Counting Aliases

Marius Bozga, Radu Iosif, Yassine Lakhnech

Counting Aliases

Marius Bozga, Radu Iosif and Yassine Lakhnech

December 30, 2016

Abstract

Our contribution is the development and study of a first order interpreted logic (κAL) in which we can express shape properties of recursive data structures and compute weakest preconditions. We show that, in general, the satisfiability problem of κAL is undecidable.

1 Alias Logic with Counters

1.1 The Storeless Heap Model

We view program heaps as directed labeled graphs in which vertices are objects and edges model pointers between objects. More precisely, we model the heap as a *finitely rooted* directed graph i.e., a graph in which the number of vertices with no incoming edges is non-zero and finite. We associate these vertices a special meaning, that of *program variables*. Next, assume that the edges of the graph are labeled with symbols from a finite alphabet Σ such that, for each vertex in the graph, the outgoing edges are labeled with different symbols. These edges model the *selector variables* i.e, the pointer fields. Consider now a designated node ι which is connected to the entry points by means of some special edges labeled $v_1, \ldots, v_k$. Let Π be the set of root labels i.e., $\Pi = \{v_1, \ldots, v_k\}$. Formally, the graph is a tuple $G = \langle V, E, \iota, \Pi, \Sigma \rangle$, where V denotes the original set of vertices ($\iota \notin V$), $E \subseteq (\{\iota\} \times \Pi \times V) \cup (V \times \Sigma \times V)$ is the edge relation, ι is the root node, Π is the root alphabet and Σ the edge alphabet.

Now each vertex $v \in V$ can be associated a deterministic finite state automaton $\mathcal{A}_v = \langle V \cup \{\iota\}, E, \iota, v \rangle$ where V and E are the set of states and transition relation respectively, ι the initial state and v the final state. The language $\mathcal{L}_v$ recognized by this automaton uniquely identifies v, and therefore the graph G can be uniquely represented (modulo isomorphism) by the set $\{\mathcal{L}_v \mid v \in V\}$. For historical reasons we call this the *storeless* representation of G. The following definition captures the concept, where $\cdot$ is the concatenation of words extended to languages pointwise:

Definition 1 (Heap) *Given a finite root alphabet Π and a selector alphabet Σ, such that $\Pi \cap \Sigma = \emptyset$, a* heap $\mathcal{M} \subseteq \mathcal{P}(\Pi \cdot \Sigma^*)$ *is either the empty set or a finite set $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n\}$ satisfying the following conditions, for all $1 \leq i, j \leq n$:*

(D1) non-emptiness*: $\mathcal{L}_i \neq \emptyset$, and (D'1)* rooting*: $\forall r \in \Pi \ . \ \{r\} \in \mathcal{M}$*

(D2) prefix closure and right regularity*:*

$$\forall x \in \mathcal{L}_i \ [\forall y, z \in \Sigma^+ [x = y \cdot z \Rightarrow \exists \ 1 \leq k \leq n \ [y \in \mathcal{L}_k \ \textit{and} \ \mathcal{L}_k \cdot z \subseteq \mathcal{L}_i]]]$$

(D3) determinism*: $i \neq j \Rightarrow \mathcal{L}_i \cap \mathcal{L}_j = \emptyset$,*

Other approaches in the literature consider the languages in the heap as equivalence classes of a prefix-closed right-regular relation (*alias*), in the sense of the Myhill-Nerode Theorem. In the following we denote by $\mathcal{S}(\Pi, \Sigma)$ the family of all heaps over the root alphabet Π and selector alphabet Σ. In the rest of the paper we shall use the sets Π and Σ, as in Definition 1, without further explanation. Special attention will be devoted to the *one-selector* case i.e., when $\|\Sigma\| = 1$.

