Plan Representation and Reasoning with Description Logics

Yolanda Gil
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey CA 90292
gil@isi.edu

Abstract

Planning problems are naturally knowledge rich. Yet, most planning research to date has focused on languages and algorithms to represent and search through the decision space in order to generate a feasible solution. This paper surveys previous work on extending planning algorithms with expressive representations of knowledge, in particular description logics, to reasoning about action, plan, and goal taxonomies. These approaches should be of interest to planning practitioners by also to Semantic Web researchers who are currently developing web service languages based on planning languages and in combination with description logics.

1 Introduction

Planning problems are naturally knowledge rich. Yet, most planning research to date has focused on languages and algorithms to represent and search through the decision space in order to generate a feasible solution. Many real-world planning applications require expressive plan representations, and often tend to not to focus on plan generation algorithms but rather plan recognition, plan evaluation, or plan matching. A challenging area of future research for the planning community is the development of rich representations of planning tasks and objectives, actions and events as part of a plan, and sophisticated relations among plans and type of plans, actions and goals.

Recent developments in web services and the composition of applications based on individual components also bring this topic to the forefront of research. Planning approaches have been used to automate service composition [McDermott 02]. Semantic web service languages are being investigated that combine description logics with service descriptions inspired by plan specification languages [Ankolenkar et al 02]. Plan representation and reasoning with description logics is of renewed interest.

This paper describes some important work on extending planning algorithms with expressive representations of knowledge, in particular description logics. Description logics are an extension of frame-based systems that support representations of classes, relations, and instances as necessary and sufficient definitions specified as logical expressions. Description logics include algorithms to automatically organize class descriptions in a *subsumption hierarchy*, where a class C1 subsumes another class C2 if its definition includes a superset of he instances included by C2. For a good overview of description logics, see [Baader et al 03] or dl.kr.org.

We can group these approaches into three topics:

1. **action taxonomies** to reason about action types
2. **plan taxonomies** to reason about plan subsumption of partially ordered plans
3. **goal taxonomies** to reason with expressive representation of goals and their parameters.

All of these approaches exploit the descriptions of objects in the domain (domain knowledge) in order to reason about action, goal, and plan descriptions.

The paper describes each of these areas using a representative system, giving examples of how description logics can be useful in planning problems. We also summarize other related research along the way that often integrates this work with other relevant areas such as knowledge representation, planning, and natural language.

2 Action Taxonomies: CLASP

One can use many levels of abstraction to describe an action. For example, driving and walking are both moving actions with specific means of locomotion. Reasoning about actions at different levels of abstraction enables a system to reason more efficiently about how different actions relate. We illustrate this idea with a specific system called CLASP.

CLASP is a knowledge representation system that builds on description logic to support plan subsumption and classification [Devanbu and Litman 94]. It was implemented as an extension of CLASSIC [Brachman et al 91] and was integrated with the LASSIE software information system [Devanbu et al 91]. A central contribution of CLASP is the subsumption and classification algorithms for plans described as action networks. Its main application was to described the behavior of a Private Branch Exchange (PBX) switching product. The examples in this section are taken from that application as described in [Devanbu and Litman 94].

CLASP used a STRIPS-like representation [Fikes and Nilsson 71] of actions in the plan, and assumes a propositional representation of planning problems with conjunctive expressions of preconditions and states. Figure 1 shows the core definitions of actions, states, and plans. Actions and states are defined as CLASSIC objects. Plans are defined in CLASP's language, and described as networks of actions that achieve a

goal from a given initial state. The action networks are partially ordered plans that include iteration and branching, and are called PLAN-EXPRESSION. A PLAN-EXPRESSION can be described with the constructs SEQUENCE, LOOP, REPEAT, TEST (conditional branching), OR (disjunctive branching), and SUBPLAN. The SUBPLAN construct supports modular definitions of plans through definitions of meaningful sub-networks.

```
(DEFINE-CONCEPT Action
   (PRIMITIVE
     (AND Classic-Thing
          (AT-LEAST 1 Actor)
          (ALL ACTOR Agent)
          (EXACTLY 1 PRECONDITION)
          (ALL PRECONDITION State)
          (EXACTLY 1 ADD-LIST)
          (ALL ADD-LIST State)
          (EXACTLY 1 DELETE-LIST)
          (ALL DELETE-LIST State)
            (EXACTLY 1 GOAL)
          (ALL GOAL STATE))))

(DEFINE-CONCEPT State
   (PRIMITIVE Classic-Thing))

(DEFINE-PLAN
   Plan
   (PRIMITIVE
     (AND Clasp-Thing
          (EXACTLY 1 INITIAL)
          (ALL INITIAL State)
          (EXACTLY 1 GOAL)
          (ALL GOAL State)
          (EXACTLY 1 PLAN-EXPRESSION)
          (ALL PLAN-EXPRESSION
               (LOOP Action)))))
```

