Preconditions of properties described in CTL for statements manipulating pointers

(Preliminary Version)

Yoshinori Tanabe[2,1], Toshinori Takai[2,1],
Toshifusa Sekizawa[2,1]
and Koichi Takahashi[1]

1: AIST CVS

2: JST CREST

Precondisions of properties described in CTL for statements manipulating pointers

Yoshinori Tanabe
Toshinori Takai
Toshifusa Sekizawa
Research Center for Verification and Semantics,
National Institute of
Advanced Industrial Science and Technology (AIST)
and
Japan Science and Technology Agency

Koichi Takahashi
Research Center for Verification and Semantics,
National Institute of
Advanced Industrial Science and Technology (AIST)

***Abstract*— In the predicate abstraction framework for verification, two key issues are computation of the weakest precondition of a predicate and satisfiability checking for a predicate. We propose to use temporal formulas as predicate for verifying properties on structures with pointers. In this paper we report that the first issue has been solved for CTL formulas.**

I. Introduction

The demand of program verification keeps increasing since programs are used ubiquitously. In particular the technology of automatic verification is important. Automatic verification based on predicate abstraction [1] has been widely studied and verification tools [2], [3] based on this technology has been developed. Most of such tools developed in the early days handle properties on the value of variables as predicates used in abstraction, and it was difficult to express properties on the shape of the heap of programs.

One of the authors proposed to use regular expressions as a method for abstracting heap structures. With the method he has verified properties on several concurrent garbage collection systems [4]. The method has then been extended to use temporal formulas instead of regular expressions [5]. We work on a (on-going) project to develop an automatic verification tool for properties on heap using predicate abstraction with the method. To achieve that we need to establish two tasks: one is to compute the weakest preconditions of predicates described in temporal formulas, the other is to decide satisfiability of such predicates. This paper gives a solution to the first task by giving an algorithm to compute such a precondition for a given predicate.

Many studies have been done on static program analysis on the shape of program heaps [6], [7]. Among them one of the closest works to ours is by Dams and Namjoshi [8] which also calculates the weakest preconditions for predicates on the shape of heaps. The difference is that their predicates ("shape predicate") is constructed from a predicate called *reach* that roughly expresses that a cell in a heap is reachable from another by following pointers, while our predicate is any CTL formula with the "pointed-by" relation as the transition relation.

This article is organaized as follows. Section II expresses the syntax and semantics of programs we analyze. We then introduce the predicates for abstraction in Section III. In Section IV we describe the procedure to compute the weakest preconditions and give a proof that the procedure is correct. Section V concludes.

II. Programs

We define a *statement* of a program in the BNF shown below. Then a *program* is defined as a finite sequence of statements.

⟨*var*⟩ ::= (any element of Var)
⟨*numeric constant*⟩ ::= (any natural number)
⟨*static pointer expression*⟩ ::= ⟨*var*⟩ | ⟨*var*⟩`.next`
⟨*pointer expression*⟩ ::= `new()` |
⟨*static pointer expression*⟩
⟨*basic condition*⟩ ::=
⟨*static pointer expression*⟩ `==` ⟨*static pointer expression*⟩ |
⟨*var*⟩`.val ==` ⟨*numeric constant*⟩
⟨*condition*⟩ ::= ⟨*basic condition*⟩ | `!`⟨*condition*⟩ |
⟨*condition*⟩ `||` ⟨*condition*⟩
⟨*statement body*⟩ ::= ⟨*var*⟩ `:=` ⟨*pointer expression*⟩ |
⟨*var*⟩`.val :=` ⟨*numeric constant*⟩ |
⟨*var*⟩`.next :=` ⟨*var*⟩ |
`if` ⟨*condition*⟩ `goto` ⟨*natural number*⟩
⟨*statement*⟩ ::= ⟨*statement body*⟩ | ⟨*label*⟩ ⟨*statement body*⟩

A *pointer structure* is a tuple (N, p, v, ρ) that satisfies:

- N is a set.
- $p : N \to N$.
- $v : N \to \omega$.
- $\rho : \mathrm{Var} \to N$.

where we denote by ω the least infinite ordinal (i.e., the set of natural numbers).

Intuitively, N is the set of "nodes." A "pointer" that points the "next" node $p(n)$ is stored in each node n. And each node keeps a natural number $v(n)$ that represents its "state." Every program variable v points a node $\rho(v)$.

We denote by PStr the class of all pointer structures.

We use the following notation:

- For a program P, we denote the i-th statement by P_i. (The first statement is P_0.)
- For a function f, the domain of f is denoted by $\mathrm{dom}(f)$.
- For a function f, new function $f' = f[s \mapsto t]$ is defined as follows:

$$\mathrm{dom}(f') = \mathrm{dom}(f) \cup \{s\}, \quad f'(x) = \begin{cases} t & \text{if } x = s \\ f(x) & \text{ow} \end{cases}$$

For a program P, we define a transition system $\mathcal{T}_P = (T_P, \rightarrow_P)$. The underlying class T_P is PStr $\times$ (len$(P)+1$), where len$(P)$is the number of statements which P consists of. The transition relation $\rightarrow_P$ will be defined so that $(S,i) \rightarrow_P (S',i')$ means "If the current pointer structure is S then executing the i-th statement of the program makes the pointer structure S' and the next program execution will be the i'-th statement." Formally, for $t = ((N,p,v,\rho),\ i) \in T_P$ and $t' \in T_P$, we denote by $t \rightarrow_P t'$ if either of the following holds:

