

Reasoning with Finite Set Constraints

Rolf Haenni & Norbert Lehmann¹

Abstract. The language of propositional logic is often insufficient for efficiently representing the knowledge about a certain problem. In this paper we present an extension of classical propositional logic to finite set constraints. The idea is that a proposition x can also be considered as Boolean variable $x \in \{0, 1\}$. The literals x and $\sim x$ are then represented by Boolean set constraints $x \in \{1\}$ and $x \in \{0\}$ respectively. More generally, arbitrary finite variables x with values in Θ_x and finite set constraints $x \in X$, $X \subseteq \Theta_x$, can be considered. This improves the expressiveness of propositional logic significantly. This paper shows how fundamental deduction techniques for propositional logic such as resolution and variable elimination can be adapted for this more general situation.

1 Introduction

Knowledge and information is often encoded in the language of propositional logic. If Σ is a set of propositional formulae describing the knowledge about a certain problem, then questions about the problem can be answered by deduction and theorem proving techniques for propositional logic. Although propositional logic is capable to describe a certain class of problems, it is often insufficient for efficiently representing the given knowledge.

One problem is that propositional variables are restricted to the truth values *true* and *false*. In contrast, real life is not only black or white, good or bad, poor or rich, etc. There are often a number of gradations. For example, a colored pencil can be red, green, blue, yellow, . . .; the weather can be sunny, rainy, cloudy, . . .; etc. Therefore, it would be more convenient to have an extension of propositional logic to discrete variables, each of them taking exactly one value from a given set of possible values. For that purpose we introduce the notion of **finite set constraints (FSC)** [3].

The idea is that a proposition x can also be considered as Boolean variable $x \in \{0, 1\}$. The literals x and $\sim x$ are then represented by Boolean set constraints $x \in \{1\}$ and $x \in \{0\}$ respectively. More generally, arbitrary finite variables x with values in Θ_x are considered. A finite set constraint $x \in X$, $X \subseteq \Theta_x$, represents then the restriction of the possible values of x to X . Finite set constraints can be used to build compound formulae using the common connectives \sim , \wedge , \vee , \rightarrow , etc. Like in propositional logic, an entailment relation \models can be defined for such formulae. In such a way, **FSC logic** is defined as a generalization of classical propositional logic.

The language of finite set constraints often allows a shorter and more structured description of the problem. This additional structure can be exploited when methods of propositional logic are generalized to FSC logic. In this paper we show how the techniques of **resolution** and **variable elimination** can be generalized to FSC logic. These two techniques are of particular importance in the domain of assumption-based reasoning, which is the authors' main topic [8, 11, 12, 13]. The techniques presented in this paper are implemented in ABEL, a general language for assumption-based modeling and reasoning under uncertainty [1, 2]². Note that method of eliminating sequentially variables is also a possible way for testing the satisfiability of FSC formulae.

The resolution principle presented in this paper has already been investigated for **signed formulae** in the domain of **many-valued logics (MVL)** [9, 15, 17, 16]. The idea behind the MVL approach is that the set of possible truth values is extended from $\{0, 1\}$ (classical propositional logic) to an arbitrary set N . Depending on properties of the set N (finite, infinite, unordered, partially ordered, totally ordered, etc.), various classes of many-valued logics can be defined [10]. The case of a finite and unordered set N leads to the concept of **signed logic**, for which the resolution principle corresponds to FSC resolution as presented in this paper. The main difference between signed logic and FSC logic is that for signed logic the same number of possible values is assumed for all variables. From this point of view, FSC logic is more general, since different sets of values are possible for different variables. The main contribution of this paper, that is the algorithm of efficiently eliminating variables from a set of FSC formulae, can be applied for signed logic as well.

Section 2 introduces the main concepts of FSC logic. Section 3 describes how the deduction techniques of resolution and variable elimination can be generalized for FSC Logic. These techniques are illustrated by an example in Section 4. Appendix A contains the proofs of the theorems.

2 FSC Logic

In propositional logic atoms can be either true or false. Although this is sufficient to describe a certain class of problems, sometimes it is not very convenient. For that purpose an extension of propositional logic called **FSC logic** is introduced. FSC logic is based on a finite set S of variables like $x, y, \dots, x_1, x_2, \dots$ called the **alphabet**. A variable $x \in S$ takes exactly one value out of a given set of values Θ_x . Θ_x is called **frame** of x . A **finite set constraint (FSC)** $x \in X$ is

¹ Institute of Informatics, University of Fribourg, Switzerland. Research supported by grant No.2100-042927.95 of the Swiss National Foundation for Research.

