The Logic of Constraint Satisfaction

by

Alan K. Mackworth

Technical Report 91-26
November 1991

Department of Computer Science
University of British Columbia
Rm 333 - 6356 Agricultural Road
Vancouver, B.C.
CANADA V6T 1Z2

The Logic of Constraint Satisfaction

Alan K. Mackworth[1]

Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1W5

November 1, 1991

Abstract

The Constraint Satisfaction Problem (CSP) formalization has been a productive tool within Artificial Intelligence and related areas. The Finite CSP (FCSP) framework is presented here as a restricted logical calculus within a space of logical representation and reasoning systems. FCSP is formulated in a variety of logical settings: theorem proving in first order predicate calculus, propositional theorem proving (and hence SAT), the Prolog and Datalog approaches, constraint network algorithms, a logical interpreter for networks of constraints, the Constraint Logic Programming (CLP) paradigm and propositional model finding (and hence SAT, again). Several standard, and some not-so-standard, logical methods can therefore be used to solve these problems. By doing this we obtain a specification of the semantics of the common approaches. This synthetic treatment also allows algorithms and results from these disparate areas to be imported, and specialized, to FCSP; the special properties of FCSP are exploited to achieve, for example, completeness and to improve efficiency. It also allows export to the related areas. By casting CSP both as a generalization of FCSP and as a specialization of CLP it is observed that some, but not all, FCSP techniques lift to CSP and, perhaps, thereby to CLP. Various new connections are uncovered, in particular between the proof-finding approaches and the alternative model-finding approaches that have arisen in depiction and diagnosis applications.

[1] Fellow, Canadian Institute for Advanced Research

1. Logical Frameworks for Constraint Satisfaction

Informally, a Constraint Satisfaction Problem (CSP) is posed as follows. Given a set of variables and a set of constraints, each specifying a relation on a particular subset of the variables, find the relation on the set of all the variables which satisfies all the given constraints. Typically, the given unary relation for each variable specifies its domain as a set of possible values; the required solution relation is a subset of the Cartesian product of the variable domains. If each domain is finite the CSP is a Finite Constraint Satisfaction Problem (FCSP).

The formulation of the CSP paradigm has yielded substantial theoretical and practical results [6,17]. It is important, though, not to conceive of the CSP paradigm in isolation but to see it in its proper context — namely, as a highly restricted logical calculus with associated properties and algorithms. The purpose of this paper is to place CSP's in that context, to redevise some old results in new, simpler ways, and to establish connections amongst the differing views of CSP's. Essentially the paper can be seen as an extended answer to the question, "Does the CSP framework make logical sense?" The ambiguity of the question lies in the fact that it can be read as "Is it sensible to isolate the CSP paradigm?" or as "Can we interpret the CSP paradigm using logical notions?" The paper can also be seen as a response to a cynical critic who asks, "Is CSP merely old wine in new bottles?" The paper is intended to lead to answers to the following questions:

- Can FCSP be posed in logical frameworks?
- Can standard logical methods be used to solve FCSP?
- Can the special properties of FCSP be exploited to get better algorithms?
- Are tractable classes of FCSP revealed?
- Do old results fall out?
- What are the relationships among the several logical views of FCSP?
- Can the approaches for FCSP be lifted to CSP?
- Do we get new results and systems?

2. An FCSP: The Canadian Flag Problem

To fix the ideas of this paper a trivial FCSP will be used as an example. Consider the well-known Canadian Flag Problem. A committee proposed a new design for the Canadian flag, shown in Figure 1. The problem is to decide how to colour the flag. Only two colours, red and white, should be used; each region should be a different colour from its neighbours, and the maple leaf should, of course, be red. The problem is so trivial that its solution requires little thought, even though the committee could not solve it, but it serves our purpose here.

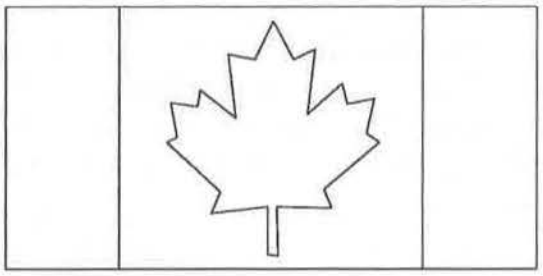


Figure 1. A Trivial FCSP: Colour The Canadian Flag

3. FCS as Theorem Proving in FOPC

The simplest standard logical CSP formulation is as theorem proving in a restricted first order predicate calculus [16]. An FCSP decision problem can be posed as $Constraints \vdash Query?$ where $Query$ has the form

$$Query: \quad \exists X_1 \exists X_2 \ldots \exists X_n QMatrix(X_1, X_2, \ldots, X_n)$$

or

$$\begin{aligned} Query: \quad \exists X_1 \exists X_2 \ldots \exists X_n p_{X_1}(X_1) \wedge p_{X_2}(X_2) \wedge \ldots \wedge p_{X_n}(X_n) \wedge \\ p_{X_1X_2}(X_1, X_2) \wedge p_{X_1X_3}(X_1, X_3) \wedge \ldots \wedge \\ p_{X_1X_2X_3}(X_1, X_2, X_3) \wedge \ldots \wedge \\ \vdots \qquad\qquad \\ p_{X_1X_2X_3 \ldots X_n}(X_1, X_2, X_3, \ldots, X_n) \end{aligned}$$

and $Constraints$ is a set of ground atoms specifying the extensions of the predicates

$$Constraints: \quad \{p_{X_{i_1}X_{i_2}\ldots X_{i_m}}(c_{i_1}, c_{i_2}, \ldots, c_{i_m}) | 1 \leq i_k < i_{k+1} \leq n\}$$

where the c_i are constants. Notice that in this formulation we are only specifying the tuples allowed by a relation, not the tuples forbidden, since $Constraints$ consists of positive literals.