Figure 1. CLASP definitions of actions, states, and plans.

```
(DEFINE-CONCEPT System-Act
   (AND Action
        (ALL ACTOR System-Agent)))

(DEFINE-CONCEPT Connect-Dialtone-Act
   (AND System-Act
     (ALL PRECONDITION
          (AND Off-Hook-State
               Idle-State))
     (All Add-LIST Dialtone-State)
     (ALL DELETE-LIST Idle-State
     (ALL GOAL
          (AND Off-Hook-State
               Dialtone-State))))

(DEFINE-CONCEPT Callee-Off-Hook-State
   (PRIMITIVE State))

(DEFINE-CONCEPT Callee-On-Hook-State
   (PRIMITIVE State))

(DEFINE-CONCEPT Callee-Off-Caller-On-State
   (AND Callee-Off-Hook-State Caller-On-Hook-State))

(DEFINE-PLAN Pots-Plan
  (AND Plan
   (ALL PLAN-EXPRESSION
       (SEQUENCE
         (SUBPLAN
            Originate-And-Dial-Plan)
         (TEST
            (Callee-On-Hook-State
             (SUBPLAN Terminate-Plan))
            (Callee-Off-Hook-State
             (SEQUENCE
                Non-Terminate-Act
                Caller-On-Hook-Act
                Disconnect Act)))))))

(DEFINE-PLAN Originate-And-Dial-Plan
   (AND Plan
        (ALL PLAN-EXPRESSION
             (SEQUENCE Caller-Off-Hook-Act
                       Connect-Dialtone-Act
                       Dial-Digits-Act))))
```

Figure 2. CLASP actions, states, and plans in the telephony domain.

```
(CREATE SCENARIO
  pots-busy-scenario
   (AND Plan
        (FILLS INITIAL state-u1on-u2off)
        (FILLS GOAL state-u1on)
        (FILLS PLAN-EXPRESSION
               (caller-off-hook-u1
                connect-dialtone-on-u1
                dial-digits-u1-to-u2
                non-terminate-on-u2
                caller-on-hook-u1
                disconnect-u1)))).

(CREATE-IND state-u1on-u2off
    (AND state-U1on State-U2off))

(CREATE-IND connect-dialtone-on-u1
    (AND Connect-Dialtone-Act
         (FILLS ACTOR switching-system)
         (FILLS PRECONDITION state-u1off-idle)))
```

Figure 3. CLASP plan, state, and action instances in the telephony domain.

Domain-specific types of states, actions, and plans are described using these core definitions. Figure 2 shows some of the definitions in the telephony switching domain. A connect dialtone action is defined as an action performed by a system (not by a user) that provides a dialtone when the phone is off the hook and idle. Subtypes of the class plan can be defined to create a taxonomy of plan types. For example, a plan for POTS (Plain Old Telephone Service) can be defined as one where a caller picks up the phone and dials, if the callee's phone is off hook the caller gets a busy signal and hangs up otherwise the call proceeds. Notice that Originate-And-Dial-Plan is a subplan that is defined separately and its plan expression is inserted in the appropriate node of the POTS plan expression.

Specific plans, states, and actions are created as instances of the classes defined in these taxonomies of states, actions, and plans. Specific plans are called *scenarios*, and they reflect different linearized sequences of actions that can be executed in the world. Figure 3 shows a scenario where the caller picks up the phone, gets a dialtone, dials and gets a busy signal, and hangs up causing the system to disconnect. The initial and goal states are defined as instances. Specific actions are defined as instances as well, for example connect-dialtone-on-u1 is performed by a switching system and requires the user to be idle.

CLASP supported subsumption and classification of plans and scenarios by extending these functions provided in CLASSIC for concepts and instances. A plan description A subsumes a plan description B if the initial state and goal state of A subsume the initial and goal states of B, and if the plan expression of A subsumes the plan expression of B. The subsumption of plan expressions was defined by considering action networks as an extension to deterministic finite automata (DFA) where the transitions are CLASSIC subsumption checks. The plan expression EA of a plan class A subsumes the plan expression EB of a plan class B if the languages accepted by their corresponding DFAs are subsumed, i.e., DEA's language is a subset of DEB's language. A scenario is an instance of a plan class if the action network of the plan expression of the scenario is accepted by the DFA defined by the plan expression of the plan class.

