. *Artificial Intelligence*, 171(10–15):619–641, 2007.

[3] E. Charniak. Bayesian Networks without Tears. *AI Magazine*, 12(4):50–63, 1991.

[4] P. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.

[5] A. Kakas and P. Moraitis. Argumentation Based Decision Making for Autonomous Agents. In *AAMAS '03: Proceedings of the 2th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 883–890, New York, NY, USA, 2003. ACM.

[6] J. Kohlas. Probabilistic Argumentation Systems a New Way to Combine Logic With Probability. *Journal of Applied Logic*, 1(3-4):225–253, 2003.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[8] S. Saha and S. Sen. A Bayes Net Approach to Argumentation. In *AAMAS '04: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1436–1437, Washington, DC, USA, 2004. IEEE Computer Society.

[9] G. Vreeswijk. Argumentation in Bayesian belief networks. *Argumentation in Multi-Agent Systems*, 3366:111–129, 2005.

Towards Toolipse 2: Tool Support for the JIAC V Agent Framework

Michael Burkhardt
DAI-Labor, TU Berlin
michael.burkhardt@dai-labor.de

Marco Lützenberger
DAI-Labor, TU Berlin
marco.luetzenberger@dai-labor.de

Nils Masuch
DAI-Labor, TU Berlin
nils.masuch@dai-labor.de

ABSTRACT

In this paper we describe our idea of supporting the multi-agent system development within the JIAC framework by a unified tool solution. We illustrate an approach of providing a development platform, which enables comfortable, quick and comprehensive multi-agent system design and provides semantic searching for available services. At this we start with our latest three feature extensions to the JIAC framework, each one developed in the scope of a diploma thesis, and describe our planned adjustments and ideas to achieve the desired functionality.

1. INTRODUCTION

Over the last decade, Agent Oriented Software Engineering (AOSE) has gained attention as a suitable methodology for providing quality assurance within software development processes [4].

In order to counter nowadays requirements, the *DAI Labor* has developed *JIAC V*, the fifth version of its *JIAC* (Java Intelligent Agent Componentware) serviceware framework, which allows the implementation of open, distributed and scalable multi-agent systems in which agents are able to offer their abilities in a service oriented way. Development is supported by a rich library, which provides services and agents with frequently required abilities. The developer can reference these agents and extend their behaviour by custom defined services. JIAC V provides a script language for this purpose. In its second incarnation, the *JIAC Agent Description Language* (*JADL++*) [5] has been geared towards the requirements of agent oriented service specification. Regarding JIAC V's comprehensive capabilities of transparent distribution, service based interaction, semantic service descriptions, generic security and management mechanisms and support for flexible and dynamic reconfiguration in distributed environments, the framework's need for tool solutions becomes apparent.

In this work, we describe our approach in developing a toolkit for the design of applications and software components based on JIAC V. At this, we make use of our latest three framework extensions — each one a result of a separate diploma thesis [5, 7, 10]. Due to the area of application of these extensions, this work focuses on the multi-agent system (MAS) design stage with particular emphasis on JIAC V's service aspects.

Cite as: Towards Toolipse 2: Tool Support for the JIAC V Agent Framework, Michael Burkhardt, Marco Lützenberger and Nils Masuch, *Proceedings of 11th European Agent Systems Summer School (EASSS 2009)*, 31.August– 4.September, 2009, Torino, Italy.

2. JIAC V TOOLS

JADLedit is an *Eclipse* based editor for JADL++ [5], a programming language which has been designed with particular focus on an easy usage in order to assure comfortable first steps in agent oriented programming. JADL++ has been developed within the diploma thesis of the first author [5]. As one of it's main features, JADL++ uses the knowledge representation language *OWL* [3] as semantic foundation for its complex data types. JADLedit is divided into two parts, a JADL++ source code editor which provides helpful features like syntax highlighting, code completion, error marking and many more. The second part is an ontology browser, which displays detailed information on the included ontologies, such as their classes and their properties.

The **Agent World Editor** [7, 8], or AWE, is a tool which supports framework independent design of multi-agent systems while a code generation feature for an extensible set of target frameworks is provided as well. AWE employs a visual system engineering paradigm and represents even complex multi-agent systems in a single diagram, whose structure is formally specified by an underlying domain model. The code generation routine is capable of translating a platform independent design into executable code of a specific framework runtime. Currently we support *JIAC IV*, JIAC V and *MicroJIAC*, however, AWE's modular architecture also holds for frameworks beyond the JIAC world.

The *JIAC* ***SEMA****ntic* ***SE****rvice* ***MA****tcher* (*SeMa*^{2}) [10] provides a matching algorithm for the comparison between service enquiries and proposed service descriptions. Since agents shall find the appropriate services in an autonomous way, the latter are described by semantic information which allows for an automatic and detailed categorisation. The JIAC SeMa^2 algorithm is based upon the OWL service description ontology *OWL-S* [9], which allows to specify the purpose of a service by offering different parameters. Besides the name of the service these are in particular input/output parameters and preconditions and effects (*IOPE*). Preconditions and effects themselves are described in the Horn-like rule language *SWRL* [6], which extends the expressiveness of OWL.

3. TOWARDS TOOLIPSE 2

The main concern of our latest feature extensions to JIAC V was to increase the framework's overall performance.

We already evaluated SeMa^2 within the last year's edition of the Semantic Service Selection Contest [1] and received positive results. An evaluation of AWE and JADLedit has been done in the context of this year's Multi Agent Contest [2], in which we supported the JIAC team developers with our tools. At the moment, we are working on a combined tool solution in which the developer can profit from each of the three presented features from a central point.

While AWE allows for the appending of services to an agent, an existence of those is still assumed. The overall MAS development process is consequentially determined by an alternating usage of JADLedit, which is used to develop the required services, and the Agent World Editor, which is used to attach the latter to agents and design the overall multi-agent system structure.

In order to increase efficiency, we are currently working on a combination of JADLedit and the Agent World Editor.

Our basic idea at this is to use JADLedit as editor for services selected in AWE. This provides not only detailed knowledge of existing services, but also allows for additional adjustments and developments from scratch, which is moreover supported by a comprehensive overall MAS representation. Since both, AWE and JADLedit, have been developed as plug-ins to the Eclipse IDE, our main task remains in defining the co-operation between both tools.

Although the combination of AWE und JADLedit makes the service oriented agent development more comfortable, the capabilities of the service paradigm — with reusability aspects in particular — are as yet not fully utilised. The development support is still limited to the implementation and the appending of existing services to agents, however, an effective search mechanism for these specific services in the framework's libraries is currently not provided. At this point we are pursuing an application of SeMa^2.

Our approach here is similar to that of the previous combination of AWE and the JADL editor. Again, we are utilising Eclipse's plug-in mechanism and encapsulate the entire service lookup feature within a separate plug-in. The plug-in will contain a visual front-end (including a search mask, a search result table and features to add the retrieved services to an agent) and the service matcher itself. In the search mask, we will provide service retrieval in different granularity. The developer will be able to search for available services (i.e. library or custom developed) by name or by an OWL-S service description, which allows for the specification of detailed parameters, such as preconditions or the service's effects. Matching results will be displayed within a table and comprise a detailed description, while buttons allow the developer to append the retrieved results to the current MAS setup.