4. FCS Decision Problems

An FCSP is specified by a $(Constraints, Query)$ pair. A common candidate formulation of the FCS decision problem is to determine if it can be shown that a solution exists or if it can be shown that a solution does not exist: $Constraints \vdash Query$, or $Constraints \vdash \neg Query$. However, given the positive form specified for $Constraints$, it is never possible to establish that $Constraints \vdash \neg Query$ so this candidate formulation is unacceptable. Later when we consider the completion of $Constraints$ we shall return to a variant of this formulation.

The FCS Decision Problem (FCSDP) is to determine if it can be shown that a solution exists or if cannot be shown that a solution exists: $Constraints \vdash Query$ or $Constraints \nvdash Query$.

If the decision problem is posed in the form of FCSDP and the constraints are supplied or discovered incrementally in the form of additional allowed tuples, extending the set $Constraints$, then the answers to FCSDP are monotonic: a 'No' may change to 'Yes' but not *vice versa.*

Proposition *FCSDP is decidable.*
Proof For an FCSP specified by the pair $(Constraints, Query)$ a decision algorithm to determine if $Constraints \vdash Query$ or $Constraints \nvdash Query$ is required. The Herbrand universe H of the theory $Constraints \bigcup \neg Query$ is

$$H = \{c \mid p_{\mathcal{V}}(\ldots, c, \ldots) \in Constraints\}$$

H is finite.

Consider the following algorithm

```
Decision Algorithm DA:
$Success \leftarrow$ No
For each tuple $(c_1, c_2, \ldots, c_n) \in H^n$
    If $Constraints \vdash QMatrix\,(c_1, c_2, \ldots, c_n)$ then $Success \leftarrow$ Yes
Report $Success$
End DA
```

where $Constraints \vdash QMatrix\,(c_1, c_2, \ldots, c_n)$ iff for each *Atom* mentioned in $QMatrix\,(c_1, c_2, \ldots, c_n)$ it is the case that $Atom \in Constraints$.

DA always terminates. It reports 'Yes' iff $Constraints \vdash Query$. It reports 'No' iff $Constraints \nvdash Query$. ■

The number of predicate evaluations made by DA is

$$(\#atoms\ in\ QMatrix(X_1, X_2, \ldots, X_n)) \times |H|^n$$

5. Completing the Constraints

Consider the completion of *Constraints* with respect to *Query*. Each predicate mentioned in *Query* can be completed [2] in the following sense:

$$completion(Constraints) =$$
$$Constraints \cup \{\neg p_V(c_1, c_2, \ldots, c_k) | c_i \in H, p_V(c_1, c_2, \ldots, c_k) \notin Constraints\}$$

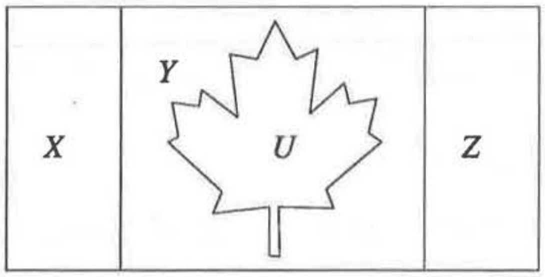
In other words, the complete extension of each k-ary predicate over H^k is specified, positively and negatively, in $completion(Constraints)$.

Notice that $Constraints \vdash Query$ iff $completion(Constraints) \vdash Query$ and $Constraints \nvdash Query$ iff $completion(Constraints) \vdash \neg Query$. Hence, DA reports 'Yes' iff $completion(Constraints) \vdash Query$ and 'No' iff $completion(Constraints) \vdash \neg Query$.

Thus, we may choose to interpret the answer from DA in the original sense of FCSDP or under the Closed World Assumption [23] that *Constraints* has been completed. Both interpretations are correct.

6. The Flag FCSP in FOPC

Using the FCSP formalism presented above we can formulate the flag problem as follows.



Query:
$\exists X \exists Y \exists Z \exists U p(X) \wedge q(Y) \wedge s(Z) \wedge t(U) \wedge ne(X, Y) \wedge ne(Y, Z) \wedge ne(Y, U)$

Constraints:
$\{p(r), p(w), q(r), q(w), s(r), s(w), t(r), ne(r, w), ne(w, r)\}$

$H = \{r, w\}$ where r stands for Red and w for White.
$H^4 = \{(r, r, r, r), (r, r, r, w), \ldots, (w, w, w, w)\}$

On the FCSP $(Query, Constraints)$ algorithm DA returns 'Yes' succeeding on the tuple (r, w, r, r).