The CLASP knowledge representation system supported several aspects of reasoning about plans, including organization of plan classes, retrieval of plan types and scenarios with description-based queries, and validation of scenarios based on type descriptions. In the telephony domain, it provided a useful way to organize the representation of various features provided by telephony systems (such as the POTS feature), analyzing which features supported certain behavior patterns, and retrieving test scenarios for specific features and behaviors.

3 Plan Taxonomies: SUDO-PLANNER

Reasoning about how two plans relate to one another enables planners to make sure that two areas of the solution space are searched only once. This results in more efficient search. It is also important for planning algorithms to exhibit a property known as systematicity, which means that they can map out in an organized, comprehensive, and non-overlapping way the search space that they explore. Typically two plans are related by the specific steps and links that they include. Plan taxonomies can relate plans in more sophisticated ways, exploiting different levels of abstraction in the plans as well as definitions of aggregate steps and domain knowledge. SUDO-PLANNER [Wellman 88] exploits plan subsumption to control the search during plan generation. It was developed to reason about tradeoffs in decision making under uncertainty, and was used in the medical domain.

SUDO-PLANNER represented actions and plans using NIKL [Moser 83]. Actions were represented as concepts, with action parameters as concept roles, and action constraints represented as role restrictions. A taxonomy of action types enabled SUDO-PLANNER to exploit inheritance and classification. Figure 4 shows several examples of how actions are described in SUDO-PLANNER. Given these descriptions, the system deduces that open-lung-biopsy is a surgery through subsumption reasoning. To simplify things, we will denote actions by letters with subscripts where subsumers have a lower number (eg. a1 subsumes a5, b2 subsumes b3). We will also denote a partial plan (including the null plan) as A*.

SUDO-PLANNER represents plans as partially ordered sets of actions. It searched through plan space search [Weld 99].

Plan subsumption was evaluated through bipartite graph matching. Figure 5 shows some examples of plans shown as a sequence of three steps (eg P is [a1 a2 a3]). The lines show subsumption of individual steps (eg a1 subsumes a3, a4 and a6.). A plan subsumes another plan if their steps have an exclusive pairwise (isomorphic) subsumption relation in the order in which they appear in each plan.

```
(defconcept surgery
   :is (:and action
             (:the route invasive-path-into-body)))

(defconcept biopsy
   :is-primitive action ...))

(defconcept open-lung-biopsy
   :is (:and biopsy
             (:the route open-lung-path)))

(defconcept open-lung-path
   :is (:and invasive-path-into-body ...))
```

Figure 4. Examples of action descriptions in SUDO-PLANNER.

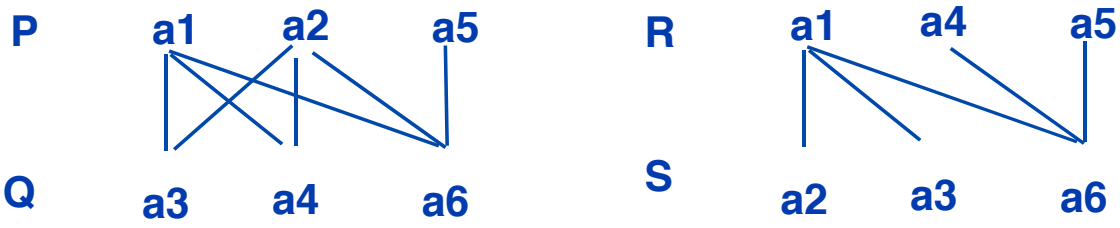

Figure 5. Plan subsumption in SUDO-PLANNER can be viewed as bipartite graph matching. Plan P subsumes plan Q because a1 subsumes a3, a2 subsumes a4, and a5 subsumes a6. Plan R does not subsume plan S because there is not a one-to-one correspondence based on subsumption for each of the three actions in the plans.

