

GrAM: Reasoning with Grounded Action Models by Combining Knowledge Representation and Data Mining

Nicolai v. Hoyningen-Huene, Bernhard Kirchlechner, and Michael Beetz

Intelligent Autonomous Systems Group
Technische Universität München, Munich, Germany
{hoyninge|kirchlec|beetz}@in.tum.de
<http://ias.cs.tum.edu>

Abstract. This paper proposes GrAM (Grounded Action Models), a novel integration of actions and action models into the knowledge representation and inference mechanisms of agents. In GrAM action models accord to agent behavior and can be specified explicitly and implicitly. The explicit representation is an action class specific set of Markov logic rules that predict action properties. Stated implicitly an action model defines a data mining problem that, when executed, computes the model's explicit representation. When inferred from an implicit representation the prediction rules predict typical behavior and are learned from a set of training examples, or, in other words, grounded in the respective experience of the agents. Therefore, GrAM allows for the functional and thus adaptive specification of concepts such as the class of situations in which a special action is typically executed successfully or the concept of agents that tend to execute certain kinds of actions.

GrAM represents actions and their models using an upgrading of the representation language OWL and equips the Java Theorem Prover (JTP), a hybrid reasoner for OWL, with additional mechanisms that allow for the automatic acquisition of action models and solving a variety of inference tasks for actions, action models and functional descriptions.

1 Introduction

Marvin L. Minsky stated 1986 in “The Society of Mind”: “we need to combine at least two different kinds of descriptions. On one side, we need structural descriptions for recognizing chairs when we see them. On the other side we need functional descriptions in order to know what we can do with chairs” [1, p. 123]. These two kinds of descriptions mirror the bifocal perspectives of all agents: the perceptual and the acting view. In the first one, objects are grouped into concepts according to similarities in the sensory input like appearance or shape, we call these a structural concept. The categories of the second are clustered by similarities of their use or function and therefore, we call them functional concepts. Even these two constructs seem orthogonal, there must be a mapping

between them in our mind or otherwise we could not categorize things we see into a functional concept like seats.

Functional concepts are based on the (typical) behavior of agents. Imagine, which things we would categorize as seats for elephants as opposed to men, or the meaninglessness of such a category for fish. Action models that describe the preconditions and typical output of an action according to a class of agents are essential for this type of descriptions. Functional concepts fit ergo perfectly to planning tasks because of their relatedness to preconditions of actions.

In this paper we describe a knowledge representation language and a system based on it, that is able to handle structural and also functional descriptions. We exemplify the system by an information agent that performs no action itself, but is capable to perceive actions of other agents and their environment as facts within a given structural concept hierarchy. This suffices to explain the intent and usage focusing on the main idea. The system could easily be utilized for active agents.

To handle functional and structural descriptions we equip agents with GrAM (Grounded Action Models) that provides means for automatically acquiring action models and for reasoning about subsumption of individuals under functional and structural concepts. GrAM provides the following principles.

1. GrAM is a representation language that uses OWL (Web Ontology Language), a knowledge representation language based on description logics. GrAM provides a basic ontology with *GrAM:actions*, *GrAM:situations*, and *GrAM:agents*, as well as action models as additional entities. Domain ontologies can easily be adapted to use GrAM's representational power. To apply GrAM to a particular application domain, we import an OWL ontology and assert that concepts in this ontology are specializations of *GrAM:actions*, *GrAM:situations*, and *GrAM:agents*. Through these assertions, the domain concepts inherit the properties of the respective GrAM concepts and GrAM inference mechanisms become applicable to these concepts. It also provides constructors for functional concepts in terms of action models.
2. GrAM represents action models as sets of Markov logic rules that correlate situation and action properties with a probability distribution. Definitions of action models can be explicit specifying the rule set extensional or implicit specifying a set of executed actions and the situations that the actions were executed in or that resulted from their execution.
3. GrAM provides a number of inference mechanisms for actions and their models which allow for the automatic acquisition of action models, predicting action properties, subsumption under functional concepts, assessing the predictive accuracy of a model, etc. Stated implicitly an action model defines a data mining problem that, when executed, computes the respective set of prediction rules that is the explicit model.

GrAM is the only knowledge representation mechanism we know of that covers action models together with the symbol grounding problem and reasons about them in sophisticated ways. GrAM integrates data mining as a key inference mechanism and provides a seamless transition between action models and

data mining tasks and results. Resource intensive inferences are performed in a demand driven and therefore resource efficient way.

GrAM is implemented extending the OWL Web Ontology Language. Actions and situations are specified as OWL classes, action models are introduced as new entities and the extended language offers new constructors for functional concepts. To realize GrAM’s reasoning mechanisms we have extended JTP, a hybrid reasoner for OWL, with additional reasoning mechanisms that allow for the automatic acquisition of the explicit representation of action models and solving a variety of inference tasks about actions and action models.

