Argumentative KGP agents for Service Composition

Francesca Toni
Department of Computing, Imperial College London, UK
ft@doc.ic.ac.uk

Abstract

We sketch a variant of the KGP agent model tailored to support service composition in service-oriented architectures. This allows to handle: user requirements, abstract and concrete workflows, possibly with annotations, inter-agent communication for supporting negotiation and persuasion, consultation of registries. We identify argumentation as core for Service-Oriented Architectures, and ground our work in the context of three concrete scenarios: e-procurement, earth observation, and business migration.

Introduction

In recent years, under the influence of the World Wide Web, many standalone software tools have evolved into locally managed but globally accessible applications within distributed environments, such as the World Wide Web, the Grid and Service-Oriented Architectures. Users are then posed the problem of selecting and/or composing these applications into more complex ones, in order to fulfil specified users' requirements.

The use of agent technology offers a solution to dynamic service composition in distributed settings such as the Grid (Foster, Jennings, & Kesselman 2004) and more generally Service-Oriented Architectures. Different services can be associated with autonomous agents that can identify and negotiate, on behalf of service requestors and providers, implementation plans that take into account the requirements of both sides.

The ARGUGRID project [1] aims at defining and deploying argumentation-based agents to support the selection and composition of services over the Grid and Service-Oriented Architectures. In (Morge *et al.* 2007a; 2007b) we have proposed an agent architecture integrating a number of argumentative modules (for various forms of decision-making), a module for interaction with other agents, "physical" modules for carrying out this interaction via communication, and several data structures. In this paper we outline a rational reconstruction of this architecture within a generalisation of the KGP agent model (Kakas *et al.* 2004). The KGP model is a general-purpose model grounded in computational logic.

[1] www.argugrid.eu

KGP agents are equipped with an internal state (with a transient knowledge base recording past actions and observations, as well as goals and actions currently planned by the agent), a catalogue of reasoning and physical capabilities used to define transitions between states, and a high-level control for determining how these transitions are streamlined and ordered. The reasoning capabilities and the control are defined within extensions of computational logic, and the control if fully flexible and dynamic. The agents we define in this paper follow the same general philosophy of state, capabilities, transitions and control, but

- are equipped with a specialised internal state, consisting of requirements, abstract or partially instantiated workflows, concrete workflows, planned communicative actions and past such actions executed by the agent or by others, and arguments
- adopt a specialised set of reasoning capabilities, to support the various forms of decision-making needed in ARGUGRID and inter-agent interaction, as well as a capability for revising the knowledge/beliefs of agents (missing in the original KGP model)
- adopt specialised physical capabilities
- adopt appropriate transitions encapsulating the new capabilities.

In this paper, we outline this specialised, argumentative KGP agent model, and illustrate it in the context of a number of simplified scenarios adapted from the ARGUGRID ones.

The paper is organised as follows. We first outline the general setting and the simplified scenarios, and argue in favour of argumentation for realising these scenarios. Then, we summarise the original KGP model. Further, we outline the proposed argumentative KGP agent model and illustrate this model for the scenarios. We finally conclude.

Scenarios and general setting

We consider the problem of service selection and composition in the context of the scenarios for the ARGUGRID project, namely e-procurement, earth observation, and business migration. Here we discuss some simplified use cases for these scenarios (see (Stournaras *et al.* 2007) for a full description).

e-procurement. We consider here a simple case whereby a user (e.g. a company or a manufacturer) needs an e-ordering system allowing it to decrease its purchasing costs. An e-ordering system can be seen simply as consisting of a computer program and an internet provider. For the e-ordering system to decrease purchasing costs, the computer program needs to have some features, e.g. allow for a self-service supplier catalogue and have a flat cost for 3-years (or more), or allow for out-of-catalogue requests and multiple product categorisations.

