ARGUING OVER MOTIVATIONS WITHIN THE V3A-ARCHITECTURE FOR SELF-ADAPTATION

Maxime Morge\textsuperscript{1}, Kostas Stathis\textsuperscript{2} and Laurent Vercouter\textsuperscript{3}

\textsuperscript{1}Dipartimento di Informatica, Universit\`a di Pisa, Largo B. Pontecorvo, 3, Italy
\textsuperscript{2}Department of Computer Science, Royal Holloway, University of London, Egham, TW20 0EX, U.K.
\textsuperscript{3}Centre G2I, Ecole des Mines de Saint Etienne, 158, cours Fauriel, F-42023 Saint-Etienne, France

Keywords: Multiagent systems, Agents models of motivation and personality, Agent architecture, Argumentation, Self-adaptation.

Abstract: The Vowel Agent Argumentation Architecture (V3A) is an abstract model by means of which an autonomous agent argues with itself to manage its motivations and arbitrate its possible internal conflicts. We propose an argumentation technique which specifies the internal dialectical process and a dialogue-game amongst internal components which can dynamically join/leave the game, thus having the potential to support the development of self-adaptive agents. We exemplify this dialectical representation of the V3A model with a scenario, whereby components of the agent’s mind called facets can be automatically downloaded to argue an agent’s motivation.

1 INTRODUCTION

Component-based engineering is a promising approach to develop adaptive systems. Adaptation is obtained by replacing some components by others or just by changing the connections between components. This approach has already been adopted to build agents (Ricordel and Demazeau, 2000; Meurisse and Briot, 2001; Vercouter, 2004) to ease the modification of the internal structure of an agent. A current challenge is to automate this process in order to provide self-adaptive agents. During this process, the coherence of an assembly of components must be warranted, i.e when the assembly is changed (addition/removal of components and/or connections) the result must be coherent. Solutions have been proposed to deal with these issues but are not completely satisfactory. The work of (Meurisse and Briot, 2001) proposes to foresee all the possible assemblies and seeks to describe what should be done in each case. This approach reduces the openness of the system only to unforeseen situations. Other works rely on human intervention (Ricordel and Demazeau, 2000) or they do not warranty the coherence of the resulting behaviour (Vercouter, 2004).

In this paper we propose the Vowel Agent Argumentation Architecture (V3A) to design self-adaptive agents. Inspired by the Vowels approach (Demazeau, 1995), V3A is built upon an intentional stance. Additionally, V3A is divided in components called facets encapsulating the different motivations of an agent. We view these motivations as arguments describing the conditional decisions to achieve goals. A facet can join the internal debate to argue for/against the adoption of a motivation. Then, the personality arbitrates the possible conflicts. Regarding self-adaptation, this personality corresponds to high-level guidelines to solve conflicts appearing when modifying the component assembly. The contribution of the work is an abstract model of agency and the definition of a dialogue-game that can be played by facets which can dynamically join or leave the game. For this purpose, we consider an argumentation framework (Dung, 1995) built to realize this internal dialectical process within this modular architecture. A scenario illustrates the use of this mechanism.

The paper is organised as follows. Section 2 introduce the walk-through example to motivate our proposal. Section 3 presents the V3A model, Section 4 presents the argumentation framework used to support it, and Section 5 describes the dialogue-game facets play. We conclude in Section 6 where we also present related work and we discuss our plans for the future.
2 WALK-THROUGH EXAMPLE

We motivate our approach with the following scenario. Max, who lives in Pisa, must reach London for a meeting. Since he does not like to spend too much time in queues in order to buy tickets, he has a PDA equipped with a personal service agent (PSA) in order to automatically buy a ticket when Max is in a train station. The PSA has been set up by the constructor to respect the user’s preferences and to use the computing system in order to assist the user.

Our starting point is the Vowels approach of multiagent oriented programming (Demazeau, 1995), which has shown to be suited for developing agent platforms e.g. see (Ricordel and Demazeau, 2000) and (Ver- couter, 2004). In this approach a multiagent system (MAS) is analysed across five dimensions: Users, Agents, Environments, Interactions, and Organizations. The Vowels approach is generally used in the analysis stage of the building process in order to divide the problem according to these dimensions.

