AI Magazine Volume 26 Number 2 (2005) (© AAAI)

Description Logics and Planning

Yolanda Gil

■ This article surveys previous work on combining planning techniques with expressive representations of knowledge in description logics to reason about tasks, plans, and goals. Description logics can reason about the logical definition of a class and automatically infer class-subclass subsumption relations as well as classify instances into classes based on their definitions. Descriptions of actions, plans, and goals can be exploited during plan generation, plan recognition, or plan evaluation. These techniques should be of interest to planning practitioners working on knowledge-rich application domains. Another emerging use of these techniques is the semantic web, where current ontology languages based on description logics need to be extended to reason about goals and capabilities for web services and agents.

Most real-world planning applications are naturally knowledge rich. Yet planning research to date has concentrated on languages and algorithms to represent and search through decision spaces in order to generate feasible plans. Typical planning systems use very small amounts of knowledge, and the representations of the domain or the planning tasks are typically not very expressive. A challenging area of future research is the integration of existing planning algorithms with rich representations of domain-specific knowledge about planning tasks and objectives, actions and events as part of a plan, and the complex relations among them. The combination of these techniques will result in planning systems that will be better equipped to address more effectively the needs of real-world planning applications.

Recent interest in web services and their composition to create distributed applications also brings this topic to the forefront of research. Planning approaches have been used as a framework for web service specification and composition (Kim, Spraragen, and Gil 2004; McDermott 2002; Ankolenkar et al. 2002). The richer the representation of the capabilities of web services and of their relations to other services, the more automation can be brought to bear for web service composition. Rich representations of goals and capabilities would also support other purposes such as matching of requests to existing services. The semantic web vision (Berners-Lee, Hendler, and Lassila 2001) brings a renewed interest in reasoning about tasks, goals, and plans using rich domain descriptions.

This article gives an overview of different uses of expressive representations of planning knowledge, focusing on description logics as expressive knowledge representation frameworks with well-understood reasoning complexity and tractability. Although these uses have been investigated in the description logics arena, they have not been incorporated within state-of-the-art planning algorithms. I discuss four main uses of description logic to represent planning knowledge, specifically: (1) object taxonomies to reason about the planning state expressed with descriptions of the different types of objects in the domain; (2) action taxonomies to reason about action types at different levels of abstraction; (3) plan taxonomies to reason about plan subsumption of partially ordered plans; and (4) goal taxonomies to reason with expressive representation of goals and their parameters.

The article draws examples from three implemented systems. The systems are only partially described here, the reader is referred to the citations for more details on the approaches

used. One of these systems is CLASP (Devanbu and Litman 1996), a system developed to reason about action taxonomies and action networks. CLASP was applied to a telephony domain and was integrated with the LaSSIE software information system (Devanbu et al. 1991). This article also draws examples from SUDO-PLANNER (Wellman 1988), which exploited plan subsumption to control the search during plan generation. It was designed to reason about trade-offs in decision making under uncertainty and was developed for medical applications. 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. I also use examples from EXPECT (Gil 1994, Swartout and Gil 1995, Gil and Melz 1996, Blythe et al. 2001), an architecture for problem solving and reasoning that supports interactive acquisition of knowledge. EXPECT exploits structured representation of goals and capabilities to support sophisticated matching during problem solving as well as facilitating the generation of natural language paraphrases of problem-solving knowledge. The same goal and capability representations were used in the PHOSPHORUS agent matchmaker used in a multiagent system for an office environment (Gil and Ramachandran 2001, Chalupsky et al. 2002).

I will begin this article with a brief introduction to description logics. I then describe how to use description logics to reason about objects, actions, plans, and goals. I use examples taken from three implemented systems and also summarize along the way other related research that often integrates this work with other techniques in knowledge representation, planning, and natural language.

A Brief Introduction to Description Logics

This section gives a short and informal introduction to description logics. See the papers by Brachman and Levesque (2004) and Baader et al. (2003) or the dl.kr.org web site for a comprehensive introduction and overview of description logics, their formal underpinnings, existing implementations, and applications. The role of description logics in the semantic web and their use in web ontology language standards are discussed in the papers by Antoniou and van Harmelen (2004); Horrocks, Patel-Schneider, and van Harmelen (2003); and Baader, Horrocks, and Sattler (2003).

Description logics are an extension of frame-based systems that can express definitions of classes and relations. Description logic languages differ in expressivity, which determines the computational complexity of the reasoning algorithms for each language. Class definitions can include disjunction and negation as well as constraints on the relations to other classes. A *relation* between a class (its *domain*) and another class (its *range*) can be constrained in cardinality and type. Relations can also be given definitions and therefore have subclasses as well. Class *partitions* can be defined by specifying a set of subclasses that represent the partitions and can be *exhaustive* if all instances of the class belong to some partition and *disjoint* if there is no overlap in the subclasses. A class can be denoted as a *primitive class* and not given a definition, and in that case its subclasses and instances must be explicitly indicated.

Description logic systems use these definitions to automatically organize class descriptions in a taxonomic hierarchy and automatically classify instances into classes whose definitions are satisfied by the features of the instance. Specifically, description logic reasoners provide two key capabilities: (1) *class subsumption,* where a class C1 subsumes another class C2 if its definition includes a superset of the instances included in C2; and (2) *instance recognition,* where an instance belongs to a class if the instance's features (roles and role values) satisfy the definition of the class. Description logic systems also have mechanisms to detect inconsistent definitions and support reasoning through inheritance and default values. Because of multiple inheritance, the taxonomic hierarchy is often a lattice.