Earth observation. We consider here a simple case where a user (e.g. an organisation) needs to monitor a fire over a large region, while keeping costs low. This will require information from a weather forecast service provider and then the choice of at least two satellites (and sensors) providing images for sub-regions of the region on fire. If the weather is forecast to be good, then (cheaper) optical sensors satellites will suffice, otherwise (more expensive) satellites with radar sensors will be needed.

Business migration. We consider here a simple case where a user (potential investor) wants to invest in the construction of a factory in some oversea location in order to decrease costs. This can be reduced to the problem of finding a location that is easily accessible and identify a good logistic plan for the construction of the factory. A location may be deemed easily accessible if it is by the sea or within 50 km of a river that is sufficiently deep. A good logistic plan may amount to finding a reasonable construction contractor and some local suppliers (e.g. of rubber soles if the factory aims at producing shoes).

General setting. All these simple cases can be supported by a multi-agent system with a user agent (the service requestor), supporting the user, and a number of agents supporting service providers (e.g. suppliers of computer programs and internet providers for e-procurement, suppliers of satellite images and weather forecast for earth observation, and suppliers of information about locations, construction contractors and local suppliers for business migration).

We provide a model for these agents. We will distinguish between *requestor agents* (*requestors* in short), e.g. the user agent in all earlier scenarios, and *provider agents* (*providers* in short), e.g. suppliers. Naturally, some agent could act in both roles, e.g. in e-procurement a provider of computer programs may act as a requestor for an internet provider if it wants to form a competitive offer to submit to the user. Agents act on behalf of users (requesting services) and providers (offering services). Agents needs to fulfil some requirements by users and providers. [2] These requirements are the *goals* of the agents. Agents need to negotiate a composition of services fulfilling the requirements (or some variation of these requirements, if persuaded to do so). Taking a service-oriented perspective, we will refer to uninstantiated or partially instantiated combinations as *abstract workflows*, and to fully instantiated combinations as *concrete workflows*. Both kinds of workflows may be annotated (by constraints and features of the combination and its components). For the purposes of this papers, we will assume that the internal representation of annotated workflows is simply in terms of conjunctions of literals in some given logical language. [3] Examples of the kind of annotated abstract workflows we use are [4]

- for e-procurement:
  $computer\_program(S_1) \wedge internet\_provider(S_2) \wedge decrease\_cost(+(S_1, S_2))$
- for earth observation:
  $weather(station, W) \wedge satellite(W, S_1) \wedge satellite(W, S_2) \wedge S_1 \neq S_2$
  where $station$ is some concrete forecasting station
- for business migration:
  $easily\_accessible(L) \wedge contractor(S_1, L) \wedge supplier(S_2, L)$

Examples of annotated concrete workflows are

- for e-procurement:
  $computer\_program(abc) \wedge internet\_provider(wind) \wedge payment(abc) = 30K \wedge payment(wind) = 20K$
  where abc and $wind$ are concrete providers
- for earth observation:
  $satellite(meteosat) \wedge satellite(JERS\_1) \wedge hourly(meteosat)$
  where $meteosat$ and $JERS\_1$ are two concrete satellites and $meteosat$ is required to provide an image every hour
- for business migration:
  $contractor(zz, hanoi) \wedge supplier(4z, hanoi)$
  where zz and $4z$ are concrete service providers in Hanoi.

Agents may get information about providers from registries and/or other agents. This information will be stored within the agents' *knowledge*.

We will see that abstract workflows are the result of the agents' own decision-making and concrete workflows are the result of the negotiation between agents. Concrete workflows (and their annotation) are used to give rise to *contracts*. In this paper, contracts are simply considered as the provision of a service (or a collection of services) from a provider to a requestor with an associated set of attributes (e.g. price

[2] We have omitted to exemplify requirements by providers in the earlier scenarios, and we will mostly focus on the view point of the requestors throughout the paper for simplicity.

