Formulating Concepts According to Purpose

Smadar T. Kedar-Cabelli
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903
ARPAnet: kedar-cabelli@rutgers.arpa

Abstract

Explanation-Based Generalization (EBG) has been recently a much-explored method of generalization. By utilizing domain knowledge, and knowledge of the concept being learned, EBG produced a valid generalization from a single example. Most EBG systems are currently provided with the concept being learned – or *target concept* – as a fixed input. A more robust generalization mechanism needs the ability to automatically formulate appropriate target concepts based on the purpose of the learning, since concepts learned for one purpose may not be appropriate for another. This paper introduced a technique and an implemented system that automatically formulate target concepts and their specialized definitions. In particular, the technique derives definitions of everyday artifacts (e.g. CUP), from information about the purpose for which agents intend to use them (e.g. to satisfy their thirst). Given two different purposes for which an agent might use a cup (e.g. as an ornament, versus to satisfy thirst), two different definitions can be derived.

I. Introduction

Explanation-Based Generalization (EBG) has been recently a much-explored method of generalization (e.g. [Mitchell *et al.*, 1986], [DeJong and Mooney, 1986]). By utilizing domain knowledge, and knowledge of the concept being learned, this method produces a valid generalization from a single example. The key power of EBG derives from its ability to extract just those features relevant to concept membership based on an explanation of how the example is a member of the concept being learned.

Most EBG systems are currently provided with the concept being learned – or *target concept* – as a fixed input. A more robust generalization mechanism needs the ability to automatically formulate appropriate target concepts based on the purpose of the learning, since concepts learned for one purpose may not be appropriate for another.

This paper introduces a technique, *purposive concept formulation*, and an implemented system, *PurForm*, that address the above limitation by using a specialized notion of *purpose* to automatically formulate target concepts and their definitions. In particular, the technique derives definitions of artifacts (e.g. CUP) from information about the purpose for which agents intend to use them (e.g. to satisfy their thirst). Given two different purposes for which an agent might use a cup (e.g. as an ornament, versus to satisfy thirst), two different definitions can be derived.

Consider the CUP scenario from [Mitchell *et al.*, 1986], based on [Winston *et al.*, 1983]: A structural definition of CUP is extracted by explaining how an example of CUP satisfies some pre-defined functional definition. Given the functional definition of CUP as a stable, liftable, open vessel, and given one example (say a green mug), the resulting structural definition states that CUPs are any light, flat-bottomed object with an upward-pointing concavity and a handle. Suppose instead that an agent wants to use the cup for the purpose of drinking *hot* liquids, or to use it as an ornament. Given explicit knowledge of purpose, the purposive concept formulation technique can automatically derive a definition appropriate for that purpose, rather than have it as a fixed input as in Winston's system. A cup for the purpose of drinking hot liquids, then, need not only be liftable, stable, and an open vessel, but also needs to insulate heat.

In the balance of the paper, we discuss the technique and the implemented system. Section II. presents an overview of the technique. Section III. describes the system, PurForm, in terms of inputs, outputs, and processes of each of the modules. Section IV. discusses related work. We conclude in section V. with limitations and some future research issues.

II. Overview

A. What is 'Purpose'?

Concepts often arise because of a need. An agent wants to achieve a goal, and needs to identify objects that would facilitate that goal. A description of an object whose properties enable an agent's goal becomes a useful concept to acquire. As a step toward automatically formulating target concepts, we examine a specialized class of concepts, those that describe everyday artifacts. Artifacts can be viewed as objects *designed* to enable agents' goals (chairs are to be seated on, pens are to write with, and so on). More

precisely, using conventional AI planning terminology, the specialized notion of *purpose* of an artifact is to enable a plan of actions to achieve an agent's goal. The artifact will enable such a plan if it satisfies those preconditions of the plan in which it is involved. The purposive concept formulation technique uses standard algorithms of planning and goal regression to compute the weakest preconditions of a plan. The target concept is then formulated by isolating those preconditions of the plan that describe properties inherent to the artifact that make it useable in the plan. Given a different goal or plan, a different target concept would be formulated.