The combination of AWE, JADLedit and SeMa^2 will support us as multi functional tool in the design of multi-agent systems, in the accompanying service selection and in their development.

4. CONCLUSION

In this paper we described our idea in combining the results of three separate diploma theses to a MAS development tool for the JIAC V framework. In doing so, we started with an introduction of the JIAC V framework with particular focus on its service feature and motivated the necessity for a supporting tool solution. Subsequently, we introduced the results of the mentioned diploma theses, namely SeMa^2 as semantic service matcher, AWE as MAS design tool and JADLedit as service development tool and described the usage of the latter two within the overall JIAC V development process. We criticised the alternating usage of both and proposed our idea and our approach in combining AWE and JADLedit to one single tool solution. In order to provide an effective search mechanism for available services, we described our intention to include SeMa^2 in this tool combination as well. As a long-term goal Toolipse 2 shall offer methodology guided support in every aspect of the JIAC V development process.

5. REFERENCES

[1] Annual International Contest S3 on Semantic Service Selection. `http://www-ags.dfki.uni-sb.de/~klusch/s3/s3c-2008.pdf`.

[2] Multi Agent Contest. `http://www.multiagentcontest.org/`.

[3] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. W3C Recommendation, February 2004. `http://www.w3.org/TR/owl-ref`.

[4] Federico Bergenti and Michael N. Huhns. *Methodologies and Software Engineering for Agent Systems*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, chapter On the use of Agents as Components of Software Systems, pages 19–32. Kluwer Academic Publishers, 2004.

[5] Michael Burkhardt. Integration von OWL in einem Interpreter als Konzept für komplexe Datentypen. Diploma thesis, Technische Universität Berlin, Berlin, Germany, March 2009.

[6] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3c member submission, World Wide Web Consortium, 2004. `http://www.w3.org/Submission/SWRL/`.

[7] Marco Lützenberger. Development of a Visual Notation and Editor for Unifying the Application Engineering within the JIAC Framework Family. Diploma thesis, Technische Universität Berlin, Berlin, Germany, March 2009.

[8] Marco Lützenberger, Tobias Küster, Axel Heßler, and Benjamin Hirsch. Unifying JIAC Agent Development with AWE. In *Proceedings of the Seventh German Conference on Multiagent System Technologies, Hamburg, Germany*, 2009.

[9] David Martin, Mark Burstein, Erry Hobbs, Ora Lassila, Drew Mcdermott, Sheila Mcilraith, Srini Narayanan, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. OWL-S: Semantic Markup for Web Services. Technical report, November 2004. `http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/`.

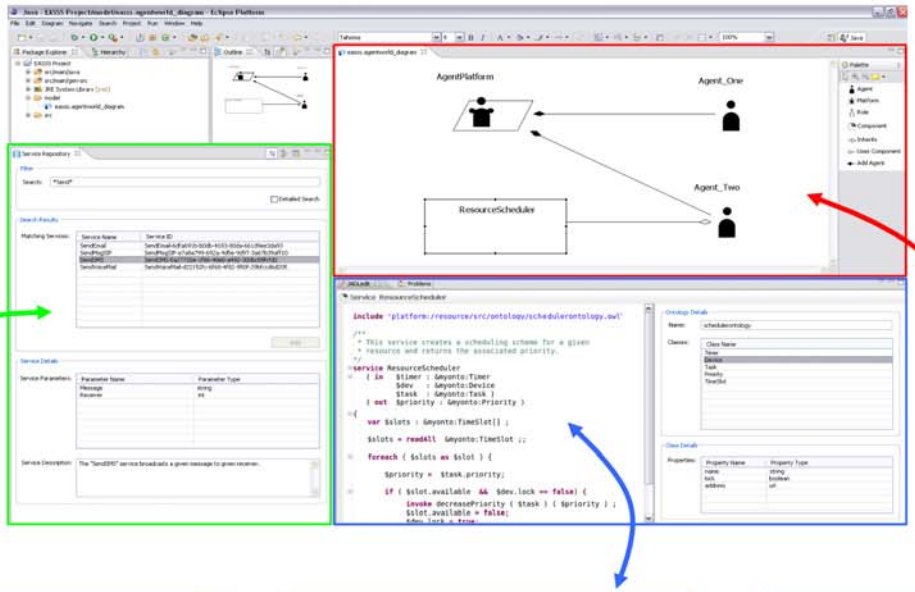
[10] Nils Masuch. Development of a Standard-Based Service Matcher Component within an Agent-Oriented Framework. Diploma thesis, Technische Universität Berlin, Berlin, Germany, April 2009.

DAI-Labor
TU Berlin

Towards Toolipse 2
Tool Support for the JIAC V Agent Framework

Motivation

JIAC V, a component-based and service-oriented agent framework, provides a graphical user interface for the simplified development of multi-agent systems with a special focus on services (Toolipse 2). Recently three framework extensions have been developed for this purpose, namely SeMa², AWE and JADLedit. In a first step, these extensions are being combined to a unified Eclipse-based IDE, which aggregates the strengths to a powerful design-time tool and is extensible to further features.



SeMa²

The **Sema**ntic **Se**rvice **Ma**tcher (SeMa²) provides a matching algorithm to match semantical enriched service descriptions at design-time and runtime.

- Uses OWL and OWL-S as semantical service description language and schema
- Provides a front-end mask for developers to search for existing services
- Enables searching for different service attributes, such as input/output parameters, preconditions and effects

JADLedit

JADLedit is a source code editor for the programming language JADL++, which enables for the service definition of software agents. At this, JADL++ utilises OWL as semantical foundation for complex data types.

- Allows ontology browsing with elective view on OWL classes and properties
- Provides code highlighting, code completion, syntactical error marking and outline view for services
- Developed as Eclipse plug-in

AWE

The **A**gent **W**orld **E**ditor is a tool which supports framework comprehensive design and deployment of multi-agent systems. At this, AWE provides visual engineering and conveniently represents even complex multi-agent systems by means of an expressive notation in a single diagram.

- Uses highly generic domain model
- Provides generation of executable code (JIAC IV, JIAC V, MircoJIAC)
- Enables comfortable framework extension
- Developed as Eclipse plug-in

Towards Toolipse 2

The figure shows an illustration of SeMa², AWE and JADLedit to a unified tool solution, which offers a comprehensive multi-agent system development at design time. In this context AWE allows to model a multi-agent system by adding different components, such as platforms, agents and services. New services can easily be developed by using JADLedit and finally added to the AWE platform. Avoiding redundancy work it is also possible to search for already existing services via the SeMa² front-end. Further, typical errors like missing imports of services can be detected at design-time.