[3] In the literature, workflows are often seen as complex compositions, e.g. by sequencing or parallel execution, of services, and are expressed within some standard notattion, e.g. DPML, possibly combined with a language for annotations such as WSMO. Here, we adopt a much simpler definition in line with the notion of workflow adopted within ARGUGRID (where the complexity of workflows arises from the workflow execution model). Also, note that services and their features typically needs to be expressed within some ontology that can be reasoned upon, e.g. using some Semantic Web technique such as OWL-S. Here, we ignore this issue for simplicity and assume a representation of ontologies within the given logical language.

[4] Below, we adopt the convention that variables start with capital letters and are implicitly existentially quantified with scope the formula where they occur, and constants start with lower-case letters or digits.

and time of delivery). Informally, for any given agent, a contract consists of the elements, that are involving the agent directly, of the agreed concrete workflow amongst all agents. For example, in the case of the simple concrete workflow given earlier:

$computer\_program(abc) \wedge internet\_provider(wind) \wedge payment(abc) = 30K \wedge payment(wind) = 20K$

the contract between the user and abc will amount to: abc providing the agreed computer program to the user for a total payment (by the user) of $30K$, and the contract between the user and $wind$ will amount to: $wind$ serving as the internet provider after a payment of $20K$.

The case for argumentation

There are several kinds of decisions that requestor agents need to make in this scenario:

- which composition of services would allow to meet its given goals, and under which constraints, which in turn may require consulting registries of available service types
- which service provider can provide these services so that the constraints are fulfilled, which in turn amounts to deciding
  - whether to consult a registry of providers, and which one
  - which provider to ask
  - which (communication) protocol/policy to use for asking
  - which offers by providers to accept.

Providers also need to make a number of decisions:

- fulfilling which request would allow it to meet its given goals, and under which constraints
- which protocol/policy to use for replying to requests.

If providers are also allowed to proactively form teams, e.g. as in e-procurement settings, then other decisions they need to make are:

- which other provider to ask, if needed
- whether to consult a registry of providers, and which one
- which protocol/policy to use for asking other providers
- which provider to accept for joining forces with.

All these decisions need to be made given information that may be partial and inconsistent (for example because this information arises from multiple sources, or because the agents use defeasible rules and facts). For example, in the e-procurement case, the user agent may have information from some softare house $kappa$ that they will provide a flat-cost for 3-years but it may have information from some other source that $kappa$ has defaulted on similar arrangements on previous occasions.

Argumentation has been long recognised as a powerful mechanism for decision making in the presence of inconsistent and partial information, and several argumentative decision making techniques exist, e.g. (Kakas & Moraitis 2003; Rahwan & Amgoud 2006; Morge & Mancarella 2007; Atkinson & Bench-Capon 2007; Toni 2007). Using argumentation, alternative decisions can be evaluated against one another, and the reasons in favour and against these decisions can be compared and contrasted. For example, in deciding whether to accept an offer by the software house $kappa$, the user agent may consider the pro-argument that is provides a flat-cost for 3-years, supported by the information provided by $kappa$, and the con-argument that it will not, supported by the information provided by other sources. This con-argument may be rebutted by further information about the unreliability of the source.

Argumentation can also be useful for explaining and justifying decisions, as may be needed to increase the success rate of our agents in finding compositions of services meeting all agents' goals. Indeed, if agents exchange arguments, they may be able to find more quickly mutually agreeable solutions where none existed prior to the exchange. For example, if a requestor agent explains the reasons for refusing an offer by a consortium, motivated by the presence of a disagreable provider in it, then the consortium may replace that provider by another one that is agreable to the requestor. Without this information the consortium may just dismantle and no solution may be found. So, argumentation can also play a role in interaction protocols/policies adopted by the agents. These protocols/policies will need to make use of argumentative decision making mechanisms.

