# Certainty Closure

## A Framework for Reliable Constraint Reasoning with Uncertainty

Neil Yorke-Smith and Carmen Gervet

IC–Parc, Imperial College London, SW7 2AZ, U.K.
{nys,cg6}@icparc.ic.ac.uk

**Abstract** Constraint problems with incomplete or erroneous data are often simplified to tractable deterministic models, or modified using error correction methods, with the aim of seeking a solution. However, this can lead us to solve the wrong problem because of the approximations made. Such an outcome is of little help to a user who expects the right problem to be tackled and reliable information returned. The *certainty closure* framework we present aims to provide the user with reliable insight by: (1) enclosing the uncertainty using what is known for sure about the data, to guarantee that the true problem is contained in the model so described, (2) deriving a closure, a set of possible solutions to the uncertain constraint problem. In this paper we first demonstrate the benefits of reliable constraint reasoning on two different case studies, and then generalise our approaches into a formal framework.

## 1 Motivation

Data uncertainties are inherent in real-world Large Scale Combinatorial Optimisation problems (LSCOs). The uncertainty can be due to the dynamic and unpredictable nature of the commercial world, but also due to the information available to those modelling the problem. In this paper we are concerned with the latter form of uncertainty, which can arise when the data is not fully known or is even erroneous.

Our work is motivated by practical issues we faced when addressing two real-world applications: energy trading [11] and network traffic analysis [12]. In both applications the data information is incomplete or erroneous. In the energy trading problem, the demand and cost profiles had evolved due to market privatisation; thus the existing simulation or stochastic models did not help address the actual problem, since no valid data trends were available. Further, the obsolete data was inconsistent with the constraint model. In the network traffic analysis problem, the overwhelming amount of information forced us to use partial data. Further, due to practical measurement difficulties (e.g. unrecorded packet loss), the data acquired in the problem was frequently erroneous.

When addressing the energy trading problem, we understood that the customer did not need nor want a solution to an approximation of his problem, but rather a guarantee that the model built was reliable, and that from it informed decisions could be made. Informally, a model and solution are *reliable* with respect to the state of the world if they accurately reflect the true problem and its possible solutions. The uncertain data is represented using what is known for sure about it, without any approximation, and no potential solutions are excluded. Our goal was to build such a model and to provide

Proceedings of CP'03, Kinsale, Ireland, September 2003. LNCS 2833 
Some minor oversights have been corrected in this version.

effective insight into the set of possible solutions. It became clear that further research was necessary to extend the potential of constraint programming to meet this goal.

Indeed, in the face of data uncertainty, existing CP approaches come from quite a different perspective. Models and methods have been proposed to tackle incomplete and dynamic data by seeking robust solutions to the problem, i.e. solutions that hold under the maximum number of possible states of the world [9]; or by reasoning upon probabilistic data distributions [5, 21]. These approaches are suited for applications where data trends are available and realistic, or where robustness is sought after: for example, dynamic scheduling problems. However, they are less suited for the reliable reasoning our motivational problems demand.

In this paper we focus on uncertainty due to incomplete or erroneous data. With the aim of providing the user with reliable insight, our threefold objective is:

1. To create a reliable model of the LSCO, i.e. remove approximations about the data and enclose the true problem in the model.
2. To compute the *full closure*, i.e. the set of all possible solutions to the model; or a subset of it as the user specifies. By possible, we mean a solution that holds for at least one realisation of the data.
3. To propose two resolution forms to solve uncertain CSPs, and give instances over specific constraint classes.

In Sect. 2, we first show on two different uncertain LSCOs how we can attain such an objective in practice. We then generalise the case studies as instances of the certainty closure framework, in Sect. 3 and 4. The framework, based on the CSP formalism, allows us to reason about uncertain problems by modelling explicitly what is known about the uncertain data in terms of an *uncertain constraint satisfaction problem* (UCSP). We define a UCSP and its full closure. Then, we formally describe two resolution forms that can derive closures in a practical way, and we give examples of the forms for various classes of UCSPs. In Sect. 5 we review and contrast with related work, and finally we conclude in Sect. 6.

## 2 Case Studies

Despite the presence of incompleteness or errors, we assume that those modelling the problem do have some definite knowledge about the data. We use this knowledge to enclose the uncertainty within an interval or a set of values. We assume further that knowledge about the data is only refined (e.g. to a subset of the initial possible values). Since the closure excludes no possible solution, we guarantee that the true solution lies in the closure whatever the state of the world. However, the closure might comprise of a large set of solutions. A key issue therefore is how informative it is in practice, and how complex it is to derive and represent.

In this section we investigate the practical benefits of reliable constraint reasoning by considering two quite different case studies which address real-world problems. The first case study is the network traffic analysis problem introduced earlier; the second is a planning problem in the aerospace domain. In both, an uncertain CSP is presented, together with a solution operator used to derive the closures. Both problems were modelled and solved using the ECL$^{\text{i}}$PS$^{\text{e}}$ CLP platform [7].

### 2.1 Network Traffic Analysis

The network traffic analysis problem poses a diagnosis question. The problem is: for a known network with incomplete and possibly erroneous traffic measurements at routers, determine guaranteed bounds for each end-to-end traffic flow. The true problem must be satisfiable, because the network exists and is executing. The complexity lies in adequately handling the data, in guaranteeing that the right problem is being solved, and in seeking tight bounds. We illustrate our approach on an example fragment of a network.

*Initial model.* Consider the fragment of a network shown in Fig. 1. Four nodes, corresponding to routers and designated A–D, are shown, together with the bidirectional traffic flow on each link. Each router makes decisions on how to direct traffic it receives, based on a routing algorithm.