To illustrate the use of these different kinds of reasoning, I use a simple example that shows how to represent office meetings and then discuss how a meeting planner could exploit these representations. Figure 1 shows a representation, not complete for lack of space, of meetings and other events in a description logic. Classes are capitalized, and relations are shown in lowercase. An activity is defined as anything that has a start time, a day, a month, and a year. A work activity is defined as an activity that has a work location, which can be one of several given meeting rooms or an office. Anything that has a start time, a day, a month, a year, and two participants is considered to be a meeting. Conversely, any meeting must have those five features. Meetings can also have a project, a presentation, a duration, and a location, but these features are not necessary nor sufficient for something to be a meeting. Relations for presentations and meeting participants and presenters are also shown. Mtg7986 is specified

as an instance, and several roles and their ranges are given, but notice that no parent class is specified.

Given these definitions, a description logic system would make the following inferences, which are in some cases quite subtle in the way they exploit the definitions. The Meeting class is a subclass of the Activity class, given that the definition of a meeting is subsumed by the definition of an activity since it has a start time, day, month, and year. A meeting may have a presentation. The relation presenter is a subclass of the relation meeting-participant. Smith is a presenter in mtg7986 and therefore is a meeting-participant. Mtg7986 is an instance of the Meeting class, given that mtg7986 complies with the definition of a meeting because it has a start time, day, month, year, and two meeting participants.

Knowledge bases developed with description logics are more modular and maintainable, since the reasoners are able to derive the taxonomy automatically and therefore the descriptions of classes and instances do not need to take into account all other definitions in the knowledge base. Note that by not stating explicitly the class of the instance mtg7986 we can decouple the definitions of the classes and the instances and therefore develop a more modular and maintainable knowledge base.

Description logic systems also facilitate the development of knowledge bases by detecting inconsistent descriptions. The reasoners would detect that mtg8897 is not consistent, since a work activity cannot have a location that is not a conference room or an office. Another important capability of description logics is to support queries that in effect form new classes on the fly. The figure shows a query for any instances that are meetings of the Trellis project, which will return mtg7986 and any others that satisfy that definition.

Some well-known description logic systems include CLASSIC (Brachman et al. 1991) (used in CLASP); NIKL (Moser 1983) (used by SUDOPLANNER); LOOM (MacGregor 1991) (used by EXPECT); FaCT (Horrocks 1998); and RACER (Haarslev and Möller 2001).

Ontology languages for the semantic web based on description logics are now de facto World Wide Web Consortium (W3C) standards. OWL, the Web Ontology Language[1] includes in its specification three flavors of increasing expressivity and complexity. OWL-DL is a description logic built as an extension of (a subset of) RDF[2] with a reasonable level of expressivity and at the same time there already are highly optimized implemented systems for its language. OWL-Lite is a simpler subset of

```
(defconcept Activity
    :is (:and Thing
              (:exactly 1 start-time)
              (:exactly 1 day)
              (:exactly 1 month)
              (:exactly 1 year))
    :constraints (:some duration)
    :exhaustive-partition Work-Activity Non-Work-Activity)

(defconcept Work-activity
    :is (:and Activity
              (:the location Work-location)))

(defconcept Conference-room
    :is-primitive (one-of 9cr 11cr 12cr 4cr 10cr)

(defconcept Office
    :is-primitive (one-of 901 919 949 934 950 913)

(defconcept Work-location
    :is (:or Conference-room Office))

       (defconcept Meeting
   :is (:and Thing
             (:exactly 1 start-time)
             (:exactly 1 day)
             (:exactly 1 month)
             (:exactly 1 year)
             (:at-least 2 meeting-participant))
   :constraints (:and (:some topic)
                      (:some project)
                      (:at-most 1 duration)
                      (:at-most 1 location)))

(defrelation presentation
   :is (:and (:the domain Meeting)
             (:the range Document)))

(defconcept Document :is-primitive thing)

 (defrelation owner
   :is (:and (:the domain thing)
             (:the range Person)))

(defrelation presenter
    :is (:composite-relation presentation owner))

(defrelation meeting-participant
   :is (:or (:and (:the domain Meeting)
                  (:the range Person))
     presenter))

(instance mtg7986
  :start-time 1500
  :day 12 :month December  :year 2003
  :project Trellis
  :presentation Trellis-kickoff-slides.pdf
  :meeting-participant Diaz
  :location 9cr)

(instance Trellis-kickoff-slides.pdf Document
  :owner Smith)

(instance mtg8897 Work-activity  :location Street)

(query ?TM (instance ?TM
              (:and Meeting
                    (:the project Trellis))))
```

Figure 1. Some Descriptions to Represent Meetings in a Description Logic.

OWL-DL. OWL-Full extends OWL-DL with additional constructs, extending on RDF as well. OWL and its predecessors have made description logic systems widely accessible and used at unprecedented levels.[3]

The following sections illustrate some of the uses of description logics in the context of diverse planning tasks and application domains.