In the remainder of the paper we proceed as follows. The next section introduces the representation of football games as our example application domain. We give an overview for OWL and describe GrAM’s extensions including a basic action ontology in section 2 as well as action models and functional concept constructors in section 3. By means of scoring chances we illustrate the strength of GrAM’s representation language. GrAM’s inference mechanisms to handle these terms are detailed afterwards in section 4. Implementational aspects contain a short guidance, how to adapt GrAM into any agent system (see 5). An overview of related work (6) and conclusions (7) complete this paper.

2 Representing Football Games

For the purpose of this paper we consider a particular application domain: the analysis of games in simulated robot football ([2], [3]). Each year more than 30 research groups participate in a competition for simulated robot football teams in the context of the RoboCup world championship. Each participating research group programs a team of simulated football players that have, at a very abstract level, fairly realistic perception and action capabilities. The competition games are logged by the simulator writing the positions and motions of all players and the ball, and the time stamped referee decisions such as offside and corner kick into log files. The ball motion implied by the ball position data is segmented into ball actions. Recognized ball actions are classified into shots, passes, and dribblings. These data form the observations of a game that are used in this paper.

Building models of ball actions in simulated football games is an interesting research task for several reasons. First, these actions are in several dimensions more realistic than models typically used in AI applications such as AI planning [4]. The spatio temporal properties are often very important. Ball actions have many different parameters: passes can be played hard or soft, deep or sideways, safe or risky, played to different receivers, etc. The effects of actions are typically affected by interactions and therefore highly situation-dependent. The outcomes are typically nondeterministic and the actions have high probability of failing. Many of these features are shared with other physical actions that are carried out by humans and animals. A consequence is that the appropriate modeling of ball actions requires particularly rich representation means.

2.1 OWL (Web Ontology Language)

We represent and reason about actions and action models using an extension of the OWL Web Ontology Language [5]. OWL is a knowledge representation language based on description logics. The basic representational means are classes and individuals, the subclass relationship and the properties of objects. Classes define a group of individuals that belong together because they share some properties. The classes together with the subclass relationship form a specialization hierarchy on classes: the ontology. Properties represent binary predicates, and can be used to state relationships between individuals or from individuals to data values. An example class definition is represented in section 2.3, figure 2 and 3 exemplify the assertions of individual facts.

The selection of description logics as representation language for structural concepts is driven by mainly two aspects: structural and relational knowledge can be naturally described and inference can be done efficiently. The particular use of OWL has several advantages over other description languages. It is specialized for ontologies in the web and proposed as a recommendation of the world wide web consortium. The exchange of knowledge and the use of existing information will become more and more important for all kinds of agents. For our demonstration domain a number of OWL ontologies are accessible through the World Wide Web including football ontologies with information about football players, teams, management, as well as time and action ontologies. We have also included concepts of the upper ontology of openCyc (<http://www.opencyc.org/>), which offers an OWL export, into our ontology.

As OWL is based on XML, taxonomies and logical expressions can be transformed into hypertext documents for browsing customized using XSLT style sheets. Comprehensive software toolboxes for processing OWL documents including APIs and parsers like Jena are available. Editors such as Protege including browsers for ontologies rendered as graphs like OntoViz ease the creation and modification of OWL knowledge bases. Using these tools, knowledge bases and action models can easily be published in the semantic web in human and machine readable form. Queries to GrAM can be stated in OWL-QL (OWL Query Language) [6].

2.2 Football ontology

Figure 1 shows an excerpt of a football ontology including the three key classes ball action, player and situation. The graph displayed in UML notation shows subclass relationships and the most important properties of actions, players and situations. Subclass relationships are represented by arcs with hollow arrow heads and properties by those with filled arrow heads. As already mentioned the class hierarchy reuses concepts like *PurposefulAction* of the openCyc upper ontology.

The most important interactions between ball actions, situations and players are the following ones. Ball actions are *doneBy* a player in a *startSituation* that correspond to a *PlayerSituation* observed by the player closest to the ball.