² The software is available on the web and can be downloaded from <http://www2-iiuf.unifr.ch/tcs/ABEL>.

denoted by $X(x)$ where X is a subset of Θ_x . $X(x)$ is a predicate which is true if and only if the true value of the variable x is in X . Finite set constraints together with the symbols \perp and \top can then be used to build compound FSC formulae. For that purpose we use the connectives \sim , \wedge , \vee , \rightarrow , and \leftrightarrow :

- (1) FSCs, \perp and \top are FSC formulae.
- (2) If f is a FSC formula, then $\sim f$ is a FSC formula.
- (3) If f and g are FSC formulae, then $(f \wedge g)$, $(f \vee g)$, $(f \rightarrow g)$, and $(f \leftrightarrow g)$ are FSC formulae.
- (4) All FSC formulae are generated by applying the above rules.

Note that in some cases unnecessary parentheses can be omitted. Given the alphabet S , the symbol \mathcal{L}_S denotes the set of FSC formulae over S . \mathcal{L}_S is also called **FSC language** over S .

The basic properties of FSCs are given by the axioms of Set Theory [7]:

$$\emptyset(x) = \perp, \quad (1)$$

$$\Theta_x(x) = \top, \quad (2)$$

$$\sim X(x) = (\Theta_x - X)(x), \quad (3)$$

$$X_1(x) \vee X_2(x) = (X_1 \cup X_2)(x), \quad (4)$$

$$X_1(x) \wedge X_2(x) = (X_1 \cap X_2)(x). \quad (5)$$

These axioms allow to simplify FSC formulae. For example (3) can be used to eliminate negations. We call a FSC $X(x)$ **proper** whenever $X \neq \emptyset$ and $X \neq \Theta_x$. A disjunction of proper FSCs $X_1(x_1) \vee \dots \vee X_q(x_q)$ where every variable occurs at most once is called a **FSC clause**. Similarly, a conjunction of proper FSCs $X_1(x_1) \wedge \dots \wedge X_q(x_q)$ where every variable occurs at most once is called a **FSC conjunction**. Arbitrary disjunctions or conjunctions of FSCs can be transformed into corresponding FSC clauses or FSC conjunctions by (1)-(5). Note that neither FSC clauses nor FSC conjunctions contain negations. They can always be eliminated according to axiom (3). Also, every formula $f \in \mathcal{L}_S$ can be transformed into a **conjunctive normal form** (CNF) consisting of FSC clauses or a **disjunctive normal form** (DNF) consisting of FSC conjunctions.

The **domain** $d(f)$ of a FSC formula f is the set of variables which occur in f . If a CNF or DNF is represented as a set $\Sigma = \{c_1, \dots, c_n\}$ of FSC clauses or FSC conjunctions, then the domain of Σ is given by

$$d(\Sigma) = \bigcup_{c \in \Sigma} d(c). \quad (6)$$

The assignment of a value $v \in \Theta_x$ to every variable x of a formula is called an **interpretation** of the formula. Under an interpretation i , the truth value of a FSC $X(x)$ is 1 (true) whenever $v \in X$ and 0 (false) otherwise. Then, given the truth values of the FSCs contained in a formula, the truth value of the formula itself can be determined in the same way as in propositional logic. An alphabet S with n different variables determines a set $N_S = \Theta_{x_1} \times \dots \times \Theta_{x_n}$ of different interpretations. $N(f) \subseteq N_S$ then denotes the set of all interpretations for which a formula f is true. If $F = \{f_1, \dots, f_n\}$ is a set of FSC formulae, then $N(F) \subseteq N_S$ denotes the set of all interpretations for which all formulae $f_i \in F$ are true.

A FSC formula f **entails** a FSC formula g (denoted by $f \models g$) if and only if g is true under all interpretations for which f is true, i.e. if $N(f) \subseteq N(g)$. Similarly, a set of FSC formulae F entails another set G (denoted by $F \models G$), if and only if $N(F) \subseteq N(G)$.

Two FSC formulae f and g are **equivalent** (denoted by $f \equiv g$) if and only if the truth values of f and g are the same under all possible interpretations of the common alphabet of f and g , i.e. $N(f) = N(g)$. Note that equivalent FSC formulae represent exactly the same information or knowledge. Similarly, two sets of FSC formulae F and G are equivalent (denoted by $F \equiv G$), if and only if $N(F) = N(G)$.

A FSC clause or a FSC conjunction c can be considered as the set $\{X_1(x_1), \dots, X_q(x_q)\}$ of its FSCs. Given a FSC variable x we write $X(x) \in c$ in order to select the corresponding FSC of the variable x .

We say that a FSC clause d_1 **absorbs** another FSC clause d_2 whenever $N(d_1) \subseteq N(d_2)$. Similarly, we say that a FSC conjunction c_1 **absorbs** another FSC conjunction c_2 whenever $N(c_1) \supseteq N(c_2)$. Absorption of FSC clauses and FSC conjunctions can be tested as follows: given that d_1 and d_2 are both FSC clauses then d_1 absorbs d_2 if and only if for every $X_1(x) \in d_1$ there is a $X_2(x) \in d_2$ such that $X_1 \subseteq X_2$; similarly, given that c_1 and c_2 are FSC conjunctions, then c_1 absorbs c_2 if and only if for every $X_1(x) \in c_1$ there is a $X_2(x) \in c_2$ such that $X_1 \supseteq X_2$.