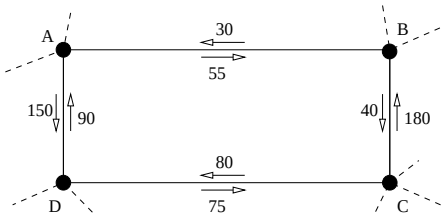


**Figure 1.** Traffic flow in a network fragment

The network was initially modelled as a classical CSP, as follows. The variables correspond to the traffic flow between any two end-points, and their domains are the non-negative reals: $V_{ij} \in \mathbb{R}^+$ is the volume of traffic entering the network at node $i$ and leaving it at node $j$. The constraints form a linear flow model. They state that the volume of traffic through each link in each direction is the sum of the traffic entering the link in that direction. There is also an upper bound (here, 64) on the flows that use only a single link, such as $V_{AB}$ or $V_{AD}$.

The traffic volume data is collected by reading router tables at each node over a given time interval (e.g. 20 minutes). As a result, the data information obtained is erroneous. On the link A→D, for example, the flow might measure as 135 at A and as 160 at D, whereas the true value, equal at both nodes, is presumably somewhere in between. A common approach therefore is to use the median value.

Another source of uncertainty comes from traffic routing. In 90% of cases, the traffic is known to be split equally when two paths are of equal cost (from the perspective of the routing algorithm). In our running example we consider this to be the case for flows between any two non-consecutive nodes, e.g. from A to C. To simplify the model, it was first assumed that the traffic is split equally in all such cases. For example, on the link A→D the traffic flow constraint generated is:

$$\text{A}\rightarrow\text{D} \qquad V_{AD} + 0.5V_{AC} + 0.5V_{BD} = 150 \tag{1}$$

The generated CSP model was unsatisfiable. Thus a data correction procedure (minimising deviation on the link traffic volumes) was employed in order to reach a satisfiable, deterministic model. The resulting model was solved using the most suitable techniques; in this case standard Linear Programming (LP). Maximum and minimum bounds were derived for each flow variable $V_i$ by solving two linear programs, with objectives $\max V_i$ and $\min V_i$ respectively.

*Certainty closure approach.* The first approach amalgamated data uncertainty and constraint satisfiability issues. Our aim was to investigate whether the approach was leading to the true problem, and hence whether the bounds obtained were reliable. We therefore removed the approximations made: we represented the uncertain flow measurements explicitly, and we modelled the splitting of traffic, actually known to be anywhere between 30–70%. We modelled the problem as an uncertain CSP. For example, on the link A→D we have the following constraint (in which $[\underline{a}, \overline{a}]$ denotes an interval):

$$\mathtt{A{\to}D} \qquad V_{AD} + [0.3, 0.7]V_{AC} + [0.3, 0.7]V_{BD} = [135, 160] \tag{2}$$

Uncertain coefficients thus represent (i) percentage of traffic going through each route, and (ii) measured flow volume on each link.

*Implementation and solving.* We modelled and solved the problem using the `ic` interval constraint library from the ECL$^i$PS$^e$ platform [7]. The library provides a *bounded real* datatype: an interval representing an unknown real value; and interval constraint solvers over numerical constraint systems of arbitrary combinations of integer and bounded real variables. Using `ic`, we can model constraints such as (2) simply by: `V_AD + 0.3__0.7 V_AC + 0.3__0.7 V_BD = 135__160`.

To calculate the closure, we first tried to solve the uncertain CSP as is, using interval and quantified CSP methods (e.g. [4, 15]). The methods proved costly or unsuited to producing tight bounds when compared with the presented method. We then considered a transformation of the uncertain model to an equivalent certain CSP, in order to benefit from existing resolution methods for standard CSPs. We defined a transform operator and proved its correctness using methods from interval linear programming [8]. A full description of the transformation can be found in [22].

Hereafter, we illustrate the transform in the three variable case for simplicity. Let $V_1$, $V_2$ and $V_3$ be variables with domains in $\mathbb{R}^+$. Then the constraints have the form $c$: $\mathbf{a_1}V_1 + \mathbf{a_2}V_2 + \mathbf{a_3}V_3 \leq \mathbf{a_4}$, where $\mathbf{a_i} = [\underline{\mathbf{a_i}}, \overline{\mathbf{a_i}}]$ are real, closed intervals. Each uncertain flow constraint $c$ can be transformed into a certain constraint $\tau(c)$ as follows:

$$\tau(c) = \begin{cases} \underline{\mathbf{a_1}}V_1 + \underline{\mathbf{a_2}}V_2 + \underline{\mathbf{a_3}}V_3 \leq \overline{\mathbf{a_4}} & \text{if } \underline{\mathbf{a_3}} \geq 0 \\ \underline{\mathbf{a_1}}V_1 + \underline{\mathbf{a_2}}V_2 + \overline{\mathbf{a_3}}V_3 \leq \underline{\mathbf{a_4}} & \text{if } 0 \in \mathbf{a_3} \\ \overline{\mathbf{a_1}}V_1 + \overline{\mathbf{a_2}}V_2 + \overline{\mathbf{a_3}}V_3 \leq \underline{\mathbf{a_4}} & \text{if } \overline{\mathbf{a_3}} < 0 \end{cases} \tag{3}$$

By convexity, it suffices to operate on the bounds of the data values. The transformation operates only on linear inequalities. Thus as a prelude to the transform, each equality constraint is replaced by a pair of inequalities; the decision variables remain unchanged. For example, the constraint (2) above is transformed to:

$$(V_{AD} + 0.7V_{AC} + 0.7V_{BD} \geq 135) \wedge (V_{AD} + 0.3V_{AC} + 0.3V_{BD} \leq 160) \tag{4}$$

The resulting model, like the initial model, describes a standard LP problem. Thus we can solve it using the same method, but now we obtain guaranteed bounds. For the example above, we obtain the following intervals, which represent the projection of the closure onto the variable domains: $V_{AC} \in [0, 150]$, $V_{AD} \in [30, 64]$, $V_{BD} \in [0, 133]$, $V_{BA} \in [0, 20]$, $V_{DC} \in [0, 40]$, $V_{DB} \in [32, 200]$, $V_{CA} \in [0, 133]$, $V_{CB} \in [17, 64]$ (omitting the four single-link flows in the clockwise direction).