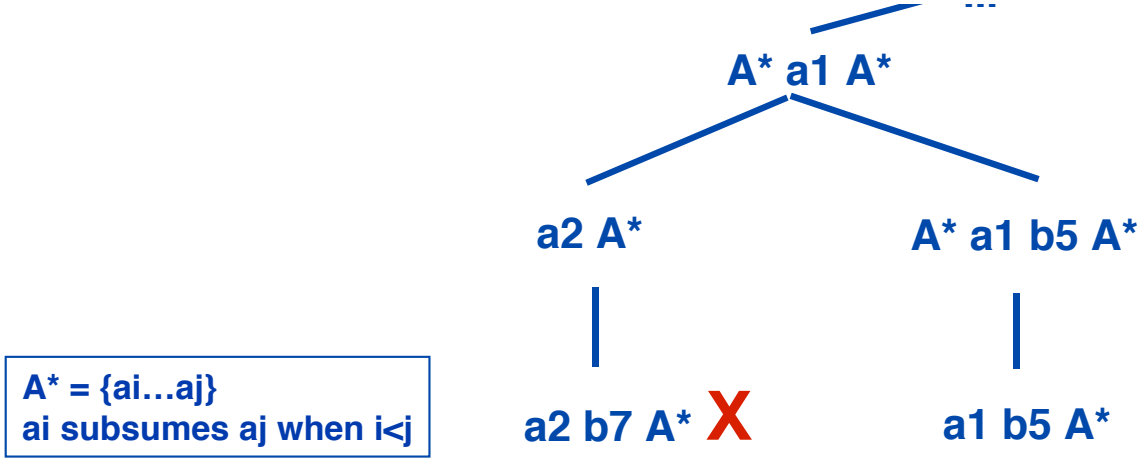

Figure 6. The search for a plan in SUDO-PLANNER is guided by a strategy called *dominance proving*. Search nodes whose plan is dominated (i.e., subsumed) by other nodes are eliminated. Here, [a2 b7 A*] is subsumed by [a1 b5 A*], so the node n is eliminated from the search

The search algorithm uses plan subsumption to eliminate nodes from the search space. A node is eliminated if its plan is subsumed (dominated) by another node. New search nodes are created by adding constraints to a parent node, either by making a step more specific (according to the action taxonomy) or by eliminating a step. For example, a node with a plan A* a1 A* could be expanded to a plan a2 A* or A* a1 b5 A*.

Figure 6 illustrates how redundant paths are eliminated. In this example, a2 b7 A* is subsumed by a1 b5 A*, and thus eliminated.

SUDO-PLANNER showed that plan taxonomies are useful to reason about relations between plans. It also showed how action taxonomies, such as those used in CLASP, can be exploited to reason about subsumption of plans. SUDO-PLANNER had other features not described here, including uncertainty reasoning and partial goal satisfaction, policy constraints to relate actions to external events, conditional effects, and qualitative probabilistic networks.

4 Goal Taxonomies: EXPECT

An important issue in reasoning about plans, processes, and activities is the description of the desired goals (or objectives) as well as which actions (or tasks, or agents) have the capability to achieve them. Typically, goals are described as a flat predicate with a predicate name and several arguments and only limited reasoning is done about them. The EXPECT architecture [Gil 94; Gil and Melz 96; Blythe et al 01], build on top of the LOOM description logic system [MacGregor 91], uses a structured representation of capabilities that has been useful in several tasks and tools, including a problem solver, a plan editor, a plan evaluation and critiquing tool, and an agent matchmaker. The main features of this approach are the declarative representation of

qualification parameters (in addition to data passing parameters) for goals and capabilities, and flexible matching techniques that go beyond exact goal match, such as goal subsumption and reformulation. A theme of the work on EXPECT has been that the representations be understandable to users, based on earlier work on explanation [Swartout 91; Swartout and Moore 93]. Many of the applications developed with EXPECT are plan evaluation and critiquing systems that support human planners in several military domains.

EXPECT is a reasoning system that supports acquisition of problem-solving knowledge through a number of different techniques. These include maintaining a dependency model of any knowledge-based system (KBS) that is built with EXPECT, scripting tools that can guide a user through a multi-step modification to a KBS and the use of background knowledge about generic tasks. Here we focus on EXPECT 's representation of tasks and subtasks within a KBS. More details about the overall reasoning and EXPECT's knowledge acquisition tools can be found in [Blythe et al 01].

EXPECT's representation of goals and capabilities is inspired in natural language work. They are represented as verb clauses using a case-grammar style of formalism [Fillmore 68]. Each capability consists of a verb, that specifies what is to be done, and a number of roles, or slots, which specify the parameters to be used in the action. The parameters use terms that are defined in a domain ontology. For example, the goal of estimating the closure date of a particular transportation movement would be specified roughly as:

```
estimate OBJ closure-date OF transportation-movement-1
```

Here, `estimate` is the verb, and the roles are indicated in upper case. The roles are filled by concepts and instances taken from the domain ontology.