KGP agents

KGP agents (Kakas *et al.* 2004) are equipped with the following components:

- an *internal (or mental) state*, consisting of the agent's *knowledge* (beliefs), *goals* and *plans*; goals and actions in plans have associated times and temporal constraints, inducing a partial order; goals and plans are organised within a tree structure, linking actions and sub-goals to goals they contribute to achieving; beliefs are structured within a number of bases, supporting the various reasoning capabilities, and also includes a dynamic part, KB_0, updated when the agent observes the environment and executes actions in plans (via the physical capabilities)
- *reasoning capabilities*, including planning, reactivity, temporal reasoning, goal decision, and temporal constraint satisfiability
- *physical capabilities*, allowing agents to sense their environment and act upon it
- *transition rules*, changing the agent's state and including PI (plan Introduction), RE (reactivity), GI (Goal Introduction), SI (sensing Introduction), OI (passive or active Observation Introduction), AE (Action Execution), SR (State Revision, performing menial revision tasks over the state); the transition rules are defined in terms of the capabilities, and their effect is dependent on the concrete time of their application
- a set of *selection functions* to select inputs to transitions from the state

- a *control*, for deciding which enabled transition should be next, based on the selection functions, the current time, and the previous transition.

The application of a transition T at time τ, mapping state S onto S' given (a possibly empty) input X, is represented as $T(S, X, S', \tau)$. It is assumed for simplicity that the application of transitions is instantaneous, namely τ is also the time when S' is generated.

The control of the agent is responsible for its behaviour, in that it induces an *operational trace*, namely a (typically infinite) sequence of transitions

$T_1(S_0, X_1, S_1, \tau_1), \ldots, T_i(S_{i-1}, X_i, S_i, \tau_i), \ldots$

such that S_0 is the given initial state, and for each $i \geq 1$, $\tau_i < \tau_{i+i}$, (namely time increases).

In the KGP model, goals and actions are timed literals, with all time variables being existentially quantified within the agent's state. Actions may be "physical", communicative or sensing, and fluents may be "mental" (to be brought about by plans) or sensing (to be observed).

The operational traces are not fixed a priori, as in conventional agent architectures, but are determined dynamically by reasoning with declarative cycle theories, giving a form of flexible control whereby the decision as to which transition should be applied depend on a number of conditions on the current state of the agent and preferences this has.

Argumentive KGP agents

Agents only need to be able to perform communicative actions (for requesting services, accepting or refusing the provision of services, etc) and actions for consulting registries, inquiring about services and their providers. In their internal state, agents store (a selection of) all communicative acts they have participated in, as either speakers or receivers, as well as the set of their current commitments, namely the contracts they have committed to.

An agent is characterised by

- a (transient) state, consisting of
  - a *knowledge base*, called KB_0 as for the KGP model, but holding communicative acts by or to the agent, acts for consulting registries by the agent, as well as contracts [5]
  - a set of *goals*, namely requirements by the user "owning" the agent
  - a set of *decisions*, of different kinds (to get services of known types from some yet-to-be-decided providers or from some known providers, or a decision to utter something, or a decision to consult some registry)
  - a set of arguments, providing justifications and reasons for goals and decisions in the state
- a number of reasoning capabilities, namely *abstract decision-making*, *social decision-making*, *communicative reactivity*, *registry consultation*; each capability is supported by an appropriate argumentation system (base)
- a *revision* capability, for modifying the argumentation systems supporting the various reasoning capabilities
- *physical capabilities*, namely *listening*, *talking*, and *consulting*
- a set of transitions, namely ADM (using the abstract decision-making capability), SDM (using the social decision-making capability), CR (using the communicative reactivity capability), RC (using the registry consultation capability), R (using the revision capability), Li, Ta, Con (using the listening, talking and consulting capabilities, respectively)
- a control, in the form of a conditional policy, that, for each transition, gives one or more possible next transitions depending on whether a number of conditions hold or not.

Here, the consulting capability is intended for accessing information in registries. The reasoning capabilities correspond to the IDM (individual decision-making), social-decision making (SDM), and social interaction (SI) modules in (Morge *et al.* 2007a; 2007b). The listening and talking capabilities are special cases of sensing and actuating in the KGP model.