7. Logical Representation and Reasoning Systems

Faced with a problem in representation and reasoning, a wide spectrum of logical representation systems is available to us. In choosing an appropriate system we can rely on two sets of criteria: descriptive and procedural adequacy criteria [24]. These are often in conflict. The best advice is to use Occam's Razor: choose the simplest system with the level of descriptive adequacy required. Some representation and reasoning systems are shown, organized as a DAG, in Figure 2. If there is an downward arc from system A to system B then A's descriptive capabilities are a strict superset of B's. In the previous section, for example, FCS was shown to be equivalent to theorem-proving in a very restricted form of FOPC. Horn FOPC restricts FOPC in only allowing Horn clauses, with at most one positive literal. Definite Clause Programs (DCP), without predicate completion, restrict Horn FOPC by allowing only one negative clause, the query. Datalog restricts DCP by eliminating function symbols. FCS restricts Datalog by disallowing rules, mixed Horn clauses. There are several further restrictions on FCS possible with corresponding gains in tractability and some generalizations of FCS with gains in expressive power. We shall examine various logical formulations of FCS and investigate some of their interrelationships.

8. FCS as Theorem Proving in Propositional Calculus

The algorithm DA can be interpreted as implementing a view of FCS as theorem proving in the propositional calculus. *Query* is a theorem to be proved.

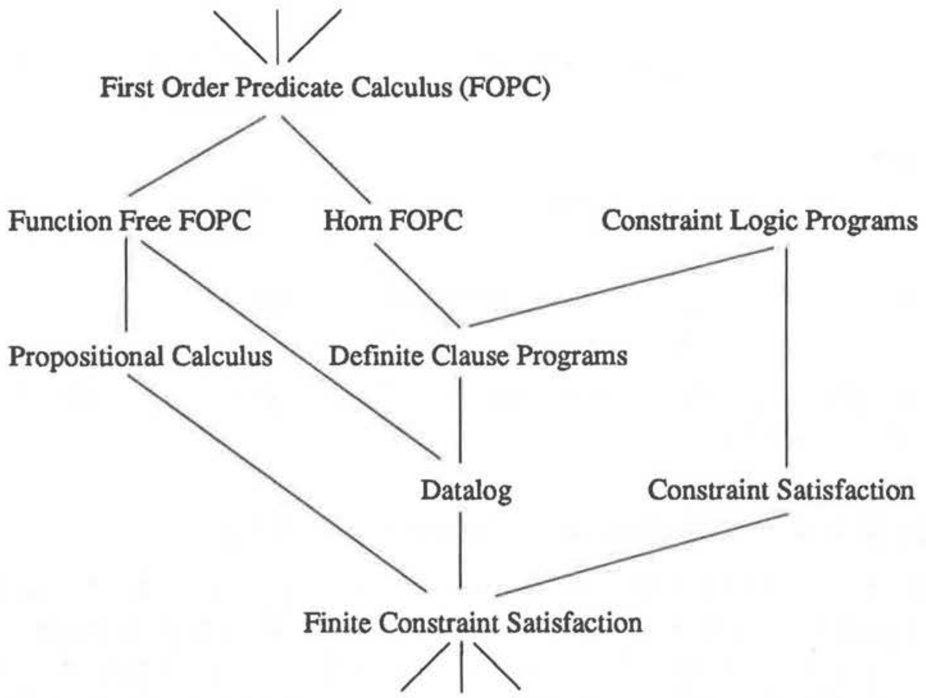


Figure 2. Some Logical Representation and Reasoning Systems

If a solution exists the theory $Constraints \bigcup \neg Query$ leads to a contradiction.

$$\begin{array}{ll} \neg Query: & \neg\exists X_1 \exists X_2 \ldots \exists X_n QMatrix(X_1, \ldots, X_n) \\ & \forall X_1 \forall X_2 \ldots \forall X_n \neg QMatrix(X_1, \ldots, X_n) \end{array}$$

A solution exists iff $Constraints \bigcup \neg Query$ has no (Herbrand) models. There are no universal quantifiers in $Query$ and so there are no existential quantifiers in the theory $Constraints \bigcup \neg Query$. Hence no Skolem functions are introduced when the theory is converted to clausal form. This important restriction guarantees that the Herbrand universe H is finite. This allows us to replace each of the universal quantifiers by the conjunction of the $\neg QMatrix(X_1, X_2, \ldots, X_n)$ clauses instantiated over H^n. This rewritten theory has the same set of Herbrand models as the original theory.

For the flag example the rewritten theory is:

$$\{p(r), q(w), q(r), q(w), s(r), s(w), t(r), ne(r,w), ne(w,r)\}$$
$$\bigcup$$
$$\{\neg p(r), \vee \neg q(r) \vee \neg s(r) \vee \neg t(r) \vee \neg ne(r,r) \vee \neg ne(r,r),$$
$$\neg p(r) \vee \neg q(r) \vee \neg s(r) \vee \neg t(w) \vee \neg ne(r,r) \vee \neg ne(r,w),$$
$$\vdots$$
$$\neg p(r) \vee \neg q(w) \vee \neg s(r) \vee \neg t(r) \vee \neg ne(r,w) \vee \neg ne(w,r), \qquad \star$$
$$\vdots$$
$$\neg p(w) \vee \neg q(w) \vee \neg s(w) \vee \neg t(w) \vee \neg ne(w,w) \vee \neg ne(w,w)\}$$