Roles can be filled in several different ways, which allows considerable flexibility in specifying a task to be performed. A role can be filled by a specific instance:

```
add OBJ 3 TO 5
```

which allows us to specify particular instances that are to be used in an action. A role can be filled by a concept:

```
compute OBJ (spec-of factorial) of 7
```

In this case, the concept `factorial` is used to specify the kind of task that is to be performed. The data required to perform the computation are specified as parameters (in this case the number 7), while these additional task parameters allow us to express what needs to be done with that data in an explicit way and are not strictly necessary to perform the computation itself. The fact that roles can be used both to specify the parameters or objects that will be involved in a task and to further describe or specify the task itself is one of the key capabilities that this representation supports, providing a rich language for specifying goals.

Roles can be a type of an instance, as in:

```
divide OBJ number BY 2
```

This expresses a generic goal that can be instantiated with any elements of that type.

Roles can also be filled by extensional sets as in:

```
find OBJ (spec-of maximum) OF 42 2 99)
```

or they can be filled by intensional sets, where the set is described by a concept:

```
find OBJ (set-of (spec-of violated-constraint))
     IN configuration
```

Finally, it is possible to use descriptions (which are similar to the definitions of Loom concepts) in roles:

```
estimate OBJ support-personnel
         IN (and location (exactly 0 seaports))
```

This is a goal to estimate the support personnel in a location with no seaports.

Goals and capabilities are translated into LOOM definitions, following an algorithm described in [Gil and Gonzalez 96]. For example, `(compute (obj (spec-of factorial)) (of (5 7)))` is translated into:

```
(defconcept compute-factorial-of-numbers
  :is (:and compute
            (:the obj (:and concept-description factorial))
            (:the of (:and number extensional-instance-set
                           (:filled-by instance-name 5)
                           (:filled-by instance-name 7)))))
```

LOOM's classifier is now able to reason with this definition. Every term used in the parameters have their own definitions, provided in the ontologies, and LOOM will use them in reasoning about capability subsumption. Notice that these terms and their definitions can be domain independent (e.g., violated-constraints, maximum) or domain dependent (e.g., location, closure-date).

Given a set of capabilities expressed in this language and the corresponding LOOM definitions, LOOM's classifier reasons about these definitions and places them in a lattice, where more general definitions subsume more specific ones. An example is shown in Figure 7. Notice that subsumption reasoning uses the definitions of the domain terms and ontologies that are contained in the domain knowledge bases. As a result, the capability to ``move cargo with a vehicle'' will subsume one to ``move cargo with an aircraft'', because according to the domain ontologies vehicle subsumes aircraft. The

capabilities are automatically organized according to their definitions, and they can be compared based on their place in the lattice.

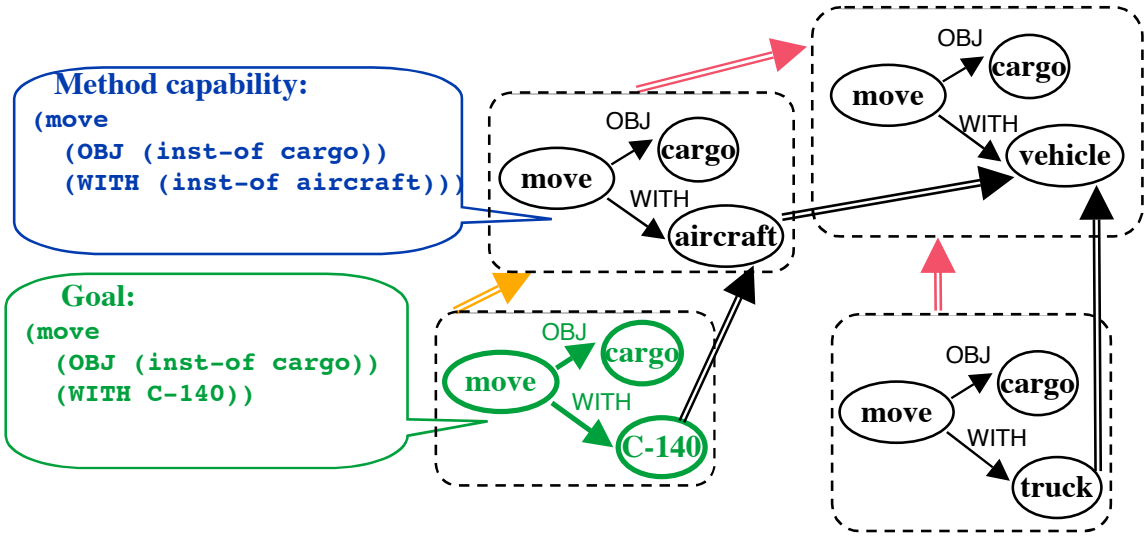

Figure 7. Goal matching in EXPECT using the LOOM classifier.

