KI Revisited: Working Note 5

Peter Clark and Bruce Porter
Dept. CS, UT Austin
{pclark,porter}@cs.utexas.edu

1 Introduction

This note gives a brief review of Murray's work on Knowledge Integration and the KI system [Murray and Porter, 1989, Murray, 1990b, Murray, 1990a]. First, we review how KI restricted inference, to avoid computing the deductive closure of the KB for every new piece of knowledge which is added. Second, we summarize the learning mechanisms which Murray proposed, and try to draw out the general principles he used from the detailed examples in the published papers. Finally, we show how the inference rules used in the "cuticle" example can be expressed in the programming language LIFE.

2 Recognition and Elaboration

2.1 Summary

KI's reasoning process involves "growing" a semantic net describing a particular scenario (hence is similar to the methods described in our earlier Working Notes). An initial "seed" graph is provided by the user, for example (in textual form):

```
epidermis001--cover-->cuticle001--composed_of-->cutin001
```

Although there may be a large number of statements about `epidermis`, `cuticle` and `cutin` in the KB which could be used to elaborate this description, only a few are added to this graph. To select these few, *views*[1] are defined which identify particular facts about general concepts. The initial graph is elaborated by selecting and instantiating a view in it. A view is selected heuristically, based on (i) the degree of overlap of the view and the graph (ii) some heuristic measures of 'interestingness'.

As views restrict which facts from the KB are added to the graph, they can be seen as "filters" of knowledge, as illustrated in Figure 1. As views are selected (partly) based on degree of overlap with the current graph, the views perform a kind-of "complete the picture" operation.

Rules then exhaustively forward chain to add relationships to this graph. The whole process is then repeated until the user is satisfied.

2.2 Objects and Relationships

Although not documented in the papers, these two cycles (applying views, applying rules) have two very distinct purposes:[2]

[1] NB Murray uses a different notion of views to that in Liane Acker's work

[2] Thanks to Jeff Rickel for these comments.