- P_i is x`:=`y and $t' = ((N,p,v,\rho[x \mapsto \rho(y)]),\ i+1)$.
- P_i is x`:=`y`.next` and $t' = ((N,p,v,\rho[x \mapsto p(\rho(y))]),\ i+1)$.
- P_i is x`:=new()` and there exists $n \notin N$ such that $t' = ((N \cup \{n\}, p[n \mapsto n], v[n \mapsto 0], \rho[x \mapsto n]),\ i+1)$.
- P_i is x`.val:=`m and $t' = ((N,p,v[\rho(x) \mapsto m],\rho),\ i+1)$.
- P_i is x`.next:=`y and $t' = ((N,p[\rho(x) \mapsto \rho(y)],v,\rho),\ i+1)$.
- P_i is `if` c `goto` j and either
  - $(N,p,v,\rho) \in [\![c]\!]$ and $t' = ((N,p,v,\rho),\ j)$, or
  - $(N,p,v,\rho) \notin [\![c]\!]$ and $t' = ((N,p,v,\rho),\ i+1)$.

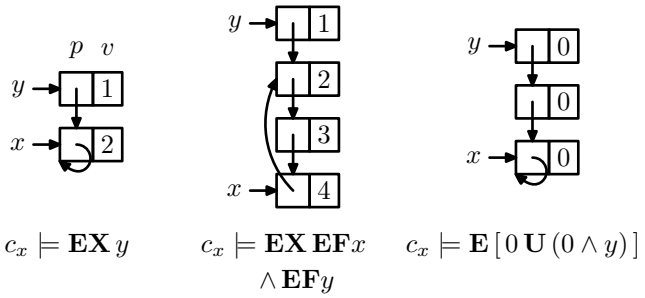
where $[\![c]\!] \subseteq$ PStr is the class of all pointer structures that "satisifies" c, or more precisely, is defined as follows:

- $[\![e_1$`==`$e_2]\!] = \{S \in \text{PStr} \mid {e_1}^S = {e_2}^S\}$ where for a variable x,
  - $x^{(N,p,v,\rho)} = \rho(x)$.
  - x`.next`$^{(N,p,v,\rho)} = p(\rho(x))$.
- $[\![x$`.val==`$m]\!]$ $= \{((N,p,v,\rho),i) \in T_P \mid v(\rho(x)) = m\}$.
- $[\![$`!`$c]\!] = T_P \setminus [\![c]\!]$.
- $[\![c_1$ `||` $c_2]\!] = [\![c_1]\!] \cup [\![c_2]\!]$.

For example, consider the following program P.

```
0: y = nil
1: if (x == nil) goto 7
2:   t = y
3:   y = x
4:   x = x.next
5:   y.next = t
6: if (x == x) goto 1
7: x = x
```

If S is a pointer structure that variable `x` points a list, which we here define as a sequence of a node that the function p points the next node and the final node points the node pointed by variable `nil` (which we call nil). Let S' be the pointer structure obtained by reversing the list and having variable `y` point the list and variable `x` nil. Then there is a sequence of elements of T_P that starts with $(S,0)$, ends with $(S',7)$, and the adjacent elements have the relation $\rightarrow_P$.



$c_x \models \mathbf{EX}\, y$ $\quad$ $c_x \models \mathbf{EX}\,\mathbf{EF}x \wedge \mathbf{EF}y$ $\quad$ $c_x \models \mathbf{E}\,[\,0\,\mathbf{U}\,(0 \wedge y)\,]$

Fig. 1. Example Formulas

III. Predicates

CTL formulas are formulas constructed from atomic proposition using boolean connectives and temporal connectives $\mathbf{EX}$, $\mathbf{AX}$, $\mathbf{EU}$, $\mathbf{AU}$, $\mathbf{ER}$ and $\mathbf{AR}$. In the rest of the paper, we only consider CTL formulas which temporal connectives $\mathbf{AX}$, $\mathbf{AU}$ and $\mathbf{AR}$ do not appear in. It is not a restriction because for any CTL formula we can find a formula with the property. Also we take the set AP of atomic propositions to be Var $\cup\, \omega$, the union of the set of program variables and the set of natural numbers.

For a pointer structure $S = (N,p,v,\rho)$, we denote by $K(S) = (N,R,\lambda)$ the Kripke structure that corresponds to S, where $R \subseteq N \times N$ and $\lambda : N \rightarrow \mathscr{P}(\text{AP})$ is defined as follows:

- $R = \{(n,n') \mid p(n') = n\}$
- $\lambda(n) = \{x \in \text{Var} \mid \rho(x) = n\} \cup \{v(n)\}$

Note that the transition relations in $K(S)$ are defined as the reverse direction of the pointers in S.

In the following examples, for variable v we denote by c_v the node pointed by v. See figure 1.

- $K(S), c_x \models \mathbf{EX}\, y$ means that c_y points to c_x.
- $K(S), c_x \models \mathbf{EX}\,\mathbf{EF}\, x \wedge \mathbf{EF}\, y$ means that By following the pointers, we can reach a loop from c_y.
- $K(S), c_x \models \mathbf{E}\,[0\,\mathbf{U}\,(0 \wedge y)]$ means that By following the pointers, we can reach from c_y to c_x and the value of all the nodes between c_y and c_x is zero.

We define *predicates for abstraction* (or *predicates* for short) as follows:

- For a CTL formula φ, $\exists\varphi$ is a predicate for abstraction.
- When Q_1 and Q_2 are predicattes for abstraction, so are $\neg Q_1$ and $Q_1 \wedge Q_2$.

We denote $\neg(\exists\neg\varphi)$ by $\forall\varphi$. We also introduce $Q_1 \wedge Q_2$, $Q_1 \rightarrow Q_2$, $Q_1 \leftrightarrow Q_2$ as abbreviation with obvious definitions.

