(37)

RECOGNITION AND CLASSIFICATION BY EXPLORATION

by

TIMOTHY A. CHKLOVSKI

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING AND
MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1998

Signature of Author _____
Department of Electrical Engineering and Computer Science
February 2, 1998

Certified by _____
Marvin Lee Minsky
Toshiba Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

RECOGNITION AND CLASSIFICATION BY EXPLORATION

by

Timothy A. Chklovski

Submitted to the department of Electrical Engineering and Computer Science
on February 2, 1998 in Partial Fulfillment of the
Requirements for the Degree of Bachelor of Science in Computer Science and
Engineering and Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents an *approach to selecting an alternative from a collection of plausible ones*. We describe an agent that implements the approach with an iterative algorithm that causes one alternative to stand out as a clear leader.

The approach is applied to the more specific problem of interactive recognition. The presented algorithm locates appropriate candidates for recognition and either selects one as a successful recognition or formulates a question that will most aid disambiguation, poses it to the user (in a larger system, it could pose it to the agent's environment) and iterates with the result. The algorithm is constructed to cope with noisy inputs as well as uncertain and unsound reasoning. The performance is exemplified in a small domain.

The presented algorithm is designed to account for additional biases on its performance that are likely to surface when it is used by a larger reasoning system. The mechanisms that allow biasing the behavior of the algorithm are discussed in some detail. Much attention is paid to how the algorithm could be integrated into a larger system and the role it would play in it.

Thesis Supervisor: Marvin Lee Minsky
Title: Toshiba Professor of Media Arts and Sciences

Acknowledgments

I would like to especially thank my parents, Anatoli and Lioubov Chklovski, who played the key role in my intellectual development from the earliest days and who have always remained my dear friends. I also thank my advisor, Professor Marvin Minsky, and my colleagues Nicholas Cassimatis and Pushpinder Singh. They have provided much more inspiration, guidance, and intellectual challenge than their call of duty.
These people, as well as many others, have strengthened my endeavor to make machines intelligent.

Contents

Figures

Chapter 1. Introduction

1.1 Selection viewed as a race

This thesis describes *an approach to selecting an alternative from a collection of plausible ones*. The approach is presented as an agent that uses an iterative algorithm to accomplish the disambiguation. The algorithm *races* the evolving set of alternatives by posing questions to the outside environment until the best alternative emerges as a clear leader.

By a *race* between the alternatives we will mean iterating thorough the following steps:

Step 1. Select a set of plausible alternatives based on available information

Step 2. Estimate appropriateness of each alternative independently of others

Step 3. Decide whether one alternative convincingly stands out above the rest. If so, return it. Otherwise, use the current landscape of alternatives to determine the action that will bring the most clarity to the current set of alternatives.

Step 4. Take that action and incorporate the information it returns, setting up for the iteration

Racing can carry out the selection in conditions with noisy sensors, unsound reasoning, and limited resources. Racing the alternatives against each other is appropriate whenever arguments supporting each alternative can only be obtained at a cost, and it is desirable to keep that cost low while developing sufficient confidence that one alternative is better than the rest.

Flexible behavior based on assessment of the intermediate situation is what places racing

on a level above simple feedback systems. Their ability to tolerate noise, we believe, makes racing an appropriate tool in many tasks where assumption of perfectly correct inferences being drawn on the basis of perfectly correct information is not realistic.

The need to select from a set of alternatives is germane to many AI tasks, not just recognition. Here are examples of some such tasks:

- Action selection: the process of deciding whether to rest or to satisfy hunger or to satisfy thirst can be mapped nicely onto racing, as outlined above, the goals of "rest," "satisfy hunger," and "satisfy thirst".
- In determining whether a given statement is true or false, there is also an implicit race between the alternatives "true" and "false" – effectively, racing in this situation amounts to weighing the pros and cons. An answer is produced only once we are sufficiently confident of it.
- In all problems that can be formulated as a search though a tree, the key decision at each step is which node in the tree to expand next. It is desirable to select a good node to expand while keeping the evaluation costs low. Therefore, the racing algorithm applies.
- Rule-based systems, when they are scaled up, run into problems of needing to select a rule to fire out of several rules that match the current situation. The need for a powerful conflict resolution strategy can be addressed by forming a race set from the matching rules and using control knowledge as the source of information for making the choice.
- More directly related to recognition is the problem of learning a new concept. A new concept can be defined in terms of similarities and differences from a known concept, which is used as a prototype. This learning can be pretty much reduced to the process of recognizing the novel object as a known one by racing and noting the differences in the process.

Since racing is widely applicable, this thesis can be viewed as a study of applying racing

to the particular task of feature-based recognition. We will use the acronym RACE, which stands for Recognition And Classification by Exploration, to refer to the application of the more general notion of *racing* to recognition and classification. The particular details developed in this thesis will remain quite relevant in adapting the RACE algorithm to other tasks. The capacity of racing to offset an alternative against others and to function even in noisy situations makes it a valuable tool in many situations.

1.2 Background

The tasks of recognition and of classification overlap. Both involve examining an unknown object or situation and deciding to what known category it is most likely to belong. Processes dubbed "recognition" more often carry connotation of dealing with only partial or imperfect information. Classification, on the other hand, usually assumes that sufficient information is present and focuses on making finer distinctions to place a concept into a provided taxonomy of concepts. The task of classification is usually approached as a top-down process that iteratively narrows the set of possibilities.

In other words, recognition and classification refer to roughly the same task, but the terms have different connotations about the coarseness of the process and the amount of information available. The Recognition and Classification by Exploration (RACE) algorithm hence lives up to its name by initially coarsely recognizing the unknown and then querying for more information to classify the object more finely. Based on the amount and quality of provided information, the same algorithm can appear to be in the recognition or in the classification regime, progressing from the former to the latter if possible.

The task we describe can be most directly mapped onto the recognition involved in high level vision. The approach of studying the high level without designing for integration into a larger system has received much criticism. Two comments about RACE are in order here.

First, the entire process of vision can be thought of as a hierarchy of RACE agents (with much additional machinery surely present). Some queries produced by the topmost agents need to percolate down all the way to the lowest levels to be answered. For an excellent discussion of counterstreams of information in the vision process, see [Ullman 1996].

Second, the sets of alternatives that form in high-level tasks and the questions that can be posed to gain information probably tend to be more varied. At the lowest levels, for example, about the only way to gain more information is to foveate on the area of interest. Therefore, we prefer to demonstrate the flexibility of the algorithm on higher-level tasks.

The task we address is important in domains such as vision, where a feature is descriptive independently of its relationship with other features. The approach taken in this thesis is not as directly applicable to domains in which features need to be structured in a particular way to be fully meaningful. Examples of domains with such underlying structure are speech recognition and plan recognition. Just as words "life" and "file" differ only in order of the letters (the uniform structure of the features) and not in the letters (the features) themselves, objects in these domains are not described simply as a set of their features. In speech recognition, the common structure is the temporal

sequencing of the phonemes that are the features of the words. In plan recognition, steps of a plan form a chain from the given situation to the desired. When such a common structure is present, custom techniques for dealing with it are appropriate, as hidden Markov models are for speech recognition.

We now turn to an overview of related problems and known approaches to these problems.

1.2.1 The Society of Mind

Marvin Minsky's *The Society of Mind* presents a broad array of machinery out of which the larger mechanisms of the human mind can be constructed. Most directly applicable to this thesis machinery is: Cross-Exclusion [Minsky 1986c], as well as Recognizers and Closing the Ring [Minsky 1986d].

1.2.2 Machine Learning

The notion of competition between the possible answers is present in k-nearest neighbor classification and related methods. In both approaches, adding a new object to the domain will cause some inputs to be classified as the new object rather than one of the old ones. In a way, the new object will beat out the competition on the inputs that are similar to it. However, the notion of posing a question to improve the recognition is not present in the standard k-nearest neighbors paradigm.

The most popular approach that does include the notion of interactively posing questions is decision trees. In decision trees, the queries to pose have been pre-computed on known

data. The RACE algorithm computes the queries dynamically. Computing the query at run-time allows the algorithm to account for many aspects of the current configuration even when the space of configurations is large. By taking advantage of as much information as possible at once, we improve the noise tolerance of the algorithm.

Noise tolerance is present in another class of approaches – autocorellational approaches such as Hopfield nets and Boltzmann machines. The drawbacks Hopfield nets and Boltzmann machines entail is that they do not readily accommodate revision of the knowledge they encode; in standard approaches, they need to be retrained from scratch. Arguably, lack of an explicit representation also makes them less amenable to manipulation by a larger system.

For an algorithm that selects a classifier based on features of the data set see [Brodley 1995].

1.2.3 Symbolic Classifiers

An important component of the knowledge representation architectures of the KL-ONE family (such as LOOM and CLASSIC) is a component called a classifier or a recognizer. However, these systems typically assume that sound inference is performed on noise-free information. Hence the designs of the algorithms are quite different. Designers of LOOM stress importance of backward chaining in the recognition process [MacGregor 1988], [MacGregor & Brill 1992].

Classification is often carried out in a top-down fashion [Brachman et al. 1991]. In the conditions where noise might be present this approach has the drawback of not taking

advantage of all available information right away. It also often requires imposing a more restrictive structure (such as a single inheritance hierarchy) on the knowledge. We stress that the approach of this thesis is applicable to the same tasks, hopefully overcoming the difficulties that the other methods run into.

In general, in matchers dealing with more complicated structures the notion of competition is usually not given as much attention. In reality, the *dynamic* computation of the set of competing alternatives becomes even more important in the situations where the matching itself is elaborate. There is an interactive, incremental classifier for KL-ONE architectures [Silverman 1984]. It proceeds in a top-down fashion through a tree of concepts. In selecting a question it primarily uses user-supplied rankings, resorting to a greedy worst-case minimization when the rankings are not available.

1.2.4 Other approaches to Recognition and Classification

A process related to recognition and classification is *categorization*. While classification places an object into a pre-supplied ontology, categorization (as we use the term) entails partitioning a *set* of objects into a (possibly hierarchically structured) set of categories. Categorization can be viewed as a process constructing the knowledge base that is used by recognition and classification algorithms. A relevant problem of what constitutes a category in the natural world and how these categories might be extracted from a set of objects has been addressed [Bobick 1987]. The work presents a categorization paradigm and an algorithm for recovering multiple levels of natural categories.