Subsumption matching can help find suitable capabilities when presented with a query, but in some cases no subsuming capability has been added to the knowledge base. In these cases it may be possible to fulfill the request by decomposing it expressing it in different terms. This allows a more flexible matching than is possible if one required an exact match for goals and methods. EXPECT supports several types of reformulations:

- A *covering reformulation* is a form of divide and conquer. It transforms a goal into a set of goals that partition the original goal. If all the goals in the set are achieved, the intent of the original goal is achieved. For example, suppose a goal of estimating support personnel has been posted, but no applicable methods have been found. Suppose further that the domain ontology indicates that the concept support personnel is partitioned into unloading personnel, seaport support personnel and airport support personnel. The original goal can then be reformulated into three new goals to estimate each type of personnel in the partition.

- A *set reformulation* is like a covering reformulation except that it involves a goal over a set of objects which is reformulated into a set of goals over individual objects.

- An *input reformulation* is somewhat similar to the support that some languages provide for polymorphic operators. This kind of reformulation occurs when a goal is specified with a general parameter and no single method is available at a sufficiently general level to handle the parameter. In that case, the goal can be reformulated into cases based on the subtypes of the parameter given in the ontology.

Goal reformulations allow us to state the description of method capabilities more independently from the statement the descriptions of the goals that are posted by other methods or by the user. The benefit is a more loosely coupling between methods and tasks, i.e., between what is to be accomplished and what are possible ways to accomplish it.

These structured representations of goals and capabilities have been used in three different and related contexts that require reasoning about goals: problem-solving goals, planning objectives, and agent capabilities.

Problem-solving knowledge can be represented in EXPECT consists of set of methods. Each method has a capability that declares what task can be achieved by the method, a body that describes how the capability is achieved and a return type that characterizes what the method produces. The method body is written in a programming language that includes basic constructs such as a conditional test and can also include other goals. These goals may be matched by the capabilities of other methods, in which case they will be used when the method is applied, resulting in a tree structure of methods. EXPECT method capability descriptions for methods are specified in a similar way to goals, except that variables may appear in the capability descriptions. These are bound when the capability descriptions are matched with goals. Because it uses structured representations of method capabilities is, EXPECT can reason about how different methods relate to each other. This is useful for organizing method libraries as well as to support the acquisition of new problem-solving methods. These representations also support natural language paraphrasing, which is useful to develop adequate knowledge acquisition tools accessible to end users with no logic or programming background.

A second use of EXPECT's goal representation is to describe steps or tasks in domain plans. Understanding tasks and reasoning about how they are subsumed by types of tasks can be useful to express concisely properties of those types of tasks, such as their duration or their cost. For example, INSPECT [Valente et al 99] is a knowledge-based system built with EXPECT that analyzes a manually created air campaign plan and checks for commonly occurring plan flaws, including incompleteness, problems with plan structure, and unfeasibility due to lack of resources. Given a plan as a set of tasks and objectives, INSPECT would point out, for example, that if one of the objectives in the plan is to gain air superiority over a certain area then there is a requirement for special facilities for storing special fuel that is currently not taken into account. By reasoning about the kinds of objectives in the plan, INSPECT could notice that gaining air superiority requires providing reconnaissance missions, which are typically flown in aircraft that need specific kinds of fuel that need to be stored in special facilities.

A third application area of EXPECT's structured representations of goals is agent matchmaking, used in the PHOSPHORUS system within the Electric Elves architecture [Gil and Ramachandran 01; Chalupsky et al 02]. Multi-agent architectures typically offer matchmaking services that an agent can query to find what other agents can perform a given task. For example, a route planning agent may invoke threat detection agents in order to make a safe choice among all possible routes. Typically, simple string matching suffices since the agent communities are relatively small and the agents that need to issue a request can be told beforehand what other agents are available and how they have to be invoked. In addition, most current multi-agent systems assume that an agent can perform a few tasks (often just a single task), where the advertisements and invocations of agents are negotiated in advance by the agent designers and thus can be significantly simplified. In large and heterogeneous communities of agents, where the agent that formulates the request would have no idea of whether and how another agent has advertised relevant services, there is a need for more sophisticated matchmaking mechanisms. The kinds of goal representations used in EXPECT provide a richer language for advertising the capabilities of agents and would support more flexible matching algorithms. In PHOSPHORUS, the agent capabilities are translated into Loom descriptions as described earlier. The matchmaker uses subsumption, reverse subsumption, and several kinds of reformulations to find agents relevant to a request.

