Searching for Solutions

Lambert Meertens*
Department of Algorithmics and Architecture, CWI, Amsterdam, and
Department of Computing Science, Utrecht University, The Netherlands
`http://www.cwi.nl/cwi/people/Lambert.Meertens.html`

Printed May 8, 1998

1 Constraints

A *constraint* over a set of variables V is a propositional sentence in some logic, with free variables drawn from a 'sufficiently large' set V. For the sake of simplicity we will assume that all variables have type $\mathbb{N}$. For example. if $V = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$, a constraint might be

$$\mathsf{x} + 1 \leq \mathsf{y} \ \wedge \ \mathsf{x} + \mathsf{y} \leq 9$$

An *assignment* on V is a function of type $A = V \to \mathbb{N}$. For example, the function $\{\mathsf{x} \mapsto 3, \mathsf{y} \mapsto 5, \mathsf{z} \mapsto 0\}$ is an assignment on $\{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$.

The meaning (semantics) of a constraint is a function of type $A \to \mathbb{B}$, that is, a predicate on the set of assignments.

We blur the distinction between constraints and predicates. More precisely, when C is a constraint, we also use C to denote the predicate on A it represents (in a context where a predicate is required) — in other words,

*This work was performed while visiting Kestrel Institute, Palo Alto.

the meaning mapping is treated like a type coercion. For example, if the symbols used have their usual interpretation, the assignment above satisfies the constraint above.

We assume that the formalism is sufficiently rich that we can express any constraint we need to express. In particular, we assume the usual constants and connectives of propositional logic, with the usual semantics:

$$\begin{array}{lcl}
\mathbf{true}(a) & \equiv & \text{true} \\
\mathbf{false}(a) & \equiv & \text{false} \\
(C \;\boldsymbol{\wedge}\; D)(a) & \equiv & C(a) \wedge D(a) \\
(C \;\boldsymbol{\vee}\; D)(a) & \equiv & C(a) \vee D(a) \\
(C \boldsymbol{\Rightarrow} D)(a) & \equiv & C(a) \Rightarrow D(a) \\
(C \boldsymbol{\equiv} D)(a) & \equiv & C(a) \equiv D(a)
\end{array}$$

Here, the connectives in the left-hand sides are constructors in the formalism used for building sentences, while in the right-hand sides they stand for the usual mathematical operations on the set $\mathbb{B}$ of truth values. (To belabour the point: we might have written something like "$\mathbf{(and}\ C\ D\mathbf{)}$" on the left-hand side.)

The notation $\models C$ means that all asignments satisfy C, or $\forall(a :: C(a))$. We say then that C is *universally valid*. Furthermore, $C \models C'$ stands for

$$\forall(a :: C(a) \Rightarrow C'(a))$$

which may also be written as $\models C \boldsymbol{\Rightarrow} C'$. Two constraints C and C' are *equivalent*, denoted by $C \simeq C'$, when $C \models C'$ and $C' \models C$. This means that they have the same meaning:

$$\forall(a :: C(a) \equiv C'(a))$$

since:

$$\begin{array}{ll}
 & C \simeq C' \\
\equiv & \quad \{\text{definition of } \simeq\} \\
 & C \models C' \;\wedge\; C' \models C \\
\equiv & \quad \{\text{definition of } \models\} \\
 & \forall(a :: C(a) \Rightarrow C'(a)) \;\wedge\; \forall(a :: C'(a) \Rightarrow C(a))
\end{array}$$

$$
\begin{array}{ll}
\equiv & \{\text{predicate calculus}\} \\
& \forall(a :: C(a) \equiv C'(a)) \\
\equiv & \{\text{definition of } \models\} \\
& \models C \boldsymbol{\equiv} C'
\end{array}
$$

Given the semantics of the connectives, we can use all the laws of conventional propositional calculus in a proof that $C \simeq C'$.