Classification is also used as a problem solving technique. Problems can sometimes be classified to locate an appropriate method of solution. Heuristic, non-hierarchical classification is applicable in this domain, as overviewed by [Clancey 1985].

1.3 What the RACE algorithm does

The algorithm, when presented with some properties of an unknown object, determines which one (of the ones in its domain) this object is. The recognition is performed with the assumption that some information may be absent, uncertain, or incorrect. Due to this assumption, the result of recognition is also not guaranteed to be correct; rather, the confidence of recognition can be traded off for the amount of information used. The algorithm is interactive in the sense that it formulates and poses questions that help it disambiguate between competing answers and build confidence in a particular answer. The input given to the algorithm and the output it produces are stated in full below.

1.3.1 Input

The input consists of the following:

- A set of assertions about the object being recognized.
- Optionally, the amount of energy allocated to the process. (If omitted, the default value is used).
- Also optionally, the caller can override the default values of the thresholds the algorithm uses. A glossary of all the controlling parameters is given in Appendix A.

1.3.2 Output

Recognition amounts to deciding which object in the system's knowledge base is the answer, i.e. deciding which object the unknown matches best. To be chosen as the answer, a candidate has to be *significantly better* than the *competing* ones. The previous sentence needs some unpacking:

- What constitutes *significantly better* is described by the relevant thresholds of Appendix A. The magnitudes of these thresholds reflect confidence in the answer.
- Two possible recognitions, as A and as B, are said to *compete* if no object can be recognized as both A and B. For example, "fruit" and "pear" do not compete while "tomato" and "pear" do. In general, if A subsumes B or vice versa, A and B are not in competition. However, if knowledge is organized in a multiple-inheritance hierarchy, it is possible for two recognitions not to compete even when they do not subsume each other. For example, recognizing that a person is a Republican does not preclude the person from being a Quaker. Combining evidence from these recognitions can pose additional problems, as in the Nixon diamond example, but processes that follow recognition are not addressed in this thesis. In our knowledge representation formalism, presented in Appendix B, A competing with B is expressed as "A isa B" having a truth value of -1 (i.e. "no").

If successful recognition is reached, the algorithm continues to explore the non-competing possible recognitions that can provide additional information. For example, having established that a person is a Republican and having some evidence that the person might be a Quaker, the algorithm will try to establish this with more confidence. Similarly, if we had a successful recognition as a "fruit" as well as evidence that this is an "apple", the algorithm would go on to refine the recognition. On the other hand, once an object is recognized as an "apple", the algorithm would not try to recognize it as a "fruit"

because that would not provide additional information.

In the example domain we use, objects are organized into a single-inheritance hierarchy, so the notion of competition becomes equivalent to subsumption. The notion of continuing to explore reduces to considering those candidates that inherit properties from the successful recognition. The main advantage of using a single-inheritance hierarchy is the following: since any two objects are either in a subsumption relationship or in competition, the result of the recognition is always a single answer. This would not be the case in the example with the Republican and the Quaker. Note, however, that the algorithm as it is described will work on a multiple-inheritance hierarchy as well.

As will become clear once the algorithm is presented, this definition of the output does not imply that recognition always proceeds from the top down. In searching for a suitable answer it is driven by the quality of possible answers rather than by the structure of the knowledge. The ability to switch to a taxonomically unrelated candidate that was suggested by the newly supplied properties is one of the strengths of the algorithm.

Finally, if the object cannot be successfully recognized as any of the objects in the hierarchy, the algorithm indicates this by outputting NIL.

1.4 Outline

The rest of the thesis is organized as follows:

- Chapter 2 presents a sample run of the algorithm to demonstrate exactly what the algorithm accomplishes.

- Chapter 3 outlines and then details the implemented recognition algorithm.

- Chapter 4 presents a version of the main algorithm for answering a specific question about the object being recognized.

- Chapter 5 outlines the directions for future work. Of particular importance there is the outline of how this algorithm may be integrated with concept acquisition and repair.

- Chapter 6, is probably the most thoughtful one; it chiefly addresses how the algorithm could be used within a larger reasoning system and describes the features of the design of the algorithm that were put in specifically to aid the integration into a larger system.

Following Chapter 7, which is a brief summary of the key points of the thesis, there are 3 appendices:

- Appendix A is a glossary of the thresholds used in the algorithm.

- Appendix B describes the knowledge representation, the domain and the inference mechanisms used for demonstrating the operation of the algorithm.

- Appendix C presents auxiliary examples that illustrate more advanced aspects of the algorithm's behavior. These examples demonstrate the necessity of the features of the algorithm that are not called upon in solving the simpler example of Chapter 2.

Chapter 2. Example

At this point, we present an example of interaction with the algorithm. All of the examples are carried out in a knowledge base with a fairly simple representation scheme. The representation and the knowledge base are described in detail in Appendix B.

The following notational convention is used: comments are in *italics*, user input is preceded by the ">" prompt, and the system queries and output are in CAPS. The algorithm has been implemented in Allegro Common Lisp 3.02 for Windows.

```
> (recognize  '(((isa edible) 1) ((isa round) 1) ((isa soft) -1)))
```

This is the initial input. We are trying to recognize something that is edible, round object but is not soft. A "1" (which can be implicit) indicates that the property is present, "-1" indicates it is absent, while "0" indicates no information is available. These numbers are truth values. The answer we are looking for is "apple". The values of all unspecified thresholds are at their default values.

```
CANDIDATES:                APPLE       TOMATO     VEGETABLE    FRUIT
SCORES:   [MAX:2.2]          2.2         1.5           1           1
------------------------------------------------------------------------
PROPERTIES & TV's:
 ISA EDIBLE     [1]       [1]  1      [1]  1       [1]  1      [1]  1
 ISA ROUND      [1]       [1]  1      [1]  1       [0]  0      [0]  0
 ISA SOFT       [-1]     [-1] .2      [1]  -.5     [0]  0      [0]  0

SUCCESSFUL RECOGNITIONS: NIL
```

The algorithm reports the race set of candidates, their scores and the maximum attainable score. The columns present truth values in square brackets; the numbers next to them are contributions of these truth values to scores. It also returns the specific truth values (in square brackets) and scores that contributes to each assertion. The successful recognitions up to this point are reported as well.

```
QUESTION: WHAT IS tv(X, (ISA FRUIT))?:
```

The agent asks a question to help disambiguate between the possibilities. In this case, we will assume that the truth value of the assertion that the unknown has this property was directly established (but the answer could be erroneous)

```
> 1
```

The user enters one of 1, 0, or -1

```
CANDIDATES:                  APPLE        FRUIT       PEAR          LEMON
SCORES:   [MAX:3.2]            3.2          2           1.7           1.5
-------------------------------------------------------------------------
PROPERTIES & TV's:
 ISA FRUIT      [1]        [1]  1       [1]  1      [1]  1        [1]  1
 ISA EDIBLE     [1]        [1]  1       [1]  1      [1]  1        [1]  1
 ISA ROUND      [1]        [1]  1       [0]  0      [-1] -.5      [-1] -.5
 ISA SOFT       [-1]       [-1] 0.2     [0]  0      [-1] 0.2      [0]  0

SUCCESSFUL RECOGNITIONS: (APPLE)


ENERGY SPENT:  2          ENERGY REMAINING:  4
ANSWER:        APPLE
```

The system re-computes the candidates and their scores. Sufficient spread between the leader Apple *and the highest scoring competing candidate* Pear *has been achieved before we ran out of energy. The answer cannot be refined, so the algorithm succeeds, returning with the answer* Apple.

Examples illustrating some of the more advanced features of the algorithm and special cases can be found in Appendix C.

Chapter 3. The Recognition and Classification by Exploration (RACE) Algorithm

The following notation will be of use in presenting the algorithm:

X	– the object we are trying to recognize
PC	– a potential candidate
C	– a candidate
p	– a property
tv(O, p)	– if *O* is an object and *p* is a property, this is the truth value of *O* having the property *p*.
I	– the set of pairs of properties and *tv* assertions that are known about the object being recognized. The pairs are of the form *<p, tv(X, p)>*, i.e. properties and the *tv*'s of X having these properties
F	– the set of objects that are successful recogntitions; initially empty
S_1	– if ***S*** is a set of property-*tv* pairs, S_1 is the subset of ***S*** containing only those pairs that have *tv*'s of 1. Subscripts *0* and *-1* can also denote restrictions on truth values

Figure 1. Notation used throughout the presentation of the algorithm.

The algorithm is outlined in brief in Figure 2. It is presented in detail in the rest of this chapter. The example of Chapter 2 is traced throughout the presentation of the algorithm.

1. *Retrieve and score potential candidates*
   if no energy remains, **then** go to step 5
   else retrieve and score potential candidates
   end if
2. *Form a race set of candidates*
   The set is formed by pruning the set of potential candidates
   if set of candidates is empty, **then** go to step 5
   else proceed to next step
   end if

3. *Decide whether successful recognition is present*
   if unknown successfully recognized as C,
   then $\boldsymbol{F} := \boldsymbol{F} \cup \{C\}$;
   go to step 1
   else proceed to next step
   end if

4. *Select and pose a question*
   Retrieve possible questions
   if such questions do not exist, **then** proceed to step 5
   else compute which question will help disambiguation the most
   pose the question and receive a *tv* for an answer
   Augment $\boldsymbol{I}$ with the new information
   go to step 1
   end if

5. *Compute the output*
   if $\boldsymbol{F}$ is empty, **then** report failure of recognition
   else remove redundancies from $\boldsymbol{F}$ and output the result
   end if

Figure 2. Outline of the algorithm

3.1 Retrieving and scoring potential candidates

The objects with at least one property from $\boldsymbol{I_1}$ are retrieved. These form the set of potential candidates. The example of Chapter 2 had:

$\boldsymbol{I}$ = { $< p_1$ = isa edible, $tv(X, p_1) = 1 >$

$< p_2$ = isa round, $tv(X, p_2) = 1 >$
$< p_3$ = isa soft, $tv(X, p_3) = -1 >\}$

Retrieval from our domain resulted in the following set of Potential Candidates (*PC*s):

{APPLE, BALL, CUCUMBER, FRUIT, LEMON, PEAR, TOMATO, VEGETABLE}

Note that the very general recognitions such as Object were not retrieved. By indexing more directly into the possible recognitions rather than working in a top-down fashion, the algorithm is able to right away take advantage of the additional properties that are not known about Object.