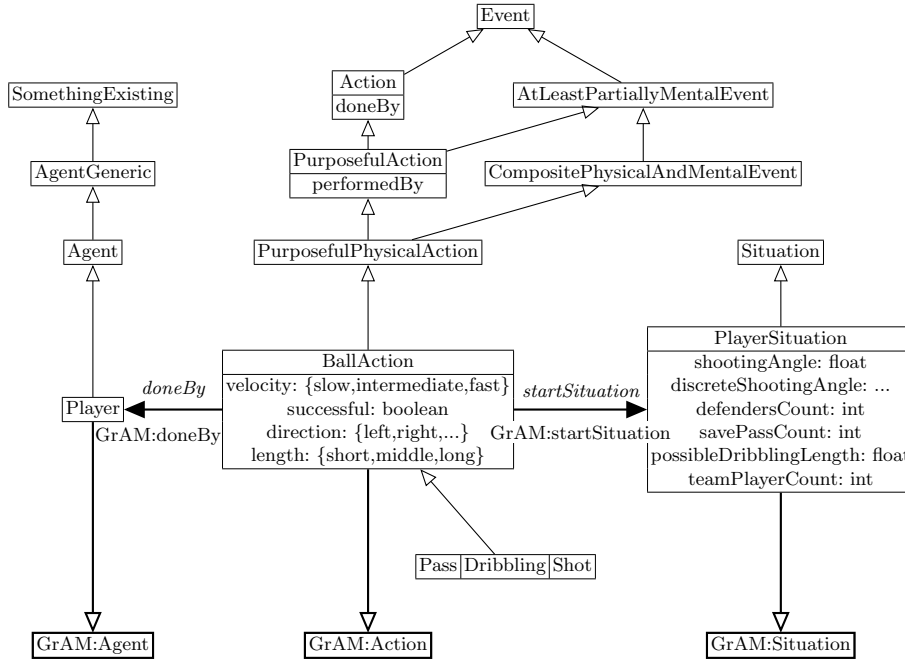


Fig. 1. Excerpt of the ontology representing the football domain and its integration in GrAM.

Subclasses of ball actions are modelled as dribblings, shots, or passes. Ball actions can be *successful* if and only if the team of the performer of the ball action keeps possession in the resulting situation or that team scored a goal. In addition, a number of parameters of the ball actions are shown as data valued properties like the velocity, direction or length.

Situations are completely determined by the properties *position*, *velocity*, and *acceleration* of the ball and the performing player. To reason about situations effectively we use, however, additional user defined situation abstractions, such as the distance of the ball to the goal (*goalDistance*), the number of possible save passes (*savePassCount*), etc. These properties of (start) situations are typically much better correlated to properties of actions such as the action outcome and are therefore needed to learn prediction models for these action properties effectively.

Figure 2 and 3 show example situations in a simulation league game. Beside an image of the scene the action and situation individuals are stated in terms of the football ontology of figure 1.

2.3 Defining scoring chances

There are a lot of other concepts in the football domain not represented in the ontology excerpt. For the rest of this paper we focus on the concept of

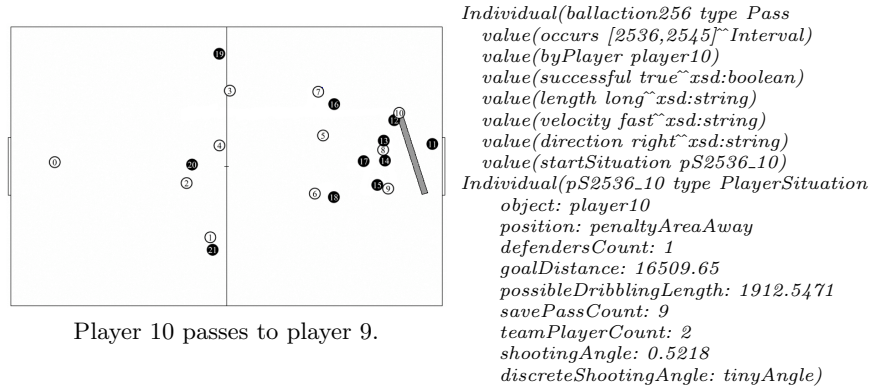


Fig. 2. Situation in a simulated RoboCup game stated in OWL using terms of the football ontology.

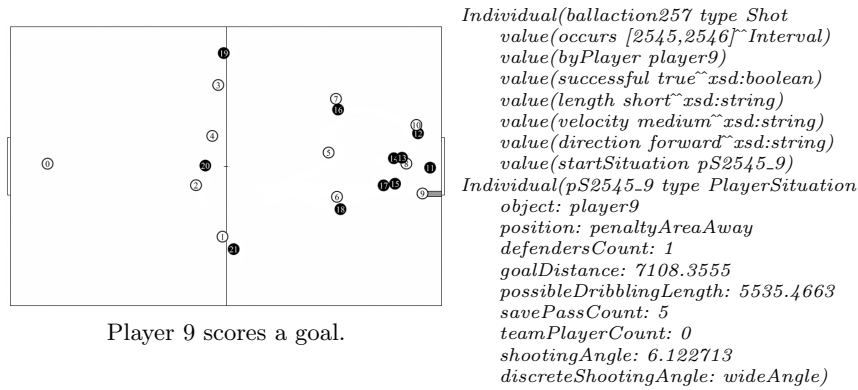


Fig. 3. Shot at the goal and (successful) scoring chance for *player9* stated in OWL.

scoring chances. This concept is obviously a subclass of situations. Stating which situations belong to scoring chances in a general form is difficult because of the high dependency of the definition in respect to the player skills. So a situation could be a scoring chance for a professional player but not for a rookie.