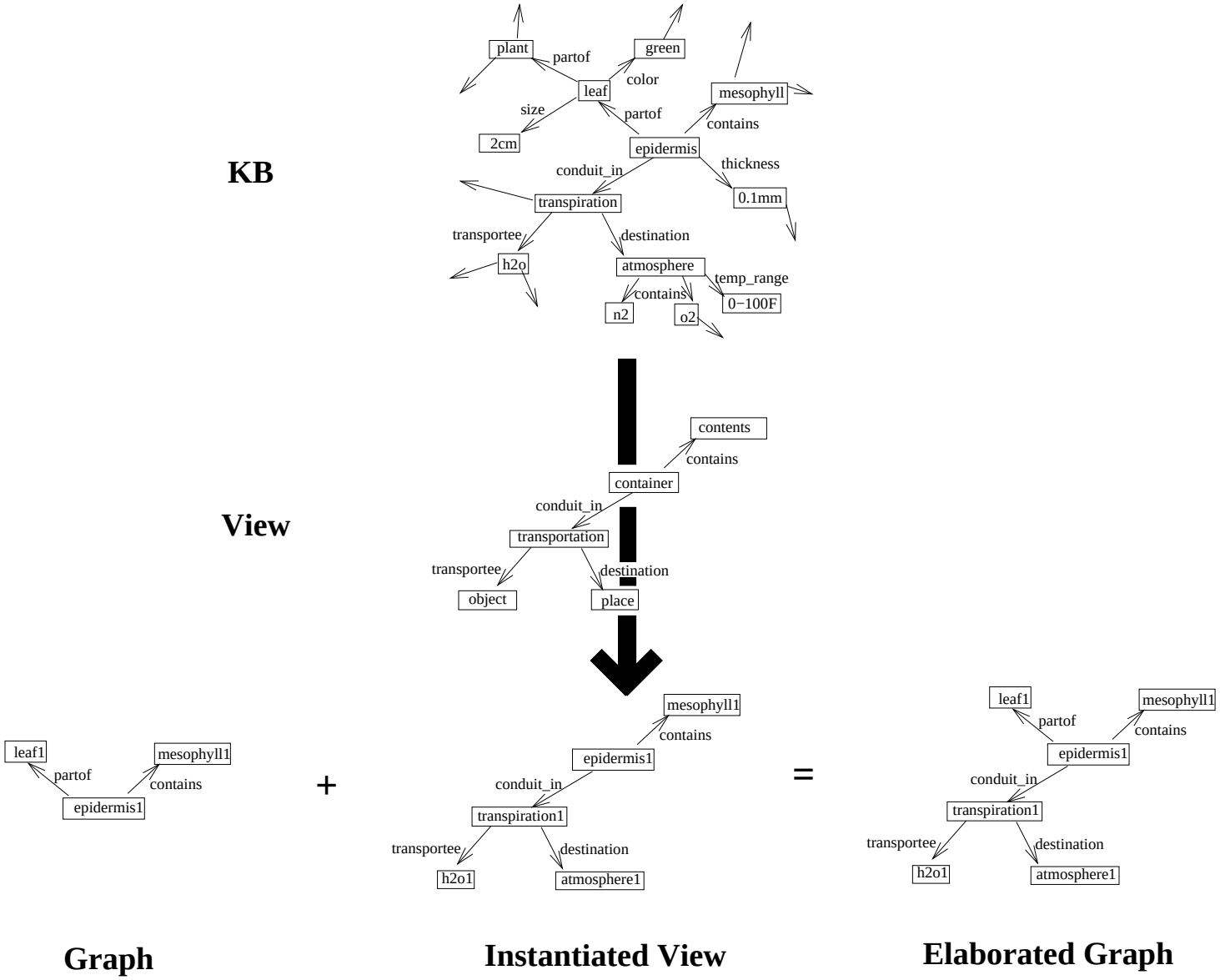

Figure 1: Elaboration of the Current Graph with an Instantiated View

Views: Introduce new *concepts* into the graph

Rules: Establish *relationships* between existing concepts in the graph.[3]

The motivation is that the important thing to control in exploring new knowledge is the introduction of new concepts: if this is controlled, then the remainder of the work (inferring relationships) can proceed with simpler reasoning techniques (exhaustive forward chaining). Views are thus a mechanism for tightly controlling introduction of new objects to the graph being grown.

3 Adaptation

3.1 Breaking Chains of Inference

As well as controlling the exploration of consequences of new knowledge, KI also aimed to fix the knowledge-base should contradictions be encountered. A contradiction requires that a previous chain of inference be "broken" somewhere so that the conflicting fact is no longer inferred. A general statement and examples of this problem are:

```
GIVEN   A -r1-> B -r2-> C
        A
        \+C
THEN "break the inference chain"

Example 1:
        cutin cover  -> impermeable -> dead
        cutin cover
        \+ dead

Example 2:
        \+ endosperm -> \+ nutrient source -> dead
        \+ endosperm
        \+ dead
```

Murray's sole mechanism for "breaking" chains of inference was by adding assumptions, assumptions which were previously considered false during the reasoning. This is the standard use of "Abnormality" predicates: All rules have an extra assumption "and there is nothing abnormal" added, and the rule can thus be broken by adding the assumption that this particular case *is* abormal in some way [Genesereth and Nilsson, 1987]:

```
X and \+ Abnormal(X) -> Y

f1(X) -> Abnormal(X)
f2(X) -> Abnormal(X)
 : : : : : : : : : :
```

where the `fi` are initially assumed false (Closed World Assumption). Adding the assumption some `fj(xi)` is true allows the inference `xi -> yi` to be retracted, rather than retracting the whole inference rule.
Murray gives two examples of this in [Murray, 1990b] and [Murray and Porter, 1989], which we summarize (in simplified form) below. From [Murray, 1990b]:

[3] Strictly, they may also introduce trivial concepts into the graph eg. numbers. The important point is that they do not introduce new concepts which are candidates for further elaboration.

```
GIVEN   X
        X and \+Y -> Z
        \+ Z
THEN    We can *assume* Y, hence block the inference of Z

Example:
GIVEN   cutin cover
        cover & \+ portals -> impermeable
        \+ impermeable
THEN    Assume portals
```

and from [Murray and Porter, 1989]:

```
GIVEN   not exists X -> Y
        \+ Y
THEN    We can *assume* there exists an X, hence block the inference of Y

Example:
GIVEN   not exists nutrient source -> dead
        \+ dead
THEN    Assume there exists a nutrient source
```

3.2 Controlling Search in Adaptation

As discussed in earlier working notes, there are often many alternative ways in which a KB can be patched so as to reachieve consistency. Making a change itself might have other repercussions (eg. cause inconsistency elsewhere), and hence it is in general not a simple task to identify the "best" way to correct a KB. The issue of how to guide this search (or even regarding it as a search problem in the first place) was not a focus of Murray's research. Other work in truth maintenance and belief revision directly addresses these issues (eg. [de Kleer, 1986, Forbus and de Kleer, 1988, Rao and Foo, 1989]).