B. Statement of the Problem

In order to more precisely define the purposive concept formulation problem, we introduce some terminology. A *state description* associates a state of the world with a list of facts, represented by ground atomic predicates. A *goal formula* is represented by a conjunction of atomic predicates that are desired to hold in some distinguished final state. An *operator* describes an action, and is represented in a STRIPS-like formalism with conjunctive formulae for precondition, add and delete lists. A *plan* is defined as a sequence of operators that transform an initial state into a state matching the goal formula. A *domain theory* is a set of rules, ground atomic predicates, and operators, that represent the general axioms, facts, and actions of the domain, respectively. A *concept* is represented as a predicate over some universe of objects, and characterizes some subset of these objects. Each *object* is described by a collection of ground atomic predicates. A *concept definition* describes sufficient conditions for concept membership. An object that satisfies the concept definition is called an *example* of that concept. A *generalization* of an example is a concept definition that describes a set containing that example. A concept definition is *functional* when it refers to the requirements on the action for which examples of the concept are used. A concept definition is *structural* when it refers to physical properties that satisfy the functional requirements.

The purposive concept formulation problem, and the technique for solving it, are defined as follows:

Purposive Concept Formulation
Given:

- Goal
- Plan to achieve goal
- Domain theory

Determine:

- Target concept
- Purposive functional definition of target concept
- Operational definition of target concept

Technique:

1. Isolate the target concept
2. Formulate purposive functional definition of target concept
3. Operationalize the definition of target concept

The purposive concept formulation technique consists of three steps. It first *isolates* a useful target concept to acquire. For example, if the agent's goal is to find something from which to drink hot tea, then an artifact that can be used to drink from becomes the target concept to acquire.

Next, given the target concept, its purpose (to enable the agent's goal) and a plan to achieve the goal, the *purposive functional definition* is *formulated* by collecting those properties inherent to the target concept that make it useable in the plan. For example, the plan for drinking hot tea might be to POUR the hot tea into a container, GRASP it with the hot tea in order to PICKUP, and finally DRINK the tea from it. The class of artifacts useable in the plan (HOT-CUPs) are those open containers that can contain hot liquid, and at the same time can be grasped and picked up by the agent, and can be emptied of the hot liquid.

In the third and final step, the purposive functional definition is reformulated, or *operationalized* [Mostow, 1983] [Keller, 1987a] into a more useable form, with the aid of an example of a HOT-CUP. The definition will be more operational if the system can use it to more efficiently recognize members of the target concept. Using a blue ceramic mug as an example of a HOT-CUP, the resulting operational definition states that HOT-CUPs are artifacts that are light weight, have an open concavity that is cylindrical, non-porous, and made of ceramic material, and have a flat bottom and a handle.

III. PurForm: A System for Purposive Concept Formulation

PurForm is the prototype system that implements the technique. In this section we describe the details of the system, including representation, inputs and outputs, and the process of each step. We illustrate each step with our case study of formulating the definition of HOT-CUP. PurForm was implemented in PROLOG.

PurForm is invoked after a problem solver has created a plan to achieve a given goal. The problem solver is a backward chaining planner (from [Nilsson, 1980] [Kowalski, 1979]). Given the goal formula

```
ingested(robbie,hot_tea,X)
```

and an initial state description in which robbie the robot is in the kitchen; mugs, bowls, glasses, et cetera, are on shelves in the kitchen; and a tea urn in the kitchen is filled with hot tea; the planner finds a plan in which mug1 is used to drink:

```
[pour(robbie,hot_tea,tea_urn1,mug1),
 grasp(robbie,mug1),
 pickup(robbie,mug1),
 ingest(robbie,hot_tea,mug1)]
```

PurForm is given the goal, and a plan that achieves the goal, as input (along with a domain theory). It performs purposive concept formulation by isolating a target concept to acquire, formulating a purposive functional definition for it, and operationalizing the definition.

Step 1: Isolating the Target Concept. PurForm first isolates a useful target concept to acquire. Any of the arguments in the goal or plan are candidate target concepts. We have simplified this step by choosing a target concept that is initially unknown (i.e. a variable argument in the goal), yet is useful in enabling the goal. Given the goal formula:

```
ingested(robbie,hot_tea,X)
```

X is the unknown argument, and becomes the target concept to acquire.