*Outcome.* Compared to the initial approach to the problem, the certainty closure has lead to more reliable quantitative results and to improved understanding of the relationship between network topology and traffic flow. For instance, if the closure is empty, we can infer that the problem is unsatisfiable due to the constraint network and not due to the data: since no approximation is considered but the data enclosure. Treating the data adequately reveals the true reasons for unsatisfiability.

### 2.2 Planning for Aerospace Equipment

The second case study arises in the aerospace domain, where future systems will be expected to achieve more complex missions with less human intervention [20]. The system must continuously operate in a changing, ill-known environment; command complicated equipment; and simultaneously fulfill mission goals and satisfy system requirements, e.g. timeliness and safety. On the whole, however, existing aerospace component design does not integrate uncertainty into autonomous planning functions [20].

Fig. 2 shows an example automaton, representing the behaviour of a thruster subsystem (a satellite 'engine'). The goal is to achieve a certain thrust performance in a given time window, while maintaining the internal temperature within given limits. Temperature, however, evolves in an ill-known way according to the heating (thrust) and cooling states. We model the thruster as a non-deterministic finite state automaton (FSA) where temperatures are attached to transitions and states of the automaton. The data uncertainty concerns the temperature increments, which are subject to both measurement errors and incomplete information. We associate with the automaton the following constraint model.

*Model.* Our constraint model is again an uncertain CSP. Here it is simplified for the sake of clarity; a full description is found in [23]. The variables are all finite domain integers. A path in the automaton is specified by transition and timing variables. The transitions $S_i \in [0, N]$ are a sequence of states, where $S_0$ is the initial state. The timings $T_i \in \mathbb{N}$ specify the duration spent in each state. We write $i \in [0, H]$ to index the states on the path, and $j \in [0, N]$ for the value of the $j^{\text{th}}$ state: i.e. $j = S_i$. Associated with the path are edge boolean variables $E_j \in \{0, 1\}$, which specify whether an edge is ever taken on the path. Finally, the temperature is modelled with variables $\Theta_i \in [0, 100]$, and the uncertain temperature increments with coefficients $\Delta_j \in [-100, 100]$.

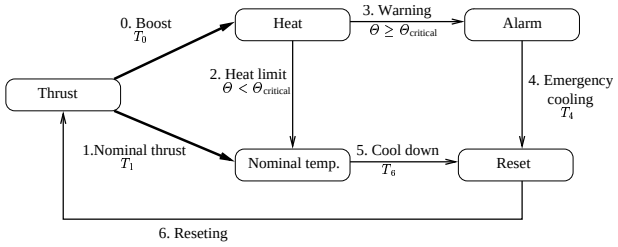


**Figure 2.** Discrete automaton representing the behaviour of a thruster sub-system. The temperature increments are uncertain in the two thrusting states and in the two cooling states.

The main constraints are of three types. The first two types are certain. Constraint (5) states flow conservation on the edge variables. For example, for state 5: $E_1 + E_2 = E_6$. Constraint (6) is an example of a constraint modelling a contingent event. Here it is the event that the temperature exceeds the warning threshold, whereupon if in state 0 we must move to state 3. The final type of constraint describes the evolution of the temperature, and is thus uncertain. For this automaton, (7) models a linear recursion.

$$\sum_{k \in e^+(j)} E_k = \sum_{k \in e^-(j)} E_k \qquad \forall j \in [0, N] \tag{5}$$

$$\Theta_j \geq \Theta_{\text{critical}} \implies S_{i+1} = j + 3 \qquad \forall j \text{ corresponding to state } 0 \tag{6}$$

$$\Theta_{i+1} = E_j \times (\Theta_i + T_i \Delta_j) \qquad \forall i \in [0, H] \tag{7}$$

*Solving.* The need to guarantee safe behaviour even in the worst case means that seeking a single plan, however optimality is measured, is inadequate for our problem.[1] Therefore, we chose to compute a *covering set closure* of feasible plans: a set containing at least one plan for every feasible realisation. Ideally this set should be of minimal size, because a smaller set in general is more compact to represent.

Given the heterogeneous nature of the constraints, we found no natural transformation from the UCSP to an equivalent CSP. We describe in [23] different enumeration methods to compute a covering set closure. For space reasons, we will outline the most efficient: enumeration using a decomposition method. The idea is to first derive a feasible plan for a given realisation, and then decompose the remainder of the UCSP by removing from future consideration all realisations covered by this plan. This decomposition method is based on the conditional decision method for mixed CSPs with full observability [9]. It uses a technique called *sub-domain subproblem extraction* [10]. Given a feasible plan (a solution), the extraction technique decomposes the set of realisations into a disjunction of two sets: one containing precisely the realisations covered by the plan. The decomposition approach is tractable because the data is discrete, and each constraint contains at most one uncertain coefficient.

To give the intuition of the approach, consider the UCSP with just one uncertain constraint: $Y = X + T \cdot \Delta$, where variables $X, Y, T \in \{0, 100\}$ and $\Delta \in \{-50, 50\}$ is an uncertain coefficient. We can find a covering set closure as follows. For each possible value $\delta$ of $\Delta$, form the *realised CSP* $P_\delta$, and solve it to find a consistent tuple for $(X, Y, T)$. For example, if $\delta = 20$, the realised CSP is $Y = X + T \cdot 20$, and a consistent tuple is $(10, 70, 3)$. A naive approach would require us to: (i) generate each realised CSP $P_\delta$, (ii) seek a solution to each, and (iii) take the union of all the solutions to derive a covering set closure. The use of decomposition allows us to consider a smaller number of realised CSPs, by eliminating realisations already covered as we progress.

*Outcome.* Contrasted to some current practice in aerospace design, the certainty closure approach enables a new expressiveness in planning and control of low-level components, by allowing us to consider the uncertainty. As a result, aerospace component behaviour can be adapted in a more reliable way to its environment, and so the behaviour and performance guarantees sought by aerospace designers can be reinforced.