DAI-Labor, Technische Universität Berlin – Fakultät IV Elektrotechnik und Informatik
www.dai-labor.de · sahin.albayrak@dai-labor.de
European Agent Systems Summer School 2009, Torino, September 2009

The Tenacity Of Social Actors

EL GEMAYEL Joseph
University of Toulouse - IRIT
2 Place du Doyen G. Marty, F-31042
Toulouse Cedex 09
0033 (0)5 61 63 35 00

joseph.el-gemayel@irit.fr

ABSTRACT

Using a formalization of a sociological theory, the Sociology of Organized Action, we describe an organization as a mathematical object that represents its structure and its components, Actors and Resources. Together with an analytic analysis of this structure, Actors are granted with a bounded rationality mechanism, that simulates the way they adapt their behaviors to others, in order to know what states are the most sociologically likely to appear. After the presentation of the meta-model of social organization and the simulation algorithm, a sensitivity analysis reveals the importance of the tenacity parameter.

Categories and Subject Descriptors

J.4 Social and Behavioral Sciences---Sociology, K.4.3 Organizational Impacts---Computer-supported collaborative work, I.2.11 Distributed Artificial Intelligence---Multi-agent systems, F.2.m Miscellaneous.

General Terms

Algorithms, Performance, Experimentation, Theory.

Keywords

Sociology of Organized Actions, Rationality of Social Actors, Cooperation, Simulations.

1. Meta Model of an Organization

The Sociology of Organized Action[2] defines the structure of an organization as a set of related Actors and Resources. Each Actor controls at least one Resource and so determines how well others (and him-self) can access this Resource. On the other hand, each Actor depends on some Resources and he has stakes in the Resources he needs to reach his goals. To analyze how the organization operates, we have to consider what the aims of an Actor lead him to do, rather than on the nature of these aims.

The Actor who controls a Resource defines its *state* inside a space of choice. This space of choice characterizes the autonomy of the Actor, and the Resource's state his degree of cooperation in the management of this Resource. Hence, the Actor controller determines the ability of other Actors to exploit the Resource to achieve their goals, i.e. their capacity of acting. An *effect* function is associated with each Resource to set this value:

$$effect_r: A * [-1; 1] \rightarrow [-10; 10]$$

where A is the set of Actors, [-1, 1] is the space of choice, and [-10, 10] is the range of the capacity of acting.

In order to assess the situation of an Actor in a state s of the organization, we consider his satisfaction, which is the means at his disposal to achieve his goals, defined as follows:

$$Satisfaction\ (a, s) = \sum_{r \in R} stake\ (a,r) * effect_r(a, s_r)$$

Another significant quantity is the ability of an Actor a to contribute the satisfaction of others thanks of the Resources he controls. This quantity is essential in the Sociology of the Organized Action, and is defined as follows:

$$Power\ (a, b, s) = \sum_{r \in R;\ a\ controls\ r} stake\ (b, r) * effect_r(b, s_r)$$

For any particular state, satisfaction and power can be computed for each Actor, allowing the sociologist to interpret the current state, according to his empirical knowledge of the organization he modelizes. However, every state may not be reachable, for cognitive or social acceptability reasons. In order to know which states are likely to appear from a sociological point of view, we simulate the behavior selection with a learning algorithm.

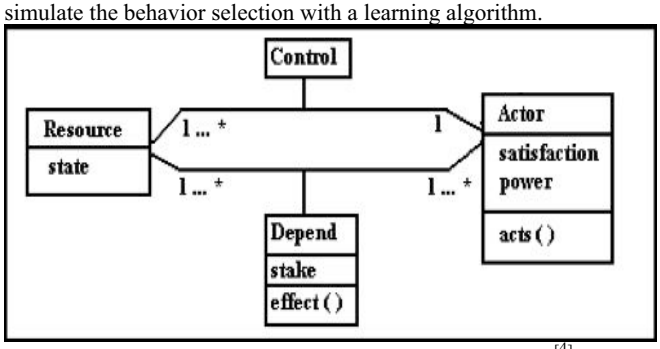


Figure 1. Meta-Model of the structure of social organization[4].

2. Simulation

The behavior selection that we compute by simulation respects the hypothesis of bounded rationality: each Actor has a strategic behavior, but he only perceives the effects of the states of the Resources he depends on, due to social Resourceship opaqueness and cognitive limitations.

That behavior has to be also quite stable, since a constitutive property of organizations is the regulation of Actors' behaviors. Finally, it has to be cooperative. Cooperation is required for the organization to sustain and to work in a proper way, and every Actor has an interest in maintaining the organization.

The assumption of rationality of social Actors leads to base this behavior on the basic cycle in which each player chooses an action according to its own objectives[3]:

- He collects information about the system state, mainly his situation.
- He decides what action to take to improve his satisfaction.
- He performs that action

until the game is stationary, i.e. that all players are satisfied.

We use a rule system to implement the learning process of each Actor. A rule is defined as {situation, action, strength, age}, where:

- situation: is the list of capacities of action provided by the

Resources the Actor depends on at the time of the rule creation.

- action: is a list of changes on the states of the Resources controlled by the Actor.
- strength: is a numerical evaluation of the effectiveness of the rule, initialized at 0, that varies according to its effect on the satisfaction of the Actor.
- age: is the age of the rules initialized at 0 and incremented at each step of the simulation.

During a simulation step, each Actor selects an action and updates the rule base as follows:

- He compares consecutive satisfaction values. The strength of the previously applied rule will be increased or decreased in proportion of the improvement.
- He selects rules, whose situation component are close to his current situation, and chooses the strongest rule.
- If the set of selected rules is empty, a new rule is created, with the current situation, a random action and a null strength.
- Once every Actor has chosen a rule, the corresponding actions are performed.

In addition, we model the ambition of an Actor with a threshold. This ambition is initialized to the maximum value of satisfaction he can reach and it will come closer to his current satisfaction during the simulation. The bigger the gap between his current satisfaction and his ambition, the more the Actor explores and undertakes vigorous actions, while he exploits his set of rules when his current satisfaction and ambition are close. The decrease of the Actor's ambition represents his resignation. When each player has a current satisfaction greater than his ambition, the simulation ends and then we consider that a regulated configuration of the organization (i.e. states set of the Resources) is reached, that is sociologically plausible: each Actor manages the resources that he controls in a way that is satisfactory for himself and others.

How quickly the ambition comes closer to the current satisfaction is defined by the Actor's tenacity parameter:

ambition + = (currentSatisfaction - ambition) * tenacity

3. Sensitivity Analysis of Tenacity

To study the influence of the tenacity on the number of steps needed for reaching a stationary state and on the levels of actors' satisfactions, we analyze the results of the simulation algorithm on the classical prisoner's dilemma.