In description logics it is straightforward to define successful scoring chances i.e. situations that lead to observed goals if the definition is based on structural concepts only. Let us look to a definition of successful scoring chances stated in OWL. It would have the following form:¹

¹ We use just the abstract syntax in this paper and forbear from spearheading the XML notation because of clarity, compactness and readability, the translation is obvious indeed.

```

Class( ScoringChance complete
  intersectionOf(
    Situation
    restriction(
      startSituation-1
      allValuesFrom(
        intersectionOf(
          Shot
          restriction(successful value(true) ) ) ) ) ) ) ) )

```

An instance of this class is stated in figure 3. In contrast to this compact definition it is difficult or not even possible to incorporate also the unsuccessful scoring chances under the concept. Scoring chances in their full meaning can only be described in a functional way: they could be defined as the situations in which some football players *tend* to shoot the ball and are *likely* to score if they shoot. The wording already suggests a probabilistic notion of the concept and we need action models, external to OWL, to be able to express intention and typical behavior of players and thus cover the whole meaning of scoring chances. This definition task will illustrate GrAM’s abilities as our running example for the rest of the paper.

3 Representation Language

Because of the limited representational power of OWL in respect to action models we need additional language constructs. GrAM’s representation language constitutes an extension of OWL by a basic ontology and constructors for action models and functional concepts, which are determined by the behavior of some agent. To equip the representation language with action models, some basic concepts and relations are necessary in advance.

3.1 GrAM’s basic action ontology

We consider actions as state transitions analogously to the situation calculus [7]. The class *Action* is therefore related to a *start* and *end situation*. Also an action is supposed to be done by some agent as an intended event. Figure 4 visualizes the three key classes and their relationships with the namespace prefix *GrAM*.

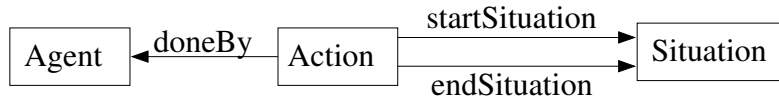


Fig. 4. The ontology representing predefined concepts and relations.

The specific concepts of the domain of interest have to be linked to these classes by stating a subclass or equivalent class or property relationship to be

processable by GrAM. This is already done for our football ontology shown in figure 1, the respective classes and relations are highlighted in the excerpt illustration.

3.2 Action models

The intention or typical behavior of an agent or agent group is represented by an action model. More precisely it corresponds to the correlation between property values of an action and its start situation with a probability distribution. The probability distribution is mostly necessary because of a not fully observable or modelled environment and the nondeterminism of action outcomes. The correlation i.e. an action model is represented as a set of Markov Logic rules.

Markov Logic Markov logic [8] is a first order predicate logic language where probabilities can be assigned to formulas. Probabilities are stated as weights that specify how strong a contradiction to the individual formula diminishes the worlds probability. Markov logic has the expressive power to represent a wide range of probabilistic representations that can be learned using statistical relational learning methods including decision and regression trees. We will state our action models explicitly in Markov logic. A Markov logic rule set for an action model can be written as

$$rule_j = \mathbf{condition}_j \xrightarrow{prob_j} \mathbf{conclusion}_j.$$

Thus, rules for predicting properties of actions that are contained in the rule set of an explicit action model have the following form:

$$\begin{aligned} &\forall ba. \text{startSituation}(ba, sit) \\ &\quad \wedge \text{cond}_{\text{situationProperty}_k}(sit) \wedge \dots \\ &\quad \wedge \text{cond}_{\text{situationProperty}_n}(sit) \\ &\xrightarrow{prob} \text{actionProperty}_i(ba, value_i) \wedge \dots \\ &\quad \wedge \text{actionProperty}(ba, value) \end{aligned}$$

The rule states that if situation sit is the enabling situation of ba and satisfies the conditions for the given properties $\text{cond}(sit)$ then the action ba will typically have the declared property values with a probability of $prob$. Predictable action properties contain also the type of an action i.e. the membership to a class.