[1] Unless it holds under all realisations, or unless we rely on online plan repair.

## 3 Uncertain CSP and its Closures

The two case studies indicate the practical value and potential benefits of reliable reasoning. We now define the certainty closure framework to provide a comprehensive and generic approach to reliable reasoning under uncertain data. After some preliminaries, we define the concepts of an uncertain CSP and its closures.

### 3.1 Preliminaries

We consider the CSP formalism since it has the generality we desire to model LSCO problems. Recall that a classical CSP is a tuple $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{V}$ is a finite set of variables, $\mathcal{D}$ is the set of corresponding domains, and $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a finite set of constraints. A solution is a complete consistent value assignment. We represent a CSP by a conjunction of its constraints $\bigwedge_i c_i$ (as opposed to the set of its allowed tuples). Similarly, we represent a solution or set of solutions to a CSP by a conjunction of constraints. These constraints should be from a simple class, e.g. unary equalities.

Recall that, with respect to a given computation domain, a constraint domain specifies the syntax and semantics of permitted constraints. It specifies the constants, functions and constraint relations. The constants we will refer to as *coefficients*. A coefficient may be *certain* (its value is known) or *uncertain* (value not known). In a classical CSP, all the coefficients are certain. We assume the user has some knowledge of the possible values for the coefficients, or bounds on their range. Call the set of possible values of a coefficient $\lambda_i$ its *uncertainty set*, denoted $U_i$. We say an *uncertain constraint* is one in which some coefficients are uncertain. Note the coefficients in an uncertain constraint are still constants; merely their exact values are unknown. For example, if the coefficient $\lambda_1$ has uncertainty set $U_1 = \{2, 3, 4\}$, the constraint $X > \lambda_1$ is uncertain.

Regarding the data, following Ben-Tal and Nemirovski [3], a *data realisation* is a fixing of the coefficients to values; in related literature, the terms *possible world* and *context space* are also used. The notation $\widehat{\cdot}$ will denote certainty. For an uncertain CSP $P$, we will say that any certain CSP $\widehat{P}$, corresponding to a data realisation of the coefficients of $P$, is a *realised* CSP, and write $\widehat{P} \in P$. Each uncertain constraint is made certain by a realisation, thus $\widehat{P} = \langle \mathcal{V}, \mathcal{D}, \widehat{\mathcal{C}} \rangle$, where $\widehat{\mathcal{C}} \in \mathcal{C}$ denotes a set of *realised* constraints. In the same way, a realisation of a constraint $c$ will be denoted $\hat{c} \in c$. It is worth noting that an uncertain constraint can have many realisations, as many as the size of the Cartesian product of the uncertainty sets involved.

### 3.2 Uncertain Constraint Satisfaction Problem

A UCSP is a simple extension to a classical CSP with an explicit description of the data:

**Definition 1 (UCSP).** *An* uncertain constraint satisfaction problem $\langle \mathcal{V}, \mathcal{D}, \Lambda, \mathcal{U}, \mathcal{C} \rangle$ *is a classical CSP* $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ *in which some of the constraints may be uncertain. The finite set of coefficients is denoted by* $\Lambda$*, and the set of corresponding uncertainty sets by* $\mathcal{U}$.

In this paper we assume the coefficients are either all discrete or all continuous. We also assume the coefficients are independent. The uncertainty set $\mathcal{U}$ is then the

Cartesian product of the uncertainty sets of the coefficients, i.e. the Cartesian product of their possible values. Other than this, there is no requirement as to the nature of the data or the representation of $\mathcal{U}$.

*Example 1.* The constraints for the network traffic analysis problem form a UCSP with the uncertainty specified by real intervals. Similarly, the constraints for the aerospace planning problem form a UCSP with the uncertainty specified by finite sets. □

For a certain CSP $\widehat{P}$, recall that its complete solution set (or space) is the set of all solutions to $\widehat{P}$, which we will denote $\mathcal{S}_{\widehat{P}}$. The extension of this concept to UCSPs will play a key role. In line with our aim of reliable reasoning, we define the complete solution set $\mathcal{S}_P$ of a UCSP $P$ as the set of all solutions supported by *at least one* realisation.

### 3.3 Closures of a UCSP

A *closure* is the resolution to a UCSP model. Depending on his application, the user might be interested in different types of closures. We distinguish several types of closures by the properties they hold. For example, a *covering set* is a set of solutions that contains at least one solution (not necessarily all solutions) for each realisation. A *most robust solution* is a solution that is supported by the greatest number of realisations. The *full* closure of a UCSP $P$ is the set of all solutions such that each is supported by at least one realisation, i.e. the complete solution space $\mathcal{S}_P$. A closure in general is a subset of the complete solution space:

**Definition 2 (Closure).** *Let $P$ be a UCSP $\langle \mathcal{V}, \mathcal{D}, \Lambda, \mathcal{U}, \mathcal{C} \rangle$. We say that a subset of the complete solution space $\mathcal{S}_P$ is a* closure *for $P$. If the closure is the entire solution space, we say it is the* full closure*, denoted* $\mathrm{Cl}(P)$.

Let $s$ denote a solution satisfying a realised CSP $\langle \mathcal{V}, \mathcal{D}, \widehat{\mathcal{C}} \rangle$ and $\langle\, s \,\rangle$ be a conjunction of constraints describing $s$. Then we can write the full closure of $P$ as the constraint:

$$\mathrm{Cl}(P) = \bigvee_{\widehat{\mathcal{C}} \in \mathcal{C}} \; \bigvee_{s \text{ satisfies } \widehat{\mathcal{C}}} \langle\, s \,\rangle \qquad (\bigstar)$$

*Example 2.* Let $X$ and $Y$ be temperature variables with integer domains $[1, 5]$ over the following constraints: $c_1 : X > \lambda_1$, $c_2 : |X - Y| = \lambda_2$, and $c_3 : Y - \lambda_1 \neq 1$. Let $\lambda_1$ and $\lambda_2$, which represent temperature increments, have uncertainty sets $U_1 = \{2, 3, 4\}$ and $U_2 = 2$ respectively. The full closure is $(X, Y) \in \{(3, 1), (3, 5), (4, 2), (5, 3)\}$; a covering set closure of minimal size is $(X, Y) \in \{(3, 1), (5, 3)\}$, since this solution set covers all three realisations. □

The different closures form a lattice under inclusion. A simple hierarchy of closures is shown in Fig. 3. The full closure is the top, and the empty closure (the empty set) the bottom. The observation that the different closures fall into a lattice hierarchy allows us to study how they relate to one another.