Let the set of variables U be a subset of V. Two assignments a and a' *agree* on U, denoted $a \approx_U a'$, whenever

$$\forall(u : u \in U : a(u) = a'(u))$$

If U is the set of free variables in C, the value of $C(a)$ only depends on the assignment a restricted to U, so then

$$\forall(a, a' : a \approx_U a' : C(a) \equiv C(a'))$$

Constraint C is a *projection* of constraint C' on U, denoted $C \succeq_U C'$, whenever

$$\forall(a :: C(a) \;\equiv\; \exists(a' : a' \approx_U a : C'(a')))$$

For example, the constraint $\mathbf{2 \leq z \leq 18}$ is a projection of the constraint $\mathbf{x+1 \leq y} \;\wedge\; \mathbf{x+y \leq 9} \;\wedge\; \mathbf{z = x+2{*}y}$ on $\{\mathbf{z}\}$. Clearly, the projections of a given constraint on U are all equivalent. Moreover, if $C \succeq_U C'$, then $C' \models C$:

$$
\begin{array}{ll}
& C' \models C \\
\equiv & \{\text{definition of } \models\} \\
& \forall(a :: C'(a) \Rightarrow C(a)) \\
\Leftarrow & \{\text{since } \approx_U \text{ is reflexive, } C'(a) \;\Rightarrow\; \exists(a' : a' \approx_U a : C'(a'))\} \\
& \forall(a :: \exists(a' : a' \approx_U a : C'(a')) \Rightarrow C(a)) \\
\Leftarrow & \{\text{propositional calculus}\} \\
& \forall(a :: C(a) \;\equiv\; \exists(a' : a' \approx_U a : C'(a'))) \\
\equiv & \{\text{definition of } \succeq\} \\
& C \succeq_U C'
\end{array}
$$

2 Solutions

There is a class of constraints called *displays*. We will not formalize the notion, but the informal idea is that a display is a constraint that is in such a form that the assignments satisfying it can be 'read off' directly. For example,

$$\mathsf{x} = \mathsf{3} \;\wedge\; \mathsf{4} \leq \mathsf{y} \leq \mathsf{6}$$

might be a display.

We assume that the set of displays is closed under semantic conjunction, disjunction, and projection. That is, for all displays D and D', and all subsets U of V, there exist displays D_c, D_d and D_p such that

$$\begin{aligned} D_c &\simeq D \wedge D' \\ D_d &\simeq D \vee D' \\ D_p &\succeq_U D \end{aligned}$$

We denote such a D_c, D_d and D_p by, respectively, $D \wedge D'$, $D \vee D'$ and $\pi_U(D)$, since we'll never be interested in the syntactic specifics of the sentence picked from the equivalence class.

Constraint S is a *solution* of constraint C whenever S is a display such that $S \models C$. For example, the display above is a solution of

$$\mathsf{x} + \mathsf{1} \leq \mathsf{y} \;\wedge\; \mathsf{x} + \mathsf{y} \leq \mathsf{9}$$

We call a solution S of C a *full* solution whenever $S \simeq C$. The display **false** is a solution of any constraint, but not a very interesting one. We'll call it (and any equivalent displays) an *empty* solution.

3 Searching

A *problem* is a pair $D \& C$, in which D is a display and C a constraint. (The symbol '&' here is just a constructor.)

By definition, the solutions to problem $D \& C$ are the solutions to the constraint $D \wedge C$. In other words, from a semantic point of view & and $\wedge$ are synonymous. The role of the two components, however, is different. Think of D as a *search space* in which the solution of C is sought for.

Initially, constraint C can be modelled as the problem $\mathbf{true}\,\&\,C$, which will be the *root* of a search process.

We extend the domain of the relations $\simeq$ and $\succeq_U$ so as to include problems, by coercing problems to the equivalent constraints. So $D\,\&\,C \;\simeq\; C'$ iff $D \wedge C \;\simeq\; C'$, etcetera.

We concentrate on finding full solutions; the adaptations for finding just any non-empty solution should be fairly obvious.

What we do next is give a framework for a 'non-deterministic' recursive searching procedure, by discussing various possible solution steps which might lead to the solution if supplemented by an appropriate strategy that determines when to take what kind of step. What constitutes a good strategy depends, of course, on the specifics of the constraint formalism we have abstracted from. Questions like whether there exists a decision procedure, and if so whether some strategy leads to an algorithm (an always terminating search procedure), are outside the scope of this investigation.

3.1 Simplification

If in the problem $D\,\&\,C$ one of the components D or C (or both) can be simplified to an equivalent component, solve instead the problem using the simplified component(s).

3.2 Trivial problems

A problem of the form $D\,\&\,\mathbf{true}$ is *trivial*: a full solution is D:

$$
\begin{array}{ll}
 & D \;\simeq\; D\,\&\,\mathbf{true} \\
\equiv & \quad \{\text{definition of } \simeq\}
\end{array}
$$

$$
\begin{aligned}
& \models D \ \equiv\ D \wedge \mathbf{true} \\
\equiv\ & \quad \{\text{propositional calculus}\} \\
& \text{true}
\end{aligned}
$$

Other trivially solvable problems are special cases of steps treated below.

