# La Logica Computazionale in sistemi basati su agenti
## *Computational Logic in Agent Based Systems*

Paolo Mancarella e Francesca Toni

## SOMMARIO/*ABSTRACT*

Viene descritto l'utilizzo della logica computazionale a supporto della formalizzazione ed implementazione di agenti in sistemi multi-agente. In questo ambito è necessario l'uso di varie forme di logica computazionale, tra le quali abduzione, argomentazione e sistemi basati su preferenze. Viene presentato a grandi linee il modello per agenti denominato KGP, nonché una sua estensione in corso di definizione per la modellazione di agenti in ambienti distribuiti quali il Grid e più in generale architetture *service-oriented*

*We describe recent work on the deployment of computational logic to support the formalisation and implementation of agents in multi-agent systems. Several forms of computational logic systems are needed in this setting, including abductive, argumentative and preference-based systems. We briefly sketch the agent model called KGP, and an ongoing extension of it which is needed to model agents in distributed settings such as the Grid and, more generally, Service-oriented architectures.*


**Keywords:** Logica computazionale, sistemi multi-agente, abduzione, argomentazione

## 1 Introduction

Computational Logic (CL) has been successfully adopted, in recent years, in modelling agents within agent based systems. The adoption of CL techniques has the advantage that formal specifications come along with their computational counterparts in the form of provably correct and executable proof procedures. The formal and computational models needed in the agent based settings require an integrated treatment of different features, which can be handled within various extensions of the basic logic programming framework, including abduction, argumentation and constraint logic programming. In this short paper we briefly summarize one such approach which has lead to the defnition of the KGP model for agency, and which is being further developed to cope with the specification of agents in service-oriented applications.

## 2 The KGP model

$KGP$ is an acronym for **K**nowledge, **G**oals and **P**lan. The model is intended to provide a modular and hierarchical specification of agents equipped with a variety of advanced reasoning features to allow intelligent decision making and behaviour. KGP agents are particularly suited to open dynamic environments where they have to adapt to changes in their environment and they have to function in circumstances where they have incomplete information. Here we give an overview of the KGP agent model and its components [4, 3]. The model relies upon

− an internal (or mental) *state*, holding the agent *K*nowledge base (beliefs), *G*oals (desires) and *P*lans (intentions),
− a set of *reasoning capabilities*,
− a set of *physical capabilities*,
− a set of *transition rules*, defining how the state of the agent changes, and defined in terms of the above capabilities,
− a set of *selection operators*, to enable and provide appropriate inputs to the transitions,
− a *cycle theory*, providing the control for deciding which transitions should be applied when, and defined using the selection operators. The model is defined in a modular fashion, in that different activities are encapsulated within different capabilities and transitions, and the control is a separate module. The model also has a hierarchical structure, depicted in figure 1.

**Internal state.** This is a tuple $\langle KB_0, \mathcal{F}, \mathcal{C}, \Sigma \rangle$, where:

- $KB_0$ holds the (dynamic) beliefs of the agent about the external world in which it is situated, as well as a record of the actions it has already executed.

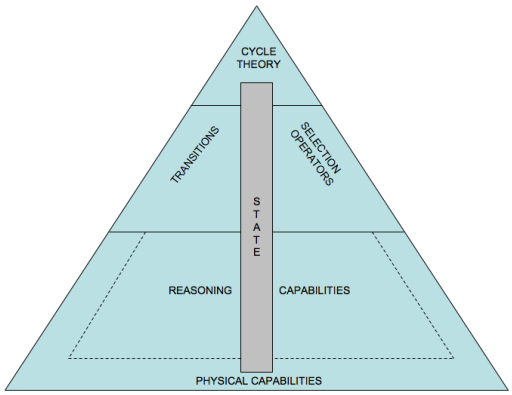


Figure 1: A graphical overview of the KGP model

- $\mathcal{F}$ is a forest of trees whose nodes are *goals*, which may be executable or not. Each tree in the forest gives a hierachical presentation of goals, in that the tree represents the construction of a plan for the root of the tree. The set of leaves of any tree in $\mathcal{F}$ forms a currently chosen plan for achieving the root of the tree. *Executable goals* are *actions* which may be *physical*, *communicative*, or *sensing*. *Non-executable goals* may be *mental* or *sensing*. Only non-executable mental goals may have children, forming (partial) plans for them. Actions have no children in any trees in $\mathcal{F}$. The roots of trees in $\mathcal{F}$ are referred to as *top-level goals*, the executable goals are referred to as *actions*, and non-executable goals which are not top-level goals are referred to as *sub-goals*. Top-level goals are classified as *reactive* or *non-reactive*. Roughly speaking, reactive goals are generated in response to observations, e.g. communications received from other agents and changes in the environment, for example to repair plans that have already been generated. Non-reactive goals, on the other hand, are the chosen desires of the agent. Note that some top-level (reactive) goals may be actions.