Prisoner's dilemma is a simple model consisting of two actors: "A" which controls the relation "Ra" and "B" which controls the relation "Rb." Each player places his stakes in a way symmetrical to the other; the sum of stakes for each actor is normalized to 10. In this model, each actor put 1 on the relation controlled by himself and 9 on the relation controlled by the other actor. The Effect functions of the actors are linear (slope 1 on the relation controlled by himself and -1 on the other), which represents the value -1 for cooperation and 1 for non-cooperation. This simple model requires cooperation between the two actors in order that each actor reaches his maximum value of satisfaction (80). This model is symmetric, i.e. the reward, punishment, temptation or payoff are the same for each actor, and payoffs have only ordinal significance, i.e. they indicate whether one payoff is better than another or not.

Both figures 2 and 3 result from a sensitivity analysis, of the Prisoner's Dilemma model, with 100 experiences varying the tenacity of both actors from 900 to 999. Each experiment was conducted with 50 runs of simulation.

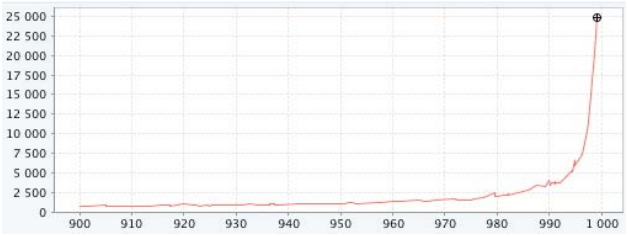


Figure 2. The variation of the average number of steps needed until the convergence is reached according to the variation of the actors' tenacity.

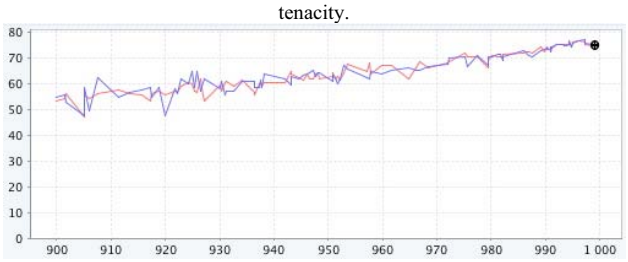


Figure 3. The variation of the average satisfaction of actors according to the variation of their tenacity.

The satisfactions of the two actors are quite similar, due to the symmetry of the system. The influence of tenacity on satisfaction and the number of steps required for convergence is due to the ambition technique used for the resignation of the actors. A high value clearly increases the satisfaction of the actors (Figure 3), as they perform a better exploration of the state space that entails a higher number of steps for convergence (Figure 2). More details may be found in [1].

4. Conclusion

This algorithm requires very little knowledge about the state and the structure of the organization, respectively low skills, keeping this mechanism socially and cognitively likely, and it does not prejudge how social Actors determine their behavior. The behavior of the algorithm is constrained by several other parameters that are also investigated by a sensitivity analysis.

5. REFERENCES

[1] Chapron, P., El-Gemayel, J., Sibertin-Blanc, C., *Impact of Tenacity upon the Behaviors of Social Actors,* submitted to publication.

[2] Crozier, M., *The Bureaucratic Phenomenon*, University of Chicago Press, *Le phénomène bureaucratique*. Edition du Seuil, Paris, 1963.

[3] Mailliard, M., Roggero, P., Sibertin-Blanc, C., *Un modèle de la rationalité limitée des acteurs sociaux*. Dans : Journées Francophones sur les systèmes Multi-Agents (JFSMA'06), Annecy, 18/10/2006 – 20/10/2006, V. Chevrier, M-P Huget (Eds.), p. 95-99, Hermès, 2006.

[4] Sibertin-Blanc, C., Amblard, F., Mailliard, M., *A coordination Framework based on the Sociology of Organized Action.* Dans : Coordination, Organizations, Institutions and Norms in Multi-Agent Systems. O. Boissier, J. Padget, V. Dignum, and G. Lindemann (Eds.), Lecture Notes in Computer Sciences, V. 3913, p. 3-17, Springer, 2006.

The Tenacity Of Social Actors

EL GEMAYEL Joseph, PhD Student
University of Toulouse - IRIT, 2 Place du Doyen G. Marty, F-31042 Toulouse Cedex 09
joseph.el-gemayel@irit.fr

Abstract

Using a formalization of a sociological theory, the Sociology of Organized Action, we describe an organization as a mathematical object that represents its structure and its components, Actors and Resources. After an analytic analysis of this structure, Actors are granted with a bounded rationality mechanism, that simulates the way they adapt their behaviors to others, in order to know what states are the most sociologically likely to appear. The meta-model and the algorithm principle are introduced.

Meta Model of an Organization

The Sociology of Organized Action defines the structure of an organization as a set of related Actors and Resources. Each Actor controls at least one Resource and so determines how well others (and him-self) can access this Resource. On the other hand, each Actor depends on some Resources and he has stakes in the Resources he needs to reach his goals. To analyze how the organization operates, we have to consider what the aims of an Actor lead him to do, rather than on the nature of these aims.

The Actor who controls a Resource defines its *state* inside a space of choice. This space of choice characterizes the leeway of the Actor, and the Resource's state his degree of cooperation in the management of this Resource. Hence, the Actor controller determines the ability of other Actors to exploit the Resource to achieve their goals, i.e. their capacity of acting. An *effect* function is associated with each Resource to set this value:

$$effect_r: A * [-1; 1] \rightarrow [-10; 10]$$

where [-1, 1] is the space of choice and [-10, 10] is the range of the capacity of acting.

In order to assess the situation of an Actor in a state s of the organization, we consider his satisfaction, that is the means at his disposal to achieve his goals, defined as follows:

$$Satisfaction\ (a, s) = \sum_{r \in R} stake\ (a,r) * effect_r(a, s_r)$$

Another significant quantity is the ability of an Actor a to contribute the satisfaction of others thanks of the Resources he controls. This quantity is essential in the Sociology of the Organized Action, and is defined as follows:

$$Power\ (a, b, s) = \sum_{r \in R;\ a\ controls\ r} stake\ (b, r) * effect_r(b, s_r)$$

For any particular state, satisfaction and power can be computed for each Actor, allowing the sociologist to interpret the current state, according to his empirical knowledge of the organization he modelizes. However, every state may not be reachable, for cognitive or social acceptability reasons. In order to know which states are likely to appear from a sociological point of view, we simulate the behavior selection with a learning algorithm.

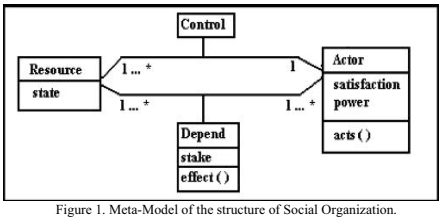


Figure 1. Meta-Model of the structure of Social Organization.

Simulation

The behavior selection that we compute by simulation respects the hypothesis of bounded rationality: each Actor has a strategic behavior, but he only perceives the effects of the states of the Resources he depends on, due to social Resourceship opaqueness and cognitive limitations.

That behavior has to be also quite stable, since a constitutive property of organizations is the regulation of Actors' behaviors. Finally, it has to be cooperative. Cooperation is required for the organization to sustain and to work in a proper way, and every Actor has an interest in maintaining the organization.