For a pointer structure $S = (N,p,v,\rho)$ and a predicate Q, we define a relation $S \models Q$ as follows:

- $S \models \exists\varphi \iff$ There exists $n \in N$ such that $K(S), n \models \varphi$.

- $S \models \neg Q \iff S \not\models Q$.
- $S \models Q_1 \vee Q_2 \iff S \models Q_1$ or $S \models Q_2$.

IV. Preconditions

Theorem 1:

(1) There is a procedure which, for given condition c, calculates a predicate Q which satisfies $[\![c]\!] = \{S \in \text{PStr} \mid S \models Q\}$.

(2) There is a procedure which, for given statement s other than an `if` statement and for given predicate Q', calculates a predicate Q such that whenever the i-th statement P_i of a program P is s, for any pointer structures S and S' with $(S, i) \to_P (S', i+1)$, $S \models Q \iff S' \models Q'$ holds.

Proof:

(1) The procedure can easily be constructed by induction on the construction of c. For example, if c is x`.next==`y`.next`, Q can be $\exists(\mathbf{EX}\, x \wedge \mathbf{EX}\, y)$. The other cases are similar. ■

To prove (2), we prepare two lemmas.

Lemma 2: Suppose $S = (N, p, v, \rho), S' = (N', p', v', \rho') \in$ PStr, $(S, i) \to_P (S', i+1)$, P is a program and its i-th statement P_i is x`:=new()`, and $N' = N \cup \{n\}$. Let φ' be a CTL formula. Then whether $K(S'), n \models \varphi'$ holds or not can be decided depending only on φ' (independent from S, S'), and there is a procedure decides which is the case. ■

Lemma 3: There is a procedure which, for any statement s other than an `if` statement and for any CTL formula φ', finds predicates $Q_1, \ldots, Q_k$ and CTL formulas $\varphi_1, \ldots, \varphi_k$ which satisifies the following coditions:

- $\bigvee_{j=1}^{k} Q_j$ is valid and $Q_j \wedge Q_{j'}$ is unsatisfiable for $j \neq j'$.
- If the i-th statement P_i of a program P is s and $(S, i) \to_P (S', i+1)$ holds for $S, S' \in$ PStr, then the followings are equivalent:
  - $K(S'), n \models \varphi'$.
  - There exists j with $0 \leq j < k$ such that $S \models Q_j$ and $K(S), n \models \varphi_j$. ■

With Lemmas 2 and 3, Theorem 1 (2) can be proved as follows: the procedure is recursively defined with respect to the construction of the predicate Q'. When Q' is either $\neg Q'_1$ or $\neg Q'_1 \vee Q'_2$, then Q will be Q_1 or $Q_1 \vee Q_2$ respectively, where Q_1 and Q_2 are the predicates obtained from the induction hypothesis. Suppose that Q' is $\exists \varphi'$. If s is not in the form of x`:=new()`, take Q_j and φ_j $(0 \leq j < k)$ in Lemma 3 and $Q = \bigvee_{j=1}^{k}(Q_j \wedge \exists \varphi_j)$ is as required. If s is in the form of x`:=new()`, apply the procedure in Lemma 2 to decide decide whether $K(S'), n \models \varphi'$ holds or not. If it holds, take **true** as Q. If it does not hold, take Q_j and φ_j $(0 \leq j < k)$ in Lemma 3 and $Q = \bigvee_{j=1}^{k}(Q_j \wedge \exists \varphi_j)$ is as required.

Proof of Lemma 2:

The procedure is defined by induction on the construction of φ. When φ is an variable v, $n \models v$ holds if and only if $v = x$. When φ is an natural number m, $n \models v$ holds if and only if $m = 0$. The rest of the cases are as follows:

- $n \models \neg\psi \iff n \not\models \psi$.
- $n \models \psi_1 \vee \psi_2 \iff n \models \psi_1$ or $n \models \psi_2$.
- $n \models \mathbf{EX}\, \psi \iff n \models \psi$.
- $n \models \mathbf{E}\,[\psi_1\, \mathbf{U}\, \psi_2] \iff n \models \psi_2$.
- $n \models \mathbf{E}\,[\psi_1\, \mathbf{R}\, \psi_2] \iff n \models \psi_2$.

The cases for logical operators are trivial. The cases for temporal operators follow from the fact that the pointer points to itself in the added node n. ■

Proof of Lemma 3:

Except for the statement x`.next:=`y, the procedure is rather simple. In the following we denote by $\varphi[e/x]$ the formula obtained from φ with x replaced by e.

- For x`:=`y, take $k = 1$, $Q_1 =$ **true**, $\varphi_1 = \varphi'[y/x]$.
- For x`:=`y`.next`, take $k = 1$, $Q_1 =$ **true**, $\varphi_1 = \varphi'[\mathbf{EX}\, y/x]$.
- For x`.val:=`m, take $k = 1, Q_1 =$**true**, and φ_1 be φ' with m replaced by $m \vee x$ and n replaced by $n \wedge \neg x$ where $n \in \omega$, $n \neq m$.
- For x`:=new()`, take $k = 1, Q_1 =$ **true**, $\varphi_1 = \varphi'[\bot/x]$.

All the cases above can be shown by easy induction on the construction of φ'.

It remains the case where s is in the form of x`.next:=`y. We construct a procedure by induction on the construction of φ'. In the following, let k^1, Q^1_j, ψ^1_j $(0 \leq j < k^1)$ and k^2, Q^2_l, ψ^2_l $(0 \leq l < k^2)$ be the ones obtained from the induction hypothesis applied to formulas ψ'_1 and ψ'_2 respectively.