- $\mathcal{C}$ is the Temporal Constraint Store, namely a set of constraint atoms in some given underlying constraint language. These basically constrain the time variables of the goals in $\mathcal{F}$. For example, they may specify a time window over which the time of an action can be instantiated, at execution time.

- $\Sigma$ is a set of equalities instantiating time variables with time constants. For example, when the time variables of actions are instantiated at action execution time, records of the instantiations are kept in $\Sigma$.

**Reasoning capabilities.** These are:

- *Planning*, which generates plans for mental goals given as input. These plans consist of temporally constrained sub-goals and actions designed for achieving the input goals.

- *Reactivity*, which is used to provide new *reactive* top-level goals, as a reaction to perceived changes in the environment and the current plans held by the agent.

- *Goal Decision*, which is used to revise the *non-reactive* top-level goals, adapting the agent's state to changes in its own preferences and in the environment.

- *Identification of Preconditions* and *Identification of Effects* for actions, which are used to determine appropriate sensing actions for checking whether actions may be safely executed (if their preconditions are known to hold) and whether recently executed actions have been successful (by checking that some of their known effects hold).

- *Temporal Reasoning*, which allows the agent to reason about the evolving environment, and to make predictions about properties, including non-executable goals, holding in the environment, based on the (partial) information the agent acquires over its life-time.

- *Constraint Solving*, which allows the agent to reason about the satisfiability of the temporal constraints in $\mathcal{C}$ and $\Sigma$.

In the concrete realisation of the KGP model, we have chosen to realise the above capabilities in various extensions of the logic programming paradigm. In particular, we use (conventional) logic programming for Identification of Preconditions and Effects, abductive logic programming with constraints for Planning, Reactivity and Temporal Reasoning, and logic programming with priorities for Goal Decision.

**Physical capabilities.** In addition to the reasoning capabilities, the agent is equipped with "physical" capabilities, linking the agent to its environment, consisting of

- A *Sensing* capability, allowing the agent to observe that properties hold or do not hold, and that other agents have executed actions.

- An *Actuating* capability, for executing (physical and communicative) actions.

**Transitions.** The state $\langle KB_0, \mathcal{F}, \mathcal{C}, \Sigma \rangle$ of an agent evolves by applying transition rules, which employ the capabilities as follows:

- *Goal Introduction (GI)*, possibly changing the top-level goals in $\mathcal{F}$, and using Goal Decision.

- *Plan Introduction (PI)*, possibly changing $\mathcal{F}$ and $\mathcal{C}$ and using Planning.

- *Reactivity (RE)*, possibly changing the reactive top-level goals in $\mathcal{F}$ and possibly $\mathcal{C}$, and using the Reactivity capability.
- *Sensing Introduction (SI)*, possibly introducing new sensing actions in $\mathcal{F}$ for checking the preconditions of actions already in $\mathcal{F}$.
- *Passive Observation Introduction (POI)*, changing $KB_0$ by recording unsolicited information coming from the environment, and using Sensing.
- *Active Observation Introduction (AOI)*, possibly changing $\Sigma$ and $KB_0$, by recording the outcome of (actively sought) sensing actions, and using Sensing.
- *Action Execution (AE)*, executing all types of actions and as a consequence changing $KB_0$ and $\Sigma$, and using Actuating.
- *State Revision (SR)*, possibly revising $\mathcal{F}$, and using Temporal Reasoning and Constraint Solving.

**Cycle and Selection operators.** The behaviour of an agent is given by the application of transitions in sequences, repeatedly changing the state of the agent. These sequences are not determined by fixed cycles of behaviour, as in conventional agent architectures, but rather by reasoning with *cycle theories*. Cycle theories define preference policies over the order of application of transitions, which may depend on the environment and the internal state of an agent. They rely upon the use of *selection operators* for detecting which transitions are enabled and what their inputs should be, as follows:
$-$*action selection* for inputs to AE;
this selection operator uses the Temporal Reasoning and Constraint Solving capabilities; $-$ *goal selection* for inputs to PI;
this selection operator uses the Temporal Reasoning and Constraint Solving capabilities; $-$ *effect selection* for inputs to AOI; this selection operator uses the Identification of Effect reasoning capability;
$-$ *precondition selection* for inputs to SI; this selection operator uses the Identification of Preconditions, Temporal Reasoning and Constraint Solving capabilities.
The provision of a declarative control for agents in the form of cycle theories is a highly novel feature of the model, which could, in principle, be imported into other agent systems. In the concrete realisation of the KGP model, we have chosen to realise cycle theories in the same framework of logic programming with priorities and constraints that we also use for Goal Decision.