The result of removing all absorbed FSC clauses or FSC conjunctions from a CNF or DNF f is denoted by $\mu(f)$. If the CNF or DNF is given by a set Σ of FSC clauses or FSC conjunctions, then it is denoted as $\mu(\Sigma)$.

3 Resolution and Variable Elimination

Given a set $\Sigma = \{d_1, d_2, \dots, d_n\}$ or FSC clauses the problem is to find another set Σ' of clauses in which all occurrences of a given variable x have been eliminated. Therefore, we have $d(\Sigma') \subseteq d(\Sigma) - \{x\}$. For every formula $f \in \mathcal{L}_Q$, $Q \subseteq d(\Sigma')$, Σ' has to satisfy

$$\begin{aligned} (C1) \quad & \Sigma \models \Sigma', \\ (C2) \quad & \Sigma \models f \implies \Sigma' \models f. \end{aligned}$$

These two conditions are equivalent to

$$\Sigma \models f \iff \Sigma' \models f, \quad \text{for every } f \in \mathcal{L}_Q.$$

Therefore, all formulae f , $d(f) \subseteq d(\Sigma')$, which can be proved in Σ' can also be proved in Σ . Additionally, if f can not be deduced from Σ' , then it can not be deduced in Σ .

Given a set Σ of propositional or FSC clauses and a domain $D' \subseteq d(\Sigma)$, the result of eliminating all occurrences of propositional variables in $d(\Sigma) - D'$ from Σ is denoted by $\Sigma^{\downarrow D'}$. It is called a **marginal** of Σ with respect to D' , and the operation of computing $\Sigma^{\downarrow D'}$ is called **marginalization**. Note that computing $\Sigma^{\downarrow \emptyset}$ corresponds to testing the **satisfiability** of Σ .

When several variables are eliminated one after another, then different elimination sequences may produce different marginals. Of course, these marginals must all be (logically) equivalent. Therefore, for every domain D'' , $d(\Sigma) \supseteq D'' \supseteq D'$, the following condition must hold:

$$(C3) \quad \Sigma^{\downarrow D'} \equiv (\Sigma^{\downarrow D''})^{\downarrow D'}.$$

Now we will show how the variable elimination method for propositional logic (also known as **Davis-Putnam Procedure** [4, 5, 14]) can be generalized to FSC logic. At least two different approaches to compute Σ' are possible:

- (i) The set Σ is transformed into an equivalent set of clauses where every variable x has been replaced by some new propositional symbols as explained in [3]. Then the classical variable elimination method for propositional logic can be applied.
- (ii) Resolution can be generalized to FSC logic. A new variable elimination method for FSC logic can then be defined.

In this paper only the second approach is discussed, that is we show how resolution can be generalized to FSC logic. The idea is the same as in [6] where resolution is defined for a general logic. Note that the same principle has already been applied for signed formulae in the domain of multi-valued logics [9, 15, 17, 16, 10].

Definition 1 Let $d_1 = X_1(x) \vee \vartheta_1$ and $d_2 = X_2(x) \vee \vartheta_2$ be two FSC clauses. FSC resolution over x is defined by

$$\rho_x(d_1, d_2) = (X_1 \cap X_2)(x) \vee \vartheta_1 \vee \vartheta_2. \quad (7)$$

The result $\rho_x(d_1, d_2)$ is called **resolvent** of d_1 and d_2 over x . The FSC $(X_1 \cap X_2)(x)$ above is called **residue**. FSC resolution is illustrated in Figure 1.

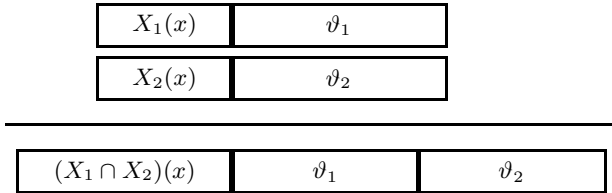


Figure 1. Resolution of two FSC clauses.

It is easy to see that FSC resolution over a fixed variable x is **commutative**, **associative** and **idempotent**, i.e.

$$\rho_x(d_1, d_2) = \rho_x(d_2, d_1), \quad (8)$$

$$\rho_x(\rho_x(d_1, d_2), d_3) = \rho_x(d_1, \rho_x(d_2, d_3)), \quad (9)$$

$$\rho_x(d_1, d_1) = d_1. \quad (10)$$

Theorem 1 For every pair (d_1, d_2) of FSC clauses we have

$$d_1 \wedge d_2 \models \rho_x(d_1, d_2). \quad (11)$$

If X_1 and X_2 in (7) are disjoint, then the residue will be contradictory because of $(X_1 \cap X_2) = \emptyset$ and $\emptyset(x) = \perp$. In this case the variable x does not appear in $\rho_x(d_1, d_2)$ any more.