Explicit Definition of Action Models The set of rules of action models predict the characteristics of actions in situation specific ways. We define for example an action model $exampleAM$ consisting of just one rule:

$$\begin{aligned} &\forall ba. \text{startSituation}(ba, sit) \\ &\quad \wedge \forall x. \text{distanceToGoal}(sit, x) \wedge x \leq 10000 \\ &\quad \wedge \forall x. \text{defendersCount}(sit, x) \wedge x \leq 1 \\ &\quad \xrightarrow{0.8} \text{successful}(ba, true) \end{aligned}$$

It describes the shot behavior and states that a shot would succeed with a

probability of 80 percent in a situation, where the shooter is closer than 10.000 mm to the goal and there is at most one defender in the direction to the goal.

The condition *startSituation*(*ba*, *sit*) is contained in every rule and needs not to be assigned explicitly. In abstract syntax the definition would look like the following:

```

ExtensionalActionmodel(exampleAM
  withRule( Rule( Body(
    restriction(distanceToGoal lessThanOrEqual(10000))
    restriction(defenderCount lessThanOrEqual(1))
    Head(restriction(successful value(true)))
    withProbability(0.8)))

```

Implicit Definition of Action Models The explicit definition of action models can be quite cumbersome because the behavior has first to be analyzed to extract a rule representation. Also even one agent could show different behaviors depending on the environment like the performance of an expert player versus a beginner differs from its performance against an opponent of equal strength. GrAM allows an action model to be defined implicitly by a set of actions and their start situations. This set is assumed to contain observed examples of the behavior that should be described by the action model. The action model is asserted to hold for this set i.e. the model describes the correlations in the given set correctly. To focus only on some aspects of the correlation the situation and action properties can optionally be narrowed down by enumerating only the relevant.

Suppose we want to define the scoring chances for a particular player, say ‘*player9*’. The specialization of this situation class requires a model of the typical shot behavior of *player9*, the respective definition of the implicit action model has the following form:

```

IntensionalActionModel(typicalBehavior9
  forAction(restriction(doneBy value(player9)))
  observable(goalDistance position teamPlayerCount
    savePassCount possibleDribblingLength defendersCount)
  predictable(type successful))

```

This model named *typicalBehavior9* represents the correlation between the stated observable situation properties on the one side and the *type* of action that is chosen and its outcome (*successful*) based on all observed actions of *player9* (denoted by *forAction*) on the other.

3.3 Functional concept definitions and action related concepts

GrAM allows to construct specific subclasses of *Situation* functionally according to an action model and a set of actions. A subclass, constructed by this way, contains the start situations of the action set for which the given action model predicts the respective action property values correctly according to a specified

threshold. This threshold declares the minimal probability for the truth of the Markov logic rules combined with the action and situation values as facts. So the subsumption of a situation individual under a specific situation class has no probability distribution but is assigned only to a boolean value. That is fundamental to integrate the constructed concept into the description logic which does not deal with probability distributions in its inference. For this reason GrAM has the strength to handle nondeterministic action models on the one side and to include concepts based on these models in a description logic like OWL on the other side.

Other action related concepts offered by GrAM are agent class constructors and predictions. Agent classes describe agents that show the same behavior as represented in a given action model for a specified set of actions. A threshold for minimal probability as in the definition of situation classes is also obligatory and is used with the same semantics.

Predictions are primarily used in queries to find out which action property values would be typical for a given behavior in a specified situation.

Using the functional situation class constructor we are able to define scoring chances for *player9* as the class of situations that would lead with a probability higher than 70 percent to a successful shot according to the typical behavior of *player9* (defined in section 3.2). The situation class constructor has the following form:

```
SituationClass(ScoringChance
  byActionmodel(typicalBehavior9)
  forAction(Shot restriction(successful value(true)))
  minProbability(0.7))
```

To exemplify the use of agent class constructors as an additional functional description, we define the concept *ExampleAgent* as all agents showing a specific behavior namely *exampleAM*, which was introduced in section 3.2, for actions in the first half of a football game:

```
AgentClass(ExampleAgent
  byActionmodel(exampleAM)
  forAction(ActionsFirstHalf)
  minProbability(0.6))
```

GrAM offers also constructors for predictions, which make only sense as incomplete definitions in premises of queries and are introduced therefore in the next section 4.

4 Inference

GrAM offers not only a representational framework for action models and related concepts but also mechanisms to automatically infer explicit representations of action models, subsumption for related concepts, and predicted values.

The subsumption of individuals under situation classes is inferred by application of Markov rules with a higher probability than the given threshold to

start situations of the given actions and by comparison of the produced action property values with the asserted ones. If they are equal, the situation individual is assumed to be a member of the situation class.