Next, for each potential candidate *PC* a score $score_{PC}$ is computed according to

$$score_{PC} = \sum_{p} SC(tv(PC, p), tv(X, p))$$

Where p ranges over the properties in the input set, and the $SC(candidate_{tv}, unknown_{tv})$ function is computed as follows:

Candidate / Unknown	1	0	-1
1	1	0	-.5
0	0	0	0
-1	-.5	0	.2

Figure 3. Computing the score from the *tv*s of the candidate and the unknown object.

The low value of *SC(-1, -1)*, .2, captures the intuition that not having a property is not really convincing evidence at all. Just as a red car is neither blue nor green nor yellow, there are many negative assertions for every positive one. Therefore, a candidate and the unknown may share the negative property by chance. The negative *tv*s serve mainly to

signal conflicts with the positive ones, not to a have strong effect when they match.

For example, if the candidate and the unknown have *tv* of 1 for the property "isa round", 1 will be added to the candidate's score.

In the above example, the potential candidates will get the following scores:

	PRELIMINARY CANDIDATES							
Properties	**Apple**	**Tomato**	**Vege-table**	**Fruit**	**Cucum-ber**	**Pear**	**Ball**	**Lemon**
Isa edible [1]	1	1	1	1	1	1	-.5	1
Isa round [1]	1	1	0	0	-.5	-.5	1	-.5
Isa soft [-1]	.2	-.5	0	0	.2	.2	0	0
SCORES:	**2.2**	**1.5**	**1**	**1**	**0.7**	**0.7**	**0.5**	**0.5**

Figure 4. Scoring the preliminary candidates.

Note that the table reflects *contributions* from properties, not the objects' *tv*s for these properties. The contributions are computed according to Figure 2 with respect to the following set of *tv*s: { *tv(X, isa round)* = 1, *tv(X,* isa hard*)* = 1, *tv(X,* isa red*)* = -1 }.

3.2 Forming a race set of candidates

The candidates are selected from preliminary candidates. The formation of the race set of candidates is carried out using the following 3 constraints:

- Candidate absolute threshold
- Candidate relative threshold, which is the fraction of the maximum score a preliminary candidate must attain to become a candidate
- If some successful recognitions are present, leave only the candidates that add new information but do not compete with the previous ones

Throughout our examples the two thresholds are set at .5, and .33, respectively. The maximum attainable score in the above example is 1+1+.2 = 2.2, so the second constraint will reject all candidates with scores below .33 * 2.2 = .726. This quantity dominates the first constraint of .5.

To understand the third constraint better, let us consider an example. If *fruit* were a successful recognition, it would eliminate such candidates as *tomato*, *vegetable*, and *fruit* from the set of candidates. However, it would not eliminate *apple* if it were a candidate. If *apple* were a successful recognition, it would eliminate absolutely all candidates that come from our domain.

Formally, the constraint translates to admitting only such preliminary candidates *PC* that *tv(PC*, isa *f)* = 0, for any *f* such that $f \in \boldsymbol{F}$. Note that this means that no successful recognition will be admitted to the set of candidates again, since in our knowledge base

$$\forall Y \; tv(Y, isa\, Y) = 1$$

Since the set $\boldsymbol{F}$ is initially empty this constraint has no effect in our running example. Thus, the set of candidate-score pairs in the example will consist of the *PC*s with scores above .726, i.e. the set of candidate-score pairs will be

```
{<APPLE, 2.2>, <TOMATO, 1.5>, <VEGETABLE, 1>, <FRUIT, 1>}
```

Special cases:

- All preliminary candidates have been eliminated, and the candidate set is empty. In this case the algorithm proceeds to the output stage (described in Section 3.5) with the

set ***F***.

3.3 Deciding whether a successful recognition is present

We will refer to the candidate with the highest score as the *leader*. Only the leader can be the result of a successful round of recognition. Some of the tests for whether the leader is good enough compare it to the *runner-up*, which is the highest-scoring candidate leader is in competition with. Two objects *Y* and *Z* are in competition if and only if

$$tv(Y, isa\ Z) = -1,$$

which in our knowledge representation is equivalent to

$$tv(Z, isa\ Y) = -1$$

If there is more than one candidate tied for the highest score, ties are broken arbitrarily. A leader is determined to be a successful recognition when these constraints are met:

- Leader's score is above the leader absolute threshold (defaults to 2). This and other thresholds are also listed in Appendix C. This parameter is related to the prior belief that the unknown indeed is one of the objects the system knows about. The higher the chances of the object being novel, the higher this parameter should be.
- Leader's score is greater than a certain fraction of the maximum attainable score. This fraction is called the leader relative threshold, and it defaults to .6. The maximum attainable score is computed by scoring the ideal match to the known properties of the object. The idea is that if a leader only accumulates 7 points while a perfect match would have accumulated 100, the leader is a poor match to the unknown object regardless of scores of the other candidates. This threshold should be kept low if the objects are described as implicit disjunctions of their properties, i.e.

when particular instances strongly mismatch their prototypical representations in the knowledge base.

- Leader's score exceeds that of the runner-up by the leader absolute spread, which defaults to 1.5.
- Leader's score exceeds that of the runner-up by a certain fraction of the leader's score. This threshold is called the leader fractional spread (it defaults to .3). This constraint is present because a win of 10 over 5 (which is 50%) is much more convincing then a win of 100 over 95 (which is 5%). This parameter should be high when the sensors are noisy.

In our example, the first time around the leader is APPLE with the score of 2.2, which is also the maximum score. The runner up is TOMATO with the score of 1.5. The difference of their scores, 0.7 is below the leader absolute spread threshold of 1.5, so no leader is found. After the question is asked, the score for APPLE becomes 3.2, even though FRUIT has the score of 2, it does not compete with APPLE. The actual runner up is PEAR with 1.7 points. All the thresholds are cleared in this situation and APPLE is a successful recognition.

Special cases:

- What if there are no candidates? We return with an error message. See section on "what to do when got stuck" below for a discussion on how it might be followed up.
- There are no candidates that compete with the leader. This situation can arise either because the candidate set is a singleton or because no candidate competes with the leader. In this case, the leader still has to exceed the "leader absolute threshold" and the leader relative threshold.

If a leader is found, it is inserted into the output set ***F***, and the retrieval and scoring the candidates is repeated using input set ***I*** and the updated set ***F***.

If a leader is not found, the algorithm computes a new question as described in the next section.

3.4 Computing a question to pose

The properties of candidates that are not mentioned in ***I*** form a pool from which a property to query about will be selected. Each candidate contributes some properties to this set. In our example, we have:

		CANDIDATES			
		Apple	**Tomato**	**Vegetable**	**Fruit**
Known Properties	isa edible	1	1	1	1
	isa round	1	1	0	0
	isa soft	-1	1	0	0
Retrieved Properties	isa tasty				S
	isa juicy				S
	isa object			S	S
	isa vegetable		S	S	
	isa sweet	S			
	isa red	S	S		
	isa hard	S			
	isa fruit	S			S
	has stem	S			

Figure 5. Candidates, known properties, and additional properties retrieved for them. An "S" indicates the object was the source of that property.

How many properties and what particular properties are retrieved is an important design decision, one that will influence the exploration strategy of the algorithm. We retrieve

only the properties that are obtained by direct lookup and do not require any inference.

We take a union of the sets of properties retrieved for each candidate and remove the properties that were already queried about. Note that in a system with continuous rather than discrete certainties re-asking a question to raise confidence is quite sensible, so the questions already asked would also have to compete. Incorporating two different answers to the same question, however, would require special handling.

Each property in the resulting set is made into a "candidate" question, and the most appropriate question is selected. The general race set methodology is appropriate here. Because we believe that relatively little energy is allocated to this subprocess, its behavior is approximated as its quick special case: the algorithm poses the question that seems best after one pass of scoring.

In a more sophisticated system, the following factors could play a role in assessing how good a question is:

- How much the question will help us discriminate between the candidates (more rigorously, whether it lowers the entropy).
- How likely is it that the answer to this question will be obtained at all.
- What is the cost of obtaining an answer to the question.

In our implementation, we do not account for latter two factors; these estimates should be subgoaled to other experts or mechanisms.

Hence, we assess the usefulness of a particular question by estimating its impact on the total entropy. Yet, different answers to a particular question will have different effects on the entropy. We combine these separate assessments in a simple way: by assessing the effects of each answer and taking a weighted sum. How the weights and the summands

are derived is described in turn.

Estimating the weights (i.e. making a guess as to how likely each answer is) is done from the candidates themselves. This is an excellent example of how the model biases our view of the world, a topic taken up in discussion. That is, the weight $W_{v,p}$ given to a particular value v of $tv(X, p)$ is computed as follows:

$$W_{v,p} = \frac{\sum_{C \in Cand} score_C \cdot P(v, tv(C, p))}{\sum_{C \in Cand} score_C} \quad (1)$$

Here, *Cand* is the set of candidates (the race set), $score_C$ is the score of a candidate C, and P is a probability function

P: *truth value* × *truth value* → [0,1]

that is computed as shown in Figure 6. This figure presents probabilities of the unknown having the given *tv*, given the *tv* of the candidate. Since the answer is always one of the listed options, the columns of the table sum to 1.

Candidate expected answer	1	0	-1
1	.8	0	0
0	.2*	1	.2*
-1	0	0	.8

Figure 6. The probabilities used for predicting *tv* of the answer.

* These reflect that there is some chance that we will not be able to establish the property.

An overall comment on the functions used in this algorithm: they were chosen to satisfy

the design criteria (which are stated as the functions are presented). However, no claim is made that these functions are optimal. In this case, linear weighting by $score_C$ of the probabilities returned by P seems to work well.