The operational trace is given by applying transitions according to the control, as in the KGP model.

Communicative Actions. We assume a *communicative language*, namely a set of utterances $\mathcal{U}$, shared amongst agents and allowing them to communicate with one another. Communicative actions are actions in this communicative language. We assume that utterances in $\mathcal{U}$ are of the form $U(S, R, C, T)$ where U is the type of the utterance, S is the speaker of the utterance, R is the receiver, C is the content, and T is the time of the utterance. For example, U may be $inform$ or $request$. Thus, the specification of utterances in $\mathcal{U}$ requires

- a vocabulary of *agent names*, $\mathcal{A}$ (e.g. any finite set)
- a language for expressing content, $\mathcal{C}$
- a timeline representation, $\mathcal{T}$ (e.g. the set of all natural numbers).

The vocabulary $\mathcal{A}$ can change over time, as the agent becomes aware of new agents by consulting registries or by being told by other agents it knows of. The syntax of the content of each utterance will depend on U. For example, if U is $inform$ then C may be a single sentence; if U is a $request$ then C may be a pair, consisting of a resource/service being requested and a set of constraints on the resource/service. However, all components of C will be expressed in the same language $\mathcal{C}$. We assume that all agents adopt the same $\mathcal{C}$, serving as a lingua franca. [6] We also assume that $\mathcal{U}$ includes all utterances $U(S, R, C, T)$ such that $U(C)$ consists of:

- $request(S, P)$ for requesting a service S with properties in P (here and below, this may be empty)

[5] These could be re-constructed from the communicative acts, but it is useful for the agent to be able to represent them explicitly.

[6] Note that our communicative actions are intended to be abstractions of utterances as represented in standard agent communication languages, e.g. FIPA ACL.

- $why(request(S, P))$ for asking reasons for requesting a service S with properties in P
- $because(request(S, P), E)$ for giving reasons E for requesting a service S with properties in P
- $accept(request(S, P))$ for accepting to provide a service S with properties in P
- $refusing(request(S, P))$ for refusing to provide a service S with all properties in P
- $why(refuse(request(S, P)))$ for asking explanations as to the reasons for refusing to provide a service S with properties in P
- $because(refuse(request(S, P)), E)$ for giving explanations E as to the reasons for refusing to provide a service S with properties in P

Actions for consulting registries. We assume a *registry query language*, for modelling all actions of this type, consisting of queries of the form $consult(A, R, Q)$ where $A \in \mathcal{A}$, $R \in \mathcal{R}$ for some given finite vocabulary $\mathcal{R}$ of registry names, and $Q \in \mathcal{Q}$ for some given language $\mathcal{Q}$ for representing queries to registries. For example, an action in this language may be

$$consult(bob, r_{35}, \exists X[constructor(X, hanoi)]).$$

Reasoning capabilities. For each of the reasoning capabilities c (this is one of abs for abstract decision-making, soc for social-decision-making, com for communicative reactivity and reg for registry consultation), we assume a logic-based *language* $\mathcal{L}_c$ with subsets:

- a set of *goal* sentences $\mathcal{G}_c$
- a set of *decision* sentences $\mathcal{D}_c$

These $\mathcal{L}_c$s are internal to the agent.

In the case of abstract decision-making, the goals are user requirements and the decisions are given by annotated abstract workflows. In the case of social-decision-making, the goals are annotated abstract workflows and the decisions are annotated concrete workflows. In the case of communicative reactivity, the decisions are specific communicative actions and the goals may be to have a particular attitude towards some agents (e.g. to be "nice" to them so as to gain a better reputation). These decisions are the outcome of reasoning with protocols/policies to adopt as well as communicative actions that these protocols/polices allow (in case these protocols/policies are non-deterministic). An example of policy might be:

- when asked for a service (with some properties) the agent cannot provide, it should always attempt to obtain an explanation (uttering a why) before refusing (uttering a $refuse$), unless the explanation has already been asked and obtained without being useful
- when asked for an explanation (for a request or refusal) the agent should always truthfully provide it (uttering a $because$)

In the case of registry consultation, the decisions are specific registry consultation actions and the goals may be preferences between different registries.

