Question-Generation in Constraint-Based Expert Systems

James Bowen and Chavalit Likitvivatanavong

Cork Constraint Computation Centre, UCC, Cork, Ireland
{j.bowen,chavalit}@4c.ucc.ie
http://www.4c.ucc.ie

Abstract. If-then rules are the core knowledge representation technology in currently deployed expert systems. However, if we replace rules by constraints, we can greatly extend both the expressive and reasoning functionality of such systems. An open issue, however, is how best should constraint-based expert systems decide which questions to ask their users during a consultation – how can an interactive constraint-based expert system extract needed data from its users while imposing the smallest possible burden on those users. In this paper we consider three types of approach to the task.

1 Introduction

Expert systems constitute one of the most successful application areas for AI techniques - they have been assimilated into the mainstream where they are widely deployed, frequently in concert with non-AI software technologies [9].

The core AI technology used in currently deployed expert systems is rule-based knowledge representation - expert knowledge about the problem domain is represented in the form of if-then rules. Constraint-based reasoning offers the prospect of greatly extending the functionality of expert systems, in terms of both knowledge representation and inferential competence [2–4]. Their greater expressiveness in knowledge representation and greater competence in drawing inferences from available knowledge suggest that constraint-based technologies will find widespread application in future expert systems.

When eliciting data from its user, an expert system should ask as few questions as possible and should avoid questions whose answers would require difficult, time-consuming or expensive investigation by the user. There are two ways to establish the strategy that is to be used by an expert system for choosing which questions to ask: the programmer who develops the system can programme the strategy explicitly or the strategy can be determined automatically by the run-time system for the formalism in which the expert system is implemented. Explicitly programmed control of input/output requires an imperative formalism – for example, imperative rule-based systems [6]. In purely declarative formalisms, such as declarative rule-based systems, question generation must be performed automatically by the run-time system for the formalism. Constraints are declarative constructs and, therefore, cannot be used by an expert system programmer to specify input/output. Instead, in constraint-based expert systems, question-generation must be performed automatically by the run-time system.

Most constraints research has been directed at the development of efficient constraint propagation and/or search algorithms [12]. Little has been done on the interface issues important in constraint-based expert systems. While explanation has received some attention [1, 8, 5, 7], nothing has yet been published on algorithms for question-generation in constraint-based reasoning. This paper aims to break the ground in this new area.

2 Question-Generation

It is quite easy for an expert system based on declarative rules to decide what questions to ask its user. The system's hypotheses comprise the roots of a set of interlaced trees whose leaves represent possible data points; when considering a hypothesis, the system backward-chains from the root to its leaves, asking questions about these.

Deciding what questions to ask is more complicated for a constraint-based expert system. To illustrate this point simply, we will consider the following party game. One guest is given a set of four coloured pencils and a map of Europe, on which the international borders are drawn but on which the countries are not coloured. He colours the countries, respecting the rule that no two adjacent countries may have the same colour; then he chooses some country and asks his fellow guests to determine the colour which he has used for this country. They are allowed to ask him what colours he has used for other countries but the point of the game is that they try to ask as few questions as possible when seeking information to help them determine the colour of the country he has chosen.

Consider a restricted version of this game, involving only three colours (red, green and blue) and a map which contains only four countries (Austria, Germany, Switzerland and Italy); the adjacency relations in this map are shown in Figure 1 (a). Suppose that the guest who has coloured the map has asked you to determine what colour he has used for Italy. How many countries would you ask him about, while collecting information to allow you to determine Italy's colour? Which country should you ask about first? Should your first question be about one of the neighbours of Italy – Switzerland or Austria?

In fact, you need to ask only one question and that question should be about Germany, which is not a neighbour of Italy at all! Germany must have a different colour from Austria and Switzerland, whose colours must be different from each other. Italy must also have a different colour from Austria and Switzerland. Since there are only three colours, Italy must have the same colour as Germany.

The task posed by the above party game may be characterised as follows. Given a constraint network and a targeted parameter from that network, determine the smallest possible subset of the other parameters from the network so that, if the values of these other parameters are known, the value of the targeted parameter can be determined.