Next, we normalize the scores of candidates to obtain $score_{NORM\ C}$:

$$score_{NORM\ C} = \frac{score_C}{\sum_{C \in Cand} score_C} \tag{2}$$

The entropy S of the race set is calculated from the scores of the candidates according to:

$$S_{CAND} = -\sum_{C \in Cand} (\log score_{NORM\ C}) \cdot score_{NORM\ C} \tag{3}$$

The score for each question is then computed by taking the weighted sum over the *tv*'s of the changes in entropy each answer will effect. That involves rescoring (but not re-selecting) the candidates and recomputing the entropies using the sets of property-assertion pairs

$$\boldsymbol{I} \cup \{<p,\ tv(X,\ p) = v>\}$$

for values of $v = \{1, 0, -1\}$, rather than using just $\boldsymbol{I}$ as we did in computing S_{CAND}.

$$score_p = -\sum_{v \in \{1,0,-1\}} W_{v,p} \cdot \left(S^v_{CAND} - S_{CAND}\right) \tag{4}$$

Here, $score_p$ denotes the score of the property p, S_{CAND} denotes the entropy of the candidate set as is explained above, and S^v_{CAND} denotes the entropy of the candidate set rescored under the assumption that $tv(X,\ p) = v$. Note that the formula negates the entropy differentials to give the highest score to the question that lowers entropy the

most.

In the running example, the candidate set is

```
{<APPLE, 2.2>, <TOMATO, 1.5>, <VEGETABLE, 1>, <FRUIT, 1>};
```

its entropy is 1.33, the winning property, "isa fruit" results in entropies 1.19, 1.33, and 1.30 for v equal to 1, 0, and -1, respectively. These entropies are weighted by 0.45, 0.2, and .35, resulting in the score of .072 for this property. The next highest scoring property is "isa vegetable" with the score of .060.

The score for that question is lower because the more likely negative answer to "isa vegetable" punishes by .5 and rewards by .2, whereas the positive answer to "isa fruit" rewards by 1 and punishes by .5, causing a bigger change in entropy.

Next, The question with the highest score is posed. The *tv* that is the answer to it is obtained. We add the result of the query to the set of known properties $\boldsymbol{I}$ and iterate with this updated set $\boldsymbol{I}$ and the old set $\boldsymbol{F}$, starting with the candidate selection stage again. Note that *we do not keep pruning the same candidate set, but allow new competition to enter.* This allows the set of candidates to evolve as new information comes in.

Special cases:

- There may be no more questions to pose. In this case the algorithm proceeds directly to the output step.
- If the candidate set consists of only one candidate, the entropy computations will not

yield very interesting scores for questions. However, the selected question will provide more information about the sole candidate, which is a reasonable behavior in that situation.

3.5 Computing the output

This step processes the list of successful recognitions ***F*** and outputs it. The set can contain recognitions that are redundant. To eliminate the redundancy, we leave an answer in the answer set if and only if it is such that it does not subsume any other candidate in the answer set:

$$(Y \in Output) \Leftrightarrow \forall C((C \in (Cand - \{Y\})) \Rightarrow tv(C, isa\ Y) \neq 1)$$

As was mentioned in 1.3.2, in a single inheritance hierarchy such elimination will always result in a single answer.

The other alternative, of course, is that the set of successful recognitions is empty. In this case the failure of recognition is reported. In our example, the recognition set ***F*** is the singleton {APPLE}, and so the answer produced is also APPLE.

Chapter 4. Goal-directed Version of RACE Algorithm

4.1 The purpose of goal-directed recognition

Why do we perform recognition? Recognition can be viewed as collapsing uncertainty that accumulates after making some uncertain steps, either of reasoning or sensing. Having performed recognition, we are much more confident about what properties the object does and does not have. However, sometimes such full recognition is more than what is needed. Sometimes all that is wanted to establish is a particular property. For example, to decide whether an object has a stem we do not need to know what the object is, we simply need to know if it is *one of* the objects with this property. (In our example, apples, pears, and cucumbers are the ones that do). We will call the version of the RACE algorithm that only tries to establish a given property *goal-directed RACE.*

The goal-directed RACE requires an additional input stating what property we are trying to establish. The output is not one of the objects in the knowledge base, but the *tv* of the goal property. The system either outputs 1, -1, or reports failure to establish the property which can be though of as equivalent to 0. (For sample runs of the algorithm see the second part of Appendix C).

The following additional notation will be used:

- g – the goal property, i.e the property the goal-directed RACE algorithm tries to establish
- tv_g – the truth value with respect to the goal property. I.e., the tv_g of the obhect O refers to $tv(O, g)$

The algorithm proceeds similarly to the original RACE algorithm, except it never has to refine its answer. Therefore, the notion of F is irrelevant here, and the algorithm returns as soon as it has developed sufficient confidence that the *tv* is 1 or is -1.

4.2 Determining success and scoring a question in goal-directed RACE

Implementing the algorithm requires that we replace two measures: the measure for whether a satisfactory answer has been found and the measure for estimating how useful a question is.

When do we have an answer? As it turns out, we can use the same procedure for establishing this as is described in Section 3.3, having only redefined what a *leader* and a *runner-up* are. The leader is the highest scoring candidate that informs us about the goal property. The runner up is the highest candidate with a *tv* different than that of the leader. That is, we designate as the *leader* the highest scoring candidate L such that

$tv(C, g) \neq 0$

and then we designate as the runner-up the highest scoring candidate R such that

$tv(R, g) \neq tv(C, g)$

Note that with this definition, it is possible for the runner-up to have a higher score than the leader (this happens whenever the highest-scoring candidate has the tv_g of 0).

When we estimate how good each question is, we also redefine which candidates affect the entropy. As in the main algorithm, we rescore the candidates, but then select the

highest with the tv_g of 1 and the highest with the tv_g of -1 and estimate the entropy of the set of the scores the two. If one of the alternatives is not to be found in the candidate set, the situation reduces to finding out more about the one that is preset. Introducing a measure that also factors in the score of the highest scoring candidate with tv_g of 0 could decrease the number of questions posed in a subset of cases.

Chapter 5. Future Work

5.1 Required Properties

Sometimes we do not need to weigh evidence carefully to find a leader. Sometimes a single property in which we are certain, i.e. which is *required*, can allow us to eliminate a particular candidate, or even eliminate all candidates but one.

For example, when we would like to recognize something that is round, hard, tasty, but is definitely not an apple, if we query the current algorithm with:

$$
\begin{aligned}
I = \quad \{ \quad & < p_1 = \text{isa round}, \quad tv(X, p_1) = 1 >, \\
& < p_2 = \text{isa hard}, \quad tv(X, p_2) = 1 >, \\
& < p_3 = \text{isa tasty}, \quad tv(X, p_3) = 1 >, \\
& < p_4 = \text{isa apple}, \quad tv(X, p_4) = -1 > \},
\end{aligned}
$$

APPLE would still be a strong candidate. This is because the query we formed did not capture the special status of p_4 being required.

Another example illustrating the usefulness of required properties is this: if we are absolutely certain that the object being recognized has the property "can moo" and the recognition takes place in the domain of farm animals, all the animals except for the cow – horses, pigs, goats, sheep, etc. – can be eliminated without further deliberation. So, recognition was possible based on presence of one key property.

Thus, adding even a simple mechanism that makes some properties required would make for recognition that:

- allows for more precise queries

- converges on an answer having asked fewer questions

In the current system, such behavior can be approximated by duplicating a given property in the input set, thereby making it more important than other properties. A more careful approach would introduce a parameter r for each property to indicate whether the property is required. Either a discrete r,

$$r \in \{0, 1\},$$

or a continuous r,

$$r \in [0, 1]$$

can be used. Every candidate is required to have the properties with r of 1, i.e. if C is a candidate, p is a property such that $r(p) = 1$, and

$$tv(C, p) \neq tv(X, p),$$

then the candidate is removed from the candidate set. This mechanism, however, will hurt the noise tolerance of the system.

5.2 Learning

The process of recognition queries about the properties of an object that belongs to the outside world. If the knowledge base is to accurately reflect the outside world, it needs to modify its knowledge in response to this information. We discuss in turn three directions for learning: repairing the knowledge base, refining knowledge base in the trouble areas, and learning about which properties are more useful than others.

Information about how an object mismatches a successful recognition can be used to repair the description of the object in the knowledge base (see [Rao 1991] for an investigation of knowledge repair). For example, successful recognition of a *green apple* as an *apple* in our domain could be used to learn that apples can be green as well as red.

More generally, the system can have a B-brain type sensor (see [Minsky 1986a] for a discussion on B-brains) that monitors the amount of energy that was spent on recognition. This would provide a measure of how good a model of the world the system has. If recognition takes much energy, the some of the current objects need to be removed or outfitted with additional properties that would ease recognition.

The system can also learn about questions. Some questions the algorithm poses can be harder to answer than others. This would manifest in the hard questions always returning a *tv* of zero. The system could empirically discover the useful and useless to ask questions and account for this in posing questions in future recognitions. Learning more complicated heuristics, such as "to compare apples and tomatoes, pay attention to hardness" is also possible.

5.3 Exploration strategy

Currently, the algorithm selects the question that produces the largest expected reduction of entropy. Perhaps, a different measure is more appropriate. There are several possibilities, out of which we mention two extremes:

- A measure that focuses more on verifying whether the leader can be made into a successful recognition (this is a depth-first exploration of sorts). One can use only the leader as the source of properties. Scoring the properties would also only account for their effect on the leader, ignoring the effect on the other candidates.

- On the opposite end of the spectrum is a measure that stresses elimination of the weak candidates, assigning higher scores to the questions that are more likely to add information about the weakest of the candidates.

The number of questions that must be posed can serve as a measure for comparing the performance of various strategies.

Chapter 6. Discussion:

The recognition algorithm presented in this thesis is a first-pass approximation of the human capacity of recognition. In building a larger system that routinely resorts to recognition, it will, most likely, be impossible to dispatch a task to the recognition module as easily as one calls a "square root" subroutine in a convention program. For successful integration, we believe much attention has to be paid to:

- control of the behavior of the agent by the larger system,
- patterns of usage of the agent in a larger system.