For example, consider the UCSP depicted in Fig. 4. The full closure at the top of the lattice hierarchy is the set $\top = \{a, b, c, d, e\}$. The most robust solution is $b$; and there are two covering sets of minimal cardinality, $\{a, b\}$ and $\{b, c\}$.

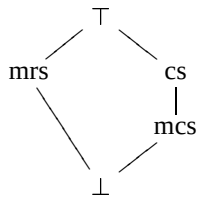


**Figure 3.** Simple hierarchy of closures. At the top of the lattice is the full closure, at the bottom the empty closure. Illustrated in the middle are a covering set (cs), a minimal covering set (mcs), and the most robust solution (mrs).

| | |
|---|---|
| $\widehat{P}_1$ | $a\ b\quad d$ |
| $\widehat{P}_2$ | $a\quad c$ |
| $\widehat{P}_3$ | $\quad b\qquad e$ |
| $\widehat{P}_4$ | $\quad b\ c$ |

**Figure 4.** Realised CSPs (denoted $\widehat{P}_1$–$\widehat{P}_4$) and their feasible solutions (denoted $a$–$e$).

## 4 Resolution Forms

A UCSP adds expressive power and flexibility to a CSP but, depending on the closure demanded, is harder to solve. Indeed, the complexity of deriving a closure from an UCSP is in the worst case that of finding at least one solution to a realised CSP, times the size of the Cartesian product of all the uncertainty sets.[2] Thus we cannot expect to derive the full closure, for example, by a generic practical approach, unless we restrict to a modest number of uncertain coefficients or accept approximation. Rather, we will look at two resolution forms — two possibilities to move from a UCSP to a closure — and we will instantiate the resolution forms to specific constraint domains. Where possible, we would like to exploit existing methods for CSP solving. Each case study in Sect. 2 is an instances of one of the resolution forms.

The first resolution form is to transform the UCSP: we find and then solve an equivalent certain CSP. The set of all its solutions is the closure to the UCSP. The second form, enumeration, applies when there are a finite number of realisations. Each realisation gives rise to a certain CSP, which we solve, and the closure is then the union of all the solutions to the satisfiable CSPs. We show how this approach relates to methods for handling disjunctions.

### 4.1 Comparing Uncertain and Certain Constraints

For both resolution forms, we reason about uncertain constraints in terms of certain constraints. This section describes the algebraic structure over which we perform the reasoning. The central idea is that uncertain constraints form a lattice:

**Proposition 3 (Constraint lattice).** *Let $\mathbb{C}$ be the set of all constraints, certain and uncertain, that can arise with respect to a computation domain. With conjunction and disjunction as meet and join, $\mathbb{C}$ is a distributive lattice. With logical implication of constraints, $\mathbb{C}$ has a natural partial order. Let $\widehat{\mathbb{C}} \subset \mathbb{C}$ be the subset of certain constraints; then $\widehat{\mathbb{C}}$ forms a sublattice of $\mathbb{C}$.* □

[2] UCSP solving is a highly specialised case of quantifier elimination (i.e. computing an equivalent, quantifier-free version of a first-order formula), which is known as an exceedingly difficult problem in the general case [15].

The conjunction, disjunction and implication operations, defined as usual for certain constraints (i.e. on $\widehat{\mathbb{C}}$), need to be extended to $\mathbb{C}$. For space reasons, we will only present the extension of the implication operation. In line with our aim for reliable reasoning, we say that an assignment satisfies an uncertain constraint if it satisfies *at least one* realisation. Hence implication is defined by: if every assignment that holds under some realisation of $c_1$ also holds under some realisation of $c_2$ (not necessarily the same), then $c_1$ implies $c_2$.

Recall that any UCSP can be represented by the conjunction of its constraints. Prop. 3 tells us this conjunction is an element of a suitable constraint lattice. Moreover, since solutions to a CSP can also be represented by a conjunction of constraints, every closure of a UCSP can likewise be described as an element of $\mathbb{C}$. Depending on the constraint class, this element may be a disjunction. For example, the full closure $\mathrm{Cl}(P)$ is described by the constraint ($\bigstar$). A well-chosen representation of a closure is crucial in any practical application.

As a consequence, firstly and importantly, mappings from $\mathbb{C}$ to itself can encapsulate the solving process. Reliable solutions in the certainty closure framework will be guaranteed by properties of the mappings. Secondly, knowledge refinement can be seen in terms of a *subsumed-by* order on solutions. Should we learn more about the data, the revised closure will be subsumed by the old. We say that a constraint $c_2$ *subsumes* a constraint $c_1$ if the complete solution set of $c_2$ contains that of $c_1$:

**Definition 4 (Order).** *Recall the subsumed-by partial order on $\widehat{\mathbb{C}}$, defined by Tsang [17].*[3] *Let $\preceq$ be an extension of the order to $\mathbb{C}$ such that $c_2 \in \mathbb{C}$* subsumes *$c_1 \in \mathbb{C}$, written $c_1 \preceq c_2$, if and only if $\mathrm{Cl}(c_2)$ subsumes $\mathrm{Cl}(c_1)$.*

This partial order is well-defined because $\mathrm{Cl}(c)$ is always a certain constraint. It is compatible with and extends the natural order that arises from constraint implication.

Along with the lattice $\mathbb{C}$, we need a notion of equivalence to be able to compare the solution sets of UCSPs (which we seek) and CSPs (which we use to describe a closure). The subsumed-by relation of Def. 4 provides this notion.