**Users.** Max has configured his PSA for this travel and does not allow his PSA for paying extra commissions since he will not be refunded for these. When Max arrives in a railway station, the PSA automatically downloads the four following components.

**Agents.** The PSA has a representation of the available seller agent (SA).

**Environments.** The PSA have the knowledge about the current location, the current time, the possible traffic troubles, and the train time schedule.

**Interactions.** The PSA has a representation of the protocol required for communicating with the SA.

**Organizations.** The PSA has a representation of the norms adopted by the system for the automatic payment.

In this way, when Max arrives in the railway station of Pisa, the PSA is able to request a train ticket to reach Pisa airport. Since the train is not fully booked the SA sells one to the PSA and the payment is performed within the computing system. Therefore, Max has a reservation and can show its PDA to the controller agent on board. When Max arrives in Stansted, the PSA automatically replaces the four previous components. Since the SA overcharges the train ticket price with extra fees, the PSA does not register the user. Therefore, Max will pay the train ticket on board to the controller agent.

3 THE V3A MODEL

Our agent architecture (Fig. 1) consists of: a **KBase** (KB) partitioned according to the vowel dimensions (KBuser, KBagt, KBenv, KBint, KBorg), an **Argumentation State** (AS) and a personality (pers). The KBase is a repository of rules and assumptions possibly conflicting. The parameter related to a dimension can still be too large and needs to be decomposed in different facets (f1, ..., f7). We partition further the dimensions linked to the external world to distinguish perceptions and capabilities. For instance, KBenv contains the representation of the physical laws in the environment (KBenv1) and the observations/actions perceived/performed by the body (KBenv2). Similarly, KBint contains the protocols (KBint1) and the messages received/sent by the agent (KBint2). Facets joint (respectively leave) the game for adding (respectively deleting) particular aspects of the whole possible agenda of the agent, which taken together will comprise every aspects that the agent can address.

![Figure 1: The V3A agent architecture.](image-url)

The interaction between facets is organized as an argumentation game, consisting of dialogues about statements. The dialogue is regulated by procedural rules. AS is a shared memory structure. Its current state includes a (partial) argument not yet defeated or subsumed. At the end of the game, AS contains the action(s) to perform. That is, each facet argues for or against motivations. The facet objectives consist of promoting (or demoting) the goals to reach, the actions to performed (or not) and the beliefs to grant (or not) in order to actively achieve some aspects of the agent’s agenda or avoid some situations that would be detrimental. In addition, facets monitor the statements proposed by other ones in AS to determine whether these statements interact with their own agenda. Essentially each facet can have an opinion of what is best for the agent as a whole, but from its limited viewpoint. A facet must argue its case against/with other possibly com-
peting/completing views for this to become incorporated into the agent overt behavior. This behavior is determined by the personality that specifies how to give priority to the facets and arbitrate amongst them to resolve the possible conflicts.

4 ARGUMENTATION FRAMEWORK

Our argumentation framework (AF) is based on the opposition calculus of (Dung, 1995), where arguments are reasons which can be defeated by other arguments.

Definition 1 (AF). An argumentation framework is a pair \(AF = (A, \text{defeats})\) where \(A\) is a finite set of arguments and \(\text{defeats}\) is a binary relation over \(A\). We say that an argument \(b\) defeats an argument \(a\) if \((b, a) \in \text{defeats}\). Additionally, we say that a set of arguments \(S\) defeats \(a\) if \((b, a) \in \text{defeats}\) and \(b \in S\).

(Dung, 1995) analysis if a set of arguments is collectively justified.

Definition 2 (Semantics). A set of arguments \(S \subseteq A\) is:

- conflict-free if \(\forall a, b \in S\) it is not the case that \(a\) defeats \(b\);
- admissible if \(S\) is conflict-free and \(S\) defeats every argument \(a\) which defeats some arguments in \(S\).