1.2 Syntax and Semantics

$$
\begin{array}{rcl}
a, b & \in & \Pi \cup \Sigma \\
x, y & \in & \Sigma^* \\
\kappa, \lambda & \in & Vars \\
t & := & \kappa \mid n \in \mathbb{N} \mid t_1 + t_2 \\
\sigma & := & a \mid x^t \mid \sigma_1 \cdot \sigma_2 \mid \sigma_1^{-1}\sigma_2 \\
\varphi & := & t_1 = t_2 \mid \sigma_1 \leq \sigma_2 \mid \sigma_1 \diamond \sigma_2 \\
 & & \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \forall\kappa.\varphi(\kappa)
\end{array}
$$

(a)

$$
\begin{array}{rcl}
\mathcal{M} & \in & \mathcal{S}(\Pi, \Sigma) \\
\nu & : & Vars \to \mathbb{N} \\
[\![t_1 = t_2]\!]_{\mathcal{M},\nu} & = & 1 \iff \nu(t_1) = \nu(t_2) \\
[\![\sigma_1 \leq \sigma_2]\!]_{\mathcal{M},\nu} & = & 1 \iff \nu(\sigma_1) \leq \nu(\sigma_2) \\
[\![\sigma_1 \diamond \sigma_2]\!]_{\mathcal{M},\nu} & = & 1 \iff \exists X \in \mathcal{M} \,.\, \nu(\sigma_1), \nu(\sigma_2) \in X \\
[\![\varphi_1 \wedge \varphi_2]\!]_{\mathcal{M},\nu} & = & \min([\![\varphi_1]\!]_{\mathcal{M},\nu}, [\![\varphi_2]\!]_{\mathcal{M},\nu}) \\
[\![\neg\varphi]\!]_{\mathcal{M},\nu} & = & 1 - [\![\varphi]\!]_{\mathcal{M},\nu} \\
[\![\forall\kappa.\varphi(\kappa)]\!]_{\mathcal{M},\nu} & = & \min\{[\![\varphi(\kappa)]\!]_{\mathcal{M},\nu[\kappa \to i]} \mid i \in \mathbb{N}\}
\end{array}
$$

(b)

Figure 1: Alias Logic with Counters: Syntax and Semantics

We denote by $\delta(\sigma) \stackrel{\Delta}{=} \sigma\diamond\sigma$ the fact that σ is a defined path in the heap i.e., $[\![\delta(\sigma)]\!]_{\mathcal{M},\nu}$ if there exists a node $X \in \mathcal{M}$ such that $\nu(\sigma) \in X$.

1.3 Expressing Heap Properties

One can characterize any finite heap structure (up to isomorphism) in the purely propositional fragment of κAL, that is, using only formulas that do not contain counter variables. In other words, for any given structure $\mathcal{M}$ we can build a formula $\phi_{\mathcal{M}}$ whose unique model is $\mathcal{M}$.

However, without using first order quantification, there is no way to express properties involving the presence of paths of unbounded length in the heap. Without attempting to be exhaustive, this section shows how such properties can be specified in alias logic with counters. We first give some properties of paths, and then combine them into specifications of recursive data structures.

Path properties

A basic property is reachability between two given nodes. Due to the basic nature of our logic, we can express the existence of a path composed of an unbounded number of repetitions of a given sequence. With the conventions from Figure 1 (a), we have:

$$REACH(\sigma, \tau, x) \stackrel{\Delta}{=} \exists\kappa \,.\, \sigma x^\kappa \diamond \tau$$

Obviously, more complicated reachability schemes can be described with the aid of more existentially quantified variables. On the other hand, it is impossible to write a formula such that, given two sequences with no counters $\sigma, \tau \in \Pi \cdot \Sigma^*$, is satisfied by every heap in which the node

pointed to by τ is reachable from the node pointed to by σ. This shows a basic limitation of our logic. Experience shows however, that in reasoning about data structures that occur during executions of programs, the capability of characterizing general reachability is not needed as much as the ability of describing the succession of selector fields occurring along a certain path. The $REACH(\sigma, \tau, x)$ predicate can be strengthened by stating the existence of a unique path between σ and τ which is a repeating sequence of x's:

$$REACH!(\sigma, \tau, x) \stackrel{\Delta}{=} \exists \kappa \ . \ \sigma x^{\kappa} \Diamond \tau \wedge \forall \lambda \ . \ \lambda < \kappa \rightarrow \bigwedge_{i=0}^{|x|} \bigwedge_{a \in \Sigma \setminus \{x_i\}} \neg \sigma x^{\lambda} x_{0 \ldots i-1} a \Diamond \sigma x^{\lambda} x_{0 \ldots i-1} a$$

The existence of a non-trivial cycle containing a node pointed to by σ and composed of more than one repetitions of sequence x can be stated as follows:

$$CYCLE(\sigma, x) \stackrel{\Delta}{=} \exists \kappa \ . \ \kappa > 0 \wedge \sigma x^{\kappa} \Diamond \sigma \wedge \forall \lambda < \kappa \ . \ \neg \sigma x^{\lambda} \Diamond \sigma$$

A strict version of this predicate, call it $CYCLE!(\sigma, x)$ is obtained following the example of $REACH!(\sigma, \tau, x)$.