The situation shown in figure 2 would be classified as scoring chance for *player9* because there exists an applicable Markov Logic rule of action model *typicalBehavior9*. For example,

$$\begin{aligned} & \forall ba. \text{startSituation}(ba, sit) \\ & \wedge \forall x. \text{distanceToGoal}(sit, x) \wedge x \leq 17203.732) \\ & \wedge \forall x. \text{defendersCount}(sit, x) \wedge x \leq 1) \\ & \xrightarrow{0.8} \text{type}(ba, \text{Shot}) \\ & \wedge \text{successful}(ba, true) \end{aligned}$$

applies to this situation and has a higher probability than the specified threshold of 0.7: The closest player to the ball in the considered situation is not further away than seventeen meter from the goal and there are no defenders towards the goal, so *player9* would perform a successful shot in this situation.

Membership of agent individuals to an agent class is inferred in a similar way. An agent belongs to an agent class if it conforms to the behavior specified by the respective action model in all considered actions. The inference process ensures this by verifying that the property values of all actions of the specified set done by the agent equal the ones, generated by application of the action model rules with higher probability than the threshold.

The predictions are inferred also via application of Markov Logic rules. A generated action with predicted properties and the probability of the used rule is appended to the prediction individual and can be queried. Only those values are generated that are predicted by an applicable rule.

Say we want to examine, how *player9* would act in a specific situation named *hypoSituation*, where *player8* failed to score. First, an incomplete definition of a prediction individual is made:

$$\begin{aligned} & \text{Individual}(\text{examplePred Prediction} \\ & \text{byActionmodel}(\text{typicalBehavior9}) \\ & \text{forSituation}(\text{hypoSituation})) \end{aligned}$$

Then, the most likely values together with their probability can be received. An OWL-QL query to achieve the outcome would be stated as follows:

$$\begin{aligned} & (\text{and } (\text{toAction examplePred ?predAction}) \\ & (\text{withProbability examplePred ?probability}) \\ & (\text{successful ?predAction ?outcome})) \end{aligned}$$

The variable *?outcome* would be bound to *true* or *false* according on which Markov logic rule of the action model *typicalBehavior9* would be applicable for *hypoSituation* to predict the outcome. The probability of the prediction will be assigned to *?probability*, *?predAction* contains just a generated name identifying the predicted action.

For all deductions concerning action related concepts the inference process needs an explicit representation of the action model. The Markov Logic rules are therefore needed to be available also for implicitly defined action models. GrAM

is able to extract the rules out of implicit definitions of action models by solving a data mining problem. All observed action and start situation pairs form the training data for that problem. The inference process first builds a matrix filled with all predictable action and respective situation property values. Decision tree learning [9] is applied to these data if action properties are discrete. Because we can not assume the action properties to be uncorrelated in general, the prediction of action property values is not learned separately for every property but at once. The learned decision tree is flattened to rules that belong to every possible path in the tree, taking the conjunction of the inner nodes as condition and the leaf as conclusion. The corresponding probability value is computed by multiplying confidence and support of the classification in the leaf (which can also be transformed to Markov logic weights). For continuous action properties GrAM infers the rules by REPTree regression tree learning [10] and analogous transformation of the resulting tree.

The mentioned inference mechanisms in combination with a calculus for OWL are sound in respect to the defined semantics of the concepts. Completeness, however, can not be ensured because of the rule generation process by data mining, where a lot of rule sets that holds for the training data exist but only one specific is mined.

We provided the introduced inference mechanisms by extending a theorem prover for first order logics called Java Theorem Prover (JTP). JTP is an object-oriented modular hybrid reasoning system implemented in Java [11]. It has a simple and general reasoning architecture consisting of modules called reasoners. The modular character of the architecture makes it easy to extend the system by adding new reasoners. JTP is equipped with special purpose reasoners for OWL expressions and knowledge bases, satisfiability in interval temporal logics, and others.

We have added inference mechanisms for action models in the form of reasoners. Namely we have equipped JTP with reasoners for solving data mining tasks, prediction by data mining models, and subsumption of individuals under functionally defined and action related concepts.

All inferences in GrAM including the generation of the explicit rule form are computed in a lazy manner or on demand. That is the respective data mining tasks are only solved if the respective explicit model has been queried directly or indirectly.

5 Work with GrAM

Starting from an existing knowledge base of an application domain which contains an ontology and facts, GrAM offers an easy way to build an information system based on this knowledge, that can be integrated into agent systems. The implemented system architecture of GrAM is shown schematically in figure 5. To adapt a domain to GrAM, the domain ontology has to be linked to GrAM's action ontology (see the arrow marked with 1 in figure 5). The facts representing the observed agent behaviors and additional knowledge can be transferred with-