This theory, now a propositional formula in CNF, has a particular form; it consists only of a set of unit positive clauses, arising from the constraints, and a set of negative clauses, from the query. There are no mixed clauses. Note that it is also always Horn. It is unsatisfiable iff the FCSP has a solution. Since it is Horn SAT is linear time in the size of the formula [8], but, of course, there are $|H|^n$ negative clauses in the formula. Also note unit resolution alone is complete for this class of formulas. For the flag example, repeated unit resolution on the clause marked $\star$ reduces it to the empty clause $\Box$, corresponding to the solution $\{X=r, Y=w, Z=r, U=r\}$. Using subsumption (whereby if clauses of the form C and $C \vee D$ are both present, $C \vee D$ is deleted) does not affect the completeness result. Iterating unit resolution followed by subsumption leaves invariant the special properties of the formula (only negative or unit positive clauses) and, moreover, only decreases its size. Hence, it terminates (correctly). As, of course, does the linear time HornSAT algorithm. The HornSAT algorithm exactly mimics the algorithm DA. The propositional variable in each unit positive literal is set to T and each negative clause is checked: if any clause has each (negative) literal required to be F then the formula is unsatisfiable otherwise it is satisfiable.

9. FCS as Theorem Proving in Definite Theories

The methods discussed so far are not serious candidates for actually solving an FCSP: they simply serve to clarify the semantics and methods of the serious candidates. One such candidate is a Prolog interpreter, which is a theorem prover for theories consisting of definite clauses. Since FCS is pure Prolog SLD-resolution is a sound solution method. SLD-resolution is not, in general, complete for Definite Clause Programs but it is for FCS. It is also, generally speaking, more efficient than other resolution methods such as the one embodied in algorithm DA. For the flag example, we can assert the constraints as ground facts in the Prolog database then define and pose the conjunctive query to Prolog:

```
%prolog
| ?- [user].
| p(r). p(w). q(r). q(w). s(r). s(w). t(r).
| ne(r,w). ne(w,r).
yes
| ?- p(X),q(Y),s(Z),t(U),ne(X,Y),ne(Y,Z),ne(Y,U).
X = Z = U = r,
Y = w
```

In finding the one solution the interpreter essentially checks every possible set of bindings for the variables X, Y, Z and U. By permuting the query one may reduce the size of the search space: a partially completed set of bindings can be rejected by a single failure. For the query

```
| ?- p(X),q(Y),ne(X,Y),s(Z),ne(Y,Z),t(U),ne(Y,U)
```

the search tree is somewhat smaller. Heuristics, such as instantiating the most constrained variable next, can be used to re-order the query dynamically but, on realistic problems, this tactic is doomed. In general, no variable ordering can avoid *thrashing* by repeatedly rediscovering incompatible variable bindings [16].

Just as for the algorithm DA, we may interpret Prolog's failure to find a proof as meaning either $Constraints \nvdash Query$ or $completion(Constraints) \vdash \neg Query$

10. FCS as Datalog

Since FCS is a restriction of Datalog, the techniques developed in the relational database community are available [19]. The solution relation is the natural join of the relations for the individual constraints. The consistency techniques discussed below can be interpreted similarly; for example, making an arc consistent in a constraint network can be interpreted as a semijoin. Results that exploit this interpretation can be found in [21,1,24,6,17,28].

11. FCS in Constraint Networks

Consideration of the drawbacks of the SLD-resolution approach mentioned above leads to a view of FCS in *constraint networks*. A constraint network represents each variable in the query as a vertex. The unary constraint $p_X(X)$ establishes the domain of X, and each binary constraint $p_{XY}(X, Y)$ is represented as the edge (X,Y), composed of arc (X,Y) and arc (Y,X). This easily generalizes to k–ary predicates using hypergraphs. The network for the flag problem is shown in Figure 3.

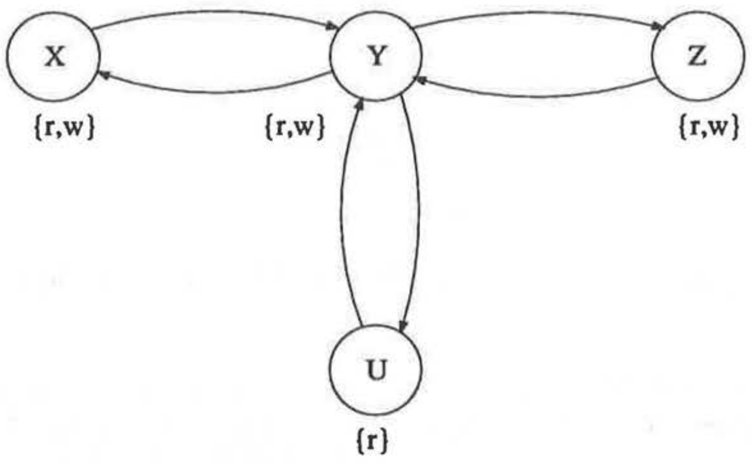


Figure 3. Constraint Network for the Flag Problem

An arc (X,Y) is *consistent* iff

$$\forall X\{p_X(X) \rightarrow \exists Y[p_Y(Y) \wedge p_{XY}(X,Y)]\}$$

A network is arc consistent if all its arcs are consistent. An arc (X,Y) may be made consistent without affecting the total set of solutions by deleting the values from the domain of X that are not consistent with some value in Y. The original flag network is not arc consistent because the single arc (Y,U) is inconsistent. Delete r from the domain of Y. This now makes arcs (X,Y) and

(Z, Y) inconsistent. They can be made consistent by deleting w from the domains of both X and Z, making the network arc consistent as shown in Figure 4.