In the upcoming discussion we shall also need the predicate $SHARE(\sigma, \tau, x) \stackrel{\Delta}{=} \exists \kappa, \lambda \ . \ \sigma x^{\kappa} \Diamond \tau x^{\lambda}$ expressing the fact that σ and τ share a common node via x-paths.

Recursive data structures

A more interesting application of the κAL logic is the characterization of an infinite class of heaps corresponding to a recursive data type specification. For instance, the class of simply linked lists with forward selector n and pointed to by σ are characterized by the following predicates:

$$LIST_l(\sigma, n) \stackrel{\Delta}{=} REACH(\sigma, \sigma n^l, n) \quad LIST(\sigma, n) \stackrel{\Delta}{=} \exists \kappa \ . \ LIST_{\kappa}(\sigma, n)$$
$$NCLIST_l(\sigma, n) \stackrel{\Delta}{=} LIST_l(\sigma, n) \wedge \neg \sigma n^{l+1} \Diamond \sigma n^{l+1}$$

The $LIST_l$ predicate states the existence of a path of length l starting with σ consisting of a succession of l cells linked by n selectors, the $LIST$ predicate states that σ points to some simply linked n-list, and $NCLIST_l$ ensures that the list is non-circular, by asserting the finiteness of the n-path.

Doubly linked lists are expressed as follows. The forward pointer is denoted in the following by n and the backward pointer by p.

$$DLIST_l(\sigma, n, p) \quad \stackrel{\Delta}{=} \quad \neg \sigma p \Diamond \sigma p \wedge LIST_l(\sigma, n) \wedge \forall \lambda \ . \ 0 < \lambda \leq l \rightarrow \sigma n^{\lambda} p \Diamond \sigma n^{\lambda - 1}$$

In analogy with simply linked lists, a non-cyclic doubly linked list can be specified by adding the $\neg \sigma n^{l+1} \Diamond \sigma n^{l+1}$ conjunct at the end of the definition.

However, trees are difficult to specify using this logic. Some classes of trees can be specified as extensions of lists. These trees can only grow unbounded along a fixed (bounded) number of directions. Again, this shows a limitation of the expressive power of our logic.

Reverse List Loop Invariant

Consider the program **REVERSE** in Figure 2. The input of the program is a non-cyclic list $NCLIST_l(i, n)$ and the output is a non-cyclic list $NCLIST_l(j, n)$ containing the cells of the first list in reverse order. Here l is a free logical variable that represents the length of the input list. In order to check that both input and output lists have the same length l, one needs to assert the following loop invariant, at line [1]:

$$\exists \kappa', \kappa'' \ . \ NCLIST_{\kappa'}(i,n) \wedge NCLIST_{\kappa''}(j,n) \wedge \kappa' + \kappa'' = l \wedge \neg SHARE(i,j,n)$$

Running the program on few small inputs is enough to convince the reader that the above is indeed the loop invariant needed for proving the validity of the triple

$$\{NCLIST_l(i,n)\} \ \textbf{REVERSE} \ \{NCLIST_l(j,n)\}$$

```
0   j := null;
1   while i ≠ null do
2         k := i.n;
3         i.n := j;
4         j := i;
5         i := k;
6   od
```

Figure 2: List Reversal Program

2 Undecidability of the Full Logic

In this section we prove the undecidability of the satisfiability problem for the κAL logic by reduction to the theory of integer arithmetic with addition and multiplication. In the following, let $\mathcal{N} = \langle \mathbb{N}, 0, S, +, \cdot \ \rangle$ be the theory of natural numbers, where S is the successor function $x \mapsto x + 1$ and $0, +, \cdot$ are interpreted as zero, addition and multiplication in $\mathbb{N}$. A first-order sentence is *valid* in $\mathcal{N}$ if and only if it is true when all the arithmetic symbols occurring in the sentence are interpreted in $\mathcal{N}$[1]. It is well known that the set of all valid sentences over $\mathcal{N}$ is not recursive, or, in other words, that the theory is undecidable.