Reasoning about Object Taxonomies

Planning systems often reason about how actions transform an initial state of the world into a final state, changing the domain objects as actions are executed. State information is often expressed as a set of ground predicates, in some cases organizing objects into small type hierarchies. Expressive representations and reasoners can be used in planning systems to define complex domain terms and objects. For example, a meeting planner could reason about events and differentiate meetings from other activities and pose dynamically formulated queries about new kinds of meetings with different combinations of features, such as meetings relevant to specific projects, using the definitions shown in figure 1. These descriptions can also be used to reason about actions based on types of objects. For example, a meeting planner could use different criteria to handle meetings that are relevant to multiple projects.

CLASP, SUDO-PLANNER, and EXPECT all represented the objects in their respective domains in description logics, and as a result they were able to represent actions, plans, and goals in terms of object classes and the kinds of reasoning that can be supported with them. State-of-the-art planning algorithms could be extended with state descriptions expressed in description logics and could support precondition matching through description logic reasoners.

The TINO mobile robot uses description logic to generate high-level plans (De Giacomo et al. 1996). 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.

The representations of actions using the representation of actions (RAT) using terminological logics planning framework (Heinsohn et al. 1992), used to design multimodal presentations in WIP (Wahlster et al. 1993), 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.

Artale and Franconi (1998) define a temporal description logic that incorporates Allen's interval relations (Allen 1984) and that can be used to specify descriptions of actions and plans in terms of temporal relations to world states and other actions.

Reasoning about Action Taxonomies

Actions can be described at many levels of abstraction. For example, driving and flying are both moving actions with specific means of locomotion. Walking is also a moving action that does not require a transportation vehicle. If we describe moving actions in terms of their properties and constraints, they can be organized in a taxonomy that will then enable more efficient reasoning about actions through abstract classes. Simple action taxonomies that do not utilize description logics have been used for case-based planning (Alterman 1986), plan generation (Tenenberg 1989), and plan recognition (Kautz 1991). 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. This can make the taxonomies and the approaches more scalable to problems with large numbers of complex actions.

I illustrate this idea with CLASP. CLASP used a STRIPS-like representation (Fikes and Nilsson 1971) of actions in the plan and assumed a propositional representation of planning problems with conjunctive expressions of preconditions and states. Figure 2 shows the core definition of actions and states, as well as some example actions and states in the telephony switching domain. The core definition of an action shows that it has actors, preconditions, and add and delete lists. Domain-specific types of states and actions are described using the corresponding core definitions. A system action is defined as any action performed by the system (not by a user). A connect dialtone action is defined as an action performed by a system that provides a dialtone when the telephone is off the hook and idle. Specific states and actions are created as instances of the classes defined in these taxonomies. 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 used CLASSIC's classifier to reason about action taxonomies. For example, it would infer that Connect-Dialtone-Act is a subclass of System-Act based on their definitions. It would also infer that connect-dialtone-on-u1 is a System-Act, given that a switching system is a kind of system and is the agent of that action.

In SUDO-PLANNER, 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 3 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.

Action taxonomies are particularly useful for organizing plan taxonomies, a topic I will discuss in the next section.

Reasoning about Plan Taxonomies

Plan taxonomies can be built in various ways. This section illustrates how plan taxonomies can be built based on the actions that compose the plan, based on the initial conditions and goals achieved, based on how they change the initial state over time. Because the actions in the plan can be related through complex constructs, reasoning about plan subsumption is often done by extending a description logic with additional mechanisms.

Plan taxonomies have a variety of uses in planning. Plan taxonomies can support several aspects of reasoning about plans, including organization of plan classes, retrieval of plan types and instances with description-based queries, and validation of plans based on descriptions of valid classes of plans. This should be very useful in applications in which large amounts of complex plan instances need to be managed. For example, planning systems could retrieve relevant plans based on description-based queries that refer to types of features of the current planning problem.

One use of plan taxonomies is to assist during plan generation. Some plans can be described as a network of actions, and hierarchical task network (HTN) planning approaches use these networks to generate plans (Ghallab, Nau, and Traverso 2004). Plan taxonomies could help organize plans based on the types of actions used within these decompositions. Other planning approaches use plan space search by incrementally adding new actions to the plan (Weld 1999). Reasoning about how two plans relate to one another is important in order to ensure that any 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

```
(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-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))

(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 2. Core Definitions and Examples of Actions and States in CLASP (from Devanbu and Litman [1996]).