In the following discussion of the algorithm we pay particular attention to these demands of integration.

6.1 Functionality provided by the agent to a larger system

In invoking a recognition algorithm, a system can count on it having the following properties:

1. Ability to switch to a recognition completely different from its initial hypotheses in light of new information. This property comes from re-retrieving candidates after receiving an answer to a question posed by the system. An example of such behavior is given in Appendix C.
2. Doing more work in difficult areas. That is, when faced with needing to

disambiguate between two good matches, the algorithm will zoom in on the distinctions between the objects.

3. Providing a measure of the quality of the resulting recognition. This information is reflected in the fraction of the maximum score achieved by the leader, as well as in the margin by which it defeated its competition.
4. Ability to accommodate dynamically changing demands on the quality of recognition. For example, as time runs out, the larger system might want to settle for a weaker leader. This ability to cope with changing thresholds is due to the iterative nature of the selection process, each iteration being able to use its own parameters.

Routine usage of the RACE algorithm for accessing a knowledge base (KB) provides the service of tying the knowledge together. Without the algorithm, a KB is a collection of disparate objects with their descriptions. Which properties are important is not clear from examining the KB. With the algorithm, the knowledge is integrated. The similar objects are brought out in candidate sets. The features that distinguish objects from each other are the features the algorithm queries about.

With the recognition algorithm, the system does not have to as clearly commit whether an object is defined as a conjunction or a disjunction of its properties. An object is defined not directly by its properties, but by its emerging as a winner of the recognition process. Thus, an object's definition becomes less localized: properties of other objects affect when this object is the result of recognition, hence properties of other objects effectively affect the definition of the object.

6.2 When is recognition completed?

The goal of recognition is to gain predictive power. Recognition is an incremental

process, so the more information we have about the unknown object, the more, generally speaking, we can predict about it. In this thesis, we have presented algorithms for the two extremes: the standard algorithm of Chapter 3 is useful before anything is known about the predictions we are trying to make, and the goal-directed recognition is useful the prediction to establish or negate is explicitly stated.

The two presented algorithms have different criteria for deciding when the recognition is complete. A system of human-level sophistication is probably capable of adjusting the recognition to finer degrees of knowledge about the goal of recognition. A larger system could accomplish this by means of a separate agency providing a measure if how satisfactory the current result is. The output of this agency combined with data on feasibility of refining the recognition can determine when the recognition is to be declared a success and halted.

It may seem unsettling that recognition is refined as long as there are non-competing informative candidates. The behavior of the whole system can be more natural than the behavior of the agent alone for the following reasons:

- As discussed above, it can be tied to an agency assessing whether the needed amount of curiosity has been satisfied and whether the known objectives of recognition have been achieved.
- A manager can call repeatedly initiate the same recognition process each time folding in the properties interactively acquired in the previous loop. This would tie the amount of recognition to the total amount of energy spent on the loops, allowing the manager agent to control it. Even though the algorithm would proceed until the

energy runs out, the overall system will not. This solution is rather similar to the above one.

- A richer system can track base-level activations of each object in the system, preventing candidates with lower activations from being retrieved when the match to them is not very strong. Then the overly specific categories will not even enter the candidate set unless there is a very specific match to them. Such an approach will help the system model the notion of basic-level categories. A related idea is that of Level-Bands [Minsky 1986b].

6.3 Correlated properties

In Chapter 5, we discussed the possibility of using different exploration strategies. Here, we do not discuss change in the scores we try to effect with a question. Rather, we discuss more generally the issues involved in posing questions altogether.

A hurdle for the algorithm is its failure to "understand" the properties. This can become a problem in the following situation: the system knows that both a candidate and the unknown have the property "isa hard". If the system discovers that they are, additionally, not soft, it will give a higher score to the match with this candidate. The failure to understand that

$$tv(C, \text{isa hard}) = 1 \quad \textit{implies} \quad tv(C, \text{isa soft}) = -1$$

leads to unjustifiably increased confidence. Our algorithm makes the simplifying assumption that the impact of a property can be estimated independently of other properties. While ability to cope with noise ameliorates the problem, there is a strong need for ability to reason about correlation of properties. The correlation of properties

goes beyond pairwise correlation, because collections of properties can imply assertions that are not implied by any single property.

More generally, the set of properties the system knows about determines how novel objects are explored. In Heidegger's terminology, this is called the forestructure of the understanding. Situations are possible in which the choice of the clarifying questions that the system poses determines which candidate will succeed as a recognition. In other words, it is possible for the system to select a set of questions, correlated or not, that would lead it to favor an objectively weaker candidate over a stronger one. The danger of this is increased if recognition is combined with some unsupervised reinforcement learning about which questions to pose.

In order to keep the system from falling into such a trap, the system has to be able to reason about which properties are acceptable for recognition and which properties are correlated. Computing the correlations with Bayesian networks is one approach, but this route simply shifts the burden to learning the dependence network.

6.4 Actions a larger system can take if recognition fails

As they stand, the presented recognition algorithms do not have a way to cope with failing to recognize. A larger system should rarely, if ever halt with failure.

When recognition fails, one option is to simply rerun the process with increased energy. Akin to yanking harder on a door that has failed to open on the first try, it is a useful but simplistic method. It is appropriate when there is reason to believe that not enough energy was allocated the first time around.

If there are too many candidates, the larger system can aid the recognition module by eliminating some candidates on different grounds. This may be done by examining each candidate in turn and trying to find a proof of why the unknown cannot be this candidate. Derivation of more general "required properties" is also possible. Clearly, this is appropriate when failure stems from multitude of candidates, not their dearth.

Finally, recognition may fail due to dearth of candidates as well. In addition to reruning with more energy, another simple solution is to lower the candidate absolute and candidate relative thresholds, hoping that a larger set of candidates will be produce a leader.

Dearth of candidates may also be a sign of problems with the knowledge, not with the process. It may be that we are being exposed to a novel object which should not be recognized as any object in the knowledge base, or it may be that we went astray in determining the properties of the object. An example of the earlier is exposing the system with our domain to a *Peach*. Section 5.2, Learning, addressed this. Examples of the latter can arise especially if the properties themselves were results of other recognition processes. For example if we observe that the unknown "isa human" and "has wings", the larger system should backtrack and possibly re-recognize the "wings" as a "cape". Simply re-trying with more energy will be of little help.

6.5 Selection sets as interfaces

There is a unifying theme to several aspects of the algorithm. The theme is that of not committing to an answer prematurely. This section sketches how this notion can be

captured by shifting from subprocedures returning single answers to returning something we will call a selection set.

Recall that a race set is a collection of options, paired with scores, to which processes that will lead to selection of one are applied. Selection sets are also collections of alternatives, paired with scores. The difference is that the operations applied to a selection set are designed to integrate multiple sources of information in calculating the goodness of each answer.

Selection sets could be used within the algorithm to modularize the different reasons for filtering out the candidates. Each constraint could be represented as a separate agent acting on a selection set made up of preliminary candidates. The same could be done with constraints on properties.

The next section addresses what biases it might be useful to introduce into the recognition algorithm by means of selection sets. These biases aim to make the process of recognition appropriate in a wider range of situations.

6.6 Biasing the RACE

We can let the system bias the behavior of the algorithm by allowing prior processes of the larger system to influence activations of objects. These activations can be used to affect the initial selection of candidates, with the highly primed candidates being selected more easily. This mechanism can be thought of as making the RACE take place in the current context rather than in some idealized, fixed situation.

Sometimes the system may need to find a particular object rather than to recognize an unknown object. Such search for an object can also be viewed a RACE, biased in a particular way. The RACE process should be biased by always setting the leader of the RACE to be the *target* that we are trying to find. Proceeding with the algorithm will result in the RACE agent either recognizing the unknown as the target or failing. Because the general recognition is not performed, much energy can be saved by not thinking about something that is not of interest anyhow.

For example, if the one is trying to find a tomato, one can disable the leader selection, fix the tomato as the leader, and try to get the tomato to win the race. This will avoid posing questions that disambiguate between two uninteresting alternatives, focusing on the questions establishing the "tomato-ness" of the object. A different measure for failure of recognition could further speed up the process.

The recognition agency itself should, perhaps, expose its set of candidates for manipulation by other agents. An outside agent could factor in the prior frequencies of objects to account for the prior belief that the unknown could be a given candidate. For example, apples are more common than plums, so if one is told that there is a piece of fruit in a box, that fruit is more likely to be an apple than a plum.

As mentioned in Section 3.4, the system can bias its selection of a question based on come estimate of the cost of answering the question. Again, the selection set methodology is appropriate here.

Finally, the RACE can also be biased by setting values of the thresholds of the agent. By setting higher the relative and absolute leader spreads the system can make the recognition more noise-tolerant. By setting higher the absolute and relative leader thresholds, the system can require a more precise match of the unknown and the successful recognition.

Chapter 7. Summary

This thesis has presented the notion of race sets and argued for their wide applicability in AI. We have shown how race sets can be applied to selecting a leader in a noise-tolerant way. In the process, we have presented a method for the question that will help disambiguation the most.

There are two important directions for extending the algorithm.

The first is to take the RACE system beyond operating in a static, pre-supplied domain. Akin to an expert system, it is currently the job of the human designer to select the objects and their properties. This can be overcome by integrating the RACE methodology with some concept acquisition and concept repair mechanisms (perhaps as it was suggested in the section on future work).

The second direction is to make the RACE system deal with correlated properties. Given redundant or nearly redundant evidence, the algorithm adds it up as if it were independent; it does not collapse it. The needed extension is to incorporating mechanisms for understanding which properties imply which, as is addressed in discussion.

Appendix A. Glossary of Thresholds

We present a glossary of thresholds used in the system. The value in square brackets is the default value of the parameter. The explanations of thresholds are stated as constraints because the filtering in the algorithm is a conjunction of these constraints; exceeding a particular threshold does not imply being selected as a candidate or as the leader.

Energy [6] – Sets an upper bound of how much effort the algorithm is to spend on recognition. This energy reflects the *cost* of computation. If energy runs out and recognition has not been completed, the algorithm proceeds to the output step, which prunes the answer set or reports failure of recognition if the set is empty. Allocating a certain amount of energy is important in controlling the depth of recognition.

