How to Solve Set Constraints

Jens Palsberg

October 1, 2008

We will explain how to solve a particular class of set constraints in cubic time.

1 Set Constraints

Let $\mathcal{C}$ be a finite set of constants, and let $\mathcal{V}$ be a set of variables. A set constraint is a conjunction of constraints of the forms:

$$
\begin{aligned}
&c \in v \\
&c \in v \Rightarrow v' \subseteq v''
\end{aligned}
$$

where $c \in \mathcal{C}$ and $v, v', v'' \in \mathcal{V}$. For a particular set constraint, we use C to denote the finite set of constants that occur in that set constraint, and we use V to denote the finite set of variables that occur in that set constraint.

We use 2^C to denote the powerset of C. A set constraint has solution $\varphi : V \to 2^C$ if

- for each conjunct of the form $c \in v$, we have $c \in \varphi(v)$ and
- for each conjunct of the form $c \in v \Rightarrow v' \subseteq v''$, we have $c \in \varphi(v) \Rightarrow \varphi(v') \subseteq \varphi(v'')$.

We say that a set constraint is satisfiable if it has a solution.

Theorem 1 *Every set constraint is satisfiable.*

Proof. The mapping $\lambda v : V.C$ is a solution of the set constraint. □

For two mappings $\varphi, \psi : V \to 2^C$, we define the binary intersection $\varphi \sqcap \psi$ as:

$$\varphi \sqcap \psi = \lambda v : V.(\varphi(v) \cap \psi(v))$$

Theorem 2 *For a given set constraint, the binary intersection of two solutions is itself a solution.*

Proof. Suppose $\varphi, \psi : V \to 2^C$ are both solutions. Let us examine each of the conjuncts of the set constraint.

- For a conjunct of the form $c \in v$, we have from φ, ψ being solutions that $c \in \varphi(v)$ and $c \in \psi(v)$. From $c \in \varphi(v)$ and $c \in \psi(v)$ we have $c \in (\varphi(v) \cap \psi(v))$ which is equivalent to $c \in (\varphi \sqcap \psi)(v)$.

- for each conjunct of the form $c \in v \Rightarrow v' \subseteq v''$, we have from φ, ψ being solutions that $c \in \varphi(v) \Rightarrow \varphi(v') \subseteq \varphi(v'')$ and $c \in \psi(v) \Rightarrow \psi(v') \subseteq \psi(v'')$. We want to show $c \in (\varphi \sqcap \psi)(v) \Rightarrow (\varphi \sqcap \psi)(v') \subseteq (\varphi \sqcap \psi)(v'')$. Suppose $c \in (\varphi \sqcap \psi)(v)$. From $c \in (\varphi \sqcap \psi)(v)$ and the definition of $\sqcap$, we have $c \in (\varphi(v) \cap \psi(v))$, so we have $c \in \varphi(v)$ and $c \in \psi(v)$. From $c \in \varphi(v)$ and $c \in \varphi(v) \Rightarrow \varphi(v') \subseteq \varphi(v'')$, we have $\varphi(v') \subseteq \varphi(v'')$. From $c \in \psi(v)$ and $c \in \psi(v) \Rightarrow \psi(v') \subseteq \psi(v'')$, we have $\psi(v') \subseteq \psi(v'')$. From $\varphi(v') \subseteq \varphi(v'')$ and $\psi(v') \subseteq \psi(v'')$, we have $(\varphi(v') \cap \psi(v')) \subseteq (\varphi(v'') \cap \psi(v''))$, which is equivalent to $(\varphi \sqcap \psi)(v') \subseteq (\varphi \sqcap \psi)(v'')$, and that is the desired result.

□

For the space $V \to 2^C$, we define an ordering $\subseteq$ as follows. We say that $\varphi \subseteq \psi$ if and only if for all $v \in V : \varphi(v) \subseteq \psi(v)$. For a set $S \subseteq (V \to 2^C)$, we say that an element $\varphi \in S$ is the $\subseteq$-least element of S if for all $\psi \in S : \varphi \subseteq \psi$.