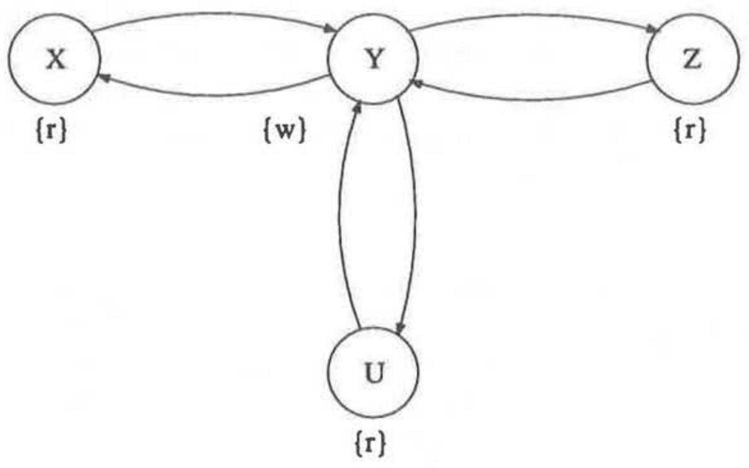


Figure 4. Arc Consistent Network for the Flag Problem

Arc consistency can be enforced in time linear in the number of binary constraints; moreover, if the constraint graph is a tree, as it is for the flag, then arc consistency alone suffices as a decision procedure for the FCSP [18]. Various other graph theoretic properties of the constraint network can be used to characterize and solve FCSPs [22,11,6].

12. Logical Interpreters for FCSP

Using these ideas we can implement an interpreter for FCS [16,27] which could be called LINC (a Logical Interpreter for a Network of Constraints). Given

$Query:$
$\exists X \exists Y \exists Z \exists U p(X) \wedge q(Y) \wedge s(Z) \wedge t(U) \wedge ne(X, Y) \wedge ne(Y, Z) \wedge ne(Y, U)$

then, following [2], for LINC we choose to complete each predicate in $Constraints$ and represent it by its *definition*, its necessary and sufficient conditions, so

$$\begin{aligned} \textit{Constraints}: \quad & p(X) \leftrightarrow ((X{=}r) \lor (X{=}w)) \\ & q(Y) \leftrightarrow ((Y{=}r) \lor (Y{=}w)) \\ & s(Z) \leftrightarrow ((Z{=}r) \lor (Z{=}w)) \\ & t(U) \leftrightarrow (U{=}r) \\ & ne(X,Y) \leftrightarrow (X{=}r \land Y{=}w) \lor (X{=}w \land Y{=}r) \end{aligned}$$

Restricting LINC to arc consistency, it non-deterministically rewrites *Constraints* using the *AC rewrite rule*:

$$p_X(X) \Leftarrow p_X(X) \land \exists Y[p_Y(Y) \land p_{XY}(X,Y)]$$

Here:

$$\begin{aligned} q(Y) &\Leftarrow q(Y) \land \exists U[t(U) \land ne(Y,U)] \\ &\Leftarrow (Y{=}r \lor Y{=}w) \land \exists U[U{=}r \land (Y{=}r \land U{=}w \lor Y{=}w \land U{=}r)] \\ &\Leftarrow (Y{=}w) \end{aligned}$$

Iterating the AC rewrite rule reduces the constraints to a fixpoint

$$\begin{aligned} & p(X) \leftrightarrow (X{=}r) \\ & q(Y) \leftrightarrow (Y{=}w) \\ & s(Z) \leftrightarrow (Z{=}r) \\ & t(U) \leftrightarrow (U{=}r) \\ & ne(X,Y) \leftrightarrow (X{=}\ r \land Y{=}w) \lor (X{=}w \land Y{=}r) \end{aligned}$$

In general, LINC must interleave the AC relaxation with some non-deterministic case analysis or higher order network consistency.

CHIP [27] is, amongst other things, an implementation of this approach. Similarly, a Connection Graph theorem prover for full FOPC, as proposed in [15], essentially performs AC relaxation on the possible sets of substitutions for variables in the literals of each clause. Using such a prover with an SLD-resolution strategy on a FCSP query would produce an effect isomorphic to using LINC.

13. CSP and CLP($\mathfrak{D}$)

The FCS constraint form is a special case of the CLP($\mathfrak{D}$) rule form [12]:

$$\begin{aligned} p(X,Y,\ldots) \leftarrow & a_1(X,Y,\ldots) \land a_2(X,Y,\ldots) \land \ldots \land \\ & p_1(X,Y,\ldots) \land p_2(X,Y,\ldots) \land \ldots \end{aligned}$$

where $a_i(.)$ is a constraint on its arguments and $p_j(.)$ is a predicate.

In Definite Clause Programs $\mathfrak{D} = H$ and the constraints are equalities on terms.