Step 2: Formulating the Purposive Concept Definition. Given the target concept, its purpose (to enable the agent's goal), and the plan, this step formulates the purposive functional definition by reasoning about the role of the target concept in enabling the plan to achieve the goal. In order to satisfy the plan, the artifact must satisfy the conjuncts in the preconditions of those plan actions in which it is involved. These conjuncts are collected together by *regressing* the goal through the plan, and then *analyzing the role* of the artifact in the regressed expression.

Given a goal formula and a plan as input, the *goal regression* algorithm [Nilsson, 1980] produces a description of all initial states such that applying the plan to any of these states produces a final state matching the goal formula. The resulting regressed expression consists of all those goal conjuncts and preconditions that the operators did not achieve, and therefore must be true even before the operator sequence is applied, that is, in the initial state.

Given the goal, with generalized arguments

```
ingested(Grasper-Robot,Hot-Drink,X)
```

and the generalized plan (with constant arguments replaced by variables), the regressed expression is:

```
open(From-Container),
can(pour(Grasper-Robot,Hot-Drink,
         From-Container,X)),
can(contain(From-Container,Hot-Drink)),
empty(X),
open(X),
can(contain(X,Hot-Drink)),
grasper_empty(Grasper-Robot),
can(grasp(Grasper-Robot,X)),
ungrasped(X),
can(pickup(Grasper-Robot,X)),
on(X,Surface),
same_loc(Grasper-Robot,X,Location),
can(ingest(Grasper-Robot,Hot-Drink,X)).
```

Analyze-Role is at the heart of the purposive concept formulation technique. It formulates the purposive functional definition of the artifact by analyzing what role the artifact plays in the regressed expression. Two heuristics are used to choose the conjuncts from the regressed expression to formulate the definition. We want to formulate the definition of the artifact using only those conjuncts of the regressed expression that describe properties *relevant* to the artifact (and not the agent, say), and that are the *intrinsic* properties of that artifact. A property is relevant to the artifact if it mentions the artifact. A property is intrinsic to an artifact if actions that manipulate that artifact do not easily create or destroy these properties. Intrinsic properties, then, are those properties that do not appear on the add or delete lists of any manipulation operators. For example, being graspable is considered intrinsic to the artifact since no manipulation operator in our database can transform an ungraspable artifact into a graspable one (e.g. build a handle). On the other hand, being at the same location as an agent is not intrinsic to an artifact since a 'MOVE-TO' manipulation operator can move the agent and artifact to the same location if they are not there already. This is similar to the notion of 'criticality' in ABSTRIPS [Sacerdoti, 1974].

Given the above regressed expression, the following conjuncts are not *relevant* to the artifact since they do not mention X as one of their arguments:

```
open(From-Container),
can(contain(From-Container,Hot-Drink)),
grasper_empty(Grasper-Robot).
```

The following conjuncts are not *intrinsic* since they appear on the add or delete list of some manipulation operator in the database:

```
empty(X) is on the add list of 'POUR',
ungrasped(X) is on the add list of 'PUTDOWN',
on(X,Surface) is on the add list of 'PUTDOWN',
same_loc(Grasper-Robot,X,Location) is on the
      add list of 'MOVE-TO'
```

The remaining conjuncts form the purposive functional definition of HOT-CUP (with a new gensym'd concept name):

```
concept22(X) ⇐
   can(pour(Grasper-Robot,Hot-Drink,
            From-Container,X)),
   open(X),
   can(contain(X,Hot-Drink)),
   can(grasp(Grasper-Robot,X)),
   can(pickup(Grasper-Robot,X)),
   can(ingest(Grasper-Robot,Hot-Drink,X)).
```

Step 3: Operationalizing the Definition. The purposive functional definition for the artifact has been formulated, yet it is not in a form that enables the system to efficiently recognize a particular example of such an artifact (it is non-operational). In our case study, recognition is assumed to be efficient if the concept definition is in observable, structural terms; inefficient – if it is in functional terms. ([Keller, 1987a] presents a more general operationality criterion.)