```
(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 ...))
```

Reproduced with permission from Michael Wellman.

Figure 3. Examples of Action Descriptions in SUDO-PLANNER (from Wellman 1988).

can map out the search space that they explore in an organized, comprehensive, and nonoverlapping way. Typically in planning algorithms two plans are considered to be related based on the specific steps and links that they include. These algorithms could use plan taxonomies to relate plans in more sophisticated ways, exploiting different levels of abstraction of actions and plans as well as definitions of aggregate steps and domain knowledge.

CLASP used plan taxonomies to organize, validate, and retrieve plans in a library. Plans are defined in CLASP's language, an extension to CLASSIC. Plans are described as networks of actions that achieve a goal from a given initial state. The action networks, denoted as PLAN-EXPRESSION, are partially ordered plans that include iteration and branching. 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 subnetworks. Figure 4 shows the core definition of a plan, as well as some domain plans to illustrate these constructs. Subtypes of the class plan can be defined to create a taxonomy of plan types. For example, a plan for plain old telephone service (POTS) can be defined as one in which a caller picks up the telephone 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 are called *scenarios,* and they reflect different linearized sequences of actions that can be executed in the world. Figure 4 also shows a scenario in which the caller picks up the phone, gets a dialtone, dials and gets a busy signal, and hangs up causing the system to disconnect.

CLASP supported subsumption and classification of plans and scenarios by extending the 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, that is, 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.

MRL also used description logics to query and index plan libraries (Koehler 1996). MRL 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. Plans in the library are indexed by features that reflect the main properties of the problem. A plan in the library is considered more specific than another if its index is more specific.

SUDO-PLANNER used plan taxonomies to guide plan generation using plan space search (Weld 1999). It represented plans as partially ordered sets of actions and used plan subsumption to detect nodes in the search that are redundant with others and therefore need to be eliminated. Plan subsumption was evaluated through bipartite graph matching. Figure 5 illustrates some examples of plans shown as sequences of three steps (for example, P is [a1 a2 a3]). To simplify things, we will denote actions by letters with subscripts where subsumers have a lower number (for example, a1 subsumes a5, b2 subsumes b3). The lines show subsumption of individual steps (for example, 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. The search algorithm used plan subsumption to eliminate redundant 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. We will denote a partial plan (including the null plan) as A*. 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 handled. In this example, a2 b7 A* is subsumed by a1 b5 A*, and thus eliminated.

Plan taxonomies can also be defined using sets of constraints, including temporal constraints. T-REX (Weida and Litman 1992) exploits action taxonomies and temporal networks of actions in order to compute plan subsumption. T-REX presents an approach to plan recognition in which, after observation of a new action instance, the entire set of hypotheses is reworked as a result of the classification of the instance.

Reasoning about Goal Taxonomies

An important issue in reasoning about plans, processes, and activities is the description of the desired goals (or objectives or tasks) as well as which actions (or procedures or agents) have the capability to achieve them. In planning systems, goals and capabilities are typically described as a predicate with a name and several arguments and are matched through straightforward variable unification. Goal taxonomies could be used to support more flexible matching approaches exploiting subsumption to relate otherwise disparate descriptions. This is especially important given the recent emphasis on distributed approaches, in which planners, agents, or services need a certain goal accomplished by others, but each may have its own way to describe the goal.

EXPECT uses goal taxonomies for matching. A theme of the work on EXPECT is that the representations in a knowledge base should be understandable to end users, since a user needs to understand how a system is solving a problem before embarking on adding new knowledge. As a result, EXPECT's representation of goals and capabilities is inspired in earlier natural language work (Swartout, Paris, and Moore 1991; Swartout and Moore 1993). They are represented as verb clauses using a case-grammar formalism (Fillmore 1968). Each capability consists of a verb that specifies what is to be done and a number of cases that specify the roles of the objects. Each case is effectively a parameter and is specified with a role name and a type using terms that are defined in a domain ontology. A typical role is a direct object (denoted as OBJ); other role names are often prepositions.

```
(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)))))

(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))))