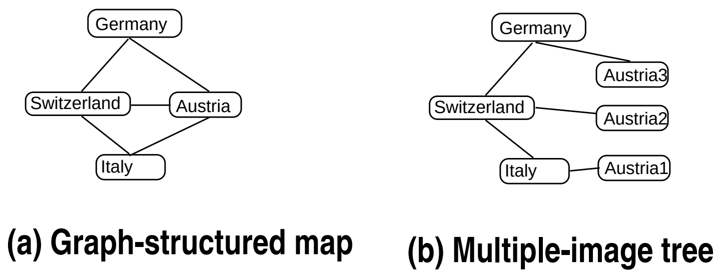

Fig. 1. Map of Central Europe

This kind of constraint-based problem, which we could call the Question-Generation Problem (QGP), is not a frivolous one. There are many economically or socially significant instances of it. Consider medical diagnosis, for example. A body of diagnostic expertise could be represented as a constraint network in which one parameter represents the disease afflicting a patient, while other parameters represent the results of possible diagnostic tests that could be performed on him. Since diagnostic tests can be dangerous, expensive and/or time-consuming, an expert diagnostician tries to minimize the number of tests that he calls for. As another example, consider an automated telephone-based sales system. Knowledge for this domain could be represented as a constraint network in which one parameter represents the product which best meets a customer's needs, while other parameters represent salient aspects of the situation in which the customer proposes to use the product that he wishes to purchase. Since a customer is likely to hang up in frustration if he is asked too many questions, a successful system will try to ask as few questions as possible.

3 Some Notation and Definitions

To facilitate the rest of this discussion, we will introduce some notation and definitions. In the literature, several different definitions are given for constraint networks, with varying degrees of formality. However, they may all be regarded as variations of the following theme:

Definition 1, Constraint Network:
A constraint network is a triple $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$. $\mathbf{D}$ is a finite set of $p > 0$ domains, the union of whose members forms a universe of discourse, $\mathcal{U}$. $\mathbf{X}$ is a finite tuple of $q > 0$ non-recurring parameters. $\mathbf{C}$ is a finite set of $r \geq q$ constraints. In each constraint $C_k(T_k) \in \mathbf{C}$, T_k is a sub-tuple of $\mathbf{X}$, of arity a_k; $C_k(T_k)$ is a relation, a subset of the a_k-ary Cartesian product $\mathcal{U}^{a_k}$. In $\mathbf{C}$ there are q unary constraints of the form $C_k(X_j) = D_i$, one for each parameter X_j in $\mathbf{X}$, restricting it to range over some domain $D_i \in \mathbf{D}$ which is called the *domain of* X_j.

The overall network constitutes an intensional specification of the simultaneous value assignments that can be assumed by the parameters. In other words, the network constitutes an intensional specification of a q-ary relation on $\mathcal{U}^q$, in which each q-tuple provides an ordered group of values, each value being an assignment for the corresponding parameter in $\mathbf{X}$. This implicitly specified relation is called the *intent* of the network:

Definition 2, The Intent of a Constraint Network:
The intent of a constraint network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$ is
$\Pi^{\mathbf{D},\mathbf{X},\mathbf{C}} = \overline{C_1}(\mathbf{X}) \cap ... \cap \overline{C_r}(\mathbf{X})$,
where, for each constraint $C_k(T_k) \in \mathbf{C}$, $\overline{C_k}(\mathbf{X})$ is its cylindrical extension[1] in $\mathcal{U}^q$.

Note that the definitions given above admit infinite domains, implying that the universe of discourse $\mathcal{U}$ may be infinite and that the constraint relations may be infinite, thereby making it possible that the intent of a network may be an infinite relation. (All such infinite sets can, of course, be expressed intensionally.)

Many different forms of constraint satisfaction problem (CSP) have been distinguished. The most common one, the Exemplification CSP, can be defined in terms of a network intent, as follows:

Definition 3, The Exemplification CSP:
Given a network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$: return nil if $\Pi^{\mathbf{D},\mathbf{X},\mathbf{C}}$ is empty; otherwise, return some tuple from $\Pi^{\mathbf{D},\mathbf{X},\mathbf{C}}$.

Another problem which will be relevant in this paper, the Enumeration CSP, can be defined as follows:

Definition 4, The Enumeration CSP:
Given a network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$: return $\Pi^{\mathbf{D},\mathbf{X},\mathbf{C}}$.

4 One Approach to Question Generation

Let us return to the task of deciding which question(s) to ask the person who has coloured the map of Central Europe we saw earlier. How could a constraint-based expert system compute that it needs to ask only one question (about Germany)? In this section, we will consider one approach, which relies on prior computation of the intent that is implicit in the constraint network. Since computing the intent of a network is computationally expensive for large networks, we will then, in later sections, move on to discuss some approaches which do not require knowledge of the network intent.

[1] The term "cylindrical extension" includes the word "cylindrical" because, when a set of points forming a circle in 2-D space is extended into 3-D space, the result is a cylinder. More generally, the cylindrical extension of a set of points in m-D space into n-D space ($m > 0$, $n > m$) is the set of points which results from generating, from each of the m-D space points, a set of points which have all possible values in the extra dimensions.