The base cases and logical connectives are as follows and their validity are clear.

- If φ' is a variable v or a natural number m, take $k = 1$, $Q_1 =$ **true**, $\varphi_1 = \varphi'$.
- If $\varphi' = \neg\psi'_1$, take $k = k^1$, $Q_j = Q^1_j$, $\varphi_j = \neg\psi^1_j$ $(0 \leq j < k^1)$.
- If $\varphi' = \psi'_1 \vee \psi'_2$, take $k = k^1 k^2$, $Q_{jk^1+l} = Q^1_j \vee Q^2_l$, $\varphi_{jk^1+l} = \psi^1_j \vee \psi^2_l$ $(0 \leq j < k^1,\ 0 \leq l < k^2)$.

Before describing procedures for the rest of the cases, we define a few notation. Let $S = (N, p, v, \rho)$, $S' = (N, p', v, \rho)$. We will write $c_x = \rho(x)$, $c_y = \rho(y)$ and $c_0 = p(c_x)$. The followings hold:

- In $K(S)$, $c_x \models x$, $c_y \models y$ and $c_0 \models \mathbf{EX}\, x$.
- In $K(S')$, $c_x \models x$, $c_y \models y$ and $c_y \models \mathbf{EX}\, x$.
- For $c \in N$, if either $c \neq c_x$, $p(c) \neq c_0$ or $p'(c) \neq c_y$, then $p(c) = p'(c)$.

And we also introduce a few more notation:

- $p_{\text{EX}}(\psi_1) = \psi_1$,
  $f^{\top}_{\text{EX}}(\psi_1) = y \vee \mathbf{EX}\,(\neg x \wedge \psi_1)$,
  $f^{\bot}_{\text{EX}}(\psi_1) = \mathbf{EX}\, \psi_1$.
- $p_{\text{EU}}(\psi_1, \psi_2) = \mathbf{E}\,[\psi_1\, \mathbf{U}\, \psi_2]$,
  $f^{\top}_{\text{EU}}(\psi_1, \psi_2) = x \vee \mathbf{E}\,[(\neg x \wedge \psi_1)\, \mathbf{U}\, (\neg x \wedge \psi_2)] \vee$
  $\mathbf{E}\,[(\neg x \wedge \psi_1)\, \mathbf{U}\, (\neg x \wedge (y \wedge \psi_1))]$,
  $f^{\bot}_{\text{EU}}(\psi_1, \psi_2) = \mathbf{E}\,[\psi_1\, \mathbf{U}\, \psi_2]$.
- $q_{\text{ER}}(\psi_1, \psi_2, v_1, v_2) =$
  $v_1 \vee \mathbf{E}\,[\{v_2 \vee \psi_1 \vee (\mathbf{EX}\, x \wedge \mathbf{AX}\, x)\}\, \mathbf{R}\, \{\psi_2 \wedge \neg x\}]$,
  $p_{\text{ER}}(\psi_1, \psi_2) =$
  $\psi_2 \wedge [\psi_1 \vee y \vee \mathbf{AX} \bot \vee \mathbf{EX}\, q_{\text{ER}}(\psi_1, \psi_2, \bot, y)]$,
  $f^{\top}_{\text{ER}}(\psi_1, \psi_2) = q_{\text{ER}}(\psi_1, \psi_2, x, y)$,
  $f^{\top}_{\text{ER}}(\psi_1, \psi_2) = q_{\text{ER}}(\psi_1, \psi_2, \bot, \bot)$.

Using them we claim:

Claim 4: Suppose for all $c \in N$ $K(S), c \models \psi_j \iff K(S'), c \models \psi'_j$ for $j = 1, 2$.
(1a) If $K(S), c_x \models p_{\mathrm{EX}}(\psi_1)$, $K(S'), c \models \mathbf{EX}\,\psi'_1 \iff K(S), c \models f^{\top}_{\mathrm{EX}}(\psi_1)$ holds for all $c \in N$.
(1b) If $K(S), c_x \not\models p_{\mathrm{EX}}(\psi_1)$, $K(S'), c \models \mathbf{EX}\,\psi'_1 \iff K(S), c \models f^{\bot}_{\mathrm{EX}}(\psi_1)$ holds for all $c \in N$.
(2a) If $K(S), c_x \models p_{\mathrm{EU}}(\psi_1, \psi_2)$, $K(S'), c \models \mathbf{E}\,[\psi'_1\,\mathbf{U}\,\psi'_2] \iff K(S), c \models f^{\top}_{\mathrm{EU}}(\psi_1)$ holds for all $c \in N$.
(2b) If $K(S), c_x \not\models p_{\mathrm{EU}}(\psi_1, \psi_2)$, $K(S'), c \models \mathbf{E}\,[\psi'_1\,\mathbf{U}\,\psi'_2] \iff K(S), c \models f^{\bot}_{\mathrm{EU}}(\psi_1)$ holds for all $c \in N$.
(3a) If $K(S), c_x \models p_{\mathrm{ER}}(\psi_1, \psi_2)$, $K(S'), c \models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2] \iff K(S), c \models f^{\top}_{\mathrm{ER}}(\psi_1)$ holds for all $c \in N$.
(3b) If $K(S), c_x \not\models p_{\mathrm{ER}}(\psi_1, \psi_2)$, $K(S'), c \models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2] \iff K(S), c \models f^{\bot}_{\mathrm{ER}}(\psi_1)$ holds for all $c \in N$. ■