4 Additional Learning Mechanisms

Murray also suggests three other learning mechanisms which, although the details were not worked out completely, are given here for completeness.

4.1 Justifying Facts in the KB

We can sometimes justify facts in the KB from other facts which are known. Knowing the principles underlying "shallow" rules in the KB may allow better explanations to be produced. The general form and example Murray gives in [Murray, 1990a] is:

```
GIVEN   A -> B
        A -> C, C -> B
THEN    we can justify A -> B by the chain A -> C -> B

Example:
   leaf is green
   leaf contains chlorophyll -> leaf contains green thing -> leaf is green
```

4.2 Explanation-Based Generalization (EBG)

Murray gives an example of doing a variant of EBG, where he generalizes an abductive rather than deductive conclusion (ABG?). The general form of EBG, and of Murray's "abduction-based generalization", and an example taken from [Murray, 1990b], are:

```
Normal EBG:
GIVEN      A -> B
           B -> C
           a
           c
DEDUCE     a -> c
GENERALIZE A -> C

Murray's abductive variant:
GIVEN      A and B -> C
           a, gensym'ed from (the also given) A
           c, gensym'ed from (the also given) C
ABDUCE     b
GENERALIZE B

GIVEN   X has chlorophyll & X has transluscent cover -> X photosynthesises
        leaf1 has chlorophyll1
        leaf1 covered by cutin1
        leaf1 photosynthesises
ABDUCES cutin1 is transluscent
GENERALIZES cutin is transluscent
```

Most of KI's reasoning is performed at the level of instances (eg. Figure 1). This generalization method can be seen as a mechanism by which KI can re-establish generality of its conclusions and add them back into the KB.

4.3 Generalization of Facts with Teleological Consequences

KI also could perform generalization of facts which contribute to some desirable purpose: If an instance of a desirable mechanism is discovered, then it is reasonable that this mechanism is used in other instances too. The general form and the example from [Murray, 1990a] is:

```
GIVEN   X Reln Y -> helps fulfil X's purpose(s)
        x1 Reln y1
ABDUCE  ALL xi Reln y1.

Example:
plant-part has cover -> restricts water loss
leaf has cutin-cover
all plant-parts have cutin-cover.
```

5 The Cutin Example, Represented in LIFE

For interest, we also rework the 'cutin' example in [Murray, 1990b] using the programming language LIFE. Our goal here is to show how we can express the rules' complex logical expressions as graphs. As well as (hopefully) making the expressions more readable, the graphical form simplifies inference: to apply a rule, we check that the graph

representing the rule's condition subsumes the graph we are currently constructing. The graphical formulation embodies the notion that reasoning involves moving pointers to concepts within a large network, a philosophical underpinning of many of the semantic net representation languages.

5.1 Background

5.1.1 Paths and Coreferentiality

Often we need to express that two values in our representation are coreferential (ie. the same node in the underlying graph). We can do this in LIFE using either paths or a common varable tagged to the values, as illustrated below:

```
% "The color of a person's car is his/her favorite color."
X:car(
   owned_by=>person(
               favorite_color=>C:color)
   color=>X.owned_by.favorite_color).
```

or alternatively with a common variable:

```
% "The color of a person's car is his/her favorite color."
car(
   owned_by=>person(
               favorite_color=>C:color)
   color=>C).
```

LIFE considers these two alternatives identical.

5.1.2 Definitionally Necessary Properties

A second KR requirement is to be able to state "definitionally necessary properties" of a concept, the properties which an object must have in order to be a member of that concept. Representing an if...then... rule as a concept, the rule's <condition> part becomes the definitionally neccessary properties and the <conclusion> part becomes the 'assertional' properties of the concept.

Below, we denote definitionally necessary properties with a `*`, for example the rule `IF telephone and color=black THEN cost=expensive` would be written:

```
% "Black telephones are expensive."
telephone(
   color=>black,                                % *
   cost=>expensive).
```

Implementationally, we can identify the definitionally necessary parts to the LIFE interpreter by repeating them in a separate clause (`fdefn/2`):

```
fdefn(1, telephone(color=>black)).
frame(1, telephone(color=>black,color=>expensive)).
```

The inference engine then forward chains, looking for the `fdefn` patterns in the instance graph being grown. If one is found, then the whole graph (in the `frame/2 clause`) is unified into the growing graph. This is equivalent to first identifying that a rule can fire, and then adding its conclusions to a working memory of assertions about the concept of interest.