If d_1 does not contain the variable x then for every clause d_2 the resolvent $\rho_x(d_1, d_2)$ is absorbed by d_1 . More generally,

for two FSC clauses $d_1 = X_1(x) \vee \vartheta_1$ and $d_2 = X_2(x) \vee \vartheta_2$ with $X_1 \subseteq X_2$, we have

$$d_1 \models \rho_x(d_1, d_2).$$

Similarly, $\rho_x(d_1, d_2)$ is absorbed by d_2 whenever $X_2 \subseteq X_1$.

Let Σ be the set of FSC clauses from which all occurrences of the variable x have to be eliminated. Without loss of generality the elements of Σ can always be renamed and arranged in such a way that the FSC clauses containing x appear first in Σ . Therefore, $\Sigma = \{d_1, \dots, d_m, d_{m+1}, \dots, d_n\}$ can be decomposed into

$$\begin{aligned} \Sigma_+ &= \{d_1, \dots, d_m\} = \{d_i \in \Sigma : x \in d(d_i)\}, \\ \Sigma_* &= \{d_{m+1}, \dots, d_n\} = \{d_i \in \Sigma : x \notin d(d_i)\} \\ &= \Sigma - \Sigma_+. \end{aligned}$$

Every $d_i \in \Sigma_+$ can be written as $X_i(x) \vee \vartheta_i$ with ϑ_i being a FSC clause. To eliminate x from Σ_+ the FSC resolution defined in (7) can be used. When resolving two FSC clauses, often the residue is not empty and therefore the resolvent still contains the variable x . Then it may be possible to eliminate x performing additional resolutions.

Definition 2 Let $\Sigma_+ = \{d_1, \dots, d_m\}$ with $d_i = X_i(x) \vee \vartheta_i$. FSC resolution $\hat{\rho}_x$ for a set $\Upsilon \subseteq \Sigma_+$ of FSC clauses relative to a variable x is defined by

$$\hat{\rho}_x(\Upsilon) = \left(\bigcap_{d_i \in \Upsilon} X_i \right) (x) \vee \left(\bigvee_{d_i \in \Upsilon} \vartheta_i \right). \quad (12)$$

Figure 2 illustrates FSC resolution for a given set of FSC clauses. Note that $\hat{\rho}_x(\Upsilon)$ resolves all $d \in \Upsilon$ in one step. Clearly, $\hat{\rho}_x(\Upsilon)$ can be calculated incrementally by performing several resolutions ρ_x , each of them only resolving two FSC clauses. Because ρ_x is associative and commutative there are different ways to get to the same final result.

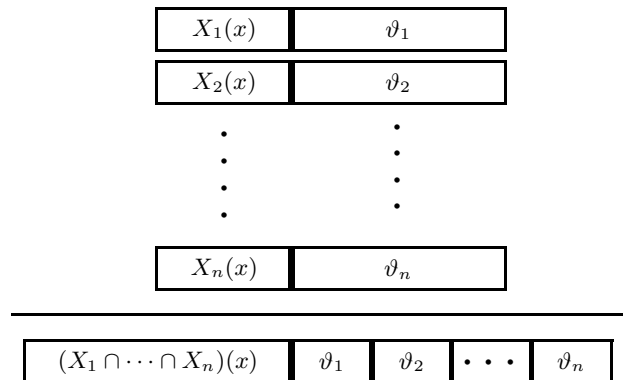


Figure 2. Resolution of a set of FSC clauses.

Theorem 2 For every set $\Upsilon \subseteq \Sigma_+$ we have

$$\bigwedge_{d_i \in \Upsilon} d_i \models \hat{\rho}_x(\Upsilon). \quad (13)$$

Depending on Υ , the variable x may still be contained in $\hat{\rho}_x(\Upsilon)$. Of particular interest are those sets Υ for which $\hat{\rho}_x(\Upsilon)$ does not contain x any more.

Definition 3 Given the set Σ_+ , $R_x(\Sigma_+)$ is defined to be the set of resolvents of subsets Υ of Σ_+ which do not contain x any more:

$$R_x(\Sigma_+) = \{\hat{\rho}_x(\Upsilon) : \Upsilon \subseteq \Sigma_+, x \notin d(\hat{\rho}_x(\Upsilon))\}. \quad (14)$$

The new set Σ' is obtained by

$$\Sigma' = \mu\{R_x(\Sigma_+) \cup \Sigma_*\}. \quad (15)$$

Theorem 3 The above variable elimination method for FSC logic satisfies conditions (C1), (C2), and (C3) at the top of Section 3.