Suppose that, in the network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$ for this example, the parameters in tuple $\mathbf{X}$ are ordered $\langle$ Germany, Switzerland, Austria, Italy $\rangle$. The intent $\Pi^{\mathbf{D},\mathbf{X},\mathbf{C}}$ that is implicitly defined in Figure 1 (a) is, then, $\{ \langle r, g, b, r \rangle$, $\langle r, b, g, r \rangle$, $\langle g, r, b, g \rangle$, $\langle g, b, r, g \rangle$, $\langle b, r, g, b \rangle$, $\langle b, g, r, b \rangle \}$. If the system knew that the contents of $\Pi^{\mathbf{D},\mathbf{X},\mathbf{C}}$ are as just given, it could compute from this that, for each possible value of Germany, only one value is allowed for Italy. It could also compute that, for each possible value of Switzerland, two values are allowed for Italy. Similarly, it could also compute that each possible value of Austria admits two values for Italy. Therefore, the system would know that, if it wants to identify the value of Italy as quickly as possible, it should ask about Germany.

It will be useful to cast this argument in terms of probabilities and expected values of domain sizes. If the system knew that the contents of $\Pi^{\mathbf{D},\mathbf{X},\mathbf{C}}$ are as given above, it could, *before asking any question*, compute an expected domain size for parameter Italy after learning the answer to each possible first question. Let $prob(X = y)$ denote the probability that, if the system asks for the colour of some country X, it will receive the answer y. The expected domain size for Italy after asking about Germany is

$$(prob(Germany = r) * 1) + (prob(Germany = g) * 1) + (prob(Germany = b) * 1).$$

Simplifying this, we get

$$(prob(Germany = r) + prob(Germany = g) + prob(Germany = b)) * 1$$

which equals 1, since the probabilities for the different values of Germany must sum to unity. Similarly, the expected domain size for Italy after asking about Austria is

$$(prob(Austria = r) * 2) + (prob(Austria = g) * 2) + (prob(Austria = b) * 2)$$

which evaluates to 2. Finally, the expected domain size for Italy after asking about Switzerland, which is

$$(prob(Switz = r) * 2) + (prob(Switz = g) * 2) + (prob(Switz = b) * 2)$$

also evaluates to 2. Thus, the expected domain size, after asking one question, is smallest if that question is about Germany. Therefore, the system knows that, if it is interested in narrowing the possibilities for Italy as quickly as possible, its best chance of doing so is to ask about Germany first.

5 Sampling the Intent

Although the rationale for question selection given above was expressed in terms of the expected domain size for the target parameter, the same approach can be explained from a different perspective. We could have used Shannon's Information Theory [11]: choosing between questions on the basis of their effectiveness in reducing entropy would lead to exactly the same questioning strategy. Indeed, our use of expected domain size in choosing questions is similar to the use of entropy reduction in Decision Tree learning [10].

In fact, an alternative approach to our task is prompted by the observation that Question Generation in constraint networks is similar to Decision Tree Learning. The approach to Question Generation which we considered above was based on the assumption that the system knew the intent of the network – in other words, the system must have previously solved the Enumeration CSP for the network. Instead, as we shall see below, the approach prompted by the analogy with Decision Tree Learning requires solving the Exemplification CSP.

A decision tree is a tool for classifying instances from some population. A non-leaf node in the tree is a question whose children are connected to it by edges which correspond to the possible answers for the question. A leaf node in the tree is a classification. When an instance from the population is to be classified, it is analysed according to the questions encountered along the path, from the root node, that correspond to the answers which the attributes of the instance give to the questions; this path eventually leads to a leaf-node which is the classification for the instance.

An interactive consultation with a constraint-based expert system can be compared with the process by which a decision tree classifies an instance. The domain of the target parameter in a constraint-based expert system corresponds to the range of possible classifications that may result from a decision tree analysis of an instance. The non-target parameters in a constraint-based expert system correspond to the instance attributes that are examined by the question nodes in a decision tree. In fact, the population of instances which a decision tree is supposed to be capable of classifying corresponds to the intent of the network which underlies a constraint-based expert system: each tuple in the constraint network intent contains the description of an instance and its corresponding classification – the value, in the tuple, of the target parameter corresponds to the classification, while the values of the non-target parameters correspond to the attribute-values of the instance.

While decision trees could be constructed manually, they are normally constructed by machine learning algorithms. Such an algorithm is given a *sample* from the population along with, for each member of the sample, its appropriate classification. We have seen that the population of instances to be classified by a decision tree corresponds to the intent of the network for a constraint-based expert system. Since a decision tree learning algorithm needs only a sample of the population, this suggests that we should be able to generate questions for a constraint-based expert system if we know only a sample of the network intent[2].