*Example 3.* Consider constraints $c_1$: $X > \{2,3,4\}$ and $\hat{c}_1$: $X > 2$. $c_1$ and $\hat{c}_1$ are equivalent under $\preceq$: they describe the same set of possible values for $X$. Note that $\hat{c}_1$ is precisely the full closure of $c_1$. □

### 4.2 Solution Operators

Recall that a classical CSP is solved by propagation and search: one calculates the fixed-point of some local consistency operators and (if necessary) explores the search space. Since we wish to consider both discrete and continuous CSPs, we encapsulate fully solving a CSP by a *solution operator*. The specific methods used to solve CSPs are not relevant: the essential point is to guarantee that the inferences are correct.

We define a solution operator as a map from $\widehat{\mathbb{C}}$ to itself that provides the conjunction of a set of solutions to a CSP $\widehat{P}$. The conjunction may be empty, indeed must be if $\widehat{P}$ is inconsistent. A *complete* solution operator is one that yields the set of all solutions.

[3] Intuitively, $\hat{c}_1 \in \widehat{\mathbb{C}}$ is *subsumed-by* $\hat{c}_2 \in \widehat{\mathbb{C}}$ if for every satisfying tuple $t_1$ to $\hat{c}_1$ there exists a satisfying tuple $t_2$ to $\hat{c}_2$ such that $t_1$ is a projection of $t_2$.

**Definition 5 (Solution operator).** *Let $\widehat{P}$ be a certain CSP. Let $\phi : \widehat{\mathbb{C}} \to \widehat{\mathbb{C}}$ be a map such that $\phi(\mathcal{C})$ describes a set of solutions to the CSP. If $\phi$ obeys:*

1. *Contraction* — *The final state is a subset of the initial state: $\phi(\mathcal{C}) \preceq \mathcal{C}$*
2. *Monotone* — *Subsumed-by order respected: $\mathcal{C}_1 \preceq \mathcal{C}_2 \Longrightarrow \phi(\mathcal{C}_1) \preceq \phi(\mathcal{C}_2)$*
3. *Idempotence* — *Further application of $\phi$ yields no further solutions*

*Then we say that $\phi$ is a* solution operator.[4] *If further $\phi(\mathcal{C})$ describes the set of all solutions to $\widehat{P}$, we say $\phi$ is* complete *for $\widehat{P}$.*

*Example 4.* Consider a solution operator for finite domain CSPs. Let $\phi_1$ be the map that corresponds to naive backtrack search. If we insist that the whole search tree be explored, then $\phi_1$ will give all solutions; this makes it complete. □

Similarly, a solution operator for uncertain CSPs is a map that yields a closure when given a UCSP $P$. A *complete* uncertain solution operator is one that yields the full closure $\mathrm{Cl}(P)$. Formally, it is defined as a mapping from $\mathbb{C}$ to $\widehat{\mathbb{C}}$:

**Definition 6 (Uncertain solution operator).** *Let $P$ be a UCSP. An* uncertain solution operator *is a map $\rho : \mathbb{C} \to \widehat{\mathbb{C}}$ such that $\rho(\mathcal{C}) \subseteq \mathrm{Cl}(P)$. An uncertain solution operator $\rho$ must obey the contraction, monotone and idempotence properties. If further $\rho(\mathcal{C}) = \mathrm{Cl}(P)$, we say $\rho$ is* complete *for $P$.*

This definition is stated in a simple way because it builds on the results of Sect. 4.1; the concept of a solution operator thus transfers naturally to UCSPs. Transformation to an equivalent certain CSP is one way to build an uncertain solution operator; enumeration is another. In the following sections we describe both resolution forms.

### 4.3 Solving an Equivalent CSP

The issues related to this approach are twofold: finding a CSP equivalent to the UCSP, i.e. one whose set of all solutions coincides with the sought closure to the original problem; and then solving it efficiently. We achieve the first part by seeking a transformation operator from UCSP to CSP which satisfies certain properties; for the second part we can use any existing technique appropriate to the computation domain at hand.

Unless she specifies otherwise, by default we suppose the user desires the full closure, since it excludes no possible solution. For reasons of space, we now concentrate the discussion to the case. The equivalent CSP is found using a CET:

**Definition 7 (Certain Equivalence Transform).** *A map $\tau : \mathbb{C} \to \widehat{\mathbb{C}}$ is a* certain equivalence transform *if it: (1) preserves certainty, i.e. $\tau(\hat{c}) = \hat{c} \quad \forall \hat{c} \in \widehat{\mathbb{C}}$; (2) is a closure operator, i.e. is increasing, monotone and idempotent; and (3) distributes over meet.*

Preservation of certainty and the closure properties ensure that a certain constraint system is found. The third property governs the behaviour on conjunctions of constraints. Together, the properties which characterise a CET allow us to guarantee correctness of the uncertain solution operator. In other words, they ensure that the complete

[4] Note the equivalents in other theoretical frameworks, e.g. Apt's reduction functions [1].

solution set of the equivalent CSP contains the full closure to the original problem. Further, if the solution sets are equivalent, then $\tau$ is a *tight* CET. If $\tau$ is a non-tight CET, we obtain only an outer approximation to the closure. There is often value in such an approximation, if suitably close, since correctness is retained.

Prop. 8 sums up the result: an uncertain solution operator $\rho$ can be defined as a composition of a tight CET $\tau$ and a solution operator $\phi$. The proof is omitted.