Candidate absolute threshold [.5] – Preliminary candidates with scores below this value do not become candidates.

Candidate relative threshold [.33] – Preliminary candidates with scores that are a lesser fraction of the maximum attainable score do not become candidates. The maximum score is the score of the perfect match. The maximum score is the score of the perfect match. For example, if the known properties are

tv(X, isa round) = 1,
tv(X, isa hard) = 1,
tv(X, isa tasty) = -1,

then the maximum obtainable score is 1 + 1 + .2 = 2.2. If this threshold is set at .3, then the preliminary candidates with scores below .3 * 2.2 = .66 will not become candidates.

Leader absolute threshold [2] – If a candidate's score is below this value, it will not be declared a leader of the set.

Leader relative threshold [.6] – If a candidate's score is a smaller fraction of the maximum obtainable score than this value, it will not be declared the leader of the set. Derivation of the maximum obtainable score is detailed in the "Candidate relative threshold" entry above.

Leader absolute spread [1.5] – The leader's score must exceed the score of any competitor by at least this amount for it to be the winner of the race set.

Leader relative spread [.3] – The leader's score must exceed the score of the competitor by at least this fraction of the leader's score for it to be the winner of the race set.

Appendix B.

B.1 Representation

The representation is centered around objects and their properties. Objects are organized into a multiple inheritance "isa" hierarchy, in which properties of the more general objects are inherited by the more specific ones. Each property consists of two words. For example, "isa apple" and "isa red" and "has taste" are all properties. The term "feature" is sometimes used to refer to the same concept as our "property".

The system contains assertions about truth value of particular objects having particular properties; we use *tv(O, p)* to refer to the truth value of the assertion that the object *O* has the property *p*. While there is room for much richness, we use a three-valued scheme:

$$tv\text{: Objects} \times \text{Properties} \rightarrow \{1, 0, -1\}$$

In human responses these would correspond most closely to, respectively, "yes", "don't know", and "no".

Another simplifying assumption we make is that the *tv* of an ambiguous statement is simply 0. For example, if there are both round and non-round ("round" is used here as a synonym of "spherical", in line with common usage) fruit (apples and pears, respectively), then

$$tv(\text{fruit, isa round}) = 0.$$

A more careful scheme would use a different *tv* to denote "don't know" and "can be either". Were our system capable of handling more complicated assertions, we could have:

$$tv(\forall \text{ X} \mid (\text{X isa fruit}) \wedge (\text{X isa spherical})) = -1, \text{ and}$$

$$tv(\exists \text{ X} \mid (\text{X isa fruit}) \wedge (\text{X isa spherical})) = 1.$$

However, considering that "X isa spherical" is neither a strong evidence for X being a fruit, nor strong evidence against it, the best decision is to make

$$tv(\text{fruit, isa spherical}) = 0.$$

For similar reasons, in our knowledge base

$$tv(\text{fruit, isa pear}) = 0.$$

Properties in our representation should not always be viewed as atomic entities.

Sometimes a property encodes a relationship between two objects, much like a link in a semantic network. For example, in the assertion

$$tv(O_1, isa\ O_2) = 1.$$

The property *isa* O_2 is not treated as atomic when, for example, we infer transitivity:

$$\frac{tv(O_1, isa\ O_2) = 1,\ tv(O_2, isa\ O_3) = 1}{tv(O_1, isa\ O_3) = 1}$$

(Inference is addressed in more detail in another section of this appendix).

In our system, each object has only a few positive properties. In the general case, (i.e. in a system with a more expressive representation and a more powerful inference mechanism) we believe that for any given object there are a great many properties that describe it. The task of focusing on the ones relevant in the current context is a challenging one but one we do not take up in this thesis. This view, however, gives rise to one phenomenon that we do model. Namely, we assume that retrieving the *tv(O, p)* for a property *p* of an object *O* takes a certain amount of energy, *en(O, p)*. If the amount of energy *E* is allocated to obtaining (this is a collective term for retrieving, deriving, or querying for) the properties of *O*, then *tv(O, p)* will be retrieved if and only if

$$en(O, p) \leq E.$$

For all other p, the system assumes $tv(O, p) = 0$.

Additionally, information about some properties being mutually exclusive is encoded in cross-exclusion objects. Cross-exclusion objects are also used to specify the inheritance hierarchy of the objects. We write down cross-exclusion objects in the following format:

Header <*connector*>
    $Member_1$
    $Member_2$
    ...
    $Member_n$

connector = value-of | isa

Value-of will be useful to us only as distinct from *isa*. Had we used *isa*, we would have had *tv*(apple, isa shape) = 1 by transitivity.
A cross-exclusion object is equivalent (and can be considered the shorthand for) the following set of assertions:

$\{\forall i\ tv(Memeber_i, connector\ Header) = 1\} \cup$

$\{\forall i\ \forall j\ tv(Member_i, isa\ Member_j) = -1\}$

For example,

vegetable <isa>
    tomato
    cucumber

Is equivalent to:

{ *tv*(tomato, isa vegetable) = 1
  tv(cucumber, isa vegetable) = 1
  tv(tomato, isa cucumber) = -1
  tv(cucumber, isa tomato) = -1 }

In a sense, the cross-exclusion object can be viewed as a trivial recognizer, with the recognition algorithm of the thesis building more powerful recognition out of these fairly primitive objects.

Any object that has properties beyond those that follow from its presence in a cross-exclusion object can potentially be the output of the recognition algorithm. In our domain, an object cannot be an answer if it does not have any properties. In a richer representation, we may have to explicitly label which objects belong to the set that are legitimate results of recognition.

B.2 Domain

The examples of recognition are all based on operating in a small domain of a few common fruits, vegetables and some other objects that fruits and vegetable may be mistaken for.

B.2.1 Structure of Objects

```
object <isa>
       fruit
       vegetable
       ball

fruit  <isa>
       apple
       pear
       orange
       lemon

vegetable     <isa>
       tomato
       cucumber
```

B.2.2 Structure of Properties

The hierarchy of the domain is represented as the following set of cross-exclusion objects:

```
shape  <value-of>
       round
       elongated
shape  <value-of>
       round
       pearshaped
hardness      <value-of>
       hard
       soft
taste  <value-of>
       sweet
       sour
color  <value-of>
       red
       green
       yellow
       orange
```

B.2.3 Assignment of Properties

We now present the direct (i.e. available without resorting to any inference) properties of objects in the domain. The following shorthand is used:

tomato

isa round

isa sweet -1

means

tv(tomato, isa round) = 1

tv(tomato, isa sweet) = -1

The set of objects that can potentially be the output of the recognition algorithm is: {Object, Fruit, Apple, Pear, Lemon, Vegetable, Tomato, Cucumber, Ball}

Such words as "red" or "juicy" cannot be the result of recognition (loosely speaking, the algorithm recognizes into nouns).

```
fruit
       isa juicy
       isa tasty
       isa edible

vegetable
       isa edible

apple
       isa round
       isa sweet
```

```
        isa red
        isa hard
        isa red
        has stem

pear
        isa elongated
        isa pearshaped
        isa sweet
        isa hard
        isa green
        has stem

lemon
        isa yellow
        isa sour
        isa elongated
        has stem -1

tomato
        isa round
        isa red
        isa soft
        isa sweet -1
        has stem 0

cucumber
        isa elongated
        isa green
        isa hard
        isa sweet -1
        has stem

ball
        isa edible -1
        isa round
        has taste -1
```

B.3 Inference

A knowledge base that has claims of being maintainable and scalable should not consist only of a set of assertions; is should also have a mechanism for inferring additional assertions from the explicitly stated ones. The mechanism could involve reasoning by analogy, could involve meta-knowledge when certain reasoning methods (e.g., the closed-world assumption) are applicable, etc. These fascinating reasoning mechanisms, however, are not the focus of the thesis. For our purposes, the inference mechanism is kept to a minimum.

Basically, the knowledge base employs one of the following two inference techniques when queried about the *tv* of a particular assertion: establishing properties by inheritance, and establishing negative properties. Below we explain each in a bit more detail. The appendix is concluded with a brief discussion of the role of energy in the inference process.

B.3.1 Establishing Properties by Inheritance

The properties are inherited in the usual way. In our system, the inheritance rule can be formulated as:

$$\frac{tv(O_1, isa\ O_2) = 1, tv(O_2, p) = v}{tv(O_1, p) = v}$$

(Here, v is one of $-1,0,1$). The problem of deciding which parent to inherit from when different parents have contradictory assertions regarding the *tv* has drawn much attention in the literature but is not addressed in this thesis. Our domain does not contain any such contradictions.

B.3.2 Establishing Negative Properties

How is it established, for example, that an apple isn't a tomato?

This is accomplished finding appropriate cross-exclusion objects through forward chaining from a word and its property.

Here is an example: to establish that

$$tv(\text{apple, isa tomato}) = -1,$$

we take the following steps:

"isa tomato" is forward-chained to "isa vegetable" "isa object". As a result, the set

$$\mathbf{F_1} = \{\text{Tomato, Vegetable, Object}\}$$

is formed.

At the same time, the assertion "isa apple" is also forward chained to yield:

$$\mathbf{F_2} = \{\text{Apple, Fruit, Object}\}.$$

Then we search for a pair of elements $E_1 \in \mathbf{F_1}$ and $E_2 \in \mathbf{F_2}$, such that

$$tv(E_1, \text{isa } E_2) = -1.$$

This can happen either because the assertion

$$tv(E_1, \text{isa } E_2) = -1$$

is given explicitly or E_1 and E_2 are both members of the same cross-exclusion object. In our example, such a pair exists, it is

$$\langle Vegetable, Fruit \rangle,$$

establishing that the original statement had the *tv* of -1.

B.3.3 On energy-bounded retrieval

Unfilled slots, if cannot be inferred, are assumed 0. Running out of energy in trying to retrieve a property would play a crucial role in a knowledge base complete search of which is not feasible. Terminating with the *tv* of 0 is the way to keep from looping and from getting infinitely deep into a sub-problem.