Amongst the semantics proposed in (Dung, 1995), we restrict ourself to the admissible one. We use AF to model the reasoning within the V3A agent architecture.

Definition 3 (KF). A knowledge representation framework is a tuple \(KF = (L, \text{Asm}, I, T, P)\) where:

- \(L\) is a formal language consisting of a finite set of sentences, called the representation language;
- \(\text{Asm}\) is a set of atoms in \(L\), called assumptions, which are taken for granted;
- \(I\) is a binary relation over atoms in \(L\), called the incompatibility relation, which is asymmetric;
- \(T\) is a finite set of rules built upon \(L\), called the theory;
- \(P \subseteq T \times T\) is a transitive, irreflexive and asymmetric relation over \(T\), called the priority relation.

\(L\) admits strong negation (classical negation) and weak negation (negation as failure). A strong literal is an atomic first-order formula, possible preceded by strong negation \(\neg\). A weak literal is a literal of the form \(\sim L\), where \(L\) is strong.

We adopt an assumption-based argumentation approach (Dung et al., 2007) to reason about beliefs, goals, decisions and priorities (Morge and Mancarella, 2007). That is, agents can reason under uncertainty. Actually, certain literals are assumable, meaning that they can be assumed to hold in the KB as long as there is no evidence to the contrary. Decisions (e.g. request(psa,sa,ticket)) as well as some beliefs (e.g. \(\sim\) strike) are assumable literals. The incomparability relation captures conflicts. We have \(L \sim L\), \(\sim L \bot L\) and \(L \sim L\). It is not the case that \(L \sim L\). For instance, paycom(psa,sa,price) \(\sim\) paycom(psa,sa,price) and \(\sim\) paycom(psa,sa,price)

We say that two sets of sentences \(\Phi_1\) and \(\Phi_2\) are incompatible (\(\Phi_1 \sim \Phi_2\)) iff there is at least one sentence \(\phi_1\) in \(\Phi_1\) and one sentence \(\phi_2\) in \(\Phi_2\) such that \(\phi_1 \sim \phi_2\).

A theory is a collection of rules with priorities over them.

Definition 4 (Theory). A theory \(T\) is an extended logic program, i.e a finite set of rules s.t. \(R: L_0 \leftarrow L_1, \ldots, L_{j-1}, \sim L_j, \ldots, \sim L_n\) with \(n \geq 0\), each \(L_j\) being a strong literal in \(L\). The literal \(L_0\), called head of the rule (denoted \(\text{head}(R)\)), is a statement. The finite set \(\{L_1, \ldots, L_n\}\), called body of the rule, is denoted \(\text{body}(R)\). \(R\), called name of the rule, is an atom in \(L\).

All variables occurring in a rule are implicitly universally quantified over the whole rule. A rule with variables is a scheme standing for all its ground instances.

For simplicity, we will assume that the names of rules are neither in the bodies nor in the head of the rules thus avoiding self-reference problems. We consider the priority relation \(P\) on the rules in \(T\), which is transitive, irreflexive and asymmetric. If \(R_1, PR_2\) can be read “\(R_1\) has priority over \(R_2\)”. There is no priority between \(R_1\) and \(R_2\), either because \(R_1\) and \(R_2\) are ex equo, or because \(R_1\) and \(R_2\) are not comparable. For instance, \(\text{user}(f_1,x_1) PR_2 \text{goal}(f_2,x_2)\) means that the rules in \(KB_{user}\) have priority over the rules in \(KB_{agt}\).

The KBases and the personality of the PSA in Pisa are depicted in Tab. 1.

Users KB. The user desires to reach the next travel step without paying any extra commissions.

Agents KB. The PSA knows that the SA can deliver tickets after requesting him. Contrary to Italy, utilising the services of a SA in UK requires to pay an extra commission. The cost of tickets also depends on the context.

Environments KB. These facets work on percepts
e.g. the time and the location of the PSA (cf. env(f4,a2)) and beliefs about the instant location, e.g. the time schedule (cf env(f3,r1)).