In other words, instead of requiring prior solution to the Enumeration CSP (in order to compute the complete intent), it should be possible to learn an appropriate set of questions from a set of solutions to the Exemplification CSP. An investigation of this approach is part of our ongoing research. The basic idea is

[2] Acknowledgement: this observation was made by Pat Langley when the relationship between decision tree learning and question generation in constraint-based expert systems was described to him.

as follows: use repeated invocations of a local search (or repair-based) algorithm to find several (hopefully randomly distributed) solutions to the Exemplification CSP; then use a decision tree learning algorithm to learn a decision tree from this sample of the overall intent of the constraint network.

6 Intent-independent Question-Generation

So far, we have outlined two approaches to question generation, one of which requires prior computation of the network intent (by solving the Enumeration CSP) while the other requires prior computation of a random sample from the network intent (by computing several randomly distributed solutions to the Exemplification CSP).

An obvious ambition, therefore, is to find some approach which does not require prior knowledge of any part of the network intent, an approach which does not require prior solution even of the Exemplification CSP, let alone prior solution of the Enumeration CSP.

Such an approach does exist. Before introducing it, is is useful to define some concepts.

6.1 Some Concepts

Definition 5, Target Shadow:
The set of values for the target parameter that are consistent with a particular value of a non-target parameter is called the *target shadow* of that value for the non-target parameter.

To see some examples of target shadows, consider, again, the network for the map-colouring party game – see Figure1 (a). Remember that, where the parameters in this network are ordered ⟨ Germany, Switzerland, Austria, Italy ⟩, the intent $\Pi^{\mathbf{D},\mathbf{X},\mathbf{C}}$ of the network is $\{ \langle r,g,b,r\rangle, \langle r,b,g,r\rangle, \langle g,r,b,g\rangle, \langle g,b,r,g\rangle, \langle b,r,g,b\rangle, \langle b,g,r,b\rangle \}$. From this, we can see that, if Austria has the value r, the target parameter, Italy, can have either the value g (tuple 4 in the intent) or b (tuple 6 in the intent). Thus, the target shadow for the value r in the domain of Austria is $\{g,b\}$. The target shadows for all the values in the domain of each non-target parameter are shown in the following table:

Germany	$r \rightarrow \{r\}$, $g \rightarrow \{g\}$, $b \rightarrow \{b\}$
Switzerland	$r \rightarrow \{g,b\}$, $g \rightarrow \{r,b\}$, $b \rightarrow \{r,g\}$
Austria	$r \rightarrow \{g,b\}$, $g \rightarrow \{r,b\}$, $b \rightarrow \{r,g\}$

To compute the expected size of the domain of the target parameter that would result if the user were to instantiate some non-target parameter P, we need to know (a) the target shadow for every value v in the domain of P and (b), in most, but not all, cases, the probability[3] that the user will select the value v for P.

[3] We say "in most but not all cases" because we do not need the probability distribution if each value in a domain has the same size of target shadow.

Definition 6, Expected Shadow Size The expected size of the domain of the target parameter that results from instantiating some non-target parameter, P, is called the *expected shadow size* of P. Its value is

$$\sum_{v \in domain(P)} prob(P = v) \times |shad(v)|$$

where, for each value v in the domain of P, $prob(P = v)$ is the probability that the user will instantiate P to this value and $shad(v)$ is the target shadow of v.

From what we have seen so far, it appears that the best question to ask the user at any time is to ask him for the value of the non-target parameter which has the minimum expected shadow size[4]. To determine this "best" question, we need to know the target shadow for every value in the domain of every non-target parameter.

We have seen that we can compute these target shadows if we know the network intent. However, we do not need to know the network intent. In what follows, we will first show that this is true in the case of tree-structured constraint networks. Then we will extend the result to cover graph-structured networks.

6.2 Target-shadow Propagation in Tree-Structured Networks

Theorem 1: If a constraint network is tree-structured, only 2-consistency information is needed to compute the target shadows for all the values of all the non-target parameters in the network.

Proof: In a tree-structured network, any node can be regarded as the root; thus it is possible to treat the target node as the root. Consider a parameter whose node is a child of the root. Using only 2-consistency information, each value in domain of this parameter can be labelled with a set which contains those values of the target parameter that are supported by this value of the child parameter – that is, each value can be labelled with its the target shadow[5]. Now consider a parameter whose node is a child of the parameter for whose values we have just computed the target shadows. Using only 2-consistency information, each value in the domain of this grand-child parameter can be labelled with a set which is the union of the target shadows of those values of the child node

[4] In fact, of course, this notion of "best" question leads to a hill-climbing approach to question generation – there are instances where it will lead to question sequences that are longer than necessary.

[5] Note that the memory cost of storing these labels is only O(qd), where q is the number of parameters and d is the maximum domain size, because each of the subsets of the domain of the target parameter can be represented by a binary number; for example, if the domain of the target parameter is $\{r, g, b\}$, we can represent the subset $\{r, b\}$ of this domain by the binary number 101.

that are supported by this value of the grand-child node; the resultant set contains the values of the target parameter that are supported by this value of the grand-child node – in other words, its target shadow. In this manner, the target shadow for every value in the domain of every non-target parameter can be determined.