3.3 Introducing new variables

Let U be the set of free variables in problem $D\&C$. If $D\&C \succeq_U D\&C'$, and S is a full solution of $D\&C'$, then $\pi_U(S)$ is a full solution of $D\&C$:

$$
\begin{aligned}
& \pi_U(S) \ \simeq\ D\&C \\
\equiv\ & \quad \{\text{definitions of } \simeq \text{ and } \pi_U\} \\
& D\&C \succeq_U S \\
\equiv\ & \quad \{S \text{ is a full solution of } D\&C'\} \\
& D\&C \succeq_U D\&C'
\end{aligned}
$$

The assumption here is that C' involves additional variables (otherwise C and C' are equivalent and the step would be pointless), which are introduced in order to bring the constraint into a more convenient form (for example, removing existential quantifiers).

From the definition of $\succeq_U$ it is easy to see that we don't loose solutions by the step, assuming sufficient expressiveness.

3.4 Propagation

If under assumption D, constraint C can be strengthened to C', we may replace problem $D\&C$ by $D\&C'$. More formally, if $D \models C \equiv C'$, then $D\&C$ is equivalent to $D\&C'$:

$$D\&C \ \simeq\ D\&C'$$

$$\equiv \quad \{\text{definition of } \simeq\}$$

$$\models D \wedge C \;\equiv\; D \wedge C'$$

$$\equiv \quad \{\text{propositional calculus}\}$$

$$\models D \Rightarrow (C \;\equiv\; C')$$

$$\equiv \quad \{\text{property of } \models\}$$

$$D \models C \;\equiv\; C'$$

3.5 Cutting

If, given problem $D \& C$, there exists a display D' such that $D \wedge C \models D'$, then $D \& C$ may be replaced by $(D \wedge D') \,\&\, C$:

$$D \& C \;\simeq\; (D \wedge D') \,\&\, C$$

$$\equiv \quad \{\text{definition of } \simeq\}$$

$$\models D \wedge C \;\equiv\; D \wedge D' \wedge C$$

$$\equiv \quad \{\text{propositional calculus}\}$$

$$\models D \wedge C \;\Rightarrow\; D'$$

$$\equiv \quad \{\text{definition of } \models\text{, semantics}\}$$

$$\forall (a :: (D \wedge C)(a) \;\Rightarrow\; D'(a))$$

$$\equiv \quad \{\text{definition of } \models\}$$

$$D \wedge C \models D'$$

An additional requirement that $D \not\models D'$ serves to ensure that the search space is effectively reduced.

3.6 Solving by cases

If $D \simeq D_1 \vee D_2 \vee \cdots \vee D_n$, to solve $D \& C$ we may find a full solution S_i of each case $D_i \& C$, giving a full solution $S_1 \vee S_2 \vee \cdots \vee S_n$. This is an immediate consequence of the fact that conjunction distributes over disjunction.

Likewise, if $D \models C \equiv C_1 \vee C_2 \vee \cdots \vee C_n$, we may solve each subproblem $D \& C_i$ giving solution S_i and combine the solutions as before.

A special case is $n = 0$, which arises when D or C is equivalent to **false**. Then any full solution of the whole problem is empty, so we may return **false**.

4 Tactics

We show that one particular solution step, *piecemeal* solving, is in fact a tactic obtained by appropriately combining solution steps introduced before.

The step is as follows. To solve $D \& (C \wedge C')$, first find a full solution S of $D \& C$, and then a full solution of $S \& C'$. This can obviously be extended to n-ary conjunctions.

Here is how this arises by combining three basic steps of the previous section:

$$
\begin{array}{ll}
 & D \& (C \wedge C') \\
\simeq & \quad \{D \& (C \wedge C') \models D \& C \models S, \text{ cutting}\} \\
 & (D \wedge S) \& (C \wedge C') \\
\simeq & \quad \{S \models D \& C \models D, \text{ simplification}\} \\
 & S \& (C \wedge C') \\
\simeq & \quad \{S \models D \& C \models C, \text{ so } S \models (C \wedge C') \equiv C', \text{ propagation}\} \\
 & S \& C'
\end{array}
$$