(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))))
```

Figure 4. Core Definition and Examples of CLASP Plans (from Devanbu and Litman [1996])

Figure 7 shows some examples of how goals and capabilities are represented in EXPECT. With the definitions given for move, move cargo by air, and airlift, the system determines that move cargo by air and airlift are equivalent. It will also infer that mv23 is an airlift, and the query for airlift instances will return mv23.

An important feature is the declarative representation of qualification parameters (in addition to data-passing parameters) for goals and capabilities. Qualification parameters express

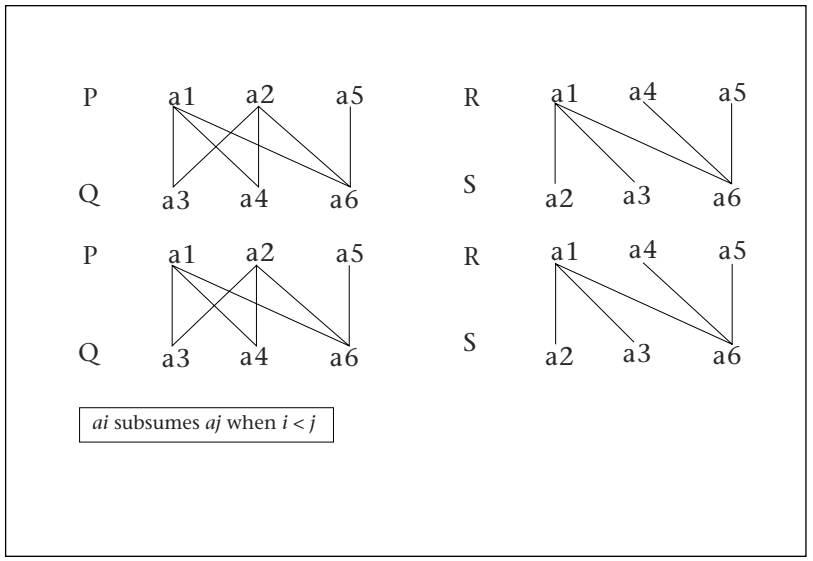

Figure 5. Plan Subsumption in SUDO-PLANNER Can Be Viewed as Bipartite Graph Matching.

Plan P subsumes plan Q since 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.

what needs to be done with data parameters in an explicit way but are not strictly necessary to carry out operations on the data parameters. Both data and qualification parameters may be single instances or concepts or sets of instances or concepts that may be intensionally or extensionally specified. For example a goal to compute the factorial of a number would be represented with a qualification parameter "factorial" and a data parameter that would be the number. The parameters in goals and capabilities may be of the following types:

a specific instance, for example, the USS *Coronado*, represented as (the-instance USS-Coronado)

an abstract concept to specify a qualification parameter, for example, air superiority, represented as (the-concept air-superiority)

an instance type, for example, barge, represented as (barge)

a concept type, which includes all its subtypes, for example, command-and-control-structure, represented as (subtype-of command-and-control-structure)

a set of instances, intensionally or extensionally specified, for example, Mexico and Canada, represented as (the-set Mexico Canada)

a set of concepts, intensionally or extensionally specified, for example, types of cargo can be referred to as (the-set Breakbulk ContainerCargo) or intensionally as (set-of (subtype-of Cargo)).

A posted goal or objective during problem solving could be to allocate cargo to a set of ships, represented as:

(assign (OBJ (the-set Breakbulk ContainerCargo))) (TO (the-set sh1 sh2 sh3)))

A method with the following capability could be applied to achieve this objective:

(assign (OBJ (set-of (subtype-of Cargo))) (TO (ship))

This is done by a matcher, which automatically translates goals and capabilities into LOOM definitions following an algorithm described by Gil and Gonzalez (1996). The bottom of figure 8 shows the concepts that would be created by the system to reflect the capability and goal just discussed. LOOM's classifier then reasons about these definitions and places them in a lattice, where more general definitions subsume more specific ones. An example is shown in figure 8. Notice that subsumption reasoning uses the definitions of the domain terms (for example, definitions of vehicles, aircrafts, and trucks) in order to build the lattice. 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.

When a goal arises during problem solving, subsumption-based matching can help find suitable capabilities, but in some cases no subsuming capability may be available. In these cases it may be possible to fulfill the goal by decomposing it and 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* transforms a goal into a set of goals that partition the original goal based on subclass partitions. If all the goals in the set are achieved, the intent of the original goal is achieved. For example, suppose a goal of moving cargo has been posted, but no applicable methods have been found. Suppose that cargo is partitioned into several subcategories such as breakbulk, container, and so on. and that there are methods to move each of these categories; the original goal can then be reformulated into several new conjunctive goals to move each type of cargo in the partition.

A *set reformulation* is like a covering reformulation except that it involves a goal over a set of objects that 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 avail-

able at a sufficiently general level to handle the parameter. In that case, the goal can be reformulated into disjunctive subgoals based on the subtypes of the parameter given in the ontology.

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 that can be represented in EXPECT consists of a 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 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, 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 this kind of structure in the goal representation is to describe steps or tasks in plans that accomplish those goals. Understanding tasks and reasoning about 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. 1999) is a knowledge-based system for plan evaluation and critiquing 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, it 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. This was done by reasoning about the kinds of objectives (goals) in the plan and their requirements based on their type and expressed through the goal taxonomy.

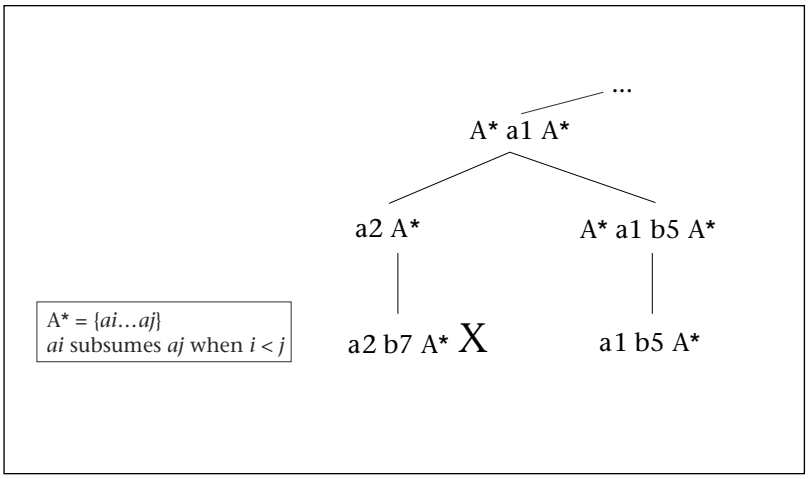

Figure 6. The Search for a Plan in SUDO-PLANNER Is Guided by a Strategy Called Dominance Proving.

Any search node whose plan is dominated (that is, subsumed) by a plan in another node is eliminated. Here, [a2 b7 A*] is subsumed by [a1 b5 A*], so its node is eliminated from the search.

A third use of this kind of structured representations of goals is agent matchmaking. This was used in the PHOSPHORUS system within the Electric Elves architecture (Gil and Ramachandran 2001, Chalupsky et al. 2002). Multiagent 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 multiagent 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

```
(defconcept Move
   :is (:and Action
             (:some obj Cargo)
             (:some with Vehicle)))

(defconcept Vehicle
   :is-primitive (:partition Aircraft Ship))

(defconcept Move-cargo-by-air
   :is (:and Move
             (:some obj Cargo)
             (:all with Aircraft)))

(defconcept Airlift
   :is (:and Move (:all with Aircraft)))

(defconcept Cargo
   :is-primitive thing)