Given the purposive functional definition and domain theory as input, this final step operationalizes the definition of the target concept. The EBG algorithm performs this step [Mitchell *et al.*, 1986]. EBG explains (proves) how a particular example is a member of the target concept, and generalizes to form an operational definition. The implementation of EBG, PROLOG-EBG, [Kedar-Cabelli and McCarty, 1987] produces an explanation and generalization in one pass, by storing both a specific and general trace of the PROLOG theorem prover as it proves that the purposive functional definition is satisfied by an example. The proof is generalized by retaining constraints only among the proof rules. The leaves of the generalized proof tree become the operational definition, and characterizes all those examples that have a proof of concept membership of the same structure (the same proof rules, applied in the same order).

Given the target concept and its purposive functional definition as above, and given an example, mug1, represented by the following attributes:

```
manufacturer(mug1,abc-co).
serial_number(mug1,72118).
color(mug1,blue).
material(mug1,ceramic).
weight(mug1,6,oz).
has_part(mug1,cylinder1).
has_part(mug1,bottom1).
has_part(mug1,handle1).
...
```

mug1 is shown to be an example of the HOT-CUP (concept22) by proving that it satisfies the purposive functional definition by certain structural characteristics (satisfied, in turn, by specific attributes). The domain theory contains axioms used to link attributes to functional requirements they satisfy (e.g. 'graspable(X) ⇐ has-handle(X)'). The essence of the proof is as follows: Since the shape of mug1 is an open cylinder, it has an open concavity, that allows hot tea to be poured into it. The ceramic material of mug1 provides a non-porous material that also insulates the heat, and the flat shape of its bottom makes it stable – all of which enable mug1 to contain the hot tea. Its handle and insulating material make it graspable. Its weight (6 oz.) makes it light, that enables it to be picked up by the agent. Finally, having an open concavity enables the ceramic mug to be emptied of the hot tea (i.e. enables the agent to drink the hot tea from the mug).

The proof is generalized, and the leaves of the proof tree become the structural (operational) definition:

```
concept22(X) ⇐
   type(C,cylinder),has_part(X,C),
   open(X),material(X,ceramic),
   type(B,flat_bottom),type(B,sealed_bottom),
   has_part(X,B),
   type(H,handle),has_part(X,H),
   weight(X,W,oz),less(W,16).
```

The proof relies on certain attributes being true of Grasper-Robot and Hot-Drink as well. The definition only holds if these additional assumptions are satisfied. These are conjoined together and associated with the definition:

```
assumptions(concept22(X),
   type(M,mouth),has_part(Grasper-Robot,M),
   type(A,arm),has_part(Grasper-Robot,A),
   type(Hot-Drink,liquid),
   temperature(Hot-Drink,hot)).
```

A. Discussion

To summarize, the novel contribution of PurForm lies in its use of standard algorithms of planning and goal regression to automatically formulate a target concept sensitive to a given plan and goal. Most EBG systems, on the other hand, are supplied with the target concept as a fixed input independent of the purpose for which it is to be used.

Several design decisions in the representation and implementation have been made to facilitate target concept formulation. We chose to represent some of the conjuncts on the precondition/add/delete lists of the operators as functional preconditions of the form 'can(P)' (e.g. 'can(grasp(_))'). This level of abstraction is a deliberate design choice to enable the definition to be formulated in functional terms. The resulting purposive functional definition covers a broader class of objects since it can be represented by alternative structural definitions.

Another design decision was to generalize the constants to variables in the goal and the plan, using type hierarchy information. This is justified since the rules that apply to the specific constant also apply to all constants of a specific type. This design choice was made so that the resulting regressed expression would be more general. As a result, the definition derived from it is also more general. For example, since any rules that apply to 'robbie' also apply to any robot with a grasper, 'robbie' was generalized to 'grasper robot'.

IV. Related Work

Only a few other research efforts have focused on automatically formulating target concepts. A parallel research effort has been [Keller, 1987b]. Keller presents a scenario for automatically formulating the target concept USEFUL-OP (operators that are useful in leading to a solution), a fixed

input in LEX2, a system that learns heuristics for symbolic integration. Keller's derivation of USEFUL-OP can be viewed as analogous to purposive concept formulation. Given a problem solver (SOLVER) that initially performs exhaustive forward search, the need to improve SOLVER's efficiency corresponds to our *goal* input. The *plan* corresponds to SOLVER's actions, that are described by a flow graph of program components. A desirable target concept is first *isolated.* It is a new filter component in SOLVER that would reduce the number of nodes expanded during search. A description of the filter is *formulated* by reasoning about its role in improving SOLVER's efficiency (by a process similar to goal regression and analyze-role). The filter should recognize operators that are useful in leading to a solution (USEFUL-OPs). Filtering just those operators would lead to a solution more often, and thus improve efficiency. To become an efficient recognizer, the filter description is *operationalized* into easily recognizable descriptions of classes of such useful operators.

