1 A Required Subformula

We describe how to add constraints to the model in order to guarantee that a particular subformula appears somewhere in the program. In this section, a *subformula* is a formula of the form

$$\bigodot_{i=1}^{n} \mathsf{p}_i, \tag{1}$$

where $\odot \in \{\wedge, \vee\}$, and $(\mathsf{p}_i)_{i=1}^n$ is a finite sequence of predicates, arguments of which are immaterial. Let Subformula (1) be precisely the required subformula.

We say that a clause has Subformula (1) if we can find indices $i_0, i_1, \dots, i_n \in \{0, \dots, \mathcal{M}_\mathcal{N} - 1\}$ such that

$$\texttt{values}[i_0] = \odot,$$

and, for each $j = 1, \dots, n$,

$$\texttt{structure}[i_j] = i_0,$$

and

$$\texttt{values}[i_j] = \mathsf{p}_j.$$

We use this idea to define several data structures for each clause, gradually building up to a Boolean indicator for whether the clause has Subformula (1). First, let $\mathbf{A}$ be an $\mathcal{M}_\mathcal{N} \times \mathcal{M}_\mathcal{N}$ Boolean matrix defined as

$$A_{i,j} \coloneqq (\texttt{values}[i] = \odot) \wedge (\texttt{structure}[j] = i),$$

i.e., the (i, j)-th element is true if node i is the root node of the subformula, and node j is its child. Next, let $\mathbf{B}$ be an $\mathcal{M}_\mathcal{N} \times n$ Boolean matrix defined as

$$B_{i,j} \coloneqq (\texttt{values}[i] = \mathsf{p}_j),$$

where the (i, j)-th element indicates whether node i contains predicate p_j. We can then use $\mathbf{A}$ and $\mathbf{B}$ to define an $\mathcal{M}_\mathcal{N} \times n$ Boolean matrix $\mathbf{C}$ where the (i, j)-th element shows whether node i is a parent of a node that contains predicate p_j. We then have that $\mathbf{C} \coloneqq \mathbf{AB}$, i.e.,

$$C_{i,j} \coloneqq \bigvee_k A_{i,k} \wedge B_{k,j}.$$

The Boolean variable f_i that indicates whether the i-th clause has the required subformula (for $i = 0, \dots, \mathcal{M}_\mathcal{C} - 1$) can then be expressed as

$$f_i \coloneqq \bigvee_j \bigwedge_k C_{j,k},$$

i.e., the clause has Subformula (1) if there is a suitable parent node that has all n predicates as its children. Note that this allows for the parent node to have more than n children, and the order of children is inconsequential. Finally, we can require the program to have the subformula simply by asking for

$$\bigvee_i f_i$$

to be satisfied.