A general Constraint Satisfaction Problem fits the CLP($\mathfrak{D}$) scheme. Consider the CSP represented by this CLP($\mathfrak{R}$) program

$$\begin{aligned} \mathit{Constraints}: \quad & p(X) \leftarrow (1 \leq X) \wedge (X \leq 3) \\ & q(Y) \leftarrow (0 \leq Y) \wedge (Y \leq 2) \\ & r(X,Y) \leftarrow X \leq Y \end{aligned}$$

and the

$$\mathit{Query}: \quad \exists X \exists Y [p(X) \wedge q(Y) \wedge r(X, Y)]$$

Using the same AC rule that LINC used for finite CSP's *Constraints* can be rewritten as

$$\begin{aligned} \mathit{Constraints}: \quad & p(X) \leftarrow (1 \leq X) \wedge (X \leq 2) \\ & q(Y) \leftarrow (1 \leq Y) \wedge (Y \leq 2) \\ & r(X,Y) \leftarrow X \leq Y \end{aligned}$$

This demonstrates that these ideas lift from FCSP to CSP and CLP. It is an open research issue to determine the limits of their applicability and their usefulness [3,26]

14. FCS as Model Finding in Propositional Logic

A radically different logical framework for FCS is as model finding in propositional logic [24,4,14,20]. In [24] an account of depiction is presented. An interpretation of an image is defined to be a logical model of a theory describing the image, the scene and the mapping between them. Under certain assumptions this theory reduces to a propositional theory whose models are identified with possible states of the world. Finding those models corresponds directly to solving an FCSP. In [4,20] the model-finding framework for FCSP is used to elucidate the connection to truth maintenance systems.

In this framework a propositional formula F is constructed for the FCSP such that each model of F corresponds to a solution of the FCSP. Each proposition in F represents a possible binding of a variable to a value. For the flag example, the proposition $X{:}r$ means that variable X takes the value r. F may be in CNF with a set of clauses representing the fact that each variable must take a value, e.g. $X{:}r \vee X{:}w$, the fact that the values are pairwise exclusive, e.g. $\neg X{:}r \vee \neg X{:}w$, and the constraints on related variables. The constraints may be encoded as clauses in any suitable fashion. A 'negative' encoding [4,20] represents only the forbidden tuples of the constraints, e.g. $\neg W{:}r \vee \neg Y{:}r$. Using that encoding for the flag example, we have

$$\begin{aligned} F =\{&X{:}r \vee X{:}w, Y{:}r \vee Y{:}w, Z{:}r \vee Z{:}w, U{:}r,\\ &\neg X{:}r \vee \neg X{:}w, \neg Y{:}r \vee \neg Y{:}w, \neg Z{:}r \vee \neg Z{:}w,\\ &\neg X{:}r \vee \neg Y{:}r, \neg X{:}w \vee \neg Y{:}w, \neg Y{:}r \vee \neg Z{:}r, \neg Y{:}w \vee \neg Z{:}w,\ \neg Y{:}r \vee \neg U{:}r\}\end{aligned}$$

In this framework we have a SAT problem again. In the propositional proof-finding framework the FCSP has a solution iff the formula has no models. Under this model-finding framework each solution corresponds to a model of F. To find the models we could use the Davis-Putnam algorithm. But we note that the SAT problem has the same special form again: there are no mixed clauses in this encoding. This can be exploited to simplify the formula before deciding if there are any models. Two inference rules can be used — a form of negative hyperresolution H_2 and a form of unit resolution U.

$$H_2: \quad \begin{array}{l} p \vee q \vee r \vee \ldots \vee u \\ \neg p \vee \neg v \\ \neg q \vee \neg v \\ \neg r \vee \neg v \\ \vdots \\ \underline{\neg u \vee \neg v \qquad} \\ \neg v \end{array}$$

$$U: \quad \begin{array}{l} p \vee q \vee r \vee \ldots \vee u \\ \underline{\neg q \qquad\qquad} \\ p \vee r \vee \ldots \vee u \end{array}$$

These rules of inference are supplemented with two subsumption rules: S_p and S_n. Given two positive clauses C_1 and C_2 where all the literals in C_1 appear in C_2 then S_p deletes C_2, the subsumed clause. S_n deletes subsumed negative clauses.

Now, we define the *AC-resolution strategy*: $(H_2\, S_n^*\, U\, S_p)^*$

A trace of AC-resolution on the flag example as it simplifies F is shown in Figure 5.