The assumption of rationality of social Actors leads to base this behavior on the basic cycle in which each player chooses an action according to its own objectives:

- He collects information about the system state, mainly his situation.
- He decides what action to take to improve his satisfaction.
- He performs that action

until the game is stationary, i.e. that all players are satisfied.

We use a rule system to implement the learning process of each Actor. A rule is defined as {situation, action, strength, age}, where:

- situation: is the list of capacities of action provided by the Resources the Actor depends on at the time of the rule creation.
- action: is a list of changes on the states of the Resources controlled by the Actor.
- strength: is a numerical evaluation of the effectiveness of the rule, initialized at 0, that varies according to its effect on the satisfaction of the Actor.
- age: is the age of the rules initialized at 0 and incremented at each step of the simulation.

During a simulation step, each Actor selects an action and updates the rule base as follows:

- He compares consecutive satisfaction values. The strength of the previously applied rule will be increased or decreased in proportion of the improvement.
- He selects rules, whose situation component are close to his current situation, and chooses the strongest rule.
- If the set of selected rules is empty, a new rule is created, with the current situation, a random action and a null strength.
- Once every Actor has chosen a rule, the corresponding actions are performed.

In addition, we model the ambition of an Actor with a threshold. This ambition is initialized to the maximum value of satisfaction he can reach and it will come closer to his current satisfaction during the simulation. The bigger the gap between his current satisfaction and his ambition, the more the Actor explores and undertakes vigorous actions, while he exploits his set of rules when his current satisfaction and ambition are close. The decrease of the Actor's ambition represents his resignation. When each player has a current satisfaction greater than his ambition, the simulation ends and then we consider that a regulated configuration of the organization (i.e. states set of the Resources) is reached, that is sociologically plausible: each Actor manages the resources that he controls in a way that is satisfactory for himself and others.

How quickly the ambition comes closer to the current satisfaction is defined by the Actor's tenacity parameter:

$$ambition\ +\ =\ (\ currentSatisfaction\ -\ ambition\ )\ *\ tenacity$$

Sensitivity Analysis of Tenacity

To study the influence of the tenacity on the number of steps needed for reaching a stationary state or on the levels of satisfaction, we analyze the results of the simulation algorithm on the classical prisoner's dilemma.

Prisoner's dilemma is a simple model consisting of two actors: "A" which controls the relation "Ra" and "B" which controls the relation "Rb." Each player places his stakes in a way symmetrical to the other; the sum of stakes for each actor is normalized to 10. In this model, each actor put 1 on the relation controlled by himself and 9 on the relation controlled by the other actor. The Effect functions of the actors are linear (slope 1 on the relation controlled by himself and -1 on the other), which represents the value -1 for cooperation and 1 for non-cooperation. This simple model requires cooperation between the two actors in order that each actor reaches its maximum value of satisfaction (80). It is noticeable that this model is symmetric, i.e. the reward, punishment, temptation or payoff is the same for each actor, and payoffs have only ordinal significance, i.e. they indicate whether one payoff is better than another or not.

Both figures 2 and 3 result from a sensitivity analysis, of the Prisoner's Dilemma model, with 100 experiences varying the tenacity of both actors from 900 to 999. Each experiment was conducted with 50 runs of simulation.

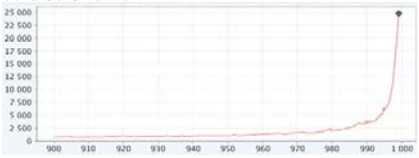


Figure 2. The variation of the average number of steps needed until the convergence is reached according to the variation of the actors' tenacity.

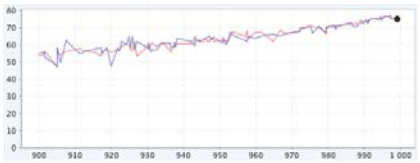


Figure 3. The variation of the average satisfaction of actors according to the variation of their tenacity.

The satisfactions of the two actors are quite similar, due to the symmetry of the system. The influence of tenacity on satisfaction and the number of steps required for convergence is due to the ambition technique used for the resignation of the actors. A high value clearly increases the satisfaction of the actors (Figure 3), as they perform a better exploration of the state space that entails a higher number of steps for convergence (Figure 2).

Conclusion

This algorithm requires very little knowledge about the state and the structure of the organization, respectively low skills, keeping this mechanism socially and cognitively likely, and it does not prejudge how Social Actors determine their behavior. The behavior of the algorithm is constrained by several other parameters that are also investigated by a sensitivity analysis.

The Impact of Routing on Traffic Congestion

Cristian Gratie, PhD student
Department of Computer Science
University "Politehnica" of Bucharest
313 Splaiul Independentei
Bucharest, Romania 060042
cgratie@yahoo.com

ABSTRACT

Traffic congestion is an important problem of today's urban life. There are various factors that cause it, such as the bad timing of traffic lights or the low throughput of intersections. Most solutions proposed for congestion are focused on one of the factors and are aimed at reducing its impact on the traffic conditions.

This paper focuses on rush hour congestion and on the choice of routes as its main cause. Indeed, many drivers tend to follow the same routes to their destination, choosing the one that is shortest or in other ways preferable.

The paper aims to analyze the impact of routing upon traffic and show that even greedy routing can reduce congestion if the routes are adapted to traffic conditions. Fixed and adaptive routing are compared by means of a traffic simulator. A multi-agent system is used for collecting traffic information from the drivers and for providing them with the best route to their destination.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Experimentation, Performance

Keywords

agent modeling, traffic congestion, adaptive routing

1. INTRODUCTION

This paper aims to investigate the impact of routing on traffic conditions and the extent to which this impact can be used for alleviating congestion.

I have developed a traffic simulator for studying the effects of two routing approaches: static and dynamic (adapted to traffic conditions). A multi-agent system is used for collecting traffic information and also for assigning routes to each vehicle, based on its current position and its destination.

Experimental results show that traffic conditions can improve significantly if the routes account for congested streets and try to avoid them.

The next section will give an overview of the work that has been done so far in the field of traffic improvement. Section 3 will introduce fixed and adaptive routing, as well as the multi-agent system. Experimental results are discussed in section 4. The paper ends with a conclusion and a few ideas for further research in section 5.

2. RELATED WORK

D. A. Roozemond suggested the use of intelligent agents for adapting traffic light controllers to changes in the traffic flow [6]. M. Wiering showed that this approach does indeed outperform non-adaptive approaches under network saturation conditions [9].

K. Dresner and P. Stone proposed an intersection management system based on reservations, in order to maximize intersection throughput [1]. Another approach, based on decentralized communication between vehicles, was proposed by Rolf Naumann and Reiner Rasche [5]. These two ideas offer a significant improvement in traffic conditions, but they both rely on autonomous vehicles.

C. Y. Liang and H. Peng have shown that the forming of platoons (one leader and followers) can increase the average driving velocity [3]. This idea was extended by J. VanderWerf et al. with a system involving continuous communication between vehicles [8] and by J. Hedrick et al. with an event-driven system. S. Hallé and B. Chaib-draa have proposed decentralized platoons as an alternative [2].

