Extensions of CAV

June 12, 2014

1 Cycles extension - copied from the CAV paper

Instead of keeping track of just *next**, we instrument the edge addition operation with a check: if the added edge is about to close a cycle, then instead of adding the edge, we keep it in a separate relation m of "cycle-inducing" edges. Two properties of lists now come into play: (1) The number of cycles reachable from program variables, and hence the size of M, is bounded by the number of program variables; (2) Any path (simple or otherwise) in the heap may utilize at most one of those edges, because once a path enters a cycle, there is no way out. In all assertions, therefore, we replace $\alpha\langle \mathit{next}^*\rangle\beta$ with: $\alpha\langle \mathit{next}^*\rangle\beta \vee \bigvee_{\langle u,v\rangle\in M}(\alpha\langle \mathit{next}^*\rangle u \wedge v\langle \mathit{next}^*\rangle\beta)$. Notice that it is possible to construct this formula thanks to the bound on the size of M; otherwise, an existential quantifier would have been required in place of the disjunction.

Cycles can also be combined with nesting, in such a way as to introduce an unbounded number of cycles. To illustrate this, consider the example of a linked list beginning at h and formed by a pointer field which we shall denote n, where each element serves as the head of a singly-linked cycle along a second pointer field m. This is basically the same as in the case of acyclic nested lists, only that the last node in every sub-chain (a list segment formed by m) is connected to the first node of that same chain.

One way to model this in a simple way is to assume that the programmer designates the last edge of each cycle; that is, the edge that goes back from the last list node to the first. We denote this designation by introducing a ghost field named c. This cycle-inducing edge is thus labeled c instead of m.

Properties of the nested data structure can be expressed with AF^R formulas as shown in Table 1. "Hierarchy" means that the primary list is contiguous, that is, there cannot be n-pointers originating from the middle of sub-lists. "Cycle edge" describes the closing of the cyclic list by a special edge c.

We were able to verify the absence of memory errors and the correct functioning of the program `flatten`, shown in Fig. 1.

All lists are acyclic	$sll(n^*) \wedge sll(m^*)$
No sharing between lists	$\forall\alpha,\beta,\gamma\colon h\langle n^*\rangle\alpha \wedge \alpha\langle m^*\rangle\beta \wedge h\langle n^*\rangle\gamma \wedge \gamma\langle m^*\rangle\beta \implies \alpha=\gamma$
Hierarchy	$\forall\alpha,\beta,\gamma : \alpha\neq\beta \wedge \beta\neq\gamma \wedge \alpha\langle m^*\rangle\beta \implies \neg\beta\langle n^*\rangle\gamma$
Cycle edge	$\forall\alpha,\beta,\gamma : \alpha\neq\beta \wedge \alpha\neq\gamma \wedge \alpha\langle n^*\rangle\beta \implies \neg\alpha\langle c\rangle\gamma$ $\forall\alpha,\beta : \beta\langle c\rangle\alpha \implies h\langle n^*\rangle\alpha \wedge \alpha\langle m^*\rangle\beta$

Table 1: Properties of a list of cyclic lists expressed in AF^R

```
Node flatten(Node h) {
  Node i = h, j = null;
  while (i != null) I1 {
    Node k = i;
    while (k != null) I2 {
      j = k; k = k.m;
    }
    j.c = null;
    i = i.n; j.m = null; j.m = i;
  }
  j.c = null; j.c := h;
  return h;
}
```

Figure 1: A program that flattens a hierarchical structure of lists into a single cyclic list.

2 Cycles

- The commands speak about n field, and $\langle n^* \rangle$ is used in the assertions.
- Behind the scenes we have an acyclic relation $\langle k^* \rangle$, and an auxiliary binary relation denoted by $\langle m \rangle$.
- $\alpha \langle n^* \rangle \beta$ in the assertions is translated to: $\alpha \langle k^* \rangle \beta \vee \vec{nk}(\alpha) \langle k^* \rangle \beta$.
- $\langle k^* \rangle$ is axiomatized with Γ_{lin} as in the CAV paper.
- Extra Axioms:
  - (A) $\forall \alpha, \beta, \gamma.\ \alpha \langle k^* \rangle \beta \wedge \alpha \langle m \rangle \gamma \rightarrow \alpha = \beta$ (there cannot be k and m from the same node)
  - (B) $\forall \alpha, \beta, \gamma.\ \alpha \langle m \rangle \beta \wedge \alpha \langle m \rangle \gamma \rightarrow \beta = \gamma$
  - (C) $\forall \alpha, \beta.\ \alpha \langle m \rangle \beta \rightarrow \beta \langle k^* \rangle \alpha$
- Additionally, two unary function symbols are used: $\vec{k}$ and $\vec{km}$.
  - $\vec{k}(\alpha)$:
    * Intended meaning: the last node reachable from α via n.
    * Axiom: $\forall \alpha, \beta.\ \alpha \langle k^* \rangle \beta \rightarrow \beta \langle k^* \rangle \vec{k}(\alpha)$
  - $\vec{km}(\alpha)$:
    * Intended meaning: the node reachable from $\vec{k}(\alpha)$ via m.
    * Axiom: $\forall \alpha, \beta, \gamma.\ \alpha \langle k^* \rangle \beta \wedge \beta \langle m \rangle \gamma \rightarrow \vec{km}(\alpha) = \gamma$
    * Axiom: $\forall \alpha.\ \vec{k}(\alpha) \langle m \rangle \vec{km}(\alpha)\ \vee\ \vec{km}(\alpha) = \vec{k}(\alpha)$