Interactions KB. These facets also work on percepts e.g. messages which will be received/sent and beliefs about protocols which depends on the context e.g. int(f6,r1(a1d1,a1d2,ticket,price,comm)).

Organisations KB. The PSA must pay a ticket to confirm its reservation.

Personality. Since the PSA has been set up by the constructor to respect the user’s preferences and assist him, the facets related to the user are preferred than those that are related to other agents. In order to utilise the computing system, the PSA prefers the organisation facet rather than the ones which are related to the environment. Contrary to other components, the personality is not embodied by facets but these preferences are encoded in the procedural rules described informally in Section 5.

In this scenario, self-adaptation is crucial since, when Max is arriving in a new location, new KBases can be downloaded and they replace the previous one. It is worth noticing that the users KB and the personality are not modified.

Due to the abductive nature of proactive agent reasoning, arguments are built by reasoning backward.

Definition 5 (Argument). An argument $a$ for a statement $\alpha \in L$ (denoted $\text{conc}(a)$) is a deduction of that conclusion whose premise is a set of rules (denoted $\text{rules}(a)$) and assumptions (denoted $\text{asm}(a)$) of $KF$.

The top-level link of $a$ (denoted $\top(a)$) is a rule s.t. its head is $\text{conc}(a)$.

The sentences of $a$ (denoted $\text{sent}(a)$) is the set of literals in $L$ in the bodies/heads of the rules including the assumptions of $a$.

In Pisa, the argument $a$ concludes motive since the PSA does not register. This argument is defined.t.

rules(a) = \{user(f1,r1), env(f3,r1), env(f4,a1), env(f4,a2), env(f3,r2(lipifi))\},

$\top(a) = \text{user(f1,r1)}$, and

$\text{asm}(a) = \{\sim \text{strike}, \text{register(lipifi,no)}, \text{be(pisa,17)}\}$.

By contrast, the argument $b$ concludes motive if the PSA request a ticket, pays it and registers. This argument is defined s.t.

rules(b) = \{user(f1,r1), env(f3,r1), env(f3,r2(lipifi)), org(f7,r1(lipifi,ticket,price,comm)), agt(f2,r1(ticket)), int(f6,r1(sa,psa,ticket,price,comm))\}

$\top(b) = \text{user(f1,r1)}$, and

$\text{asm}(b) = \{\sim \text{strike}, \text{be(pisa,17)}, \text{pay(psa,sa,4,0)}, \text{request(psa,sa,ticket)}\}$. After self-adaptation, similar arguments exist in Stansted.

We define here the defeat relation. Firstly, we define the attack relation to deal with conflicting arguments.

Definition 6 (Attacks). Let $a$ and $b$ be two arguments. $a$ attacks $b$ iff $\text{sent}(a) \sqsubseteq \text{sent}(b)$.

This relation encompasses both the rebuttal attack due to the incompatibility of conclusions, and the undermining attack, i.e. directed to a “subconclusion”. The strength of arguments depends on the priority of their sentences. In order to give a criterion that will allow to prefer one argument over another, we consider here the last link principle to promote high-level goals.

Definition 7 (Strength). Let $a$ and $b$ be two arguments. $a$ is stronger than $b$ (denoted $\text{prior}(a,b)$) if it is the case that $\top(a) \sqsubseteq \top(b)$.

The two previous relations can be combined.

Definition 8 (Defeats). Let $a$ and $b$ be two arguments. $a$ defeats $b$ iff: i) $a$ attacks $b$; ii) it is not the case that $\text{prior}(b,a)$.

In Pisa, $\{b\}$ is in an admissible set since the organisation has priority over the environment. This argument describes the motivation for registering. After the self-adaptation of the PSA in London, even if a new travel connection is considered, this argument is no more admissible since an extra commission is required. Due to the agent personality, argument $a$ is re-instantiated and the PSA does not register.

5 DIALOGUE-GAME

The result of the debate amongst facets is an argument sketched in $AS$. We consider here the procedural rules which regulate the exchanges of moves to reach an agreement. For this purpose, we instantiate a dialectical framework (Prakken, 2006).