D. E. Moriarty and P. Langley focused on highway traffic and proposed a system for cooperative lane selection [4].

K. Tumer and A. Agogino have addressed rush hour congestion by using a multi-agent system and reinforcement learning [7]. They focus on two different aspects of traffic in a commuting environment: the choice of departure times and the choice of routes. The proposed solution is cooperative and ensures congestion avoidance, with the possible drawback that some drivers will not be using the fastest routes available.

This paper, on the other hand, is focused on using the same greedy approach that some drivers are already using when choosing routes, but adapt these routes to traffic conditions so as to offer the fastest (though possibly not shortest) route available.

3. THE PROPOSED SYSTEM

The multi-agent system consists of three types of agents. A Driver agent is associated with each vehicle and is responsible with guiding it to its destination. An Intersection agent is associated with each intersection. Its duties are to collect, aggregate and forward traffic information from the Driver agents to the City agent. Finally, the City agent is the that does the actual route computations and sends

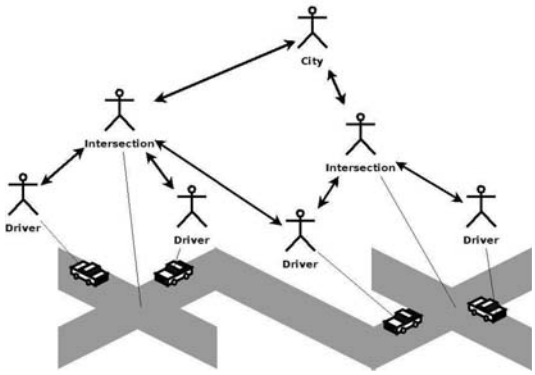


Figure 1: Overview of the multi-agent system

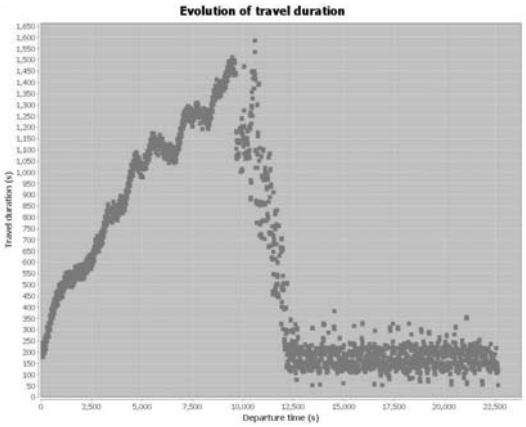


Figure 2: Evolution of the total travel time.

routes to Drivers through Intersection agents. The general architecture of the system is shown in figure 1.

The traffic information consists of the average velocities on each street, updated at regular intervals. The routing algorithm operates on the road network graph, using as cost the time needed for reaching one intersection from another. Because the velocities are in fact related to a past state, the system would provide delayed route adjustments. In order to overcome this, the second best route is computed as an alternative and are used with a probability that increases to 0.5 as the travel time of the fastest route comes closer to that of the alternative path.

Note that congestion is associated with low velocities, so with high travel times, which means that congested streets will not be part of the chosen routes if faster alternatives exist.

4. EXPERIMENTAL RESULTS

Experiments consist in generating cars at regular intervals. Each car has a predetermined destination. The multi-agent system is used for guiding the car along a route to its destination. Upon arrival, the car reports the total time it needed.

The routing approach described in the previous section is compared with a static approach that uses fixed estimates of the velocities, based on the length of the street, the speed limit and the average waiting times of the semaphores.

For a better comparison of the two approaches, experiments were run by using the static approach for half of the simulation time and the adaptive routing for the other half. The evolution of the system is described by a chart showing the total time needed for reaching the destination. Such a chart is presented in figure 2, where it is easy to see that the adaptive routing can even help the system recover from serious congestion.

5. CONCLUSIONS AND FUTURE WORK

The results of the performed experiments show that routing has a significant impact on traffic congestion and that an intelligent choice of routes can greatly improve traffic conditions.

A significant drawback of the current algorithm is that it is computationally expensive, since it runs in $O(n^3)$ time, which makes it inappropriate for deployment. Therefore, the most challenging goal for the future is to decentralize the routing algorithm, while keeping the overall system behavior similar to that of the current, centralized approach. Another extension would be to add learning capabilities to the multi-agent system.

6. REFERENCES

[1] K. Dresner and P. Stone. A Multiagent Approach to Autonomous Intersection Management. *Journal of Artificial Intelligence Research*, 31:591–656, 2008.

[2] S. Hallé and B. Chaib-draa. A collaborative driving system based on multiagent modelling and simulations. *Transportation Research Part C*, 13(4):320–345, 2005.

[3] C. LIANG and H. PENG. String Stability Analysis of Adaptive Cruise Controlled Vehicles. *JSME Int Journal. Ser C. Mech Systems, Mach Elem Manuf*, 43(3):671–677, 2000.

[4] D. Moriarty and P. Langley. Learning cooperative lane selection strategies for highways. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[5] R. Naumann and R. Rasche. *Intersection collision avoidance by means of decentralized security and communication management of autonomous vehicles.* 1997.

[6] D. Roozemond. Using intelligent agents for urban traffic control control systems. In *Proceedings of the International Conference on Artificial Intelligence in Transportation Systems and Science*, pages 69–79, 1999.

[7] K. Tumer and A. Agogino. Agent Reward Shaping for Alleviating Traffic Congestion. 2006.

[8] J. VanderWerf, S. Shladover, N. Kourjanskaia, M. Miller, and H. Krishnan. Modeling effects of driver control assistance systems on traffic. *Transportation Research Record: Journal of the Transportation Research Board*, 1748(-1):167–174, 2001.

[9] M. Wiering. Multi-Agent Reinforcement Leraning for Traffic Light Control. In *Proceedings of the Seventeenth International Conference on Machine Learning table of contents*, pages 1151–1158. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2000.

The impact of routing on traffic congestion

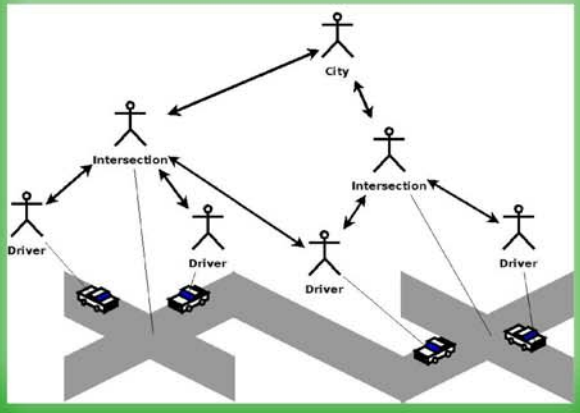
Cristian Gratie
AI-MAS Group
Politehnica University of Bucharest

Which is the best route?