A primitive algorithm for computing $R_x(\Sigma_+)$ would calculate $\hat{\rho}_x(\Upsilon)$ for every subset Υ of Σ_+ and then remove all FSC clauses still containing the variable x . A more efficient algorithm is obtained when $\hat{\rho}_x(\Upsilon)$ is calculated by reusing previous intermediate results. Consider for example the tree in Figure 3: It consists of 8 nodes. Each node contains a subset of $\Sigma_+ = \{d_1, d_2, d_3\}$ and a list of FSC clauses. The list of FSC clauses is used for the construction of the subtree rooted at the corresponding node. For every node of the tree, FSC resolution is performed for the corresponding subset by reusing the resolution performed on the predecessor node. As an example, $\hat{\rho}_x(\{d_1, d_2, d_3\})$ is calculated as $\rho_x(\hat{\rho}_x(\{d_1, d_2\}), d_3)$.

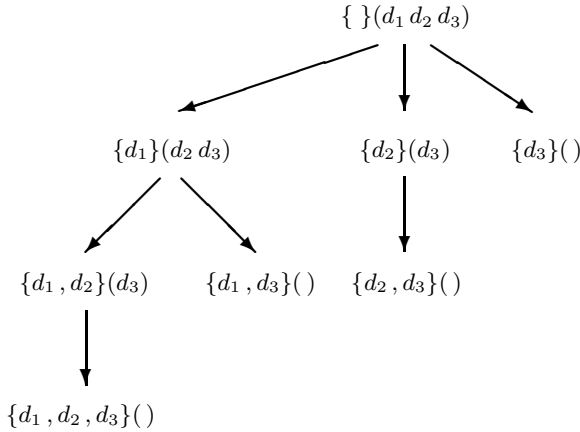


Figure 3. Subsets of $\Sigma_+ = \{d_1, d_2, d_3\}$ ordered in a tree.

For $\Upsilon \subseteq \Sigma_+$ and $d_\ell \in \Upsilon$, $\hat{\rho}_x(\Upsilon)$ is calculated as $\rho_x(\hat{\rho}_x(\Upsilon - d_\ell), d_\ell)$. Therefore, a recursive formula to compute $R_x(\Sigma_+)$ is given by

$$R_x(\Sigma_+) = \{r_x = \rho_x(\hat{\rho}_x(\Upsilon - d_\ell), d_\ell) : \begin{aligned} & d_\ell \in \Sigma_+, \Upsilon \subseteq \{d_1, \dots, d_\ell\}, \\ & d_1 \in \Upsilon, x \notin d(r_x) \\ & \cup R_x(\Sigma_+ - \{d_1\}). \end{aligned} \quad (16)$$

This improvement is not yet very practical because in general there are too many subsets Υ . Another issue is that $R_x(\Sigma_+)$

could contain many unnecessary FSC clauses which are absorbed by others. Such FSC clauses should be prevented from being produced because later they have to be removed in any case.

A further improvement is therefore obtained if instead of $R_x(\Sigma_+)$ a logically equivalent set $R'_x(\Sigma_+)$ is computed. For an ordering of the FSC clauses of Σ_+ there is a tree as shown in Figure 3. It contains the subsets of Σ_+ as nodes. Starting from the empty subset the tree is run through from left to right and $\hat{\rho}_x(\Upsilon)$ is still calculated as $\rho_x(\hat{\rho}_x(\Upsilon - d_\ell), d_\ell)$ according to (16). But now not all nodes are visited. There are three special cases which can improve the efficiency of the algorithm considerably:

- (1) $x \notin d(\hat{\rho}_x(\Upsilon))$,
- (2) $X(x) \in \hat{\rho}_x(\Upsilon - d_\ell)$ and $X_\ell(x) \in d_\ell$ such that $X \subseteq X_\ell$ or $X \supseteq X_\ell$,
- (3) $\hat{\rho}_x(\Upsilon) = \top$.

In the first case x has been eliminated. Therefore, $\hat{\rho}_x(\Upsilon)$ can be adjoined to $R'_x(\Sigma_+)$. In the second case the resolvent is absorbed by either $\hat{\rho}_x(\Upsilon - d_\ell)$ or d_ℓ . Therefore, it is not necessary to calculate it. For each of the three cases and for every $\Phi \supset \Upsilon$, $\hat{\rho}_x(\Phi)$ has not to be calculated.

Whenever one of the above tree special cases occurs, it is not necessary to run through the subtree rooted at the corresponding node. Such a subtree can be pruned.

4 Example

The two approaches of eliminating all occurrences of a given variable from a set of FSC clauses are illustrated by an example. Suppose the following set $\Sigma = \{d_1, d_2, d_3, d_4, d_5, d_6\}$ of FSC clauses is given:

$$\begin{aligned} d_1 &= (x \in \{a_4\}) \vee (z \in \{c_1, c_3\}), \\ d_2 &= (x \in \{a_1, a_2\}) \vee (y \in \{b_1\}), \\ d_3 &= (x \in \{a_1, a_3\}) \vee (y \in \{b_1, b_3\}) \vee (z \in \{c_2, c_3\}), \\ d_4 &= (x \in \{a_2, a_3, a_4\}) \vee (y \in \{b_3\}), \\ d_5 &= (y \in \{b_1, b_2\}) \vee (z \in \{c_1, c_3\}), \\ d_6 &= (y \in \{b_2, b_3\}). \end{aligned}$$