Throughout our examples, the retrieval energy is kept sufficiently high to discover all relevant positive properties in the domain via forward chaining, and all properties with the *tv* of -1 can also be established when the *tv* is explicitly queried, so anything encoded in the system is always discovered. Decreasing the retrieval energy would cause graceful degradation of the recognition algorithm.

Appendix C. Examples

This appendix presents some additional examples of the algorithm running. These examples illuminate the special and unusual cases that can arise.

C.1 Main recognition algorithm

C.1.1 Leader not in initial candidates

The system can come to a recognition that was not in its initial set of candidates at all. In this example, TOMATO is the eventual result of recognition, but it is not one of the four original candidates. We highlight in **BOLD** the important aspects of the example.

```
(recognize '((isa tasty) ((isa round) 0)
CANDIDATES:                LEMON       PEAR      APPLE       FRUIT
SCORES:    [MAX:  1]        1           1         1           1
-------------------------------------------------------------------
PROPERTIES & TV's:
 ISA TASTY       [1]        [1] 1      [1] 1      [1] 1     [1] 1
 ISA ROUND       [0]       [-1] 0     [-1] 0      [1] 0     [0] 0
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA ELONGATED))?: -1


CANDIDATES:                 APPLE       FRUIT      LEMON      PEAR
SCORES:    [MAX:1.2]          1.2         1          0.5        0.5
-------------------------------------------------------------------
PROPERTIES & TV's:
 ISA ELONGATED  [-1]      [-1] 0.2     [0] 0      [1] -.5     [1] -.5
 ISA TASTY       [1]       [1] 1       [1] 1      [1] 1       [1] 1
 ISA ROUND       [0]       [1] 0       [0] 0     [-1] 0      [-1] 0
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA RED))?: 1


CANDIDATES:               APPLE       TOMATO      FRUIT
SCORES:    [MAX:2.2]        2.2         1.2         1
-------------------------------------------------------
PROPERTIES & TV's:
 ISA RED         [1]       [1] 1       [1] 1      [0] 0
 ISA ELONGATED  [-1]      [-1] 0.2    [-1] 0.2    [0] 0
 ISA TASTY       [1]       [1] 1       [0] 0      [1] 1
 ISA ROUND       [0]       [1] 0       [1] 0      [0] 0
SUCCESSFUL RECOGNITIONS: NIL
```

TOMATO *becomes a candidate at this step.*

```
QUESTION: WHAT IS tv(X, (ISA FRUIT))?: -1


CANDIDATES:                    APPLE        TOMATO
SCORES:   [MAX:2.4]          1.7          1.4
-----------------------------------------------------
PROPERTIES & TV's:
 ISA FRUIT       [-1]      [1] -.5     [-1] 0.2
 ISA RED         [1]       [1] 1         [1] 1
 ISA ELONGATED   [-1      [-1] 0.2      [-1] 0.2
 ISA TASTY       [1]       [1] 1         [0] 0
 ISA ROUND       [0]       [1] 0         [1] 0
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA HARD))?: -1


CANDIDATES:                    TOMATO       APPLE
SCORES:   [MAX:2.6]          1.6          1.2
-----------------------------------------------------
PROPERTIES & TV's:
 ISA HARD        [-1]      [-1] 0.2      [1] -.5
 ISA FRUIT       [-1]      [-1] 0.2      [1] -.5
 ISA RED         [1]        [1] 1        [1] 1
 ISA ELONGATED   [-1]      [-1] 0.2     [-1] 0.2
 ISA TASTY       [1]        [0] 0        [1] 1
 ISA ROUND       [0]        [1] 0        [1] 0
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA VEGETABLE))?: 1


CANDIDATES:                    TOMATO       VEGETABLE
SCORES:   [MAX:3.6]          2.6          1.2
-----------------------------------------------------
PROPERTIES & TV's:
 ISA VEGETABLE   [1]        [1] 1        [1] 1
 ISA HARD        [-1]      [-1] 0.2      [0] 0
 ISA FRUIT       [-1]      [-1] 0.2     [-1] 0.2
 ISA RED         [1]        [1] 1        [0] 0
 ISA ELONGATED   [-1]      [-1] 0.2      [0] 0
 ISA TASTY       [1]        [0] 0        [0] 0
 ISA ROUND       [0]        [1] 0        [0] 0
SUCCESSFUL RECOGNITIONS: (TOMATO)


ENERGY SPENT:  6          ENERGY REMAINING:  0
ANSWER:        TOMATO
```

C.1.2 Cannot disambiguate between candidates

In some situations, recognition does not receive the answers that help to disambiguate between the candidates. In that case, it eventually runs out of energy. Recognition fails having never established a leader.

```
(recognize '(((isa juicy) -1) (isa tasty)))

CANDIDATES:                    LEMON        PEAR         APPLE        FRUIT
SCORES:    [MAX:1.2]          0.5          0.5          0.5          0.5
-------------------------------------------------------------------------
PROPERTIES & TV's:
 ISA JUICY       [-1]        [1] -.5      [1] -.5      [1] -.5      [1] -.5
 ISA TASTY       [1]         [1] 1        [1] 1        [1] 1        [1] 1
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA ELONGATED))?: 1


CANDIDATES:                    LEMON        PEAR      CUCUMBER
SCORES:    [MAX:2.2]          1.5          1.5          1
-----------------------------------------------------------
PROPERTIES & TV's:
 ISA ELONGATED   [1]         [1] 1        [1] 1        [1] 1
 ISA JUICY       [-1]        [1] -.5      [1] -.5      [0] 0
 ISA TASTY       [1]         [1] 1        [1] 1        [0] 0
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA FRUIT))?: -1


CANDIDATES:                    CUCUMBER     LEMON        PEAR
SCORES:    [MAX:2.4]          1.2          1.0          1.0
-----------------------------------------------------------
PROPERTIES & TV's:
 ISA FRUIT       [-1]       [-1] 0.2     [1] -.5      [1] -.5
 ISA ELONGATED   [1]         [1] 1        [1] 1        [1] 1
 ISA JUICY       [-1]        [0] 0        [1] -.5      [1] -.5
 ISA TASTY       [1]         [0] 0        [1] 1        [1] 1
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (HAS STEM))?: 1


CANDIDATES:                    CUCUMBER     PEAR
SCORES:    [MAX:3.4]          2.2          2.0
-----------------------------------------------
PROPERTIES & TV's:
 HAS STEM        [1]         [1] 1        [1] 1
 ISA FRUIT       [-1]       [-1] 0.2      [1] -.5
 ISA ELONGATED   [1]         [1] 1        [1] 1
 ISA JUICY       [-1]        [0] 0        [1] -.5
 ISA TASTY       [1]         [0] 0        [1] 1
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA VEGETABLE))?: -1


CANDIDATES:                    PEAR         CUCUMBER
```

```
SCORES:   [MAX:3.6]        2.2          1.7
-----------------------------------------------
PROPERTIES & TV's:
 ISA VEGETABLE  [-1]      [-1] 0.2      [1] -.5
 HAS STEM       [1]        [1] 1        [1] 1
 ISA FRUIT      [-1]       [1] -.5     [-1] 0.2
 ISA ELONGATED  [1]        [1] 1        [1] 1
 ISA JUICY      [-1]       [1] -.5      [0] 0
 ISA TASTY      [1]        [1] 1        [0] 0
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA YELLOW))?: -1


CANDIDATES:                   CUCUMBER     PEAR
SCORES:   [MAX:3.8]         1.9          1.7
-----------------------------------------------
PROPERTIES & TV's:
 ISA YELLOW     [-1]      [-1] 0.2      [1] -.5
 ISA VEGETABLE  [-1]       [1] -.5     [-1] 0.2
 HAS STEM       [1]        [1] 1        [1] 1
 ISA FRUIT      [-1]      [-1] 0.2      [1] -.5
 ISA ELONGATED  [1]        [1] 1        [1] 1
 ISA JUICY      [-1]       [0] 0        [1] -.5
 ISA TASTY      [1]        [0] 0        [1] 1
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA GREEN))?: -1

SORRY, RAN OUT OF ENERGY

ENERGY SPENT:  6          ENERGY REMAINING:  0
ANSWER:       NIL
```

C.1.3 Not similar to anything

In some cases no object in the knowledge base is a sufficiently good match. In this case recognition fails, but not due scarcity of candidates. This case is the dual of the previous one.

```
(recognize  '((isa juicy) ((isa fruit) -1))

CANDIDATES:                  LEMON       PEAR        APPLE        FRUIT
SCORES:   [MAX:1.2]        0.5          0.5         0.5          0.5
-------------------------------------------------------------------------
PROPERTIES & TV's:
 ISA JUICY      [1]        [1] 1        [1] 1       [1] 1      [1] 1
 ISA FRUIT      [-1]       [1] -.5      [1] -.5     [1] -.5      [1] -.5
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (ISA ELONGATED))?: -1

CANDIDATES:                    APPLE        FRUIT
SCORES:   [MAX:1.4]         0.7          0.5
-----------------------------------------------------
```

```
PROPERTIES & TV's:
 ISA ELONGATED  [-1]      [-1] 0.2      [0] 0
 ISA JUICY      [1]        [1] 1        [1] 1
 ISA FRUIT      [-1]       [1] -.5      [1] -.5
SUCCESSFUL RECOGNITIONS: NIL

QUESTION: WHAT IS tv(X, (HAS STEM))?: -1


CANDIDATES:
SCORES:   [MAX:1.6]
-------------------------------------------------------
PROPERTIES & TV's:
 HAS STEM       [-1]
 ISA ELONGATED  [-1]
 ISA JUICY       [1]
 ISA FRUIT      [-1]
SUCCESSFUL RECOGNITIONS: NIL

SORRY, NO CANDIDATES ARE LEFT! - QUITTING

ENERGY SPENT:  3          ENERGY REMAINING:  3
ANSWER:       NIL
```