With this claim, we can take the predicates and formulas for the case of logical connectives. For $\mathbf{EX}$, $k = 2k^1$, $Q_{2j} = Q_j \wedge \exists(x \wedge p_{\mathrm{EX}}(\psi^1_j))$, $Q_{2j+1} = Q_j \wedge \neg\exists(x \wedge p_{\mathrm{EX}}(\psi^1_j))$, $\varphi_{2j} = f^{\top}_{\mathrm{EX}}(\psi^1_j)$, $\varphi_{2j+1} = f^{\bot}_{\mathrm{EX}}(\psi^1_j)$. Note that $S \models \exists(x \wedge p_{\mathrm{EX}}(\psi^1_j))$ holds if and only if $K(S), c_x \models p_{\mathrm{EX}}(\psi^1_j)$ holds since c_x is the only element n of $K(S)$ that satisfies $n \models x$. For $\mathbf{EU}$ and $\mathbf{ER}$, we can take $k = k^1k^2$ and Q_j and φ_j similar to the case of $\mathbf{EX}$ using p_{EU}, $f^{\top}_{\mathrm{EU}}$, $f^{\bot}_{\mathrm{ER}}$, p_{ER}, $f^{\top}_{\mathrm{ER}}$ and $f^{\bot}_{\mathrm{ER}}$.

It is now easy to check the k, predicates Q_j and formulas φ_j ($0 \le j < k$) satisfies the condition. That completes the proof of Lemma 3. ■

Proof of Claim 4 (1): We only show the "⇒" part of (1a). The other direction and (1b) can be shown in a similar way.

Take $d \in N$ such that $p'(d) = c$ and $K(S'), d \models \psi'_1$. If $d = c_x$ then $c = c_y$ and hence $K(S'), c \models y$. If $d \neq c_x$ then $p(d) = p'(d) = c$. By induction hypothesis $K(S), d \models \psi_1$, therefore $K(S), c \models \mathbf{EX}\,(\neg x \wedge \psi^1_j)$. ■

We denote by $\mathrm{Path}(S)$ the set of *paths* in $K(S)$, namely $\pi \in \mathrm{Path}(S)$ if there exists $\alpha \le \omega$ such that:

1) π is a function from α to N,
2) for $i + 1 < \alpha$, $p(\pi(i + 1)) = \pi(i)$,
3) if $\alpha < \omega$, there is no $c \in N$ such that $p(c) = \pi(\alpha_1)$.

Similarly we denote by $\mathrm{Path}(S')$ the set of paths in $K(S')$ by replacing p with p' in the conditions above. If π satisfies 1) and 2), we call π a *prepath*.

For a prepath π and $j \in \mathrm{dom}(\pi)$, we denote by $\pi \restriction j$ a prepath π' such that $\mathrm{dom}(\pi') = j$ and $\pi(i) = \pi'(j)$ for all $i < j$. For a prepath π and $j < \omega$, we denote by π^j the j-th postfix of π, i.e. $\mathrm{dom}(\pi^j) = \{i \mid j + i \in \mathrm{dom}(\pi)\}$ and $\pi^j(i) = \pi(j+i)$. For a prepath π and σ with $m = \mathrm{dom}(\pi) < \omega$ and $p(\sigma(0)) = \pi(m - 1)$, we denote by $\pi \cdot \sigma$ the prepath concatenating π and σ, namely $\mathrm{dom}(\pi \cdot \sigma) = m + \mathrm{dom}(\sigma)$, $(\pi \cdot \sigma)(j) = \pi(j)$ for $j < m$ and $(\pi \cdot \sigma)(j) = \sigma(m + j)$ for $j \ge m$. For a node $c \in N$ we denote by $\langle c \rangle$ the prepath π with $\mathrm{dom}(\pi) = 1$ and $\pi(0) = c$.

Let $c \in N$ and π be a prepath, we say c *appears in* π if there exists $j \in \mathrm{dom}(\pi)$ such that $c = \pi(j)$.

For $m < \omega$ and $\sigma : m \to N$, we denote by $\sigma*$ a function $\sigma* : \omega \to N$ with $\sigma*(j) = \sigma(j)$ for $j < m$ and $\sigma*(j + m) = \sigma*(j)$ for $j < \omega$. If σ is a prepath, $m = \mathrm{dom}(\sigma)$ and $p(\sigma(0)) = \sigma(m - 1)$, then $\sigma*$ is a path.

Proof of Claim 4 (2):

For a path π and a (path) formula φ in the form of $\varphi_1\,\mathbf{U}\,\varphi_2$ such that $\pi \models \varphi$ holds, we denote by $u(\pi, \varphi)$ the least $k \in \mathrm{dom}(\pi)$ such that $\pi(k) \models \varphi_2$ and $\pi(k') \models \varphi_1$ for all $k' < k$.

First we show $K(S), c_x \models \mathbf{E}\,[\psi_1\,\mathbf{U}\,\psi_2] \iff K(S'), c_x \models \mathbf{E}\,[\psi'_1\,\mathbf{U}\,\psi'_2]$. Take $\pi \in \mathrm{Path}(S)$ such that $K(S), \pi \models \psi_1\,\mathbf{U}\,\psi_2$ and $\pi(0) = c_x$. Let k be $u(\pi, \psi_1\,\mathbf{U}\,\psi_2)$ and j be the maximum number that satisfies $k \ge j$ and $\pi(j) = c_x$. (Such j exists: take $j = 0$.) There exists $\pi' \in \mathrm{Path}(S)$ such that $\pi'(i) = \pi(j + i)$ for $0 \le i < k - j$: since c_x does not appear between $j + 1$ and k on π, $p(\pi(j + i + 1)) = \pi(j + i)$ implies $p'(\pi(i + 1)) = \pi_i$. For $i \le k - j$, we can take an arbitrary $\pi'(i + 1)$ so that $p'(\pi'(i + 1)) = \pi'(i)$ until a deadlock occurs (if ever). It is now clear that $K(S), \pi' \models \psi'_1\,\mathbf{U}\,\psi'_2$ holds using the induction hypothesis. Also $\pi'(0) = \pi(j) = c_x$. That proves $K(S), c_x \models \mathbf{E}\,[\psi_1\,\mathbf{U}\,\psi_2] \implies K(S'), c_x \models \mathbf{E}\,[\psi'_1\,\mathbf{U}\,\psi'_2]$, the other direction can be shown similarly.