Definition 9 (Dialectical Framework). Let us consider the topic, i.e. a statement in $L$, and $\emptyset \subseteq L$ a facet communication language. The dialectical framework is a tuple $DF(topic,KF) = (P, AS, \Omega_M, H, T, proto, Z)$ where:

- $P = \{p_1,\ldots,p_n\}$ is a set of $n$ players;
- $AS = \langle\text{Pro,Opp}\rangle$ is composed of two boards $\text{Pro}$ and $\text{Opp}$ which contains the literals held by the proponents and the opponents respectively;
- $\Omega_M \subseteq \emptyset \subseteq L$ is a set of well-formed moves;
- $H$ is a set of histories, the sequences of well-formed moves s.t. the speaker of a move is determined at each stage by the turn-taking function $T$ and the moves agree with the protocol $proto$;
Table 1: The KBases of the PSA in Pisa.

<table>
<thead>
<tr>
<th>Function</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>register_ticket</code></td>
<td><code>train</code></td>
<td><code>pay</code></td>
</tr>
<tr>
<td><code>train</code></td>
<td><code>pay</code></td>
<td><code>request</code></td>
</tr>
</tbody>
</table>

- `T`: $H \rightarrow P$ is the turn-taking function;
- `proto`: $H \times AS \rightarrow \Omega_M$ is the function determining the moves which are allowed to expand an history;
- `z` is the set of dialogues, i.e. the terminal histories where the proponent (respectively opponent) board is a sumable literals (respectively empty).

In the V3A architecture, the DF allows multi-party dialogues amongst facets (the players) about motive (the topic) within KF. Players claim literals during dialogues. Each dialogue is a maximally long sequence of moves. We call line the sub-sequence of moves where backtracking is ignored. Amongst players, the proponents argue for an initial claim while the opponents argue against it.

We define here the syntax and the semantics of moves. The `syntax` of moves in is in conformance with a common facet communication language, $\mathcal{F} \subset \mathcal{L}$. A move at time $t$ has an identifier, $mv_t$ is uttered by a speaker $(sp_t \in P)$; eventually $rp_t$ is the identifier of the message to which $mv_t$ responds and the speech act is composed of a location $loc_t$ and a content $content_t$. The locutions are claim, concede, oppose, deny, and unknown. The content is a set of atoms in $L$.

The `semantics` of speech acts is public since all players the same meaning to the moves. The semantics is defined in terms of pre/post-conditions.

**Definition 10 (Semantics of $\mathcal{F} \subset \mathcal{L}$).** Let be the time of a history $h$ in $H$ ($0 < t < |h|$). $AS_0 = (\{topic\}, \emptyset)$. The semantics of the utterance at the facet $f$ at time $t$ is defined s.t.:

1. $f$ may utter `unknown($\emptyset$)` and so $AS_{t+1} = AS_t$.
2. Considering $L \in P \cap \Omega$,
   
   (a) $f$ may utter `claim($P$)`, if $\exists r \in \mathcal{F}_f \text{ head}(r) = L$, $\text{ body}(r) = P$, $P \cap P_{op} = \emptyset$, and it is not the case that $P \cap \Omega = \emptyset$. Therefore, $AS_{t+1} = \langle P_{op}, \Omega \rangle$.

   (b) $f$ may utter `concede($\{L\}$)`, if $L \in Asm_f$. Therefore, $AS_{t+1} = AS_t$.

   (c) $f$ may utter `oppose($\{L'\}$)`, if $L' \in L$. Therefore, $AS_{t+1} = \langle P_{op}, \Omega \rangle$.

3. Considering $L \in \Omega$,
   
   (a) $f$ may utter `claim($P$)`, if $\exists r \in \mathcal{F}_f \text{ head}(r) = L$, $\text{ body}(r) = P$ and $P \cap \Omega = \emptyset$. Therefore, $AS_{t+1} = \langle P_{op}, \Omega \rangle$.