### 2.1 Computational model

One central distinguishing feature of the KGP model, in comparison with other models for agency, including those based on logic programming, is its modular integration within a single framework of abductive logic programming, temporal reasoning, constraint logic programming, and preference reasoning based on argumentation in order to support a diverse collection of capabilities. Each one of these is specified declaratively and equipped with its own provably correct computational counterpart. These computational models are heavily based upon proof procedures for (various extensions of) logic programming. In particular, the operational model for KGP agents relies upon CIFF [2], a proof procedure for abductive logic programming with constraints, and Gorgias [1], for logic programming with priorities. These procedures have been obtained by adapting and suitably extending two existing proof procedures for logic programming, namely Fung and Kowalskis IFF procedure for abductive logic programming for CIFF, and Kakas and Tonis argumentation-based procedure for negation as failure in logic programming for Gorgias. The overall operational models are sound and (in some cases) complete with respect to the abstract KGP model, and form a solid bridge between the KGP model and its implementation within the PROSOCS platform, a prototype implementation using Sicstus Prolog and JXTA [5].

## 3 Argumentative agents in ARGUGRID

The use of agent technology offers a solution to dynamic service composition in distributed settings such as the Grid 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 [6]. We have proposed in [7] 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. This type of argumentative agents can be seen as a variant of KGP agents. This variant relies upon the use of an argumentation decision-making tool supporting all reasoning capabilities, and it makes use of argumentative protocols for persuasion in negotiation. Argumentative agents are equipped with a specialised internal state, consisting of requirements, abstract or partially instantiated workflows, concrete workflows, planned communicative actions and actions executed in the past by the agent or by others, and arguments. The KGP modular architecture allows us to adopt a specialised set of reasoning capabilities supporting 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, which is actu-

[1] www.argugrid.eu

ally missing in the original KGP model. Specialised physical capabilities are also needed in this setting to provide suitable forms of inter-agent communications, and finally appropriate transitions encapsulate the new capabilities. In this setting, agents 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. Basically, argumentative KGP agent are 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
  − 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 extended 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), SDM (social decision making), and SI (social interaction) modules in [7]. The listening and talking capabilities are special cases of sensing and actuating in the KGP model.

## 4 Conclusions

We have briefly described work carried out in recent years and still ongoing, which aims at adopting computational logic for the description of agents in agent based systems. The use of computational logic allows us, on one hand to partially fill the gap between agent models and their computational realization. Indeed, the specification of KGP agent is a sort of *executable* specification due to the fact that the computational logic tools adopted in this setting are equipped with suitable concrete proof procedures. On teh other hand, the modularity of the KGP model allows one to extend it naturally to support new forms of reasoning, such as the ones needed in order to model the type of agents needed in service-oriented applications. Again, computational logic tools, based on various forms of argumentation, can be adopted in these settings to support the new type of capabilities needed, such as decision making and negotiation. This is still ongoing work we are carrying out within the ARGUGRID project.

## REFERENCES

[1] N. Demetriou and A. C. Kakas. Argumentation with abduction. In *Proceedings of the fourth Panhellenic Symposium on Logic*, 2003.

[2] U. Endriss, P. Mancarella, F. Sadri, G. Terreni, and F. Toni. The CIFF proof procedure for abductive logic programming with constraints. *Lecture Notes in Artificial Intelligence*, 3229:680–684, 2004.

[3] A. C. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. Declarative agent control. *Lecture Notes in Artificial Intelligence*, 3487:96–110, 2005.

[4] A.C. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. The KGP model of agency. In R. Lopez de Mantaras and L. Saitta, editors, *Proceedings of the Sixteenth European Conference on Artificial Intelligence, Valencia, Spain (ECAI 2004)*. IOS Press, August 2004.

[5] K. Stathis, A.C. Kakas, W. Lu, N. Demetriou, U. Endriss, and A. Bracciali. PROSOCS: A platform for programming software agents in computational logic. *Applied Artificial Intelligence*, 20(4-5), 2006.

[6] F. Toni. Argumentative KGP agents for service composition. Proc. AITA08, Architectures for Intelligent Theory-Based Agents, AAAI Spring Symposium, March 2008, Stanford University, CA, USA, 2008.

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

## 5 Contacts

Paolo Mancarella (Corresponding Author)
Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo, 3
56127 Pisa, Italy
Tel: +39 050 2212 710
paolo.mancarella@unipi.it

Francesca Toni
Department of Computing, Imperial College London
South Kensington Campus, Huxley Building
London SW7 2AZ, UK
Tel: +44 (0)20 7594 8228
ft@doc.ic.ac.uk