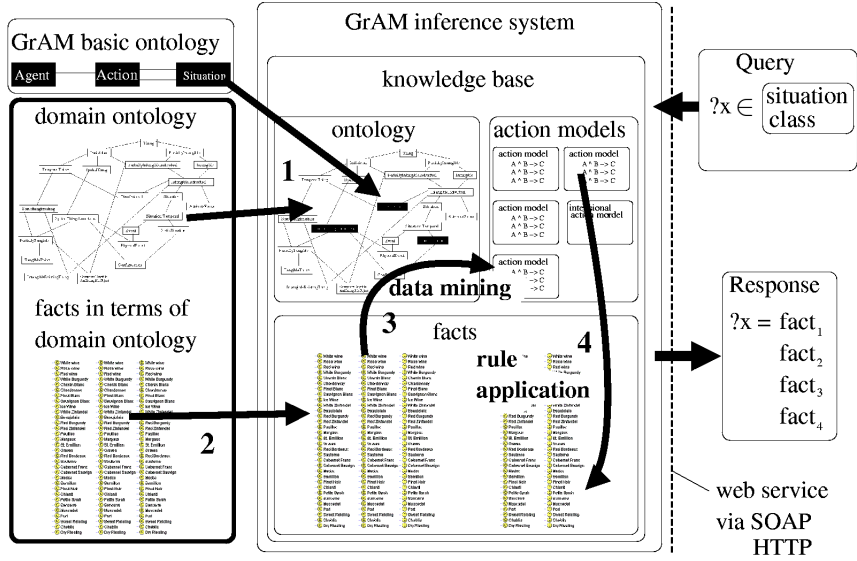


Fig. 5. GrAM’s system architecture.

out modification (see arrow 2). Facts and extended ontology form the knowledge base of an information system, which offers a web service interface to answer queries. The answer is inferred by GrAM’s inference mechanisms including data mining and Markov rule application (see arrows 3 and 4). The web service interface via SOAP over HTTP offers a flexible interface independent of programming languages and platforms.

We implemented a servlet as graphical user interface to GrAM, that can be used for human interaction with GrAM. A screen shot of a web browser showing the query input mask is depicted in figure 6.

6 Related Work

Our research on GrAM is part of a larger research effort in which we investigate computational models of embedded intelligent systems that “know what they are doing” [12]. Having action models that cover a wide range of behavior and its rationale is a necessary capability of such systems. Research on action modeling has been mainly performed in the area of reasoning about action using first-order predicate and nonmonotonic logics [13]. Those representations have only covered a small subset of the models proposed here. Symbolically specified action models yield the so-called symbol grounding problem: assigning meaning to symbols based on perception and action mechanisms of agents. While symbol grounding is not addressed in this research branch GrAM proposes a solution for it consisting of implicit and explicit action models.

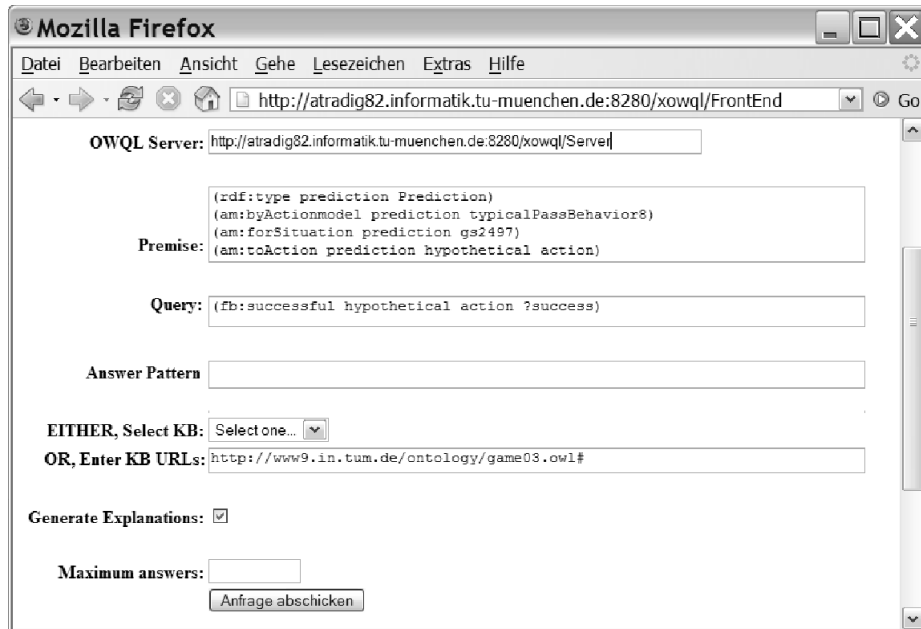


Fig. 6. Web interface to GrAM.

The acquisition of models of physical actions has so far received surprisingly little attention. Oates et al. [14] have learned models of action effects expressed in the sensor data space. Pasula et al [15] learn probabilistic Strips like rules from examples. Stulp and Beetz [16] use learned performance models in order to optimize chains of abstract actions. This branch of research does not focus on the representational issues of model acquisition.