Purposive concept formulation contrasts sharply with another system that formulates target concepts. SOAR [Laird *et al.*, 1986] formulates concepts as a by-product of problem-solving, while our technique formulates concepts following problem-solving as an intentional activity to improve problem-solving performance. Each time SOAR encounters and solves a subgoal, it formulates an implicit target concept: the general conditions under which it can reuse the solution to this subgoal. SOAR formulates and operationalizes target concepts at every impasse, without an explicit analysis of how they might be useful in improving performance.

Purposive concept formulation is related to EBG in that both are analytic techniques that reformulate knowledge from one level of description to another. Purposive concept formulation reformulates a purposive description, while EBG reformulates a functional description. Purposive concept formulation uses goal regression over a plan to produce a purposive functional definition. EBG uses generalization over a proof to produce a structural definition.

V. Future Research

The purposive concept formulation technique requires additional research to become robust. For one, we need to experiment with case studies of learning concepts for alternative purposes (e.g. cup to be used as an ornament). In addition, future work includes extensions to handle other notions of purpose (such as purpose of the agent, purpose of the learner); augmentations to formulate other useful target concepts that appear in the plan (e.g. the class of agents, the class of liquids); and techniques to formulate the fixed inputs to EBG other than the target concept.

Acknowledgments

I would like to thank Tom Mitchell, my advisor, and Rich Keller, who strongly influenced this research. Thank also go to Thorne McCarty, Jack Mostow, and Chris Tong for their thoughtful comments on the research and on drafts of this paper. In addition, Armand Prieditis, Sridhar Mahadevan, Prasad Tadepalli, Lou Steinberg, Mike Sims, Steven Minton, Kai Zercher, Juergen Koenemann, and other Rutgers colleagues provided helpful suggestions and critiques. Jan Chomicki was helpful with Quintus PROLOG, and Chun Liew was helpful with LaTeX. This work is being supported by NSF under Grant Number DCR-83-51523-02.

References

[DeJong and Mooney, 1986] G. DeJong and R. Mooney. Explanation based learning: an alternative view. *Machine Learning*, 1(2):145–176, 1986.

[Kedar-Cabelli and McCarty, 1987] S. T. Kedar-Cabelli and L. T. McCarty. Explanation-based generalization as resolution theorem proving. In *Proceedings of the Fourth International Machine Learning Workshop*, Morgan Kaufmann, University of California at Irvine, June 1987.

[Keller, 1987a] R. M. Keller. Defining operationality for explanation-based learning. In *Proceedings AAAI-87*, Seattle, WA, July 1987.

[Keller, 1987b] R. M. Keller. *The role of explicit contextual knowledge in learning concepts to improve performance.* PhD thesis, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1987.

[Kowalski, 1979] R. Kowalski. *Logic for Problem Solving.* Elsevier North Holland, New York, NY, 1979.

[Laird *et al.*, 1986] J. E. Laird, P. S. Rosenbloom, and A. Newell. Chunking in soar: the anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1986.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: a unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Mostow, 1983] D. J. Mostow. Machine transformation of advice into a heuristic search procedure. In *Machine Learning: An Artificial Intelligence Approach, Vol. 1*, Tioga, Palo Alto, CA, 1983.

[Nilsson, 1980] N. J. Nilsson. *Principles of Artificial Intelligence.* Tioga, Palo Alto, CA, 1980.

[Sacerdoti, 1974] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115–135, 1974.

[Winston *et al.*, 1983] P. H. Winston, T. O. Binford, B. Katz, and M. Lowry. Learning physical descriptions from functional definitions, examples, and precedents. In *Proceedings AAAI-83*, pages 433–439, Washington, DC, August 1983.