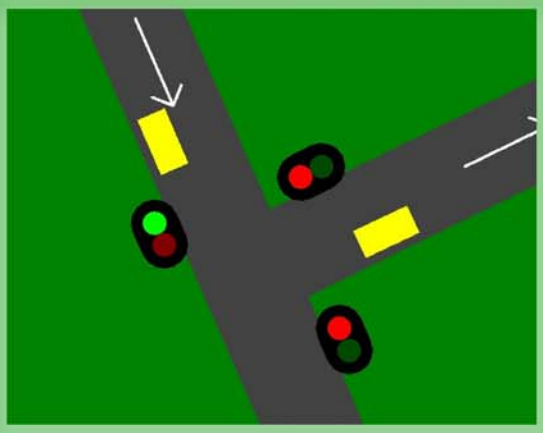
Clearly the shortest route is not always the best choice, especially when many other drivers attempt to use it as well. Whenever more drivers use the same path between two points, congestion is bound to occur. What if all drivers were able to evaluate the current traffic flow and choose their routes accordingly? Would such a behavior also generate congestion or would it rather reduce it?

A multi-agent system

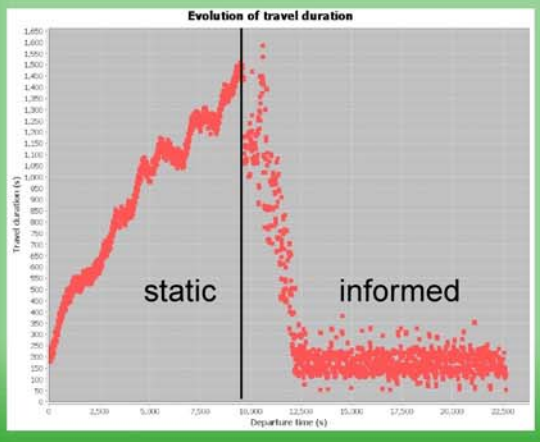
Average speeds are reported for every street segment and the instantaneous speeds are approximated and used for routing. A simple multi-agent system is used for transfering information to and from a central agent that performs the actual routing.



Simulation and results



Simulations were run with static routes then informed routing was activated in order to see the difference. The ever-growiing congestion is rapidly reduced and an equilibrium is reached. Further research will focus on turning the MAS into an intelligent, distributed system.



Contact: cgratie@yahoo.com

A Rule-Based Psychologist Agent for Improving the Performances of a Sportsman

Andrei-Horia Mogos
University Politehnica of Bucharest
313 Splaiul Independentei
060042, Bucharest, Romania
mogosandrei@yahoo.com

Monica Cristina Voinescu
University Politehnica of Bucharest
313 Splaiul Independentei
060042, Bucharest, Romania
vmc4u@yahoo.com

ABSTRACT
The analysis of the psychological characteristics of a sportsman is an important research domain. In this paper we present a rule-based system that helps a sportsman to improve his sportive value in order to obtain better performances. The central part of this system is a psychologist agent that analyzes the events and the psychological state of the sportsman and recommends some actions to handle some negative event during the sportive competitions.

Categories and Subject Descriptors
I.2.1 [**Applications and Expert Systems**]: Medicine and science; J.4 [**Social and Behavioral Sciences**]: Psychology.

Keywords
Agent, psychologist, sportsman, rule-based system, performance

1.INTRODUCTION
In this paper, we propose an intelligent coaching agent that will use affective computing to help a sportsman to keep an optimal psychological shape for achieving performance.

Although there are plenty of instances of virtual assistants and tutors in the literature, most of these solutions do not take into consideration the affective aspect of human-computer interaction. Few of the works in the affective computing literature addresses the challenge of designing affect-enabled tutoring systems or e-learning platforms.

An example of such an affect-enabled system is the AutoTutor designed by the MIT MediaLab, an Intelligent Tutoring System which is capable to detect emotions and "mindfully" select its pedagogical strategy. The AutoTutor awaits further development and empirical testing to verify the prediction of a superior learning gain [1]. Other projects aimed to prove the potential of affective computing in learning are the TAPA system for cognitive Training with Animated Pedagogical Agents [2] in children with learning disabilities, and the MYSELF e-learning platform endowed with affective computing capabilities, mainly for the training of relational skills [3].

The most closely related work in the literature to the objective of our system is the research project of Bickmore and his colleagues at the MIT Media Lab in which several "relational" skills are designed into a computer interface agent named "Laura" within the context of a healthy behavioral change towards physical activity adoption [4].

The paper is organized as follows. Section 2 contains the description of the problem. Section 3 describes the structure of our system. In Section 4 we discuss about the algorithm used by the system and about the implementation. Finally, Section 5 contains the conclusion of our paper.

2.PROBLEM DESCRIPTION
Consider a sportsman who participates in several competitions. The goal of the psychologist agent is to help the sportsman to optimize his sportive value in order to get better performances. The sportsman's sportive value at any moment in time can be evaluated based on some of his psychological characteristics and some physical characteristics. In this paper we consider only psychological characteristics.

Consider a set of psychological characteristics denoted by $\{1, \ldots, c_n\}$. These characteristics form the psychological state of the sportsman. Various competition events have been modeled using a set of events $\{e_1, ..., e_m\}$, where each event is described in terms of the psychological characteristics it modifies. For example, an event e can be represented as follows: $c_i = c_i + 8$, $c_j = c_j - 3$, $c_k = c_k$, $k \neq i, j$.

After each negative event (the values of some characteristics decrease), the psychologist agent gives the sportsman some actions to perform in order improve the sportsman psychological state. Each psychological characteristic c_i, has an associated action a_i that can increase the value of that particular characteristic. Each action a_i can also increase the values of other characteristics, but by a smaller degree than for the characteristic c_i. The actions can be represented in the same way as the events.

3.SYSTEM STRUCTURE
The structure of our system is presented in Figure 1. The Events module generates the events that model the various real world events of the competitions related to the sportsman. The Psychological State module contains the psychological characteristics of the sportsman and evaluates his current psychological state.

The Real World Module gets an event and makes the corresponding changes over the psychological state of the sportsman. It receives from the Psychological State module the characteristics, called C and returns to that module the modified characteristics, called C'.

The Psychologist agent receives the modified characteristics C' and, if necessary, finds some actions to improve the characteristics C'. Each action a is executed by the Executor and the resulting characteristics C'' are sent to the Psychological State module. When the characteristics C'' are better than the characteristics C, the system stops and waits for another event e.

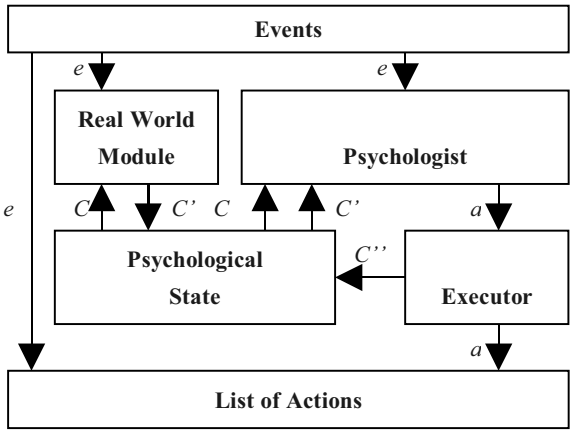


Figure 1.The system structure

The result of the system is the list of actions that can help the sportsman to neutralize the effects of a negative event he experienced.