 (defconcept Breakbulk
   is-primitive Cargo)

(defconcept ContainerCargo
   :is-primitive Cargo)

(instance mv23 Move
    (obj container23)
    (with C-140)
    (on 7-25-04))

(query ?A (instance ?A Airlift))

(defconcept AssignBB&C
   :is (:and Assign
             (:the obj (:and Extensional-Concept-Set
(:filled-by Concept-name Breakbulk)
(:filled-by Concept-name Container)
             (:the to (:and ship Extensional-instance-set
(:filled-by Instance-name sh1)
(:filled-by Instance-name sh2)
                               (:filled-by Instance-name sh3)))


(defconcept z98
   :is (:and Assign
             (:the obj (:and Cargo Intensional-Concept-Set))
             (:the to (:and ship Intensional-Concept))))
```

Figure 7. Examples of Goal and Capability Representations in EXPECT.

The last two concepts are automatically created by the matcher to represent a capability and a posted goal, respectively, that are related through subsumption.

LOOM descriptions as described earlier. The matchmaker uses subsumption, reverse subsumption, and several kinds of reformulations to find agents relevant to a request.

Other goal and action taxonomies have been developed based on linguistic theories. Di Eugenio and Webber (1992) use Jackendoff's Conceptual Structure primitives (Jackendoff 1992) 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 (1994) augments these descriptions with effects, conditions, and substeps and found them effective to interpret purpose clauses (for example, cut the square in half along the diagonal in order to make two triangles) in natural language.

In summary, structured representations of goals and capabilities support complex subsumption-based matchmaking and goal reformulations. An important benefit of goal taxonomies is loosely coupling between goals and capabilities, that is, between what is to be accomplished and what are possible ways to get it accomplished. This is a key feature as planning systems scale up and move towards distributed frameworks.

Summary and Future Prospects

Description logics have been used in several aspects of planning, including plan analysis, plan generation, plan recognition, plan retrieval, and plan evaluation and critiquing. Through expressive class descriptions and subsumption reasoning, description logic systems can support sophisticated queries about planning knowledge based on object taxonomies, action taxonomies, plan taxonomies, and goal taxonomies. These taxonomies can be combined to create powerful abstractions of planning knowledge. Description logic languages and systems can be extended to support temporal and control constructs that are central to planning problems. By incorporating these techniques, state-of-the-art planning research would be better positioned to tackle the challenges of reasoning about plans in knowledge-intensive environments in military planning, enterprise and process modeling and management, scientific research, and space operations. As the planning community continues to tackle more practical and ambitious tasks, description logics are an important ingredient to scale up and provide the kinds of knowledge representation and reasoning capabilities required by these applications.

Description logics are now central to the semantic web vision, since the adopted web ontology language OWL is based on description logics. The combination of description logics and planning techniques becomes directly relevant to reasoning about web services and agents in terms of their goals and capabilities. A knowledge-rich web with semantic underpinnings, where many simple tasks are automated and more complex tasks can be automated through their composition, is unlikely to be a reality without building on the techniques described in this article.

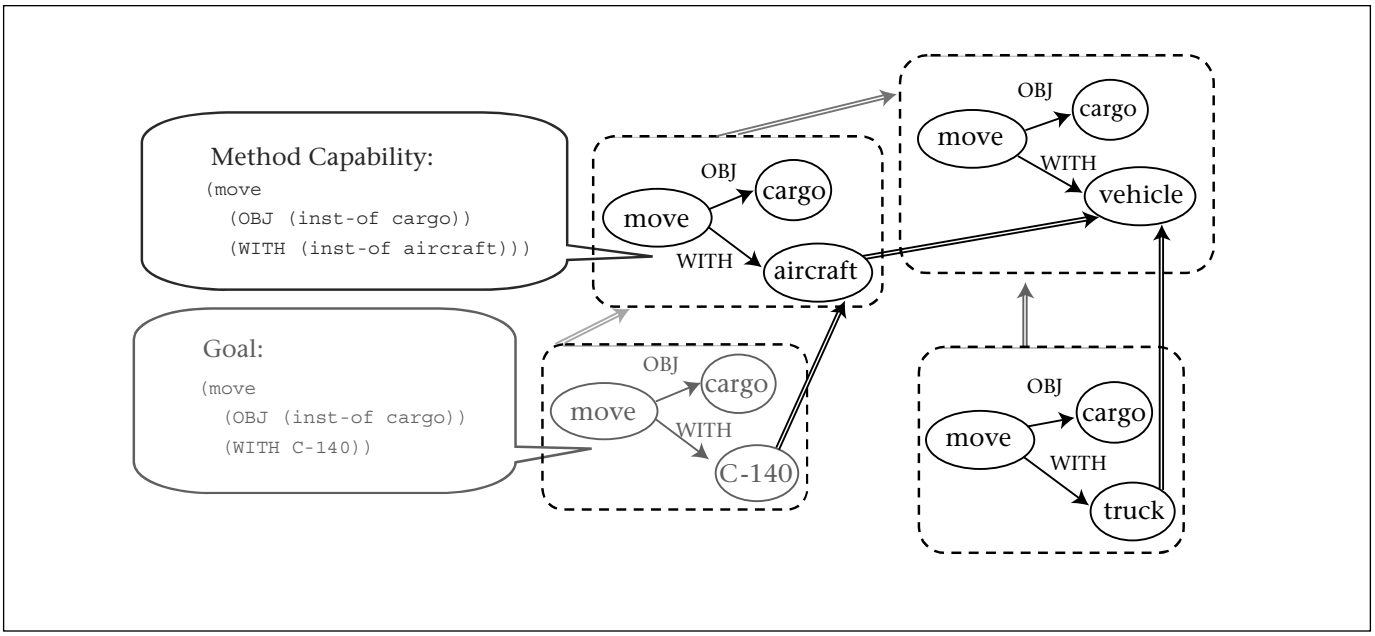

Figure 8. Goal Matching in EXPECT Using the LOOM Classifier.

Acknowledgements

The author gratefully acknowledges the Interactive Knowledge Capture group and other researchers past and present at USC/ISI for many fruitful discussions over the years, as well as the students of the graduate course on Artificial Intelligence Planning at the University of Southern California and *AI Magazine* reviewers for feedback on the article. This work was supported in part by the DARPA Personalized Agents that Learn (PAL) program under contract number NBCHD030010.

Notes