5 Other Relevant Work

Action and plan taxonomies that do not utilize description logics have been used for case-based planning [Alterman 86], plan generation [Tenenberg 89], and plan recognition [Kautz 91]. The advantage of using description logics is that the taxonomy can be constructed automatically by the system based on the class descriptions provided instead of being built by hand which is more time consuming and prone to error.

Action taxonomies have also been developed based on linguistic theories. [Di Eugenio 94], Uses Jackendoff's Conceptual Structure primitives [Jackendoff 90] and maps them to class descriptions for example carry can be defined as a kind of move with a physical means of taking an object. [Di Eugenio 94] augments these descriptions with effects conditions, and substeps, and found them effective to interpret purpose clauses in natural language (eg. cut the square in half along the diagonal in order to make two triangles).

Subsumption has also been used to reason about actions in terms of their preconditions and effects. RAT [Heinsohn et al 91] represents preconditions and effects using a description logic restricted to conjunctions of feature restrictions and equality. An action can be applied to a state if the precondition expression subsumes the current world state. The expression in the effects results in the assignment of the values or restrictions in the features mentioned. WIP [Wahlster et al 93] is a multimodal presentation system that designs presentations using, the RAT planning framework. RAT enables WIP to detect that a plan can be shortened if some portion of it is subsumed by a subplan presented earlier.

Action and plan taxonomies can be defined taking into account temporal constraints. [Artale and Franconi 98] define a temporal description logic that incorporates Allen's interval relations [Allen 91] and that can be used to specify descriptions of actions and plans in terms of temporal relations to world states and other actions.

Plan taxonomies can also be defined using sets of constraints, including temporal constraints. T-REX [Weida and Litman 92] exploits action taxonomies and temporal networks of actions in order to compute plan subsumption. T-REX presents an approach to plan recognition where after observation of a new action instance the entire set of hypothesis is reworked as a result of the classification of the instance.

Description logics have also been used to query and index plan libraries. MRL [Koehler 96] is a case-based planning system that uses conjunctive expressions of preconditions and goals to be achieved by a plan, and retrieves relevant plans to a new case by querying the plan library about plans whose preconditions and/or goals subsume the new case.

TINO is a mobile robot that uses description logic to generate high-level plans [De Giacomo et al 96]. The representation of the domain includes *static axioms*, used to represent background knowledge that does not change as actions are executed, and *dynamic axioms* that represent the changes caused by the actions. Conditional plans are generated, and during execution different branches can be selected based on sensory feedback.

6 Summary and Future Prospects

Description logics have been applied to several areas of planning, including plan analysis, plan generation, plan recognition, and plan evaluation and critiquing. The applications range from military planning, to telephony systems, to mobile robot control. Subsumption reasoning and classification support sophisticated reasoning about general types of actions and plans than other planning research. In comparison with other planning research, these systems support more sophisticated reasoning about general types of actions and plans through subsumption reasoning and classification. Another advantage is that their plan representations are integrated with the description logic representations of the objects in the domain, in comparison with the more impoverished representations used in other planning research.

The challenge of reasoning about plans in knowledge-intensive environments is already being raised in military planning, physical sciences research, enterprise and process modeling and management, and space operations. As the planning community continues to tackle more practical and ambitious tasks, description logics will be able to provide the kinds of knowledge representation and reasoning capabilities that these applications require.

References

[Alterman 86]. Alterman, R. “An Adaptive Planner”. AAAI-86.

[Ankolenkar et al 02] A. Ankolenkar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, and K. Sycara. DAML-S: Semantic Markup for Web Services. In The First International Semantic Web Conference (ISWC), Sardinia (Italy), 2002.

[Artale and Franconi 98]. Artale, A. and Franconi, E. “A Temporal Description Logic for Reasoning about Actions and Plans”, Journal of Artificial Intelligence Research, Vol. 9, 1998.

[Baader et al 03] “The Description Logic Handbook: Theory, Implementation and Applications”. Edited by Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider. Cambridge University Press, 2003.

[Blythe et al 01] Jim Blythe, Jihie Kim, Surya Ramachandran, and Yolanda Gil. “An Integrated Environment for Knowledge Acquisition." Proceedings of the 2001 International Conference on Intelligent User Interfaces (IUI-2001), Santa Fe, New Mexico, January 2001.

[Brachman et al 91] Ronald Brachman, Deborah McGuinness, Peter Patel Schneider, Lori Alperin Resnick and Alexander Borgida. "Living with CLASSIC: When and How to Use a KL-ONE-like Language", In Principles of Semantic Networks --- Explorations in the Representation of Knowledge, John F. Sowa (Ed). Morgan Kaufmann, 1991.

