Reasoning about directed graphs using separation logic

John Wickerson

8th January 2010

Consider a datastructure that contains a set of nodes X. Each node $x \in X$ comprises three fields: a value $x.\mathtt{val}$ and two (possibly null) pointers $x.\mathtt{next1}$ and $x.\mathtt{next2}$ to other nodes.

1 Trees and lists

Suppose there are no cycles, and that there is a single root node r from which all nodes can be reached. Suppose further that `next1` and `next2` always identify disjoint graphs. Then the structure would be a *tree*, and we could describe it as $\mathit{tree}(r)$, where *tree* is the strongest predicate that satisfies:

$$\begin{aligned}\mathit{tree}(x) \Leftrightarrow x = 0 \wedge \mathsf{emp} \vee \\ x \neq 0 \wedge \exists y, z.\, x.\mathtt{val} \mapsto \_ * x.\mathtt{next1} \mapsto y * \mathit{tree}(y) \\ * x.\mathtt{next2} \mapsto z * \mathit{tree}(z)\end{aligned}$$

Now suppose again there are no cycles, and that r is again the root. Suppose further that `next2` is always null. Then the structure would be a list, and we could describe it as $\mathit{list}(r)$, where *list* is the strongest predicate that satisfies:

$$\mathit{list}(x) \Leftrightarrow x = 0 \wedge \mathsf{emp} \vee x \neq 0 \wedge \exists y.\, x.\mathtt{val} \mapsto \_ * x.\mathtt{next1} \mapsto y * \mathit{list}(y)$$

In both of these cases we are able to find an *inductively-defined* predicate to describe the structure. However, we cannot describe in this way structures that contain cycles or node-sharing, because the separating conjunction that we use enforces that the subcomponents be disjoint.

2 Dags

Matt suggests describing a dag rooted at r as $\mathit{dag}(r)$, where dag is the strongest predicate that satisfies:

$$\begin{aligned}\mathit{dag}(x) \Leftrightarrow\ & x = 0 \wedge \mathtt{emp}\ \vee \\ & x \neq 0 \wedge \exists y, z, P.\, x.\mathtt{val} \mapsto \_ * x.\mathtt{next1} \mapsto y * (P \mathrel{-\!\!*} \mathit{dag}(y)) \\ & \qquad * x.\mathtt{next2} \mapsto z * (P \mathrel{-\!\!*} \mathit{dag}(z)) \\ & \qquad * P\end{aligned}$$

Here, P represents the part of the graph that is shared between the two children. Note that the predicate is, pleasingly, precise: it describes only the heap cells of the dag.

3 Directed graphs

We can describe our most general graph structure as $\mathit{graph}(X)$, where X is the set of nodes (defined above) and graph is defined like so:

$$\mathit{graph}(X) \stackrel{\text{def}}{=} \forall^* x \in X.\, \mathit{node}_X(x)$$

There's no particular ordering between the nodes in X. We just use the iterated star ($\forall^*$) to iterate over all of them, and say that each one is a *node*. We define node_X like so:

$$\begin{aligned}\mathit{node}_X(x) \stackrel{\text{def}}{=} \exists y, z.\, x.\mathtt{val} \mapsto \_ & * (x.\mathtt{next1} \mapsto y \wedge y \in X_\perp) \\ & * (x.\mathtt{next2} \mapsto z \wedge z \in X_\perp)\end{aligned}$$

where $X_\perp \stackrel{\text{def}}{=} X \cup \{0\}$. Note that the *node* predicate, unlike the others, is not inductively defined. This is because the datastructure is not inductive either.

Even though the predicate is not inductive, we can still reason using the predicate perfectly nicely. Note the following lemmas:

Lemma 1. If $X \subseteq X'$ then $\mathit{node}_X(x) \Rightarrow \mathit{node}_{X'}(x)$.

Lemma 2 (Merging two (disjoint) graphs)**.**

$$\mathit{graph}(X) * \mathit{graph}(Y) \Rightarrow \mathit{graph}(X \uplus Y)$$

Proof.

$$\begin{aligned}
\mathit{graph}(X) * \mathit{graph}(Y) &\Leftrightarrow \forall^* x \in X.\, \mathit{node}_X(x) * \forall^* x \in Y.\, \mathit{node}_Y(x) \\
&\Rightarrow \forall^* x \in X.\, \mathit{node}_{X \uplus Y}(x) * \forall^* x \in Y.\, \mathit{node}_{X \uplus Y}(x) \quad \text{(by Lemma 1)} \\
&\Leftrightarrow \forall^* x \in X \uplus Y.\, \mathit{node}_{X \uplus Y}(x) \\
&\Leftrightarrow \mathit{graph}(X \uplus Y)
\end{aligned}$$

Note that the converse doesn't hold because the RHS allows edges between nodes in X and nodes in Y, but the LHS doesn't. □

The next lemma is for the case when we have a graph and an external node (that points into the graph) that we wish to add in.

Lemma 3 (Adding a new node)**.** If $y \notin X$ then

$$\mathit{graph}(X) * \mathit{node}_X(y) \Rightarrow \mathit{graph}(X \uplus \{y\})$$

Proof.

$$\begin{aligned}
\mathit{graph}(X) * \mathit{node}_X(y) &\Leftrightarrow (\forall^* x \in X.\, \mathit{node}_X(x)) * \mathit{node}_X(y) \\
&\Rightarrow (\forall^* x \in X.\, \mathit{node}_{X \uplus \{y\}}(x)) * \mathit{node}_{X \uplus \{y\}}(y) \\
&\Rightarrow \forall^* x \in X \uplus \{y\}.\, \mathit{node}_{X \uplus \{y\}}(x)
\end{aligned}$$

Note that the converse doesn't hold because the RHS allows edges to y from nodes in X, but the LHS doesn't. □

Note that this predicate doesn't cater at all for cyclicity. That is to say, I see no elegant way to adapt the *graph* predicate to describe a dag.