To prove (2a), we can therefore assume that $K(S'), c_x \models \mathbf{E}\,[\psi'_1\,\mathbf{U}\,\psi'_2]$.

To show the "⇒" part, take an arbitrary $c \in N$ such that $K(S'), c \models \mathbf{E}\,[\psi'_1\,\mathbf{U}\,\psi'_2]$. We further assume $c \neq c_x$. Take $\pi' \in \mathrm{Path}(S')$ such that $K(S'), \pi' \models \psi'_1\,\mathbf{U}\,\psi'_2$ and $\pi'(0) = c$. Let $k = u(\pi', \psi'_1\,\mathbf{U}\,\psi'_2)$. If c_x does not appear in π', $\pi' \in \mathrm{Path}(S)$ and $K(S), \pi' \models \psi_1\,\mathbf{U}\,\psi_2$. If c_x appears in π', let j be the minimum $j \in \mathrm{dom}(\pi')$ such that $\pi'(j) = c_x$. Since $c \neq c_x$, $j \neq 0$ and $\pi'(j-1) = c_y$. From the minimality of j, c_x does not appear in $\pi' \restriction j$. Therefore there exists $\pi \in \mathrm{Path}(S)$ such that $\pi \restriction j = \pi' \restriction j$. If $k > j - 1$ then $K(S), \pi \models (\neg x \wedge \psi_1)\,\mathbf{U}\,(\neg x \wedge (y \wedge \psi_1))$. If $j - 1 \le k$ then $K(S), \pi \models (\neg x \wedge \psi_1)\,\mathbf{U}\,(\neg x \wedge \psi_2)$.

We next show the "⇐" part. When $K(S), c \models x$ holds, the conclusion clearly holds.

When there exists a path $\pi \in \mathrm{Path}(S)$ such that $K(S), \pi \models (\neg x \wedge \psi_1)\,\mathbf{U}\,(\neg x \wedge (y \wedge \psi_1))$, π is also a path in S' since c_x does not appear in π. Therefore $K(S'), \pi \models \psi_1\,\mathbf{U}\,(\psi_1 \wedge y)$. From the assumption there exists $\sigma' \in \mathrm{Path}(S')$ with $K(S'), \sigma' \models \psi'_1\,\mathbf{U}\,\psi'_2$. Let $k = u(\pi, \psi_1\,\mathbf{U}\,(\psi_1 \wedge y))$ and $\tau' = (\pi \restriction k) \cdot \sigma'$, then $K(S'), \tau' \models \psi'_1\,\mathbf{U}\,\psi'_2$.

When there exists a path $\pi \in \mathrm{Path}(S)$ such that $K(S), \pi \models (\neg x \wedge \psi_1)\,\mathbf{U}\,(\neg x \wedge \psi_2)$, c_x does not appear in π. Therefore π is also a path in S' and $K(S'), \pi \models \psi'_1\,\mathbf{U}\,\psi'_2$.

(2b) can be shown in a similar way. That completes the proof of Claim 4 (2). ■

We prepare two lemmas to prove Claim 4 (3).

Lemma 5: Suppose $c \in N$. $K(S'), c \models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2]$ holds if and only if there exists $\pi \in \mathrm{Path}(S)$ such that $\pi(0) = c$ and there is $\xi \le \omega$ that satisfies the following codition:

1) If $c = c_x$,
   a) for any $j \in \mathrm{dom}(\pi)$, if $1 \le j \le \xi$ then $\pi(j) \neq c_x$,

b) for any $j \in \mathrm{dom}(\pi)$, if $j \leq \xi$ then $K(S), \pi(j) \models \psi_2$, and
c) either of the following holds:
i) $\xi \notin \mathrm{dom}(\pi)$,
ii) $\xi \in \mathrm{dom}(\pi)$ and $K(S), \pi(\xi) \models \psi_1$,
iii) $\xi \in \mathrm{dom}(\pi)$ and $\pi(\xi) = c_y$, or
iv) $\xi \in \mathrm{dom}(\pi)$, $\pi(\xi) = c_0$, and there is no node $d \in N \setminus \{c_x\}$ such that $p(d) = c_0$.

2) If $c \neq c_x$ and $K(S'), c_x \models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2]$ holds, the condition is the same as 1) with replacing a) by:
a') for any $j \in \mathrm{dom}(\pi)$, if $j \leq \xi$ then $\pi(j) \neq c_x$,

3) If $c \neq c_x$ and $K(S'), c_x \not\models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2]$ holds, the condition is the same as 2) with removing c)iii).

Proof: We start with the "only if" part.

Case 1). Assume $c = c_x$. Let π' be a path in S' such that $\pi'(0) = c$ and $K(S'), \pi' \models \psi'_1\,\mathbf{R}\,\psi'_2$. Let ξ' be the least j such that $\pi'(j) \models \psi_1$ and $\pi'(j') \models \psi_2$ for all $j' \leq j$ if such j exists. If not let $\xi' = \omega$. Note that if $\xi' = \omega$ then $\pi'(j) \models \psi_2$ for all $j \in \mathrm{dom}(\pi')$.