1. See the OWL Web Ontology Language Overview edited by Deborah McGuinness and Frank van Harmelen (http://www.w3.org/TR/owl-features).

2. RDF Schema, edited by Dan Brickley and R. V. Guha (http://www.w3.org/TR/rdf-schema/).

3. See http://www.w3.org/2001/sw/WebOnt/ for good introductory materials, methodology for practical use, and pointers to implemented tools such as editors, parsers, and reasoners.

References

Allen, J. 1984. A General Model of Action and Time. *Artificial Intelligence* 23(2): 123–154.

Alterman, R. 1986. An Adaptive Planner. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (AAAI-86). Menlo Park, CA: AAAI Press.

Ankolenkar, A.; Burstein, M.; Hobbs, J. R.; Lassila, O.; Martin, D. L.; McDermott, D.; McIlraith, S. A.; Narayanan, S.; Paolucci, M.; Payne, T. R.; and Sycara, K. 2002. DAML-S: Semantic Markup for Web Services. In *Proceedings of the First International Semantic Web Conference* (ISWC'02). Lecture Notes in Computer Science 2342. Berlin: Springer.

Antoniou, G., and van Harmelen, F. 2004. *A Semantic Web Primer*. Cambridge, MA: The MIT Press.

Artale, A., and Franconi, E. 1998. A Temporal Description Logic for Reasoning about Actions and Plans. *Journal of Artificial Intelligence Research* 9: 463–509.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. New York: Cambridge University Press.

Baader, F.; Horrocks, I.; and Sattler, U. 2003. Description Logics as Ontology Languages for the Semantic Web. In Festschrift in Honor of Jörg Siekmann, Lecture Notes in Artificial Intelligence, ed. Dieter Hutter and Werner Stephan. Berlin: Springer-Verlag.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The Semantic Web. *Scientific American* 284(5) (May): 34–43.

Blythe, J.; Kim, J.; Ramachandran, S.; and Gil, Y. 2001. An Integrated Environment for Knowledge Acquisition. In *Proceedings of the 2001 International Conference on Intelligent User Interfaces* (IUI-2001). New York: Association for Computing Machinery.

Brachman, R. J.; and Levesque, H. J. 2004. *Knowledge Representation and Reasoning*. San Francisco: Morgan Kaufmann Publishers.

Brachman, R.; McGuinness, D.; Patel-Schneider, P.; Alperin Resnick, L.; and Borgida, A. 1991. Living with CLASSIC: When and How to Use a KL-ONE-like Language. In *Principles of Semantic Networks—Explorations in the Representation of Knowledge,* ed. John F. Sowa. San Francisco: Morgan Kaufmann Publishers.

Chalupsky, H.; Gil, Y.; Knoblock, C. A.; Lerman, K.; Oh, J.; Pynadath, D. V.; Russ, T. A.; and Tambe, M.

2002. Electric Elves: Agent Technology to Support Human Organizations. *AI Magazine* 23(2): 11–24.

De Giacomo, G.; Iocchi, L.; Nardi, D.; and Rosati, R. 1996. Moving a Robot: the KR&R Approach at Work. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning*. San Francisco: Morgan Kaufmann Publishers.

Devanbu, P. T.; Brachman, R. J.; Selfridge, P. G.; and Ballard, B. W. 1991. LaSSIE: A Knowledge-Based Software Information System. *Communications of the ACM* 34(5): 34–49.

Devanbu, P. T., and Litman, D. J. 1996. Taxonomic Plan Reasoning. *Artificial Intelligence* 84(1–2): 1–35.

Di Eugenio, B. 1994. Action Representation for Interpreting Purpose Clauses in Natural Language Instructions. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning* (KR'94). San Francisco: Morgan Kaufmann Publishers.

Di Eugenio, B.; and Webber, B. 1992. Plan Recognition in Understanding Instructions. In *Proceedings of the First International Conference on AI Planning Systems* (AIPS-92). San Francisco: Morgan Kaufmann Publishers.

Fillmore. C. J. 1968. The Case for Case. In *Universals in Linguistic Theory,* ed. Emmon Bach and Robert T. Harms. New York: Holt, Rinehart, and Winston.

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

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. San Francisco: Morgan Kaufmann Publishers.

Gil, Y. 1994. Knowledge Refinement in a Reflective Architecture. In *Proceedings of the Twelfth National Conference of Artificial Intelligence* (AAAI–94). Menlo Park, CA: AAAI Press.

Gil,Y., and Gonzalez, P. A. 1996. Subsumption-Based Matching: Bringing Semantics to Goals. In *International Workshop on Description Logics* (DL-96). AAAI Technical Report WS-96-05. Menlo Park, CA: AAAI Press.

Gil, Y., and Melz, E. 1996. Explicit Representations of Problem-Solving Strategies to Support Knowledge Acquisition. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (AAAI-96). Menlo Park, CA: AAAI Press.