4.ALGORITHM AND IMPLEMENTATION

Consider the set of psychological characteristics $C=\{c_1, \ldots, c_n\}$, the set of events $E=\{e_1, \ldots, e_m\}$, and the set of actions, $A=\{a_1, \ldots, a_n\}$. For two sets $C'=\{c_1', c_2', \ldots, c_n'\}$ and $C''=\{c_1'', c_2'', \ldots, c_n''\}$, we say that $C' >= C''$ if for every i, $1<=i<=n$, we have $c_i' >= c_i''$.

Next, we present the algorithm used by our system:

```
Algorithm (C, E, A) {
   while (true) {
     e = WaitForAnEvent(E)
     C = PsychologicalState ()
     C' = RealWorldModule (e, C)
     If (C >= C') then {
        while (C >= C') {
           a = Psychologist (e, C, C', A)
           C'' = Executor (a, C'), C' = C'' }}}}
```

The function call *Psychologist (e, C, C', A)* returns an action that improves the sportsman's characteristics. The function uses the following ideas: choose each action at most ones when handling an event, choose randomly between the actions that were not already used. One can easily find better ideas for this function. We only constructed a simple function, our main goal being to see how the system works.

For the implementation we chose a rule-based programming language, CLIPS [5]. Consider we have 5 psychological characteristics, 4 events and 5 actions. Next, we present a short example of a result of our system (the form of the result is changed to better fit with the paper):

C $\longrightarrow$ 46 61 82 41 102

After the event 4: C' $\longrightarrow$ 46 61 77 41 102

After the action 5: C' $\longrightarrow$ 46 61 77 44 110

After the action 3: C' $\longrightarrow$ 46 61 82 46 110

5.CONCLUSION

This paper presents a rule-based psychologist agent that helps a sportsman to improve his sportive value in order to obtain better performances. All the necessary parameters for our analysis, i.e. psychological characteristics, events, and actions are represented using natural numbers. In order to make the analysis we assume that this representation is already done, and we are using it as a particularity of the sportsman.

We can say that our system can be seen as a rule-based system, and after some research related to the psychology of a sportsman, our system can become a real expert system.

6.REFERENCES

[1] D'Mello, S., Jackson, T., Craig, S., Morgan, B., Chipman, P., White,H., Person, N., Kort, B., el Kaliouby, R., Picard., R.W. and Graesser, A., (2008). "AutoTutor Detects and Responds to Learners Affective and Cognitive States," Workshop on Emotional and Cognitive Issues at the International Conference of Intelligent Tutoring Systems, Montreal, Canada.

[2] Mohamad, Y., Carlos A. Velasco, C. A., Damm, S., Tebarth, H. (2004). "Cognitive Training with Animated Pedagogical Agents (TAPA) in Children with Learning Disabilities", K. Miesenberger et al. (Eds.): ICCHP 2004, LNCS 3118, pp. 187–193, Springer-Verlag Berlin Heidelberg.

[3] Anolli, L., Mantovani, F., Balestra, M., Agliati, A., Realdon, O., Zurloni, V., Mortillaro, M., Vescovo, A., Confalonieri, L., (2005). "The Potential of Affective Computing in E-Learning: MYSELF project experience", International Conference on Human-Computer Interaction (Interact 2005), Workshop on eLearning and Human-Computer Interaction: Exploring Design Synergies for more Effective Learning Experiences, Roma

[4] Bickmore, T., Gruber, A., and Picard, R.W. (2005). "Establishing the Computer-Patient Working Alliance in Automated Health Behavior Change Interventions," Patient Educational Counseling, Volume 59, No. 1, pp. 21-30.

[5] CLIPS, http://clipsrules.sourceforge.net/, accessed July 2009

A Rule-Based Psychologist Agent for Improving the Performances of a Sportsman

Andrei-Horia Mogos, Monica Cristina Voinescu
University Politehnica of Bucharest, Romania

Introduction

A rule-based system that helps a sportsman to improve his sportive value in order to obtain better performances

A psychologist agent that analyzes the events and the psychological state of the sportsman and recommends some actions to handle some negative events during the sportive competitions

Problem Description

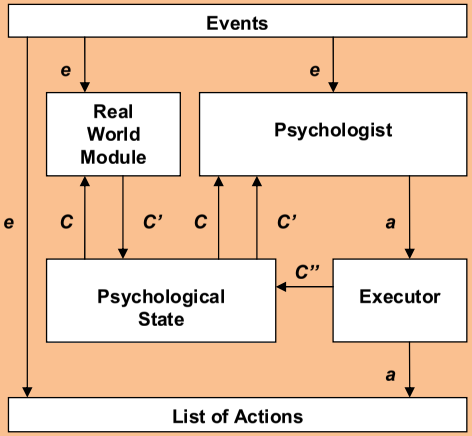
Psychological characteristics $\{c_1,...,c_n\}$: form the psychological state

Events $\{e_1,...,e_m\}$: each event is described in terms of the psychological characteristics it modifies

Ex: $e_x = \{c_i = c_i + 8, c_j = c_j - 3\}, c_k = c_k, k \neq i, j$

Actions $\{a_1,...,a_n\}$: each c_i has an associated a_i that can improve its value

System Structure



e – event
a – action
C – psychological characteristics
C' – psychological characteristics modified by *e*
C'' – psychological characteristics modified by *a*

The result of the system is the list of actions that the sportsman must done

Algorithm and Results

```
Algorithm (C, E, A) {
  while (true) {
    e = WaitForAnEvent(E)
    C = PsychologicalState ()
    C' = RealWorldModule (e, C)
    If  (C >= C') then {
       while (C >= C') {
         a = Psychologist (e, C, C', A)
         C'' = Executor (a, C'), C' = C'' }}}}
```

Results: C -> 46 61 82 41 102
After the event 4: C' -> 46 61 77 41 102
After the action 5: C' -> 46 61 77 44 110
After the action 3: C' -> 46 61 82 46 110

Implementation: CLIPS

Conclusion

Our system can be seen as an artificial psychologist that helps a sportsman to improve his performances

Contact: mogosandrei@yahoo.com, vmc4u@yahoo.com

This volume contains the papers presented at the Student Session of the 11th European Agent Systems Summer School (EASSS) held on 2nd of September 2009 at Educatorio della Providenza, Turin, Italy.
The Student Session, organised by students, is designed to encourage student interaction and feedback from the tutors. By providing the students with a conference-like setup, both in the presentation and in the review process, students have the opportunity to prepare their own submission, go through the selection process and present their work to each other and their interests to their fellow students as well as internationally leading experts in the agent field, both from the theoretical and the practical sector.

ISSN 1864-9300