We shall not reduce κAL directly to $\mathcal{N}$, but to the equivalent theory $\mathcal{D} = \langle \mathbb{N}, 0, S, +, | \ \rangle$, where $|$ is an infix binary predicate symbol, and the rest of the symbols are as before. Given $x, y \in \mathbb{N}$, $x|y$ is true if and only if x *divides* y.

To describe the reduction of $\mathcal{D}$ to κAL, let φ be any (possibly open) formula of $\mathcal{D}$. By $tr(\varphi)$ we denote the translation of φ into κAL. We denote by $x, y, \ldots$ the variables of φ and by $\kappa_x, \kappa_y, \ldots$ the corresponding variables in κAL. The letters t, u are used to denote terms in $\mathcal{D}$. Let us fix $\Pi = \{v\}$ and $\Sigma = \{a, b\}$ for the purposes of the proof.

We first translate φ into an equivalent formula φ', composed of atomic propositions of the form $v|u$, $v = u$, $x = S(v)$ or $x = v + u$, with x variable and v, u either variables or constants from $\mathbb{N}$. To do this, we first write the formula in prenex normal form $\varphi = Q_1 x_1 \ldots Q_n x_n \ . \ \psi$. Then ψ' is obtained from ψ by repeating the following steps until a fixpoint is reached:

1. choose a subterm θ of ψ of the form $S(t)$ or $t + u$.
2. let ψ' be $\psi[y/\theta] \wedge y = \theta$ where y is a fresh variable.

It is easy to see that the number of iterations is linear in the size of ψ. The resulting formula will be $\varphi' = Q_1 x_1 \ldots Q_n x_n \exists y_1 \ldots y_m \ . \ \psi'$. For example $[\forall x_1, x_2 \ S(x_1 + x_2) = S(x_1) + x_2]' = \forall x_1, x_2 \exists y_1, y_2, y_3, y_4 \ . \ y_1 = x_1 + x_2 \wedge y_2 = S(y_1) \wedge y_3 = S(x_1) \wedge y_4 = y_3 + x_2 \wedge y_2 = y_4$. Without losing generality, we can consider that φ' is in positive normal form i.e., the only negations being

[1]For simplicity we used a semantic notion of validity. Nevertheless, the proofs in this section go through also when the term *valid* means deductible from the set of Peano axioms.

relative to atomic propositions. The translation of φ into κAL is defined recursively on the structure of the equivalent formula φ'. Table 3 gives the translation of the atomic propositions and their negations. The other constructions are as follows: $tr(\exists x\ .\ \varphi) \stackrel{\Delta}{=} \exists \kappa_x\ .\ tr(\varphi)$, $tr(\varphi_1 \vee \varphi_2) \stackrel{\Delta}{=} tr(\varphi_1) \vee tr(\varphi_2)$ and $tr(\varphi_1 \wedge \varphi_2) \stackrel{\Delta}{=} tr(\varphi_1) \wedge tr(\varphi_2)$.

φ	$tr(\varphi)$	$tr(\neg\varphi)$
$x = S(v)$	$\exists\kappa\forall\lambda\ \lambda \neq \kappa_x \rightarrow \neg va^\kappa b^\lambda \Diamond va^\kappa b^{\kappa_x} \wedge va^\kappa b^{tr(v)} b \Diamond va^\kappa b^{\kappa_x}$	$\forall\kappa\ \neg va^\kappa b^{tr(v)} b \Diamond va^\kappa b^{\kappa_x}$
$x = v + u$	$\exists\kappa\forall\lambda\ \lambda \neq \kappa_x \rightarrow \neg va^\kappa b^\lambda \Diamond va^\kappa b^{\kappa_x} \wedge va^\kappa b^{tr(v)} b^{tr(u)} \Diamond va^\kappa b^{\kappa_x}$	$\forall\kappa\ \neg va^\kappa b^{tr(v)} b^{tr(u)} \Diamond va^\kappa b^{\kappa_x}$
$v \mid u$	$\exists\kappa\ \forall\lambda\ \lambda < tr(v) \rightarrow \neg va^\kappa \Diamond va^\kappa b^\lambda \wedge va^\kappa \Diamond va^\kappa b^{tr(v)} \wedge va^\kappa \Diamond va^\kappa b^{tr(u)}$	$\forall\kappa\ \neg va^\kappa b^{tr(v)} \Diamond va^\kappa b^{tr(u)}$
$v = u$	$tr(v) = tr(u)$	$tr(v) \neq tr(u)$