If there is $k \in \mathrm{dom}(\pi') \setminus \{0\}$ such that $\pi'(k) = c$, take the least such k. Note that $\pi'(k-1) = c_y$. Take $\pi \in \mathrm{Path}(S)$ with $\pi \restriction k = \pi' \restriction k$. Let $\xi = k$ if $\xi' \geq k$ and $\xi = \xi'$ otherwise. These π and ξ satisfies the condition. If $\xi' \geq k$ then c)iii) holds, otherwise c)ii) holds.

If there is no $k \in \mathrm{dom}(\pi') \setminus \{0\}$ with $\pi'(k) = c$, we define $\pi \in \mathrm{Path}(S)$ and ξ by:

- If $\mathrm{dom}(\pi') < \omega$ and $\pi'(\mathrm{dom}(\pi') - 1) = c_0$, $\pi = \pi' *$, $\xi = \mathrm{dom}(\pi') - 1$,
- otherwise, $\pi = \pi'$, $\xi = \xi'$.

If $\xi' < \omega$, c)ii) holds. Otherwise, if $\pi = \pi' *$ then c)iv) holds, if $\pi = \pi'$ then c)i) holds.

Case 2). Assume $c \neq c_x$ and $K(S'), c_x \models \mathbf{E}\,[\psi_1\,\mathbf{R}\,\psi_2]$. Almost the same proof as in 1) can be applied and in a) $\pi(0)$ cannot be c_x since $c \neq c_x$.

Case 3). Assume $c \neq c_x$ and $K(S'), c_x \not\models \mathbf{E}\,[\psi_1\,\mathbf{R}\,\psi_2]$. Still the same proof can be applied but there should not be $k \in \mathrm{dom}(\pi') \setminus \{0\}$ such that $\pi'(k) = c$ from the assumption. Therefore the condition c)iii) cannot be satisfied.

Next we prove the "only if" part. Recall that (path) formula $\varphi\,\mathbf{R}\,\psi$ is equivallent to $(\psi\,\mathbf{U}\,(\varphi \wedge \psi)) \vee \mathbf{G}\,\psi$.

Case 1). We assume $c = c_x$ and take π and ξ that satisfy the condition. One of the c)i) through c)iv) holds. If c)ii) holds, from a) we can take $\pi' \in \mathrm{Path}(S')$ with $\pi \subseteq \pi'$ (i.e. $\mathrm{dom}(\pi) \subseteq \mathrm{dom}(\pi')$ and $\pi = \pi' \restriction \mathrm{dom}(\pi)$). It satisfies $K(S'), \pi' \models \psi'_2\,\mathbf{U}\,(\psi'_1 \wedge \psi'_2)$. If c)iii) holds, let $\pi' = (\pi \restriction (\xi+1))*$. $\pi' \in \mathrm{Path}(S')$ by a) and $K(S'), \pi' \models \mathbf{G}\,\psi'_2$. If c)iv) holds, we can assume $c_0 \neq c_y$ since otherwise the case is reduced to c)iii). Then $\pi' = \pi \restriction (\xi+1)$ is a path in S' and $K(S'), \pi' \models \mathbf{G}\,\psi'_2$. Finally if c)i) holds, we can assume c_y does not appear in π since otherwise the case is reduced to c)iii) by taking a smaller ξ. Then π is also a path in S' and $K(S'), \pi \models \mathbf{G}\,\psi'_2$.

In case 2), almost the same proof as in case 1) can be applied except when c)iii) holds. When c)iii) holds, let σ' be a path in S' such that $\sigma'(0) = c_x$ and $K(S'), \sigma' \models \psi'_1\,\mathbf{R}\,\psi'_2$ holds and let $\pi' = (\pi \restriction \xi) \cdot \sigma'$. Then π' is a path in S', $\pi'(0) = c$ and $K(S'), \pi' \models \psi'_1\,\mathbf{R}\,\psi'_2$.

Case 3). The same proof as in case 1) can be applied. ■

Lemma 6: Let $c \in N \setminus \{c_x\}$
(1) $K(S'), c_x \models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2] \iff K(S), c_x \models p_{\mathrm{ER}}(\psi_1, \psi_2)$.
(2) If $K(S'), c_x \models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2]$ holds,
$K(S'), c \models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2] \iff K(S), c \models q_{\mathrm{ER}}(\psi_1, \psi_2\,\bot, y)$.
(3) If $K(S'), c_x \not\models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2]$ holds,
$K(S'), c \models \mathbf{E}\,[\psi'_1\,\mathbf{R}\,\psi'_2] \iff K(S), c \models q_{\mathrm{ER}}(\psi_1, \psi_2\,\bot, \bot)$.

Proof:

(1) First we show the "⇒" part of (1). Let π be a path satisfying the condition of Lemma 5 1). $K(S), c_x \models \psi_2$ holds by b). If $\mathrm{dom}(\pi) = 1$ then $K(S), c_x \models \mathbf{AX}\,\bot$ and we are done, therefore we can assume $\mathrm{dom}(\pi) \geq 2$. If $\xi = 0$, then either ii),iii) or iv) holds in c). In the cases of ii) and iii) we have $K(S), c_x \models \psi_1 \vee y$. In the case of iv), let $\sigma = \langle c_x \rangle$ then we have $\sigma \in \mathrm{Path}(S)$ and σ guarantees that $K(S), c_x \models \mathbf{EX}\,\mathbf{EG}\,(\psi_2 \wedge \neg x)$. Therefore we can assume $\xi \geq 1$. Under these assumptions, we show $K(S), \pi^1 \models (\psi_1 \vee y \vee (\mathbf{EX}\,x \wedge \mathbf{AX}\,x))\,\mathbf{R}\,(\psi_2 \wedge \neg x)$, using the cases in c): In case c)i), $K(S), \pi^1 \models \mathbf{G}\,(\psi_2 \wedge \neg x)$ holds, in case c)ii), $K(S), \pi^1 \models \psi_1\,\mathbf{R}\,(\psi_2 \wedge \neg x)$ holds, in case c)iii), $K(S), \pi^1 \models y\,\mathbf{R}\,(\psi_2 \wedge \neg x)$ holds, and in case c)iv), $K(S), \pi^1 \models (\mathbf{EX}\,x \wedge \mathbf{AX}\,x)\,\mathbf{R}\,(\psi_2 \wedge \neg x)$ holds.