We have $d(\Sigma) = \{x, y, z\}$. Let the frame of these variables be $\Theta_x = \{a_1, a_2, a_3, a_4\}$, $\Theta_y = \{b_1, b_2, b_3\}$, and $\Theta_z = \{c_1, c_2, c_3\}$, respectively. The problem is to compute $\Sigma' = \Sigma^{\downarrow\{x, z\}}$ by eliminating the variable x . First, the set Σ is decomposed into $\Sigma_+ = \{d_1, d_2, d_3, d_4\}$ and $\Sigma_* = \{d_5, d_6\}$. Initially, $R'_x(\Sigma_+) = \emptyset$. There are a number of FSC resolutions to be performed.

- $\hat{\rho}_x(\{d_1, d_2\}) = \rho_x(d_1, d_2) = (y \in \{b_1\}) \vee (z \in \{c_1, c_3\}) \Rightarrow R'_x(\Sigma_+) = \{(y \in \{b_1\}) \vee (z \in \{c_1, c_3\})\}$;
- $\hat{\rho}_x(\{d_1, d_3\}) = \rho_x(d_1, d_3) = (y \in \{b_1, b_3\}) \vee (z \in \{c_1, c_2, c_3\}) = \top$;
- $\hat{\rho}_x(\{d_1, d_4\}) = \rho_x(d_1, d_4)$ not necessary because of $\{a_4\} \subseteq \{a_2, a_3, a_4\}$;
- $\hat{\rho}_x(\{d_2, d_3\}) = \rho_x(d_2, d_3) = (x \in \{a_1\}) \vee (y \in \{b_1, b_3\}) \vee (z \in \{c_2, c_3\})$;
- $\hat{\rho}_x(\{d_2, d_3, d_4\}) = \rho_x(\rho_x(d_2, d_3), d_4) = (y \in \{b_1, b_3\}) \vee (z \in \{c_2, c_3\}) \Rightarrow R'_x(\Sigma_+) = \{(y \in \{b_1\}) \vee (z \in \{c_1, c_3\}), (y \in \{b_1, b_3\}) \vee (z \in \{c_2, c_3\})\}$;

- $\hat{\rho}_x(\{d_3, d_4\}) = \rho_x(d_3, d_4) = (x \in \{a_3\}) \vee (y \in \{b_1, b_3\}) \vee (z \in \{c_2, c_3\})$.

Note that it was not necessary to calculate $\hat{\rho}_x$ for some subsets of Σ_+ . For example the following resolvents have not been computed:

- $\hat{\rho}_x(\{d_1, d_2, d_3\})$ because the variable x was already eliminated in $\hat{\rho}_x(\{d_1, d_2\})$,
- $\hat{\rho}_x(\{d_1, d_3, d_4\})$ because $\hat{\rho}_x(\{d_1, d_3\})$ is equal to \top ,
- $\hat{\rho}_x(\{d_1, d_4\})$ because $\{a_4\} \subseteq \{a_2, a_3, a_4\}$.

Finally, we get

$$\begin{aligned} R'_x(\Sigma_+) &= \{(y \in \{b_1\}) \vee (z \in \{c_1, c_3\}), \\ &\quad (y \in \{b_1, b_3\}) \vee (z \in \{c_2, c_3\})\}, \\ \Sigma_* &= \{(y \in \{b_1, b_2\}) \vee (z \in \{c_1, c_3\}), (y \in \{b_2, b_3\})\}. \end{aligned}$$

The final result $\mu\{R'_x(\Sigma_+) \cup \Sigma_*\}$ consists of three FSC clauses

- (1) $(y \in \{b_1\}) \vee (z \in \{c_1, c_3\})$,
- (2) $(y \in \{b_1, b_3\}) \vee (z \in \{c_2, c_3\})$,
- (3) $(y \in \{b_2, b_3\})$.

5 Summary

The paper presents a direct way for eliminating variables in FSC logic by resolution. This technique is more natural and more efficient than the alternative way of first transforming the FSC formulae into propositional formulae and then to use the variable elimination method for propositional logic.

The use of FSC logic improves the expressiveness of propositional logic significantly. Problems consisting of variables with (non-binary) finite domains can then be modeled and solved more easily. The technique presented in this paper has been successfully implemented in ABEL and it works well for a number of examples in different domains. Future work will focus on approximation strategies for problems which are not feasible.

A Proofs

A.1 Proof of Theorem 1

Let $d_i = X_i(x) \vee \vartheta_i$, $\rho_x(d_1, d_2) = (X_1 \cap X_2)(x) \vee \vartheta_1 \vee \vartheta_2$. Every interpretation satisfying ϑ_1 or ϑ_2 also satisfies $\rho_x(d_1, d_2)$. Every interpretation satisfying neither ϑ_1 nor ϑ_2 has to satisfy both $X_1(x)$ and $X_2(x)$ in order to make $d_1 \wedge d_2$ true. Therefore, $(X_1 \cap X_2)(x)$ is satisfied, hence also $\rho_x(d_1, d_2)$.

qed.