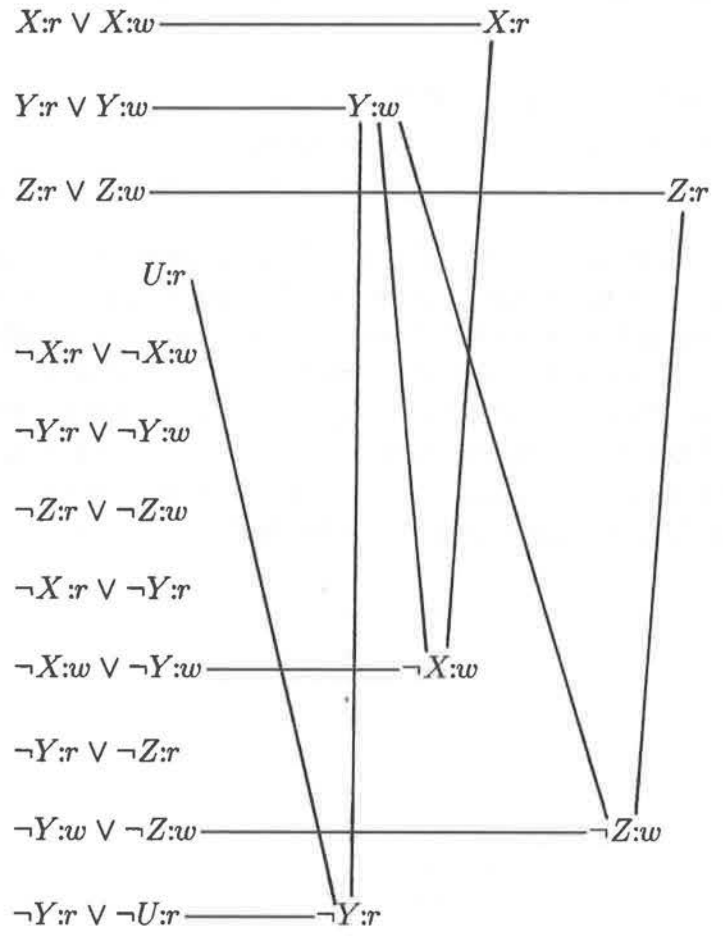


Figure 5. A Trace of AC-resolution on the Flag Problem

The resultant simplified formula is

$$F_s = \{X{:}r, Y{:}w, Z{:}r, U{:}r, \neg X{:}w, \neg Y{:}r, \neg Z{:}w\}$$

which obviously has exactly one model.

Notice that resolution and subsumption are used here in a non-standard way, namely, not to prove a theorem but to find models. It is easy to verify that the AC-resolution strategy has the following properties:

1. The set of models is invariant under AC-resolution. The models of F are the models of F_s. This follows from the Soundness Theorem [9]. $F \models F_s$ (every model of F is a model of F_s) because $F \vdash F_s$, and $F_s \models F$ because $F_s \vdash F$.
2. No mixed clauses are generated so the separation into positive and negative clauses is invariant.
3. The total number and length of clauses decreases monotonically.
4. AC-resolution is $O(e)$ where e is the number of constraints [18].
5. In general, AC-resolution is incomplete in the sense that it does not always terminate with F_s either the empty clause $\square$ or consisting only of unit literals. It must be interleaved with search, such as assigning a truth value to a proposition, or enforcing higher order network consistency, which corresponds to other forms of hyperresolution [4], to determine the models of F explicitly.
6. AC-resolution used for model-finding exactly mimics the behaviour of the LINC interpreter using the AC rewrite rule on FCSP (and the Connection Graph theorem prover) since each proposition, e.g. $X{:}r$, reifies a possible substitution for a variable in the FOPC theorem-proving framework.

The model-finding framework shows the relevance of a variety of serial and parallel complexity results on special cases of SAT, such as planar SAT [25], 2-SAT and HornSAT. If the variables in an FCSP have a maximum domain size of 2 and the constraints are only unary or binary (as is the case for the Flag Problem) then this encoding results in a 2-SAT problem which is $O(e)$ serial time [10] and poly-logarithmic parallel time since it is in NC [13,14].

Other encodings, beside the negative encoding, can be useful. Consider directed constraint networks [7] which are a specialization of FCSP. In a directed constraint network, for each constraint some subset of its variables can be considered as input variables to the constraint with the rest considered as output variables. The projection of the constraint relation on the input variables is the universal relation — that is, they are unconstrained. A constraint relation that is functional on the input variables has this form. A directed constraint that is not functional can be made functional by inventing additional input variables to discriminate between the different output values for the same values of the original input variables. The topology of the constraint network must respect this distinction between input and output variables.

A suitable mixed encoding of a directed constraint network in the model-finding framework can be arranged as a propositional theory, as follows. The theory has a positive clause for each variable and a set of negative clauses for each variable, as before, but it has definite clauses for all the multivariable constraints. If the values for the input variables to the network are known, the theory essentially collapses to become Horn as a modified HornSAT algorithm determines the entire state of the network in $O(e)$ time [8]. On the other hand, determining the input values required to produce a given output can be much harder. Diagnosis of the internal state of a causal system can also be put in this framework: the unknown states of the components constitute additional input variables [5].

15. Conclusions

In summary, all of the questions posed in Section 1 of this paper have been answered affirmatively except, of course, the one posed by the cynical critic. The basic approach has been to see Finite Constraint Satisfaction as a restricted logical calculus in a space of logical representation and reasoning systems. The FCSP framework has been formulated in a variety of logical settings: theorem proving in FOPC (which reduces to propositional theorem proving and hence SAT), the Prolog and Datalog approaches, constraint network algorithms, a logical interpreter for networks of constraints, the CLP paradigm and forms of propositional model finding (and hence SAT, again). Several standard, and some not-so-standard, logical methods can therefore be used to solve these problems. By doing this we obtain a specification of the semantics of the common approaches.