5.2 The Cutin Example

First, we illustrate representing Murray's rules as LIFE frames in the manner just described. We then show illustrate how they can be applied to an initial 'seed' graph, supplied by the user, to produce an elaborated graph in which `leaf(status=>starving)` is concluded. The issue of how this this chain of inference would then be revised (if the user points out the leaf isn't starving) isn't addressed here; instead our goal is merely to illustrate the utility of LIFE's data structures for representing this kind of knowledge.

```
% "1. If an object is composed of cutin, then it is impermeable to gases."

object(
   composed_of=>cutin,                          % *
   impermeable_to=>gas(
                     forall=>true))).

% ----------------------------------------

% "2. If an object's cover is impermeable, then the object is impermeable."

object(
   cover=>@(                                    % *
             impermeable_to=>S),                % *
   impermeable_to=>S)).

% ----------------------------------------

% "3. If conduit is impermeable to the transportee, then transportation diabled."

transportation(
   transportee=>Tee,                            % *
   conduit=>@(                                  % *
              impermeable_to=>Tee),             % *
   status=>disabled)).

% ----------------------------------------

% "4. If resource aquisition is disabled, then its utilization is disabled."

resource(
   acquired_in=>@(                              % *
                   status=>disabled),           % *
   utilized_in=>@(
                   status=>disabled))).

% ----------------------------------------

% "5. If a thing's method of attaining nutrients is disabled, then it's starving."

living_thing(
   attainer_in=>attenuation(                    % *
                   attained=>nutrient,          % *
                   status=>disabled),           % *
   status=>starving)).
```

```
% ---------------------------------------

% (required part of) ISA lattice

leaf <| living_thing.
epidermis <| object.
cuticle <| object.
co2acquisition <| acquisition.
acquisition <| transportation.
co2 <| gas.
gas <| resource.
photosynthesis <| attenuation.
sugar <| nutrient.

% ---------------------------------------

% Initial graph, denoting an instance of a leaf

leaf(
   attainer_in=>P:photosynthesis(
                  attained=>sugar),
   part=>E:epidermis(
            cover=>cuticle(
                      composed_of=>cutin),
            conduit_in=>C:co2acquisition(
                           conduit=>E,
                           acquired=>CO2:co2(
                                       acquired_in=>C,
                                       utilized_in=>P),
                           transportee=>CO2))).

% ---------------------------------------

% Final graph, showing leaf(status=>starving) is concluded

leaf(                                                          % From rule:
   attainer_in=>P:photosynthesis(
                  attained=>sugar,
                  status=>disabled),                           % [4]
   part=>E:epidermis(
             cover=>cuticle(
                       composed_of=>cutin,
                       impermeable_to=>CO2),                   % [1]
             conduit_in=>C:co2acquisition(
                            conduit=>E,
                            acquired=>CO2:co2(
                                        acquired_in=>C,
                                        forall=>true,
                                        utilized_in=>P),
                            transportee=>CO2,
                            status=>disabled),                 % [3]
             impermeable_to=>CO2),                             % [2]
   status=>starving).                                          % [5]
```

References

[de Kleer, 1986] de Kleer, J. (1986). An assumption-based TMS. *Artificial Intelligence*, 28:127–162.

[Forbus and de Kleer, 1988] Forbus, K. D. and de Kleer, J. (1988). Focussing the ATMS. In *AAAI-88*, pages 193–198.

[Genesereth and Nilsson, 1987] Genesereth, M. R. and Nilsson, N. J. (1987). *Logical Foundations for Artificial Intelligence*. Kaufmann, CA.

[Murray, 1990a] Murray, K. (1990a). Improving explanatory competence. In *12th Annual Conf. of the Cognitive Science Society*.

[Murray, 1990b] Murray, K. (1990b). Learning as knowledge integration: A case study. In *AAAI Spring Symposium on Knowledge Assimilation*.

[Murray and Porter, 1989] Murray, K. and Porter, B. (1989). Controlling search for the consequences of new information during knowledge integration. In *Proc. 6th Int. Workshop on Machine Learning*.

[Rao and Foo, 1989] Rao, A. S. and Foo, N. Y. (1989). Formal theories of belief revision. In *KR89 (1st Int Conf on Principles of KR and Reasoning)*, pages 369–380, CA. Kaufmann.