Consider, for example, the tree-structured map in Figure 2; suppose that, again, we have a palette of three colours, red, blue and green. Let Poland be the target parameter.

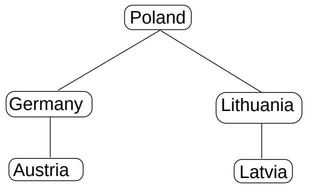

Fig. 2. Tree-structured map

The target shadows for the values in the domain of Germany are $\{r \rightarrow \{g, b\}$, $g \rightarrow \{r, b\}$, $b \rightarrow \{r, g\}\}$. The target shadows for the values in the domain of Austria can be computed from those for Germany; they are $\{r \rightarrow \{r, b\} \cup \{r, g\}$, $g \rightarrow \{g, b\} \cup \{r, g\}$, $b \rightarrow \{g, b\} \cup \{r, b\}\}$; in other words, they are $\{r \rightarrow \{r, g, b\}$, $g \rightarrow \{r, g, b\}$, $b \rightarrow \{r, g, b\}\}$. Similarly, the target shadows for the values in the domain of Lithuania are $\{r \rightarrow \{g, b\}$, $g \rightarrow \{r, b\}$, $b \rightarrow \{r, g\}\}$ and those for the values in the domain of Latvia are $\{r \rightarrow \{r, g, b\}$, $g \rightarrow \{r, g, b\}$, $b \rightarrow \{r, g, b\}\}$.

6.3 Target Shadow Propagation in Graph-Structured Networks

It is probably not surprising that we need only 2-consistency information to compute the target shadows in a tree-structured constraint network. What is more surprising is that, as we will prove below, only the same degree of consistency is needed to compute target shadows in graph-structured constraint networks. The reason that we need only 2-consistency information to compute the target shadows in a graph-structured network is that we can use hypothetical reasoning to temporarily break cycles, so that we can use the approach given above for propagating target shadows in tree-structured networks. To simplify the proof below, we will first define some terms.

Definition 7, Non-target Cycle-Cutset:
In a graph-structured network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$, there is at least one subset of the non-target parameters that are in the tuple $\mathbf{X}$ such that, if these parameters were instantiated, the remaining constraint network would be a tree. Such a set of parameters is called a non-target cycle-cutset.

Consider, for example, the map-colouring network shown in Figure 3 (a), where Lilliput is the target node. One non-target cycle-cutset is {Gilgitia}.

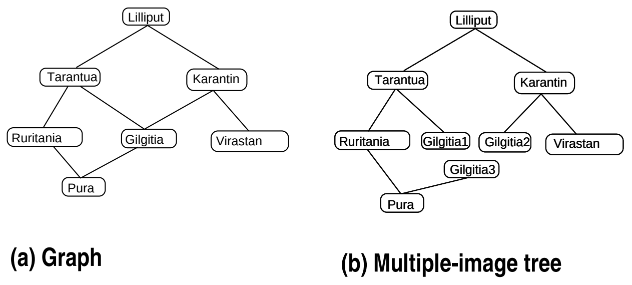

Fig. 3. Lilliput and its neighbours

Definition 8, Multiple-Image-Tree:
Together, a graph-structured network and a non-target cycle-cutset, O, for the network define a tree, called a multiple-image-tree. This tree is produced by making multiple distinct images of each parameter in O, one for each of its neighbours in the original graph-structured network.

The multiple-image-tree defined by the network in Figure 3 (a) and the non-target cycle-cutset {Gilgitia} is shown in Figure 3 (b); in this tree, there are three separate nodes which contain an image of the Gilgitia node in Figure 3 (a).

Definition 9, Multiple-Image Tree Constraint Network:
While a graph-structured network $\langle \mathbf{D}, \mathbf{X}, \mathbf{C} \rangle$ and a non-target cycle-cutset O define a unique multiple-image-tree, they define a set of constraint networks, called multiple-image-tree constraint networks. The set of such networks is defined as follows. Let the multiple-image-tree be M. Let $\mathbf{X}_o$ be that sub-tuple of $\mathbf{X}$ which contains the parameters that are in O. Let D_o be the Cartesian cross-product of the domains of the parameters in $\mathbf{X}_o$. Then there is one multiple-image-tree constraint network corresponding to each tuple t in D_o. A multiple-image-tree constraint network is a constraint network whose parameters and constraints are, respectively, the nodes and edges of M. Except for the multiple-image node parameters, the domain of each parameter is the same as its domain in the original given network. However, the domain of each multiple-image node parameter is a singleton set which contains the value in tuple t for the parameter in the original network of which the node is one of the multiple images.