**Proposition 8 (Closure by transformation).** *Let $P$ be a UCSP. If $\tau$ is a tight CET and $\phi$ is a solution operator complete for $\tau(\mathcal{C})$, then $\rho = \phi \circ \tau$ is an uncertain solution operator, complete for $P$.* □

*Example 5.* Recall how resolution by transformation was applied to the network traffic analysis problem. In Sect. 2.1 we gave a simplified form (3) of the CET $\tau$ used. It can be shown to be tight and to have the properties of Def. 7. Our use of LP as the solution operator $\phi$ likewise obeys the properties expected in Def. 5. Hence by Proposition 8, the certainty closure framework derives an enclosure guaranteed to be reliable. □

### 4.4 Enumerating Realised CSPs

Depending on the constraint class, it might not always be possible to find a CET. A second means to derive closures is by enumeration. As an essentially exhaustive technique, enumeration requires operationally there be only finitely-many, $M < \infty$, realisations of the data. We generate and solve each realised CSP, forming the closure from the solutions to all the good realisations. Contrasted with transformation, the cost of enumerating and solving $M$ possibly similar CSPs grows with $M$, which can be exponential in the size of the UCSP. This said, in a given computation domain, it may be possible to exploit knowledge of the structure of the realised problems (e.g. [16, Chapter 6]).

*Example 6.* Consider a UCSP with three variables: $X, Y, T \in \mathbb{N}$, and constraints of the form: $Y = X + T \cdot \Delta$, where $\Delta \in \mathbb{Z}$ is an uncertain coefficient. In Sect. 2.2 we showed how to derive by enumeration a covering set closure for this class of UCSPs. □

### 4.5 Approximation

In practice it might be desirable to approximate the closure, either because the user seeks a different representation, or because the complexity of deriving or representing the closure to the UCSP is too high. Approximation must not impair correctness, i.e. omit elements of a closure (since it would no longer be a reliable resolution of a UCSP model), but may forgo tightness, i.e. include non-elements of the closure. We must balance complexity and closeness of the approximation.

For example, in the network traffic analysis problem, since the user's interest is to determine safe operating capacities, he will be satisfied by reliable intervals for the traffic flow variables. Thus we can give a tight outer box approximation. This means we need not calculate a general convex polytope, which could be computationally expensive, but a simpler shape, an axis-parallel hyperbox.

### 4.6 Instances of Resolution Forms

The choice of the resolution form is driven by the constraint class, variable domains and nature of the uncertain data. We give some instances of the resolution forms for four classes of UCSPs $\langle \mathcal{V}, \mathcal{D}, \Lambda, \mathcal{U}, \mathcal{C} \rangle$. We sketch how existing solution methods can be leveraged to provide practical algorithms for deriving closures in each class.

*Transformation for UCSPs with* $\mathcal{D} = \mathbb{R}^m$, $\mathcal{U} = \mathbb{R}^\ell$ *and* $\mathcal{C} = \{$*n-ary linear, arithmetic constraints*$\}$. When the variable domains are nonnegative, i.e. $\mathcal{D} = (\mathbb{R}^+)^m$, the UCSP $P$ is an instance of a *positive orthant interval linear system*. The CET we saw in Sect. 2.1 transforms $P$ into an equivalent linear problem, solvable in polynomial time by linear programming.

A generalisation in operational research is to semi-definite problems[5] with uncertain data coefficients. In particular, for the class of UCSPs with ellipsoidal data and linear constraints, the CET transforms the UCSP to an equivalent conic quadratic problem, solvable in polynomial time by interior point methods [3].

*Enumeration for UCSPs with* $\mathcal{D} = \mathbb{R}^m$, $\mathcal{U} = \mathbb{R}^\ell$ *and* $\mathcal{C} = \{$*n-ary negatable constraints*$\}$. If reals are finitely represented (e.g. as in floating point arithmetic), enumeration is applicable to continuous data. In the field of interval constraint solving, several works seek complete, sound solution sets in the presence of universally quantified variables. At present, the constraints must be able to be negated (which excludes equalities). The combination of numerical constraint propagation and search can be thought of as a non-naive enumeration. In [4], an exact method for a single uncertain coefficient is given; in [15], an approximate method for many coefficients.

*Transformation and enumeration for UCSPs with* $\mathcal{D} = \mathbb{Z}^m$, $\mathcal{U} = \mathbb{Z}^\ell$ *and* $\mathcal{C} = \{$*basic constraints*$\}$. Over finite domains, consider the classes of basic constraints as defined in [18]. A system of uncertain monotone constraints (e.g. binary inequalities) can be transformed by a CET similar to (3) in Sect. 2.1. The constraints of the resulting CSP are monotone, and their complete solution set can be found in linear time by computing the 2D integer hull [13]. For other types of basic constraints, enumeration is available.

*Enumeration for UCSPs with* $\mathcal{D} = \mathbb{Z}^m$, $\mathcal{U} = \mathbb{Z}^\ell$. CSP algorithms have been extended to derive robust solutions for mixed CSPs [9]. These algorithms can be adapted for the discrete data case of UCSPs over finite domains, as Sect. 2.2 illustrated.

If we consider a UCSP $P$ as a disjunction of its realised CSPs, $\bigvee_i \widehat{P}_i$, then $\mathrm{Cl}(P)$ is a constraint implied by the disjunction. Specifically, in *constructive disjunction* one eliminates all domain values not supported in at least one of the disjuncts (i.e. not supported by at least one realisation) [19]. However, because each $\widehat{P}_i$ is itself a conjunction, the constraints in $P$ would have to be of simple form if the algorithms of constructive disjunction are to be applied. In a similar way, *generalised propagation* can be thought of as reasoning on a disjunction to infer a constraint that describes all solutions [14]. Depending, again, on the complexity of the constraints, the *topological branch and bound* algorithm [14] can be used to derive the full closure to $P$ by enumeration.

[5] That is, optimisation problems with semi-definite constraint matrix.

## 5 Related Work

Existing generic approaches to uncertain data in CP propose models and methods for robust solutions to the problem. The *mixed CSP* framework [9] of Fargier et. al., defined for discrete data and variables, seeks a solution that holds under the most possible realisations of the data;[6] the *stochastic CSP* framework [21] of Walsh attaches a probability distribution to parameters and seeks a solution that maximises expectation. The purpose of computing robust solutions is to ensure that whatever the real world situation, the solution holds under most cases. Robust solutions are semantically ideal for dynamic changes but inadequate for handling data errors where one is certainly not looking for a solution that satisfies as many erroneous models as possible.