Gil, Y., and Ramachandran, S. 2001. PHOSPHORUS: A Task-Based Agent Matchmaker. In *Proceedings of the Fifth International Conference on Autonomous Agents* (Agents'01). New York: Association for Computing Machinery.

Haarslev, V., and Möller, R. 2001. RACER System Description. In *Proceedings of the First International Joint Conference on Automated Reasoning*, Lecture Notes in Computer Science 2083. Berlin: Springer Verlag.

Heinsohn, J.; Kudenko, D.; Nebel, B.; and Profitlich, H.-J. 1992. RAT: Representation of Actions Using Terminological Logics. DFKI Workshop on Taxonomic Reasoning. Technical Report D-92-08. Deutsche Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), Saarbrucken / Kaiserslautern, Germany.

Horrocks, I. 1998. Using a Description Logic: Fact or Fiction? In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98).* San Francisco: Morgan Kaufmann Publishers.

Horrocks, I.; Patel-Schneider, P. F ; and van Harmelen, F. 2003. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics* 1(1): 7–26,

Jackendoff, R. 1992. *Semantic Sructures*. Cambridge, MA: The MIT Press.

Kautz, H. 1991. A Formal Theory of Plan Recognition and Its Implementation. In *Reasoning about Plans*, ed. J. F. Allen, H. A. Kautz, R. N. Pelavin, and J. D. Tennenberg. San Francisco: Morgan Kaufmann Publishers.

Kim, J.; Spraragen, M.; and Gil, Y. 2004. An Intelligent Assistant for Interactive Workflow Composition. In *Proceedings of the 2004 International Conference on Intelligent User Interfaces* (IUI-2004). New York: Association for Computing Machinery.

Koehler, J. 1996. Planning from Second Principles. *Artificial Intelligence* 87(1–2): 148–187.

MacGregor, R. M. 1991. Using a Description Classifier to Enhance Deductive Inference. In *Proceedings of the Seventh IEEE Conference on AI Applications*, 141–147. Los Alamitos, CA: IEEE Computer Society.

McDermott, D. 2002. Estimated-Regression Planning for Interactions with Web Services. In *Proceedings of the Sixth International Conference in AI Planning Systems*. Menlo Park, CA: AAAI Press.

Moser, M. 1983. An Overview of NIKL, the New Implementation of KL-ONE. In *Research in Natural Language Understanding*. Cambridge, MA: Bolt, Beranek, and Newman.

Swartout, B., and Gil, Y. 1995. EXPECT: Explicit Representations for Flexible Acquisition. Paper presented at the Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'95), Banff, Canada, February 26–March 3.

Swartout, W. R., and Moore, J. D. 1993. Explanation in Second-Generation Expert Systems. In *Second Generation Expert Systems,* ed. J.-M. David, J.-P. Krivine, and R. Simmons. Berlin: Springer-Verlag, 1993.

Swartout, W. R.; Paris, C. L.; and Moore, J. D. 1991. Design for Explainable Expert systems. *IEEE Expert* 6(3): 58–64.

Tenenberg, J. D. 1989. Inheritance in Automated Planning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning* (KR'89). San Francisco: Morgan Kaufmann Publishers.

Valente, A.; Russ, T.; MacGregor, R.; and Swartout, W. 1999. Building and (Re)Using an Ontology of Air Campaign Planning. *IEEE Intelligent Systems* 14:(1): 27–36.

Wahlster, W.; Andre, E.; Finkler, W.; Profitlich, H.; and Rist, T. 1993. Plan-Based Integration of Natural Language and Graphics Generation. *Artificial Intelligence* 63(1–2): 387–427.

Weida, R. A., and Litman, D. J. 1992. Terminological Reasoning with Constraint Networks and an Application to Plan Recognition. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning* (KR-92), 82–293. San Francisco: Morgan Kaufmann Publishers.

Weld, D. 1999. Recent Advances in AI Planning. *AI Magazine* 20(2): 93–123.

Wellman, M. P. 1988. Formulation of Tradeoffs in Planning under Uncertainty. Ph.D. dissertation, Department of Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Yolanda Gil is associate division director at the Information Sciences Institute of the University of Southern California and a research associate professor in the Computer Science Department. She received her M.S. and Ph. D. degrees in computer science from Carnegie Mellon University. Her research interests include interactive knowledge capture, intelligent user interfaces, knowledge-rich problem solving, scientific and grid computing, and the semantic web. She was cofounder and cochair of the First International Conference on Knowledge Capture (K-CAP) in 2001 and is program cochair of the International Semantic Web Conference (ISWC) in 2005. She was recently elected to the Executive Council of the American Association of Artificial Intelligence (AAAI).