Figure 3: Translation of $\mathcal{D}$ into κAL

All formulas $tr(\varphi)$ are interpreted over models of a certain kind. Precisely, we consider all a-lists starting with v, out of which some b-lists might branch. However an element of a b-list may not be the start of another a-list. This is encoded in the following:

$$\Theta \quad \stackrel{\Delta}{=} \quad \forall\kappa > 0\ \delta(va^\kappa) \rightarrow \Big(\forall\kappa' \leq \kappa\ \forall\lambda \leq 0\ \delta(va^{\kappa'} b^\lambda) \rightarrow \big(\forall\mu > 0\ \neg\delta(va^{\kappa'} b^\lambda a^\mu)\big)\Big)$$

The crux of the translation is the interpretation of the *divides* predicate. Basically, $v|u$ if there exists a non-trivial b-cycle of period v in the heap, and u is the length of some path along this cycle. If we consider the κAL formula $tr(x|y)$, the first two conjuncts mean that κ_x and κ_y are the lengths of some paths along the same b-cycle, and the third conjunct states that κ_x is the least such length i.e., the period of the cycle. All cycles needed to interpret the various occurrences of the $|$ operator in φ are linked in a simply linked a-list pointed to by v. This explains the role of the existentially quantified variable κ: it gives the position of the corresponding b-cycle in the a-list. Since κAL formulas are interpreted over models of $\mathcal{S}(\Pi, \Sigma)$, the deterministic condition (C2) from Definition 1 ensures that there is only one b-cycle from the node reached by va^κ, for a given κ. We have thus reduced an arbitrary validity problem in $\mathcal{D}$ to a satisfiability problem in κAL . A decision procedure for κAL would answer the former problem and, since $\mathcal{D}$ is equivalent to $\mathcal{N}$, contradict the fact that $\mathcal{N}$ is undecidable.

Theorem 1 *A formula φ of $\mathcal{D}$ is valid if and only if the corresponding κAL formula $\Theta \wedge tr(\varphi)$ is satisfiable. As a result, κAL is undecidable.*

Proof: We remind that an open formula $\varphi(\vec{x})$ of $\mathcal{N}$ is valid iff for every valuation of the free variables ν, $\nu(\varphi)$ holds, where $\nu(\varphi)$ is a shorthand for $\varphi[\nu(\vec{x})/\vec{x}]$. The proof goes by induction on the structure of φ. The atomic propositions and their negations are intuitive.

- if $\varphi = \exists x\ \psi$ then $\varphi(x)$ is valid iff $\varphi(n)$ is valid for some $n \in \mathbb{N}$. By induction hypothesis this is true iff $tr(\varphi(n))$ is satisfiable iff $\exists\kappa_x\ tr(\varphi)$ is satisfiable.
- if $\varphi = \psi_1 \vee \psi_2$ then for each valuation ν we have that either $\nu(\varphi_1)$ is true or $\nu(\varphi_2)$ is true. The first case is equivalent, by the induction hypothesis with $tr(\nu(\psi_1))$ being satisfiable, hence $tr(\nu(\psi_1)) \vee tr(\nu(\psi_2)) \equiv \nu(tr(\psi_1) \vee tr(\psi_2))$ is satisfiable. The second case yields the same result. Hence $tr(\psi_1) \vee tr(\psi_2)$ is satisfiable in general.
- if $\varphi = \psi_1 \wedge \psi_2$ then φ is valid iff ψ_1 is valid and ψ_2 is valid. By the induction hypothesis, this is equivalent to $tr(\psi_1)$ and $tr(\psi_2)$ being satisfiable. A model of $tr(\psi_1) \wedge tr(\psi_2)$ can

be obtained by concatenating the a-lists from the models of $tr(\psi_1)$ and $tr(\psi_2)$. It can be easily shown that this is still a model of both $tr(\psi_1)$ and $tr(\psi_2)$.

□

As a remark, the translation defined by tr proves actually the undecidability of the fragment of κAL without addition but with order relation ($>$).