[Chalupsky 02] H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ, and M. Tambe. “Agent Technology to Support Human Organizations”, AI Magazine, Vol 23, No 2, Summer 2002.

[De Giacomo et al 96] Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, and Riccardo Rosati. “Moving a robot: the KR&R approach at work” In Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning, 1996.

[Devanbu et al 91]. Devanbu, P.T., Brachman, R.J. Selfridge, P.G., and Ballard, B.W. “LaSSIE: A Knowledge-Based Software Information System”, Communications of the ACM, Vol. 34, No. 5, 1991.

[Devanbu and Litman 94]. Devanbu, P.T., and Litman, D.J. “Taxonomic Plan Reasoning”.

[Di Eugenio 94]. Di Eugenio, B. “Action Representation for Interpreting Purpose Clauses in Natural Language Instructions”. KR-94.

[Di Eugenio and Webber 92] De Eugenio, B. and Webber, B. “Plan Recognition in understanding Instructions”. AIPS-92.

[Fillmore 68] Charles J. Fillmore. “The case for case”. In Emmon Bach and Robert T. Harms, editors, Universals in Linguistic Theory, pages 1 88. Holt, Rinehart and Winston, Inc., 1968.

[Fikes and Nilsson 71] R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, 2(3-4):189--208, 1971.

[Gil 94] Yolanda Gil. "Knowledge Refinement in a Reflective Architecture." Proceedings of the Twelfth National Conference of Artificial Intelligence (AAAI-94), Seattle, WA, August 1994.

[Gil and Melz 96] Yolanda Gil and Eric Melz. "Explicit Representations of Problem-Solving Strategies to Support Knowledge Acquisition." Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), Portland, OR, August 4-8, 1996.

[Gil and Ramachandran 01] Yolanda Gil and Surya Ramachandran. “PHOSPHORUS: A Task-Based Agent Matchmaker”, Proceedings of the Fifth International Conference on Autonomous Agents (Agents'01), short paper track. Montreal, Canada, May 28-June 1, 2001.

[Koehler 96] Koehler, J. “Planning from second Principles”. Artificial Intelligence Vol. 87, 1996.

[McDermott 02] McDermott, D. “Estimated-Regression Planning for Interactions with Web Services”. in Sixth International Conference in AI Planning Systems, 2002

[MacGregor 91] Robert M. MacGregor. "Using a Description Classifier to Enhance Deductive Inference". In Proceedings Seventh IEEE Conference on AI Applications, pp. 141-147, 1991.

[Moser 83] M. Moser, "An overview of NIKL, the new implementation of KL-ONE," In Research in Natural Language Understanding, Bolt, Beranek, and Newman, Inc., Cambridge, MA, 1983.

[Swartout et al., 1991] William R. Swartout, Cecile L. Paris, and Johanna D. Moore. Design for explainable expert systems. IEEE Expert, 6(3):58-64, June 1991

[Swartout and Moore 1993] Swartout, W. R., and Moore, J. D. Explanation in second-generation expert systems. In J.-M. David, J.-P. Krivine, and R. Simmons (Eds.), Second Generation Expert Systems, Springer-Verlag, 1993.

[Swartout and Gil 1995] Swartout, B. and Gil, Y. "EXPECT: Explicit Representations for Flexible Acquisition" in Proceedings of the Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'95) Banff, Canada, February 26-March 3, 1995.

[Tenenberg 89] Tenenberg, J.D. "Inheritance in Automated Planning". KR-89.

[Valente et al 99] Andre Valente, Thomas Russ, Robert MacGregor and William Swartout. "Building and (Re)Using an Ontology of Air Campaign Planning". In IEEE Intelligent Systems 14:1, pp. 27-36, Jan-Feb, 1999.

[Wahlster et al 93]. Wahlster, W., Andre, E., Finkler, W., Profitlich, H., and Rist, T. "Plan-Based Integration of Natural Language and Graphics Generation". Artificial Intelligence Vol. 63, 1993.

[Weida and Litman 92] Weida, R. and Litman, D., "Terminological Reasoning with Constraint Networks and an Application to Plan Recognition". KR-92.

[Weld 99] Weld, D. "Recent Advances in AI Planning". AI Magazine, 1999.

[Wellman 88] Wellman, M.P., "Formulation of Tradeoffs in Planning under Uncertainty". PhD Thesis, Department of Computer Science, MIT, 1988.