C.1.4 Initial Recognition is Refined

A major feature of the algorithm is refining of recognition when candidates that add information are present. This example demonstrates how initial recognition of a FRUIT is refined to an APPLE.

```
(recognize '((isa tasty) (isa juicy) (isa edible) ((isa hard) 0)
             ((isa red) -1) ((isa elongated) -1) ((isa yellow) -1)))

CANDIDATES:                  FRUIT       APPLE       LEMON       PEAR
SCORES:   [MAX:3.6]            3           2.9         2.2         2.2
-------------------------------------------------------------------------
PROPERTIES & TV's:
 ISA TASTY      [1]        [1] 1       [1] 1       [1] 1       [1] 1
 ISA JUICY      [1]        [1] 1       [1] 1       [1] 1       [1] 1
 ISA EDIBLE     [1]        [1] 1       [1] 1       [1] 1       [1] 1
 ISA HARD       [0]        [0] 0       [1] 0       [0] 0       [1] 0
 ISA RED        [-1]       [0] 0       [1] -.5    [-1] 0.2    [-1] 0.2
 ISA ELONGATED  [-1]       [0] 0      [-1] 0.2     [1] -.5     [1] -.5
 ISA YELLOW     [-1]       [0] 0      [-1] 0.2     [1] -.5     [1] -.5
SUCCESSFUL RECOGNITIONS: (FRUIT)

QUESTION: WHAT IS tv(X, (ISA SWEET))?: 1


CANDIDATES:                  APPLE       PEAR        FRUIT       LEMON
SCORES:   [MAX:4.6]            3.9         3.2         3           1.7
-------------------------------------------------------------------------
PROPERTIES & TV's:
 ISA SWEET      [1]        [1] 1       [1] 1       [0] 0      [-1] -.5
 ISA TASTY      [1]        [1] 1       [1] 1       [1] 1       [1] 1
 ISA JUICY      [1]        [1] 1       [1] 1       [1] 1       [1] 1
 ISA EDIBLE     [1]        [1] 1       [1] 1       [1] 1       [1] 1
```

```
 ISA HARD        [0]       [1] 0       [1] 0       [0] 0       [0] 0
 ISA RED         [-1]      [1] -.5    [-1] 0.2     [0] 0      [-1] 0.2
 ISA ELONGATED   [-1]     [-1] 0.2     [1] -.5     [0] 0       [1] -.5
 ISA YELLOW      [-1]     [-1] 0.2     [1] -.5     [0] 0       [1] -.5
SUCCESSFUL RECOGNITIONS: (FRUIT)

QUESTION: WHAT IS tv(X, (HAS STEM))?: 1


CANDIDATES:                APPLE        PEAR        FRUIT
SCORES:   [MAX:5.6]        4.9         4.2         3
-------------------------------------------------------
PROPERTIES & TV's:
 HAS STEM        [1]       [1] 1       [1] 1       [0] 0
 ISA SWEET       [1]       [1] 1       [1] 1       [0] 0
 ISA TASTY       [1]       [1] 1       [1] 1       [1] 1
 ISA JUICY       [1]       [1] 1       [1] 1       [1] 1
 ISA EDIBLE      [1]       [1] 1       [1] 1       [1] 1
 ISA HARD        [0]       [1] 0       [1] 0       [0] 0
 ISA RED         [-1]      [1] -.5    [-1] 0.2     [0] 0
 ISA ELONGATED   [-1]     [-1] 0.2     [1] -.5     [0] 0
 ISA YELLOW      [-1]     [-1] 0.2     [1] -.5     [0] 0
SUCCESSFUL RECOGNITIONS: (FRUIT)

QUESTION: WHAT IS tv(X, (ISA ROUND))?: 1


CANDIDATES:                  APPLE        PEAR       FRUIT
SCORES:   [MAX:6.6]          5.9         3.7         3
-------------------------------------------------------------
PROPERTIES & TV's:
 ISA ROUND       [1]       [1] 1      [-1] -.5     [0] 0
 HAS STEM        [1]       [1] 1       [1] 1       [0] 0
 ISA SWEET       [1]       [1] 1       [1] 1       [0] 0
 ISA TASTY       [1]       [1] 1       [1] 1       [1] 1
 ISA JUICY       [1]       [1] 1       [1] 1       [1] 1
 ISA EDIBLE      [1]       [1] 1       [1] 1       [1] 1
 ISA HARD        [0]       [1] 0       [1] 0       [0] 0
 ISA RED         [-1]      [1] -.5    [-1] 0.2     [0] 0
 ISA ELONGATED   [-1]     [-1] 0.2     [1] -.5     [0] 0
 ISA YELLOW      [-1]     [-1] 0.2     [1] -.5     [0] 0
SUCCESSFUL RECOGNITIONS: (APPLE FRUIT)


ENERGY SPENT:  4          ENERGY REMAINING:  2
ANSWER:        APPLE
```

C.1.5 A clear leader, no additional work necessary

Finally, some cases are easy. A clear leader emerges in the first pass. No additional questions are necessary.

```
(recognize '((isa elongated) (isa green) (isa hard)
((isa sweet) -1) (has stem)))

CANDIDATES:                  CUCUMBER     PEAR
SCORES:   [MAX:4.2]          4.2         2.0
```

```
-----------------------------------------------------
PROPERTIES & TV's:
 ISA ELONGATED  [1]        [1] 1        [1] 1
 ISA GREEN      [1]        [1] 1       [-1] -.5
 ISA HARD       [1]        [1] 1        [1] 1
 ISA SWEET      [-1]      [-1] 0.2      [1] -.5
 HAS STEM       [1]        [1] 1        [1] 1
SUCCESSFUL RECOGNITIONS: (CUCUMBER)


ENERGY SPENT:  1          ENERGY REMAINING:  5
ANSWER:        CUCUMBER
```

C.2 Goal-directed recognition

We present two examples of goal-directed recognition. The first one illustrates that a property can be established without the need for full disambiguation. The second illustrates that a system poses different questions when given a goal that it would without a goal.

C.2.1 Full RACE unnecessary to establish a particular property

Even though two mutually exclusive candidates are tied for leadership, the goal can still be established:

```
(recognize '((isa tasty) (isa hard) (isa sweet)) :goal '(has stem))

CANDIDATES:                PEAR         APPLE        FRUIT
SCORES:    [MAX:  3]         3            3            1
-----------------------------------------------------------
PROPERTIES & TV's:
 ISA TASTY       [1]     [1] 1        [1] 1        [1] 1
 ISA HARD        [1]     [1] 1        [1] 1        [0] 0
 ISA SWEET       [1]     [1] 1        [1] 1        [0] 0

TV FOR (HAS STEM):       [1]          [1]          [0]


ENERGY SPENT:  1          ENERGY REMAINING:  5
ANSWER:       tv(X, HAS STEM) = 1
```

C.2.2 Goal-directed RACE algorithm poses different questions than the standard RACE

Here is an example where the input is inconsistent, but the inconsistency does not need to be fully eliminated to establish the goal. Note that the candidates in the final round do not even share an immediate parent:

```
(recognize '((has stem) (isa yellow) (isa vegetable))
:goal '(isa elongated))

CANDIDATES:                CUCUMBER      PEAR      VEGETABLE
SCORES:   [MAX:  3]        1.5          1.5         1
------------------------------------------------------------
PROPERTIES & TV's:
 HAS STEM       [1]     [1] 1        [1] 1        [0] 0
 ISA YELLOW     [1]    [-1] -.5      [1] 1        [0] 0
 ISA VEGETABLE  [1]     [1] 1       [-1] -.5      [1] 1

TV FOR (ISA ELONGATED):   [1]          [1]          [0]

   QUESTION: WHAT IS tv(X, (ISA HARD))?: 1
```

The question posed as a response to the same query sans the goal, i.e. to `(recognize '((has stem) (isa yellow) (isa vegetable))` *is different. It is "*`WHAT IS tv(X, (ISA FRUIT))?`*"*

```
CANDIDATES:                CUCUMBER      PEAR
SCORES:   [MAX:  4]        2.5           2.5
-----------------------------------------------
PROPERTIES & TV's:
 ISA HARD       [1]     [1] 1        [1] 1
 HAS STEM       [1]     [1] 1        [1] 1
 ISA YELLOW     [1     [-1] -.5      [1] 1
 ISA VEGETABLE  [1]     [1] 1       [-1] -.5

TV FOR (ISA ELONGATED): [1]          [1]


ENERGY SPENT:  2          ENERGY REMAINING:  4
ANSWER:       tv(X, ISA ELONGATED) = 1
```

References

Brachman R.J., McGuinness D.L., Patel-Schneider P.F., Resnick L.A., & Borgida A., 1991. Living with CLASSIC: When and How to Use a KL-ONE-Like Language. In: J. Sowa, (ed.), *Principles of Semantic Networks: Explorations in the representation of knowledge*, San Mateo: Morgan-Kaufmann, 401-459.

Bobick, A.F. 1987. Natural Object Categorization. *Revised Ph.D. Thesis, MIT AI Laboratory Technical Report 1001.*

Brodley, C., 1995. Recursive Automatic Bias Selection for Classifier Construction. *Machine Learning*, 20, 63

Clancey, W.J. 1985. Heuristic Classification. *Stanford CS report STAN-CS-85-1066.*

MacGregor R.M. 1988. A Deductive Pattern Matcher. *Proceedings of the Seventh National Conference on Artificial Intelligence, (AAAI 88),* 403-408

MacGregor R.M. & Brill D. 1992. Recognition Algorithms for the Loom Classifier. *Proceedings of the Tenth National Conference on Artificial Intelligence, (AAAI 92),* 774-779

Minsky, M. 1986a. *The Society of Mind.* Section 6.4. New York: Simon & Schuster.

Minsky, M. 1986b. *The Society of Mind.* Sections 8.5-8.7. New York: Simon & Schuster.

Minsky, M. 1986c. *The Society of Mind.* Section 16.4. New York: Simon & Schuster.

Minsky, M. 1986d. *The Society of Mind.* Sections 19.4-19.10. New York: Simon & Schuster.

Rao, S. 1991. Knowledge Repair. *Master's thesis, Department of EECS, Massachusetts Institute of Technology.*

Silverman, D.L. 1984, An Interactive, Incremental Classifier. *Thesis, Department of Computer and Information Science, University of Pennsylvania.*

Ullman, S. 1996. *High Level Vision.* Cambridge, MA:MIT Press.