For example, consider, again, the map-colouring constraint network in Figure 3 (a); suppose that the palette of colours contains the three colours $\{r, g, b\}$.

Then, using the the non-target cycle-cutset {Gilgitia}, we get three multiple-image-tree constraint networks. The topology of each of these networks is as shown in Figure 3 (b). Except for parameters Gilgitia1, Gilgitia2 and Gilgitia3, the domain of each parameter is $\{r, g, b\}$. In one multiple-image-tree constraint network, the three parameters, Gilgitia1, Gilgitia2 and Gilgitia3, all have the same singleton domain $\{r\}$; in another network they all have the same singleton domain $\{g\}$; in the third network they all have the same singleton domain $\{b\}$.

A little extra care must be taken in computing target shadows in multiple-image-tree constraint networks. Remember that all nodes in a group of image nodes that are related by virtue of being images of the same parameter in the original graph represent different facets of that parameter in the original graph. Specifically, propagating along the path between the target node to such an image node produces a set of values in the target node that are supported *via that path* by the single value in the domain of the image node. However, the target shadow must be supported along *all such paths*. Therefore, the target shadow of the single value in the domain of a group of related image nodes is the *intersection* of the sets of values that are supported along the different paths.

For example, consider one of the multiple-image-tree constraint networks based on the multiple-image-tree in Figure 3 (b) – consider the network where images nodes Gilgitia1, Gilgitia2 and Gilgitia3 all have the singleton domain $\{r\}$. The target shadow for the sole value r in this domain is the intersection of three subsets of the domain of the target parameter: the subset of the target domain that is supported by r along the path from Gilgitia1 to the target node; the subset of the target domain that is supported by r along the path from Gilgitia2 to the target node; the subset of the target domain that is supported by r along the path from Gilgitia3 to the target node.

With this infrastructure of ideas in place we can now prove that, to compute target shadows for graph-structured constraint networks, we need only 2-consistency.

Theorem 2: Only 2-consistency information is needed to compute the target shadows for all the values of all the non-target parameters in a graph-structured network.

Proof: Given a graph-structured network, take any non-target cycle-cutset for the network. The network and the non-target cycle-cutset implicitly define a set of multiple-image-tree constraint networks, each of which represents the situation that would result from one possible instantiation of the parameters in the non-target cycle-cutset. The target shadow for a value in the domain of a non-target parameter in the graph-structured network represents the set of values in the domain of the target parameter that are supported by this value in the domain of the non-target parameter. Such a target shadow is the union of the target shadows that the value has in the various multiple-image-tree constraint networks that are produced by instantiating the parameters in a non-target cycle-cutset for the graph-structured network. Theorem 1 shows that we need only 2-consistency information to compute target shadows in tree-structured networks. Therefore we need only 2-consistency information to compute target shadows in the multiple-image-tree constraint networks that result from instantiating the parameters in the non-target cycle-cutset. Therefore we need only 2-consistency information to compute target shadows in graph-structured constraint networks.

Let see this approach at work in the map-colouring network shown in Figure 1 (a), remembering that the palette contains only three colours, $\{r, g, b\}$ and that Italy is the target parameter.

A non-target cycle-cutset is {Austria}, whose corresponding multiple-image-tree is as shown in Figure 1 (b). There are three images of Austria in this tree, one for each of Austria'a neighbours in the original, graph-structured, network.

This multiple-image-tree implicitly defines three multiple-image-tree constraint networks, because there are three values in the domain of Austria. Consider the first of these multiple-image-tree constraint networks, the network in which each of the multiple images of Austria is given the singleton domain $\{r\}$. Applying arc-consistency, the domains of the other three parameters are reduced to $\{g, b\}$. The value r in the domain of Austria1 is compatible with both values that remain in the domain of Italy; that is, r supports the set $\{g, b\}$ along the path from Austria1 to Italy. Each of the two values in the domain of Switzerland, g and b, is compatible with only one value in the domain of Italy; their target shadows are $\{b\}$ and $\{g\}$, respectively. The single value, r, in the domain of Austria2 is compatible with both values in the domain of Switzerland, so, along the path from Austria2 to Italy, the value r supports the union of the target shadows of the two values in the domain of Switzerland – it supports the set $\{g, b\}$. The value g in the domain of Germany is compatible with only value in the domain of Switzerland, b, so it inherits its target shadow, namely $\{g\}$. Similarly, the value b in the domain of Germany inherits the target shadow of the value g in the domain of Switzerland, namely $\{b\}$. The single value, r, in the domain of Austria3 is compatible with both values in the domain of Germany, so, along the path from Austria3 to Italy, the value r supports the union of the target shadows for the two values in the domain of Germany – it supports the set $\{g, b\}$. The target shadow for the value r in the singleton domain that is shared by Austria1, Austria 2 and Austria3 is the intersection of the three sets that are supported along the paths from Austria1 to Italy, from Austria2 to Italy and from Austria3 to Italy; since the three sets are the same, the result is $\{g, b\}$. The above reasoning is summarised in the following table:

Alternative 1:
domain(Austria1) $\leftarrow \{r\}$
domain(Austria2) $\leftarrow \{r\}$
domain(Austria3) $\leftarrow \{r\}$
domain(Italy) $\leftarrow \{g, b\}$
domain(Switzerland) $\leftarrow \{g, b\}$
domain(Germany) $\leftarrow \{g, b\}$
Austria1: $r \rightarrow \{g, b\}$
Switzerland: $g \rightarrow \{b\}$, $b \rightarrow \{g\}$
Austria2: $r \rightarrow \{g, b\}$
Germany: $g \rightarrow \{g\}$, $b \rightarrow \{b\}$
Austria3: $r \rightarrow \{g, b\}$
So, finally:
Switzerland: $g \rightarrow \{b\}$, $b \rightarrow \{g\}$
Austria: $r \rightarrow \{g, b\}$
Germany: $g \rightarrow \{g\}$, $b \rightarrow \{b\}$

The correspondiong reasoning for the cases where Austria is given the singleton domain $\{g\}$ and the singleton domain $\{b\}$ are summarised in the next two tables.

Alternative 2:	Alternative 3:
domain(Austria1) $\leftarrow \{g\}$	domain(Austria1) $\leftarrow \{b\}$
domain(Austria2) $\leftarrow \{g\}$	domain(Austria2) $\leftarrow \{b\}$
domain(Austria3) $\leftarrow \{g\}$	domain(Austria3) $\leftarrow \{b\}$
domain(Italy) $\leftarrow \{r, b\}$	domain(Italy) $\leftarrow \{r, g\}$
domain(Switzerland) $\leftarrow \{r, b\}$	domain(Switzerland) $\leftarrow \{r, g\}$
domain(Germany) $\leftarrow \{r, b\}$	domain(Germany) $\leftarrow \{r, g\}$
Austria1: $g \rightarrow \{r, b\}$	Austria1: $b \rightarrow \{r, g\}$
Switzerland: $r \rightarrow \{b\}$, $b \rightarrow \{r\}$	Switzerland: $r \rightarrow \{g\}$, $g \rightarrow \{r\}$
Austria2: $g \rightarrow \{r, b\}$	Austria2: $b \rightarrow \{r, g\}$
Germany: $r \rightarrow \{r\}$, $b \rightarrow \{b\}$	Germany: $r \rightarrow \{r\}$, $g \rightarrow \{g\}$
Austria3: $g \rightarrow \{r, b\}$	Austria3: $b \rightarrow \{r, g\}$
So, finally:	So, finally:
Switzerland: $r \rightarrow \{b\}$, $b \rightarrow \{r\}$	Switzerland: $r \rightarrow \{g\}$, $g \rightarrow \{r\}$
Austria: $g \rightarrow \{r, b\}$	Austria: $b \rightarrow \{r, g\}$
Germany: $r \rightarrow \{r\}$, $b \rightarrow \{b\}$	Germany: $r \rightarrow \{r\}$, $g \rightarrow \{g\}$

Now unioning the corresponding target shadows from all three possible multiple-image-tree constraint networks, we get the target shadows for the original graph-structured constraint network, as follows:

Switzerland:	$r \rightarrow \{b\} \cup \{g\}$,	$g \rightarrow \{b\} \cup \{r\}$,	$b \rightarrow \{g\} \cup \{r\}$
Austria:	$r \rightarrow \{g, b\}$,	$g \rightarrow \{r, b\}$,	$b \rightarrow \{r, g\}$
Germany:	$r \rightarrow \{r\} \cup \{r\}$,	$g \rightarrow \{g\} \cup \{g\}$,	$b \rightarrow \{b\} \cup \{b\}$

In summary, therefore, the target shadow situation for the original graph-structured constraint network is as shown in the table below. Compare these results with the target shadows that were computed in Section 6.1, where knowledge of the network intent was assumed. The results are the same – as they should be, according to Theorem 2, the theorem that target shadows in graphs can be calculated without knowing the network intent.

Switzerland	$r \rightarrow \{g, b\}$, $g \rightarrow \{r, b\}$, $b \rightarrow \{r, g\}$
Austria	$r \rightarrow \{g, b\}$, $g \rightarrow \{r, b\}$, $b \rightarrow \{r, g\}$
Germany	$r \rightarrow \{r\}$, $g \rightarrow \{g\}$, $b \rightarrow \{b\}$

7 Concluding Remarks

We have introduced the notion of question generation for elicitating data from users of interactive constraint-based expert systems. We have considered three approaches to generating questions, based, respectively, on the prior computation of the network intent, on the prior computation of a sample from the network intent and on the propagation of target shadows without knowing any member of the network intent.