Each capability c is defined using some argumentative decision making tool aDM_c. As discussed earlier, there are several such tools in the literature: here, we do not commit to any concrete approach, and keep aDM_c completely abstract. We assume that aDM_c, given a goal in $\mathcal{G}_c$, returns a single decision in $\mathcal{D}_c$ and the reasons supporting it. These reasons are a set of arguments that, for simplicity, we can see as abstract, as in (Dung 1995) (but we need to assume here implicitly a function linking each set of arguments to an objective or set of objectives it fulfils). We will assume that the reasons are "acceptable" in some sense, following some semantics of argumentation (Dung 1995). Concretely, arguments may be drawn from any concrete argumentation framework, e.g. (Garcia & Simari 2004; Bondarenko *et al.* 1997). Decisions may take preferences of various kinds into account, but we ignore this issue in this paper.

Revision capability. This capability amounts to deciding, when receiving a communication act including an argument, whether to modify or not the appropriate set of arguments (in one of the underlying argumentation systems supporting the reasoning capabilities). This capability may involve intervention by the user/agent's owner and/or reasoning about the reputation of the speaker of the argument. In turn, reasoning about this reputation may require argumentation (Bentahar *et al.* 2007).

Physical capabilities. As in the KGP model, these capabilities are simply "wrappers" for the agent's sensors. In (Morge *et al.* 2007a; 2007b), they belong to the CM (Communication Module). We assume that the consulting capability is used to instantiate the variables of actions for consulting registries, in a syncronous manner.

State. KB_0 is a set of ground communicative actions and actions for consulting registries. It also contains contracts drawn from the communicative actions. For example, if the $user$ agent has uttered $accept(user, wind, internet\_provider(wind), 10)$ to the agent $wind$ at time 10, then a corresponding contract between the two agents will be in the state. Finally, the state contains a given set of goals and current decisions, drawn from the appropriate languages.

Transitions. These update the state as follows:

- $ADM(S, G, S', \tau)$: given $G \in S$ such that $G \in \mathcal{G}_{abs}$, if D is the outcome of aDM_{abs} for G, with argument A, then S' records the information that A is an argument for D given G;
- $SDM(S, G, S', \tau)$: similarly to ADM, but using sdm
- $CR(S, G, S', \tau)$: similarly to ADM, but using com
- $RC(S, G, S', \tau)$: similarly to ADM, but using reg
- $R(S, G, S', \tau)$: S' is updated according to the outcome of the revision capability

- $Li(S, G, S', \tau)$: S' is obtained by adding all incoming communicative acts "listened" to by the listening capability
- $Ta(S, G, S', \tau)$: S' is obtained by adding all outgoing communicative acts uttered by the agent by the talking capability
- $Con(S, G, S', \tau)$: the information gathered by the consulting capability (for executing registry consultation actions) is added to the argumentation systems for all reasoning capabilities.

The arguments used to extend the state can then be used during communication, to provide explanations in $because$ utterances.

Control. This decides which transitions to apply when and with which input. For example, if the abstract goal component of the state is empty, then the agent can only apply Li (as it has nothing to achieve). If the abstract goal is not empty, then it should apply ADM with that goal as an input, and then SDM with the resulting abstract decision as input. After Li, Ta should always be applied if the agent wants to be helpful towards other agents that have put forwards requests.

Conclusions

We have sketched a variant of the KGP model for modelling agents serving as requestors and/or providers of services in service-oriented architectures. This variant relies upon the use of an argumentation decision-making tool supporting all reasoning capabilities. It also needs to make use of argumentative protocols for persuasion in negotiation.