Theorem 3 *Every set constraint has a $\subseteq$-least solution.*

Proof. Let a particular set constraint be given. The space of possible solutions of the set constraint is $V \to 2^C$, which is a finite set. Let $S \subseteq (V \to 2^C)$ be the set of solutions of the set constraint. From $V \to 2^C$ being finite, we have that S is finite. From Lemma 1 we have that S is nonempty. So, S is a nonempty, finite set. Let $S = \{ \varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_n \}$. Let $\varphi = (\ldots((\varphi_1 \sqcap \varphi_2) \sqcap \varphi_3) \ldots \sqcap \varphi_n)$. From Lemma 2 we have that φ is a solution of the set constraint, so $\varphi \in S$. Additionally, we have that for all $i \in 1..n : \varphi \subseteq \varphi_i$. So, φ is the $\subseteq$-least solution of the set constraint. □

2 Solving Set Constraint Efficiently

We will translate the problem of solving a set constraint into a graph problem. For a given set constraint, the initial graph has one node for each element of V, and an empty set of edges. Additionally, each node v is associated with a bit vector B_v of length $|C|$ in which every bit is initially 0. For a bit vector B_v and a constant $c \in C$, we denote the entry for c in B_v by $B_v[c]$. Finally, every bit in every bit vector is associated with a list $L_v[c]$ of constraints of the form $v' \subseteq v''$; all those lists are initially empty.

The initial graph represents the mapping $\lambda v : V.\emptyset$. The idea is that for a given $v \in V$, the bit vector B_v represents a subset of C. When all the bits are 0, the bit vector B_v represents the empty set. An edge in the graph implies a subset relationship. If the graph has an edge from v to v', it implies that the set represented by the bit vector associated with v is a subset of the set represented by the bit vector associated with v'.

We now process each conjunct of the set constraint in turn. The processing will add edges, change bits from 0 to 1, and add constraints to the constraint lists associated with bits in the bit vectors. We will use the following two procedures propagate and insert-edge as key subroutines.

```
procedure propagate(v:Node, c:Constant) {
    if (B_v[c] == 0) {
        B_v[c] = 1
        for (each element (v' ⊆ v'') of L_v[c]) { insert-edge(v', v'') }
        for (each edge (v, v')) { propagate(v', c) }
    }
}

procedure insert-edge(v, v':Node) {
    insert an edge (v, v')
    for (c such that B_v[c] == 1) { propagate(v', c) }
}
```

For a constraint of the form $c \in v$, we execute propagate(v,c). For a constraint of the form $c \in v \Rightarrow v' \subseteq v''$, we execute:

```
if (B_v[c] == 0) { add the constraint (v' ⊆ v'') to the list L_v[c] }
else             { insert-edge(v', v'') }
```

When we have processed all the constraints, the resulting graph represents the $\subseteq$-least solution of the set constraint.

We will now analyze the time complexity of the constraint processing. We will do the analysis for a set constraint with $|C| = O(n)$, $|V| = O(n)$, $O(n)$ conjuncts of the form $c \in v$, and $O(n^2)$ conjuncts of the form $c \in v \Rightarrow v' \subseteq v''$. For each conjunct, the algorithm performs a constant amount of work except for the calls to the propagate subroutine. So, the total time is $O(n^2)$ plus the time spent by propagate. The propagate routine itself performs a constant amount of work except for recursive calls! So the key to analyzing the time spent by propagate is to determine the *number of calls* to the propagate routine. The processing of the set constraint generates $O(n^3)$ immediate calls to propagate. The recursion involved in each call to propagate stops when finding a bit that is 1. So, for each $c \in C$, the total work of all calls of the form propagate(v,c) is given by the number of edges in the graph, which is $O(n^2)$. To sum up, the total time is $O(n^2) + (O(n) \times O(n^2)) = O(n^3)$.