We know of no other practical knowledge representation mechanisms that cover the range of action models and reasons about them as GrAM does. As well the combination of structural and functional descriptions was not be addressed before. Also, GrAM's integration of the data mining mechanism and its seamless transition between action models and data mining tasks and results is, as far as we know, novel.

7 Conclusions

In this paper we have proposed GrAM as an extension of a knowledge representation system that makes action models, their acquisition and reasoning about them as an integral part of the knowledge representation system of an intelligent agent. GrAM can be used for various domains where the behavior of agents needs to be embedded into the reasoning apparatus of the system. Functional descriptions are adapted to the behavior of agent groups to learn a structural representation of the same concept.

In a companion research project we use GrAM to represent and reason about a range of action models including causal and outcome models, action selection models, parameterization models, derived models, and others [3]. We believe that the representation languages that are capable of grounding action models and acquire them through data mining will be of key importance in the development of embedded intelligent systems that “know what they are doing”.

We are also starting to apply GrAM in the context of autonomous robot learning. For this purpose GrAM is integrated into the robot learning language ROLL [17] and used to make robots “action aware” [18]. GrAM will furthermore be used in context-aware intelligent environments, especially in the kitchen scenario where we try to achieve models for everyday activities [19, 20].

References

1. Minsky, M.L.: The society of mind. Simon and Schuster (1986)
2. Beetz, M., Flossmann, S., Stammeier, T.: Motion and episode models for (simulated) football games: Acquisition, representation, and use. In: 3rd International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS). (2004)
3. Beetz, M., Kirchlechner, B., Lames, M.: Computerized real-time analysis of football games. *IEEE Pervasive Computing* **4** (2005) 33–39
4. Smith, D., ed.: Special Issue on the 3rd International Planning Competition. Volume 20 of *Journal of Artificial Intelligence Research*. (2003)
5. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.: OWL Web Ontology Language Reference (2004) W3C Recommendation.
6. Fikes, R., Hayes, P., Horrocks, I.: OWL-QL: A Language for Deductive Query Answering on the Semantic Web. Technical Report KSL 03-14, Stanford University, Stanford, CA (2003) Technical Report.
7. McCarthy, J.: Situations, actions and causal laws. Technical report, Stanford University. Reprinted 1968 in *Semantic Information Processing* (M.Minsky ed.), MIT Press (1963)
8. Domingos, P., Richardson, M.: Markov logic: A unifying framework for statistical relational learning. In: *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connection to Other Fields*, Banff, Canada, IMLS (2004) 49–54
9. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc. (1993)
10. Witten, I.H., Frank, E.: *Data Mining: Practical machine learning tools and techniques*. 2nd edn. Morgan Kaufmann, San Francisco (2005)
11. Frank, G., Fikes, R., Jenkins, J.: JTP: A system architecture and component library for hybrid reasoning. In: *Procs of the 7th World Multiconf. Systemics, Cybernetics, and Informatics.*, Orlando, Florida, USA (2003)
12. Brachman, R.: Systems that know what they’re doing. *IEEE Intelligent Systems* (2002) 67 – 71
13. Allen, J., Ferguson, G.: Actions and events in interval temporal logic. *Journal of Logic and Computation* **4** (1994) 531–579

14. Oates, T., Schmill, M., Cohen, P.: Identifying qualitatively different outcomes of actions: Gaining autonomy through learning. In: Proceedings of the Fourth International Conference on Autonomous Agents, Barcelona, Spain, ACM Press (2000) 110–111
15. Pasula, H., Zettlemoyer, L., Kaelbling, L.: Learning probabilistic relational planning rules. In: Procs. of the 14th International Conference on Planning and Scheduling. (2004)
16. Stulp, F., Beetz, M.: Optimized execution of action chains using learned performance models of abstract actions. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI). (2005)
17. Beetz, M., Kirsch, A., Müller, A.: RPL-LEARN: Extending an autonomous robot control language to perform experience-based learning. In: 3rd International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS). (2004)
18. Stulp, F., Beetz, M.: Action awareness – enabling agents to optimize, transform, and coordinate plans. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2006)
19. Kranz, M., Rusu, R.B., Maldonado, A., Beetz, M., Schmidt, A.: A player/stage system for context-aware intelligent environments. In: Proceedings of UbiSys’06, System Support for Ubiquitous Computing Workshop, at the 8th Annual Conference on Ubiquitous Computing (UbiComp 2006), Orange County California, September 17-21, 2006. (2006)
20. Rusu, R.B.: Acquiring models of everyday activities for robotic control in ‘current PhD research in pervasive computing’. Technical Reports - University of Munich, Department of Computer Science, Media Informatics Group **LMU-MI-2005-3** (2006)