A.2 Proof of Theorem 2

The proof is analogous to the previous one. Let $\Upsilon = \{d_1, \dots, d_n\}$, $d_i = X_i(x) \vee \vartheta_i$. Every interpretation satisfying one of the ϑ_i also satisfies $\hat{\rho}_x(\Upsilon)$. Every other interpretation has to satisfy all $X_i(x)$ in order to make $\bigwedge_{d_i \in \Upsilon} d_i$ true. Therefore, $\bigcap\{X_j : d_j \in \Upsilon\}(x)$ is satisfied, hence also $\hat{\rho}_x(\Upsilon)$.

qed.

A.3 Proof of Theorem 3

There are three conditions to prove:

- (1) $\Sigma \models \Sigma'$,
- (2) for every $h \in \mathcal{L}_{d(\Sigma')}$ such that $\Sigma \models h$ we have $\Sigma' \models h$,
- (3) $\Sigma^{\perp D''} = (\Sigma^{\perp D'})^{\perp D''}$.

Ad (1): $\Sigma' = \mu\{R'_x(\Sigma_+) \cup \Sigma_*\}$ is equivalent to $\mu\{R_x(\Sigma_+) \cup \Sigma_*\}$. We know that

- (i) $\Sigma \models \Sigma_*$
- (ii) $\Sigma \models R_x(\Sigma_+)$ because for every $\gamma \in R_x(\Sigma_+)$ there is a $\Upsilon \subseteq \Sigma$ such that

$$\Sigma \models \bigwedge_{d_i \in \Upsilon} d_i \models \hat{\rho}_x(\Upsilon) = \gamma.$$

(i) together with (ii) implies $\Sigma \models \Sigma'$.

Ad (2): If $\Sigma_+ = \emptyset$ then $\Sigma' = \Sigma$ and the condition is trivially fulfilled. Otherwise let $I = \{1, \dots, m\}$ be the index set of Σ_+ . For $J \subseteq I$ let's define

$$\begin{aligned} X_J &:= \bigcap_{i \in J} X_i, \\ S_{\top} &:= \{J : J \subseteq I, X_J \neq \emptyset\}, \\ S_{\perp} &:= \{J : J \subseteq I, X_J = \emptyset\}. \end{aligned}$$

$R'_x(\Sigma_+)$ is logically equivalent to $R_x(\Sigma_+)$. Σ_+ and $R_x(\Sigma_+)$ can be written as

$$\begin{aligned} \Sigma_+ &= \bigwedge_{i \in I} (X_i(x) \vee \vartheta_i) \\ &= \bigvee_{J \in S_{\perp}} \left(X_J(x) \wedge \left(\bigwedge_{i \in I-J} \vartheta_i \right) \right) \vee \\ &\quad \bigvee_{J \in S_{\top}} \left(X_J(x) \wedge \left(\bigwedge_{i \in I-J} \vartheta_i \right) \right) \\ &= \bigvee_{J \in S_{\top}} \left(X_J(x) \wedge \left(\bigwedge_{i \in I-J} \vartheta_i \right) \right), \\ R_x(\Sigma_+) &= \bigwedge_{J \in S_{\perp}} \left(\bigvee_{i \in J} \vartheta_i \right). \end{aligned}$$

Given $\Sigma_+ \neq \emptyset$ then the condition is equivalent to the following statement:

For every $h \in \mathcal{L}_{d(\Sigma')}$ with $\Sigma_+, \Sigma_*, \sim h \models \perp$ we have $R_x(\Sigma_+), \Sigma_*, \sim h \models \perp$.

Then it is sufficient to prove the following statement:

For every $h^* \in \mathcal{L}_{d(\Sigma')}$ with $\Sigma_+, h^* \models \perp$ we have $R_x(\Sigma_+), h^* \models \perp$.

Two cases can be distinguished:

- (i) Interpretations which do not satisfy h^* ;
- (ii) Interpretations which satisfy h^* .

In the first case, the implication is trivially fulfilled. For the interpretations of the second case, Σ_+ must be false. Therefore, for every $J \in S_\top$,

$$X_J(x) \wedge \left(\bigwedge_{j \in I-J} \vartheta_j \right) \text{ is false.}$$

For every $j \in I - J$, $J \in S_\top$, the variable x does not occur in ϑ_j and $X_J(x)$ can always be satisfied. Therefore,

$$\bigvee_{J \in S_\top} \left(\bigwedge_{j \in I-J} \vartheta_j \right) \text{ is false.}$$

This disjunction of conjunctions can be transformed into an equivalent conjunction of disjunctions. Then, for some set S ,

$$\bigwedge_{J \in S} \left(\bigvee_{i \in J} \vartheta_i \right) \text{ is false.}$$

It is sufficient to show that $S \subseteq S_\perp$: Let $S_\ell \in S$. It is either $S_\ell \in S_\top$ or $S_\ell \in S_\perp$. $S_\ell \in S_\top$ is not possible because in this case, due to the construction of S , S_ℓ would contain at least one element of $I - S_\ell$. Therefore, it is $S_\ell \in S_\perp$.