We have proven that target shadow propagation can be done on the basis of information that need be only 2-consistent – we have proven that this is the case in graph-structured constraint networks as well as in tree-structured networks. However, while target shadow propagation in tree-structured networks is relatively inexpensive, there are several factors which may make it an expensive tool when used in graph-structured networks – prior computation of a non-target cycle-cutset for the graph-structured network is required[6] and, more significantly, target shadow propagation must be performed for each of the multiple-image-tree constraint networks which result[7].

However, it must be pointed out that the situation, vis-a-vis graph-structured networks, is not as black as might be deemed from the above remark. Remember that an expert system will be used repeatedly, on many occassions, by many users. It is, therefore, worthwhile spending a lot of effort processing the network before it is released to its user community. Such pre-processing effort has two kinds of cost: the memory space needed and the time required. The fact that only 2-consistent information is required for target shadow propagation in graph-structured networks is good news on the memory front. If a complete questioning strategy (which, in most real-world applications, will require that multiple questions be asked of the user) can be computed before the expert system is released to the user community, the time cost of target shadow propagation may not be a problem either. (It is, of course, always possible to compute the best[8] candidate for the first question, before releasing an expert system to

[6] There is no known polynomial cost algorithm for computing the minimum non-target cycle-cutset. However, non-minimum sets can be computed at reasonable cost.

[7] This, of course, means that the best non-target cycle-cutset is one whose domains have the smallest cross-product of all non-target cycle-cutsets. While no polynomial cost algorithm for computing the optimal non-target cycle-cutset is known, heuristics exist which enable the optimal set to be computed at reasonable cost in many networks. These heuristics are beyond the scope of this paper.

[8] Modulo the remark, made earlier, about hill-climbing.

its users. However, there are some unexplored issues related to the complexity of using target shadow propagation for computing an optimal *series* of questions.)

There are several issues that we have not touched on in this paper, including the following. What is the appropriate probability distribution to use for the possible answers that a user could give when asked for the value of a parameter? Do there exist heuristics for reducing the extent of target shadow propagation needed in trees. Do heuristics exist for reducing the number of multiple-image-tree constraint networks that need be considered for graphs? What additional complexity is introduced by the fact that, in many real-world applications, some questions may be more expensive to answer than others? We have some results in each of these areas, but they are beyond the scope of this paper.

In conclusion: if constraints are to replace declarative if-then rules for knowledge representation in expert systems, question-generation will be an important feature of such systems; we have started to explore this new topic; much remains to be done.

References

1. Bowen J and Bahler D, 1992, "Frames, Quantification, Perspectives and Negotiation in Constraint Networks for Life-Cycle Engineering", *Intn'l Journal of AI in Engineering*, 7, 199-226.
2. Bowen J and Bahler D, 1991, "Conditional Existence of Variables in Generalized Constraint Networks", *Proc. Nat. Conf. of the American Association for Artificial Intelligence.*
3. Bowen J and Bahler D, 1992, "Lexical Imprecision in Fuzzy Constraint Networks", *Proc. Nat. Conf. of the American Association for Artificial Intelligence.*
4. Bowen J, 2002, "Constraint Processing Offers Improved Expressiveness and Inference for Interactive Expert Systems", Technical Report, Department of Computer Science, UCC, Cork, Ireland.
5. Freuder E, Likitvivatanavong C and Wallace R, 2000, "A Case Study in Explanation and Implication", *Proc. CP-2000 Workshop on Analysis and Vizualization of Constraint Programs and Solvers.*
6. Jackson P, 1999, *Introduction to Expert Systems*, 3rd. Edition, Addison Wesley Longman.
7. Junker U, 2001, "Quickxplain: Conflict detection for arbitrary constraint propagation algorithms", *IJCAI-2001 Workshop on Modeling and Solving Problems with Constraints.*
8. Jussien N and Barichard V, 2000, "The PaLM system: explanation-based constraint programming", *CP-2000 Workshop on Techniques for Implementing Constraint Programming Systems.*
9. Rasmus D, 2000, "Knowledge Management Trends: The Role of Knowledge in E-Business", *PCAI Magazine*, 14(4), Special issue on Knowledge Management, Expert Systems and E-Business.
10. Mitchell M, 1997, *Machine Learning*, McGraw-Hill.
11. Shannon C and Weaver W, 1949, *The Mathematical Theory of Communication*, University of Ilinois Press.
12. Tsang, E, 1993, *Foundations of Constraint Satisfaction*, Academic Press.