   (b) $f$ may utter `deny($\{L'\}$)`, if $L' \in L$. Therefore, $AS_{t+1} = \langle P_{op}, \Omega \rangle$.

The rules to update $AS$ incorporates a filtering (in case 2a and 3a) to be more efficient. Concretely, the set of literals in $P_{op}$ and $\Omega$ are filtered, so they are not repeated more than once, and finally the literals in $P_{op}$ are not incompatible with each other. The speech act `unknown($\emptyset$)` has no preconditions. Neither concessions nor pleases of ignorance have effect on $AS$.

In order to be uttered, a move must be well-formed. The initial moves are initial claims and pleases of ignorance: $mv_0 \in \Omega$ iff $\text{ loc}(mv_0) = \text{ claim}$ or $\text{ loc}(mv_0) = \text{ unknown}$. The replaying moves are well-formed iff they refer to an earlier move: $mv_j \in \Omega$ iff $rp_j = mv_i$ with $0 < i < j$. Notice that backtracking is allowed. Each dialogue is a sequence $h = (mv_0, ..., mv_{|h|-1})$ with $\text{ proto}(h) = \emptyset$. In this way, the set 2 of dialogues is a set of maximally long histories, i.e. which cannot be expanded even if backtracking is allowed.

The turn-taking function $T$ determines the speaker of each move. If $h \in \mathcal{H}$, $sp_f = f$, and $j - i = |h| \text{ mod } n$, then $T(h) = f$.

The protocol `proto` consists of a set of sequence rules (e.g. $sr_1, ..., sr_4$ represented in Tab. 2) specifying the legal replaying moves. For example, $sr_1$ specifies the legal moves replying to a previous claim (claim($P$)). The speech acts resist or surrender, i.e. close the line. Players resist as much as possible. The locutions concede and unknown are utilised to manage the sequence of moves since they surrender, and so close the line but not necessarily the dialogue (backtracking is allowed). By contrast, a claim (claim($P'$)) and an opposition (oppose($\{L'\}$)) resist to the previous claim. The moves replying to a deny (deny($\{L'\}$)) are the same as the replaying move of a
ARGUING OVER MOTIVATIONS WITHIN THE V3A-ARCHITECTURE FOR SELF-ADAPTATION

6 CONCLUSIONS

We have proposed a dialectical argumentation framework allowing an agent to argue with itself about its motivations. The framework relies upon the admissibility semantics and uses an assumption-based argumentation approach to support reasoning about the knowledge, goals, and decisions held in the agent's mental facets. These modules interact via a dialogue-game which is formally defined and exemplified via a concrete scenario. The contribution of the work is a modular model that allows the facets and the personality of an agent to be specified declaratively, manages potential conflicts and replaces components at runtime, thus avoiding to restart the agent’s reasoning process whenever a component joins or leaves the game.

Some of the concepts utilized here have been introduced in the AAA model (Witkowski and Stathis, 2004). However, here we provide a formal definition of the argumentation game that the original AAA model abstracted away from. We have also reinterpreted the original model by using the Vowels approach, which has an agent-oriented software engineering foundation.

Our work is also related to the KGP (Kakas et al., 2004b) model of agency and in particular the modelling of the personality of the agent (Kakas et al., 2004a) through preferences. One important difference with (Kakas et al., 2004a) comes from our decomposition which distinguishes explicitly the different aspects, possibly conflicting, that the agent must arbitrate. These aspects are embodied by faculties that are more amenable to be plug-and-play components at run-time using a multi-threaded implementation.

Future work includes investigating the properties of different dialogue-games for different semantics and properties. We also plan to extend the current prototype using CaSAPI 1 to allow an internal dialectic that is multi-threaded and relies on facets that are interpreted by different proof systems implementing different kinds of reasoning such as epistemic reasoning, practical reasoning and normative reasoning.

ACKNOWLEDGEMENTS

This work is supported by the Sixth Framework IST 035200 ARGUGRID project.

REFERENCES


1 http://www.doc.ic.ac.uk/~dg00/casapi.html