The paper describes ongoing work, with many issues that need addressing, including the provision of appropriate persuasion protocols, and whether we single out a single argumentative decision making tool supporting all reasoning capabilities. Also, we have adopted an informal "conditional-policy" formulation of the control: this will need to be formalised, either using cycle theories as in the KGP model or otherwise.

Several works exist on argumentation-based negotiation (Rahwan *et al.* 2003). For example, (van Veenen & Prakken 2006) propose a protocol and a communication language for dealing with refusals in negotiation. It would be useful to see how this protocol and communication language may be used to support service selection and composition. Also, (Amgoud, Dimopolous, & Moraitis 2007) presents an abstract negotiation framework whereby agents use abstract argumentation internally and with each other. The framework presented here can be seen to some extent as a specialisation of that approach, tailored to service composition.

Acknowledgements

The author has been supported by the Sixth Framework IST programme of the EC, under the ARGUGRID project.

References

Amgoud, L.; Dimopolous, Y.; and Moraitis, P. 2007. A unified and general framework for argumentation-based negotiation. In *Proc. AAMAS'2007*.

Atkinson, K., and Bench-Capon, T. 2007. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence* 171(10–15):855–874.

Bentahar, J.; Toni, F.; Meyer, J.-J. C.; and Labban, J. 2007. A security framework for agent-based systems. *International Journal of Web Information Systems* 3(4):341–362.

Bondarenko, A.; Dung, P.; Kowalski, R.; and Toni, F. 1997. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence* 93(1-2):63–101.

Dung, P. 1995. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77:321–357.

Foster, I.; Jennings, N.; and Kesselman, C. 2004. Brain meets brawn: why grid and agents need each other. In *Proc. AAMAS'2004*.

Garcia, A., and Simari, G. 2004. Defeasible logic programming: An argumentative approach. *Journal of Theory and Practice of Logic Prog.* 4(1-2):95–138.

Kakas, A., and Moraitis, P. 2003. Argumentation based decision making for autonomous agents. In *Proc. AAMAS'03*.

Kakas, A. C.; Mancarella, P.; Sadri, F.; Stathis, K.; and Toni, F. 2004. The KGP model of Agency. In *Proc. ECAI 2004*.

Morge, M., and Mancarella, P. 2007. The hedgehog and the fox: An argumentation-based decision support system. In *Proc. ArgMAS'2007*.

Morge, M.; Mancarella, P.; Toni, F.; McGinnis, J.; Bromuri, S.; and Stathis, K. 2007a. Toward a modular architecture of argumentative agents to compose services. In *Proc. JFSMA 2007*.

Morge, M.; McGinnis, J.; Bromuri, S.; Toni, F.; Mancarella, P.; and Stathis, K. 2007b. Toward a modular architecture of argumentative agents to compose services. In *Proc. EUMAS-2007*.

Rahwan, I., and Amgoud, L. 2006. An argumentation-based approach for practical reasoning. In *Proc. AAMAS'06*.

Rahwan, I.; Ramchurn, S.; Jennings, N.; McBurney, P.; Parsons, S.; and Sonenberg, L. 2003. Argumentation-based negotiation. *The Knowledge Engineering Review* 18(4):343 – 375.

Stournaras, T.; Dimitrelos, D.; Tabasco, A.; Barba, J.; Pedrazzani, D.; Yage, M.; An, T.; Dung, P.; Hung, N.; Khoi, V. D.; and Thang, P. M. 2007. e-business application scenarios. In Stournaras, T., ed., *ARGUGRID deliverable D.1.2*.

Toni, F. 2007. Assumption-based argumentation for selection and composition of services. In *Proc. CLIMA VIII*.

van Veenen, J., and Prakken, H. 2006. A protocol for arguing about rejections in negotiation. In *Proc. ArgMAS'2006*.