    Their properties ensure they can be used in EPR (these properties **follow** from the previous ones and the general theory for $\langle k^* \rangle$):
    * Idempotence: $\forall \alpha.\ \vec{k}(\vec{k}(\alpha)) = \vec{k}(\alpha)$
    * Idempotence: $\forall \alpha.\ \vec{km}(\vec{km}(\alpha)) = \vec{km}(\alpha)$
    * $\forall \alpha.\ \vec{km}(\vec{k}(\alpha)) = \vec{km}(\alpha)$
    * $\forall \alpha.\ \vec{k}(\vec{km}(\alpha)) = \vec{k}(\alpha)$
- A predicate $Between(\alpha, \beta, \gamma)$ standing for "there is a simple path from α to γ through β" can be defined by:

  $$Between(\alpha, \beta, \gamma) := (\alpha \langle k^* \rangle \beta \langle k^* \rangle \gamma) \vee (\vec{km}(\alpha) \langle k^* \rangle \beta \langle k^* \rangle \gamma) \vee (\alpha \langle k^* \rangle \beta \langle k^* \rangle \vec{k}(\alpha) \wedge \vec{km}(\alpha) \langle k^* \rangle \gamma)$$

  Using $Between(\alpha, \beta, \gamma)$ only, the programmer is not aware of our internal troubles with k, m and $\vec{km}, \vec{k}$.
- A predicate $OnCycle(\alpha)$ standing for "α is on a cycle" is defined by: $(\vec{k}(\alpha) \langle m \rangle \vec{km}(\alpha) \wedge \vec{km}(\alpha) \langle k^* \rangle \alpha)$

- To compute the weakest pre-conditions of $x.n := null$ use a microcode:

```
if OnCycle(x)
    s := k⃗(x)
    t := km⃗(x)
    x.k := null
    s.m := null
    if s ≠ x
        s.k := t (we may assume that s.k = null)
else
    x.k := null
```

- To compute the weakest pre-conditions of $x.n := y$ use a microcode (assuming that $x.n := null$ was performed immediately before)

```
if y⟨k*⟩x
    x.m := y (we may assume that x.m = null)
else
    x.k := y (we may assume that x.k = null)
```

- To compute the weakest pre-conditions of $x := y.n$ use a microcode

```
if k⃗(y) = y
    x := y.m
else
    x := y.k
```

- additions to wp for function symbols:
  - $x.k := y$ (assuming that $x.k = x.m = null$):
    * substitute $\vec{k}(\alpha)$ by $ite(\vec{k}(\alpha) = x, \vec{k}(y), \vec{k}(\alpha))$
    * substitute $\vec{km}(\alpha)$ by $ite(\vec{k}(\alpha) = x, \vec{km}(y), \vec{km}(\alpha))$
  - $x.m := y$ (assuming that $x.k = x.m = null$):
    * no change in $\vec{k}(\alpha)$
    * substitute $\vec{km}(\alpha)$ by $ite(\vec{k}(\alpha) = x, y, \vec{km}(\alpha))$
  - $x.k := null$:
    * substitute $\vec{k}(\alpha)$ by $ite(\alpha\langle k^*\rangle x, x, \vec{k}(\alpha))$
    * substitute $\vec{km}(\alpha)$ by $ite(\vec{k}(x) \neq x \wedge \alpha\langle k^*\rangle x, x, \vec{km}(\alpha))$
  - $x.m := null$
    * no change in $\vec{k}(\alpha)$
    * substitute $\vec{km}(\alpha)$ by $ite(\vec{k}(\alpha) = x, x, \vec{km}(\alpha))$

3 Nested Linked List

- The commands speak about d (down) and r (right) field.
- The assertions may include $\langle d^*\rangle$, $\langle r^*\rangle$, $\langle (d \cup r)^*\rangle$.
- This only works when there is no sharing — if $r(\alpha) = \beta$ then: $r(\gamma) = \beta$ iff $\gamma = \alpha$, and $d(\gamma) \neq \beta$ for every γ.
  - Add to the pre-condition of $x.r := y$ the fact that y does not become shared: $\forall\alpha.\ (\alpha\langle d^*\rangle y \rightarrow \alpha = y) \wedge (\alpha\langle r^*\rangle y \rightarrow \alpha = y)$. Make this a conjunct in the wp of $x.r := y$.
  - Add to the pre-condition of $x.d := y$ the fact that y does not become shared: $\forall\alpha.\ (\alpha\langle r^*\rangle y \rightarrow \alpha = y)$. Make this a conjunct in the wp of $x.d := y$.
- One unary function symbol are used: $\overleftarrow{r}$.

- $\overleftarrow{r}(\alpha)$:
  - Intended meaning: the first node that can reach α via n.
  - Axiom: $\forall \alpha, \beta . \beta\langle r^*\rangle\alpha \rightarrow \overleftarrow{r}(\alpha)\langle r^*\rangle\beta$
  - Idempotence: $\forall \alpha . \overleftarrow{r}(\overleftarrow{r}(\alpha)) = \overleftarrow{r}(\alpha)$

- $\alpha\langle(d \cup r)^*\rangle\beta$ in the assertions is translated to: $\alpha\langle n^*\rangle\overleftarrow{r}(\beta)$.