This synthetic treatment also allows algorithms and results from many of these disparate areas to be imported, and specialized, to FCSP; the special properties of FCSP are exploited to achieve, for example, completeness and to improve efficiency. It also allows export to the related areas. By casting CSP both as a generalization of FCSP and a specialization of CLP it was observed that some, but not all, FCSP techniques lift to CSP and, perhaps, thereby to CLP. Various new connections have been uncovered, in particular between the proof-finding approaches and the alternative model-finding approaches that have arisen in depiction and diagnosis applications.

Acknowledgments

The contents of this paper benefitted from discussions with Johan de Kleer, Alex Kean, Nick Pippenger, David Poole, Ray Reiter, Ron Rensink and Zhang Ying. This research was supported by the Canadian Institute for Advanced Research, the Natural Sciences and Engineering Research Council of Canada and the Institute for Robotics and Intelligent Systems Network of Centres of Excellence.

References

[1] Bibel, W. Constraint satisfaction from a deductive viewpoint. *Artificial Intelligence 35* (1988), 401–413.

[2] Clark, K. L. Negation as failure. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum Press, New York, 1978, pp. 293–322.

[3] Davis, E. Constraint propagation with interval labels. *Artificial Intelligence 32* (1987), 281–331.

[4] de Kleer, J. A comparison of ATMS and CSP techniques. In *IJCAI-89* (Detroit, Michigan, Aug. 1989), IJCAI, pp. 290–296.

[5] de Kleer, J., Mackworth, A. K., and Reiter, R. Characterizing diagnoses. In *Proc. 8th National Conference on Artificial Intelligence* (Boston, July 1990), pp. 324–330.

[6] Dechter, R. Constraint networks: A survey. In *The Encyclopedia of AI*, S. Shapiro, Ed., second ed. John Wiley, New York, 1991.

[7] Dechter, R., and Pearl, J. Directed constraint networks: A relational framework for causal modeling. Tech. Rep. R-153, UCLA, Los Angeles, California, Oct. 1990.

[8] Dowling, W. F., and Gallier, J. H. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming 3* (1984), 267–284.

[9] Enderton, H. B. *A Mathematical Introduction to Logic.* Academic Press, Orlando, Florida, 1972.

[10] Even, S., Itai, A., and Shamir, A. On the complexity of timetable and multicommodity flow problems. *SIAMJournal of Computing 5*, 4 (1976), 691–703.

[11] Freuder, E. C. Complexity of k-tree structured constraint satisfaction problems. In *Proc. 8th National Conference on Artificial Intelligence* (Cambridge, Massachusetts, July 1990), pp. 4–9.

[12] Jaffar, J., and Lassez, J. L. Constraint logic programming. In *Proc. 14th ACM Principles of Programming Languages Conf.* (Munich, 1987), pp. 111–119.

[13] Jones, N. D., Lien, Y. E., and Laaser, W. T. New problems complete for nondeterministic log space. *Mathematical Systems Theory 10* (1976), 1–17.

[14] Kasif, S. Parallel solutions to constraint satisfaction problems. In *Proc. First International Conf. on Principles of Knowledge Representation and Reasoning* (Toronto, May 1989), pp. 180–188.

[15] Kowalski, R. A proof procedure using connection graphs. *JACM 22*, 4 (Oct. 1975), 572–595.

[16] Mackworth, A. K. Consistency in networks of relations. *Artificial Intelligence 8* (1977), 99–118.

[17] Mackworth, A. K. Constraint satisfaction. In *The Encyclopedia of AI*, second ed. John Wiley, New York, 1991.

[18] Mackworth, A. K., and Freuder, E. C. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence 25* (1985), 65–74.

[19] Maier, D. *The Theory of Relational Databases*. Computer Science Press, Rockville, Maryland, 1983.

[20] McAllester, D. Truth maintenance. In *Proc. 8th National Conference on Artificial Intelligence* (Boston, July 1990), pp. 1109–1116.

[21] Montanari, U. Networks of constraints: fundamental properties and applications to picture processing. *Information Sciences 7* (1974), 95–132.

[22] Montanari, U., and Rossi, F. Constraint relaxation may be perfect. Tech. Rep. TR-21/89, University of Pisa, Pisa, Italy, 1989.

[23] Reiter, R. Nonmonotonic reasoning. In *Exploring Artificial Intelligence*, H. E. Shrobe, Ed. Morgan Kaufmann, San Mateo, Ca, 1988, pp. 439–482.

[24] Reiter, R., and Mackworth, A. K. A logical framework for depiction and image interpretation. *Artificial Intelligence 41* (1989), 125–155.

[25] Seidel, R. A new method for solving constraint satisfaction problems. In *Proc. 7th International Joint Conf. on Artificial Intelligence* (Vancouver, BC, 1981), pp. 338–342.

[26] Sidebottom, G., and Havens, W. S. Hierarchical arc consistency applied to numeric processing in constraint logic programming. Tech. Rep. CSS-IS TR 91-06, Simon Fraser University, Burnaby, B. C., 1991.

[27] van Hentenryck, P. *Consistency techniques in logic programming*. Thesis, University of Notre-dame, Namur-Belgium, 1987.

[28] Zhang, Y., and Mackworth, A. K. Parallel and distributed algorithms for constraint satisfaction problems. In *Proc. 3rd IEEE Symposium on Parallel and Distributed Processing* (Dallas, Texas, Dec. 1991).