In dealing with unsatisfiability, the potential data issue is not considered in CP. The approach most widely used consists of reasoning at the constraint level: when the model is unsatisfiable, the usual interpretation is that the problem is over-constrained. Thus, most of the research has focused on relaxing constraints and setting priorities (e.g. [6]).

Besides work on quantified constraints over the reals [4], we are not aware of any work in CP aimed at building reliable solution sets in the presence of uncertain data. The closest parallels are the meta-solution reasoning of generalised propagation [14] and constructive disjunction [19].

While our work is defined with CP modelling in mind, in concept it is more closely related to work in control theory and operational research on continuous problems. In particular, *convex modelling* (of which interval analysis over the reals is a simple instance) is used to obtain a closure guaranteed to contain the true solution [2, 3, 8].

## 6 Discussion and Future Work

In this paper we have investigated how the successes of CP can be extended to real-world problems with data uncertainty. We introduced the certainty closure as a generic framework to allow the modelling of incomplete and erroneous data, both discrete and continuous. It guarantees reliable reasoning in that, whatever the true value of the data, the solution to the corresponding realised CSP is contained within the full closure.

A formal framework does not suffice unless its application to real LSCOs is practical. We derive a reliable solution set by solving standard CSPs, to make use of the most appropriate specific resolution techniques for the problem at hand. We have demonstrated the use of the framework by showing the benefits of reliable constraint reasoning on case studies from network traffic analysis and aerospace planning.

Most models with data uncertainty presently assume independence of the data (e.g. [9, 21]). For the certainty closure, assuming independence retains correctness but loses tightness. Future work will include study of how to extend the framework to account for dependency. We also wish to study new instances of the resolution forms for different

[6] If we restrict to finite domains and discrete data, a UCSP $\langle \mathcal{V}, \mathcal{D}, \Lambda, \mathcal{U}, \mathcal{C} \rangle$ can be viewed as a *mixed CSP* $\langle \Lambda, L, \mathcal{V}, \mathcal{D}, \mathcal{K}, \mathcal{C} \rangle$ with $\mathcal{U}$ being the complete solution set of the CSP $\langle \Lambda, L, \mathcal{K} \rangle$ induced by the parameters. However, as we discussed, the objectives (and algorithms, in general) of the two frameworks are quite different.

uncertain constraint classes. In particular we will consider the hybrid case where the data uncertainty is both discrete and continuous.

**Acknowledgements.** The authors thank P. Brisset, C. Guettier, S. Ratschan, M. Wallace, and the participants of the UICS'02 workshop for their discussions; and the reviewers for their recommendations. This work was partially supported by the EPSRC under grant GR/N64373/01.

## References

[1] K. R. Apt. The essence of constraint propagation. *TCS*, 221(1–2), 1999.
[2] Y. Ben-Haim and I. Elishakoff. *Convex Models of Uncertainty in Applied Mechanics*. Elsevier Science Publishers, Amsterdam, 1990.
[3] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23, 1998.
[4] F. Benhamou and F. Goualard. Universally quantified interval constraints. In *CP-2000*.
[5] T. Benoist, E. Bourreau, Y. Caseau, and B. Rottembourg. Towards stochastic constraint programming: A study of online multi-choice knapsack with deadlines. In *Proc. of CP'01*.
[6] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Basic properties and comparison. In *LNCS 1106*. 1996.
[7] A. Cheadle, W. Harvey, A. Sadler, J. Schimpf, K. Shen, and M. Wallace. *ECLiPSe: An Introduction*. IC–Parc Technical Report IC-Parc-03-1, 2003.
[8] J. W. Chinneck and K. Ramadan. Linear programming with interval coefficients. *J. Operational Research Society*, 51(2), 2000.
[9] H. Fargier, J. Lang, and T. Schiex. Mixed constraint satisfaction: A framework for decision problems under incomplete knowledge. In *Proc. of AAAI-96*, pages 175–180, 1996.
[10] E. Freuder and P. Hubbe. Extracting constraint satisfaction subproblems. In *Proc. of IJCAI-95*, pages 548–557, 1995.
[11] C. Gervet, Y. Caseau, and D. Montaut. On refining ill-defined constraint problems: A case study in iterative prototyping. In *Proc. of PACLP'99*, pages 255–275, 1999.
[12] C. Gervet and R. Rodošek. *RiskWise-2 problem definition*. IC–Parc Internal Report, 2000.
[13] W. Harvey. Computing two-dimensional integer hulls. *SIAM J. Computing*, 28(6), 1999.
[14] T. Le Provost and M. Wallace. Generalized constraint propagation over the CLP scheme. *J. Logic Programming*, 16(3), 1993.
[15] S. Ratschan. Continuous first-order constraint satisfaction. In *LNCS 2385*, 2002.
[16] I. Tsamardinos. *Constraint-Based Temporal Reasoning Algorithms with Applications to Planning*. Ph.D. Thesis, University of Pittsburgh, 2001.
[17] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London, 1993.
[18] P. Van Hentenryck, Y. Deville, and C.-M. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57(2–3), 1992.
[19] P. Van Hentenryck, V. Saraswat, and Y. Deville. Design, implementation, and evaluation of the constraint language cc(FD). In *LNCS 910*, 1994.
[20] G. Verfaillie. What kind of planning and scheduling tools for the future autonomous spacecraft? In *Proc. ESA Workshop on On-Board Autonomy*, 2001.
[21] T. Walsh. Stochastic constraint programming. In *Proc. of AAAI'01 Fall Symposium on Using Uncertainty within Computation*, pages 129–135, 2001.
[22] N. Yorke-Smith and C. Gervet. Data uncertainty in constraint programming: A non-probabilistic approach. In *Proc. of Using Uncertainty within Computation*, 2001.
[23] N. Yorke-Smith and C. Guettier. Towards automatic robust planning for the discrete commanding of aerospace equipment. In *Proc. of IEEE ISIC'03*, Oct. 2003. To appear.