Ad (3): The variable elimination method satisfies conditions (1) and (2) which are proved above. For all $h \in \mathcal{L}_{D''}$ the following two statements are therefore valid:

$$\begin{aligned} \Sigma \models h &\iff \Sigma^{\downarrow D''} \models h, \\ \Sigma \models h &\iff (\Sigma^{\downarrow D'})^{\downarrow D''} \models h. \end{aligned}$$

From these two statements it follows for all $h \in \mathcal{L}_{D''}$ that

$$\Sigma^{\downarrow D''} \models h \iff (\Sigma^{\downarrow D'})^{\downarrow D''} \models h.$$

This allows to conclude that

$$\Sigma^{\downarrow D''} = (\Sigma^{\downarrow D'})^{\downarrow D''}.$$

qed.

REFERENCES

- [1] B. Anrig, R. Haenni, J. Kohlas, and N. Lehmann, ‘Assumption-based modeling using ABEL’, in *First International Joint Conference on Qualitative and Quantitative Practical Reasoning; ECSQARU-FAPR’97*, eds., D. Gabbay, R. Kruse, A. Nonnengart, and H.J. Ohlbach. Lecture Notes in Artif. Intell., Springer, (1997).
- [2] B. Anrig, R. Haenni, and N. Lehmann, ‘ABEL – a new language for assumption-based evidential reasoning under uncertainty’, Technical Report 97-01, University of Fribourg, Institute of Informatics, (1997).
- [3] B. Anrig, N. Lehmann, and R. Haenni, ‘Reasoning with finite set constraints’, Working Paper 97-11, Institute of Informatics, University of Fribourg, (1997).
- [4] M. Davis and H. Putnam, ‘A computing procedure for quantification theory’, *Journal of the ACM*, **5**, 394–397, (1962).
- [5] M. Davis and H. Putnam, ‘A computing procedure for quantification theory’, in *Automation of Reasoning 1: Classical Papers on Computational Logic 1957-1966*, eds., J. Siekmann and G. Wrightson, 125–139, Springer, Berlin, Heidelberg, (1983).
- [6] N. Eisinger and H.J. Ohlbach, ‘Deduction systems based on resolution’, in *Handbook of Logic in Artificial Intelligence and Logic Programming*, eds., D.M. Gabbay, C.J. Hogger, and J.A. Robinson, 184–263, Oxford Science Publications, (1993).
- [7] D. Gries and F.B. Schneider, *A Logical Approach to Discrete Math*, Springer-Verlag, 1993.
- [8] R. Haenni, *Propositional Argumentation Systems and Symbolic Evidence Theory*, Ph.D. dissertation, Institute of Informatics, University of Fribourg, 1996.
- [9] Reiner Hähnle, ‘Short conjunctive normal forms in finitely-valued logics’, *Journal of Logic and Computation*, **4**(6), 905–927, (1994).
- [10] Reiner Hähnle and Gonzalo Escalada-Imaz, ‘Deduction in many-valued logics: a survey’, *Mathware & Soft Computing*, **IV**(2), 69–97, (1997).
- [11] J. Kohlas and P.A. Monney, ‘Probabilistic assumption-based reasoning’, in *Proc. 9th Conf. on Uncertainty in Artificial Intelligence*, eds., Heckerman and Mamdani. Kaufmann, Morgan Publ., (1993).
- [12] J. Kohlas and P.A. Monney, *A Mathematical Theory of Hints. An Approach to the Dempster-Shafer Theory of Evidence*, volume 425 of *Lecture Notes in Economics and Mathematical Systems*, Springer, 1995.
- [13] J. Kohlas, P.A. Monney, B. Anrig, and R. Haenni, ‘Model-based diagnostics and probabilistic assumption-based reasoning’, Technical Report 96-09, University of Fribourg, Institute of Informatics, (1996).
- [14] J. Kohlas and S. Moral, ‘Propositional information system’, Working paper, Institute of Informatics, University of Fribourg, (1995).
- [15] J. Lu, N. Murray, and E. Rosenthal, ‘A framework for automated reasoning in multiple-valued logics’, Technical Report 94-04, State University of New York at Albany, (1994).
- [16] James J. Lu, ‘Logic programming with signs and annotations’, *Journal of Logic and Computation*, **6**(6), 755–778, (1996).
- [17] Neil V. Murray and Erik Rosenthal, ‘Adapting classical inference techniques to multiple-valued logics using signed formulas’, *Fundamenta Informaticae*, **21**(3), 237–253, (1994).