Next we show the "⇐" part of (1). To do that we need to construct $\pi \in \mathrm{Path}(S)$ that satisfies the condition 1) in Lemma 5. If $K(S), c_x \models \psi_1 \vee y$ holds, π can be any path with $\pi(0) = c_x$ and take $\xi = 0$. If $K(S), c_x \models \mathbf{AX}\,\bot$ holds, take $\pi = \langle c_x \rangle$ and $\xi = 1$. If neither of the above holds, there exists $d \in N$ with $p(d) = c_x$ and $K(S), d \models \mathbf{E}\,[\{\psi_1 \vee y \vee (\mathbf{EX}\,x \wedge \mathbf{AX}\,x)\}\,\mathbf{R}\,\{\psi_2 \wedge \neg x\}]$ holds. Take a path $\sigma \in \mathrm{Path}(S)$ that guarantees it and let $\pi' = \langle c_x \rangle \cdot \sigma$. Clearly $\pi'(0) = c_x$ and it is fairly straight forward to show that π' is a path in S' $K(S'), \pi \models \psi'_1\,\mathbf{R}\,\psi'_2$.

We can show (2) and (3) in a similar way, using Lemma 5 2) and 3) respectively. ■

Now Claim 4 (3) is clear from Lemmas 5 and 6, and that completes proof of Claim 4, Lemma 3 and Theorem 1. ■

V. Conclusion and Future Works

We show a method to compute the weakest precondition of predicates on the pointer structure described by CTL formulas for statements manipulating pointers.

Our goal is to build a tool that automatically verifies properties on programs manipulating pointers based on predicate abstraction. In the tool the method we describe in this paper will be implemented and combined with satisfiability solver for CTL formulas.

Another direction of future work would be to extend the range of formula. We already devloped algorithms of satisfiability checking for a variety of formula systems that are wider than CTL, such as two-way CTL and alternation-free two-way μ calculus [9]. By studying weakest preconditions for these formulas, a wider range of properties are expected to be verified using predicate abstraction.

ACKNOWLEDGMENT

This research was supported by Core Research for Evolutional Science and Technology (CREST) Program "New High-performance Information Processing Technology Supporting Information-oriented Society" of Japan Science and Technology Agency (JST).

REFERENCES

[1] S. Graf and H. Saidi, "Construction of abstract state graphs with PVS," in *Proc. 9th INternational Conference on Computer Aided Verification (CAV'97)*, O. Grumberg, Ed., vol. 1254. Springer Verlag, 1997, pp. 72–83.

[2] T. Ball and S. K. Rajamani, "Automatically validating temporal safety properties of interfaces," in *Proceedings of the 8th international SPIN workshop on Model checking of software*. Springer-Verlag New York, Inc., 2001, pp. 103–122.

[3] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre, "Software verification with Blast," in *Proceedings of the Tenth International Workshop on Model Checking of Software (SPIN)*, ser. Lecture Notes in Computer Science, vol. 2648, 2003, pp. 235–239.

[4] K. Takahashi and M. Hagiya, "Abstraction of link structures by regular expressions and abstract model checking of concurrent garbage collection," in *First Asian Workshop on Programming Languages and Systems*, 2000, pp. 1–8.

[5] ——, "Abstraction of graph transformation using temporal formulas," in *Supplemental Volume of the 2003 International Conference on Dependable Systems and Networks (DSN-2003)*, 2003, pp. W–65 to W–66.

[6] N. D. Jones and S. S. Muchnick, *Flow analysis and optimization of Lisp-like structures*, 1981, ch. 4, pp. 102–131.

[7] M. Sagiv, T. Reps, and R. Wilhelm, "Parametric shape analysis via 3-valued logic," vol. 24, no. 3, pp. 217–298, May 2002.

[8] D. Dams and K. S. Namjoshi, "Shape analysis through predicate abstraction and model checking," in *Fourth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI03)*, 2003, pp. 310–324.

[9] Y. Tanabe, K. Takahashi, M. Yamamoto, T. Sato, A. Tozawa, and M. Hagiya, "Decision procedures for satisfiability checking of modal logics implementable with BDD (in Japanese)," in *Seventh Workshop on Programming and Programming Languages (PPL2005)*, 2005.

ポインタに関して CTL で表現された性質の実行文に対する前条件(in English)
(算譜科学研究速報)
発行日：2005 年 3 月 31 日
編集・発行：独立行政法人産業技術総合研究所関西センター尼崎事業所
システム検証研究センター
同連絡先：〒661-0974 兵庫県尼崎市若王寺 3-11-46
e-mail：informatics-inquiry@m.aist.go.jp

Preconditions of properties described in CTL for statements manipulating pointers
(Programming Science Technical Report)
March 31, 2005
Research Center for Verification and Semantics (CVS)
AIST Kansai, Amagasaki Site
National Institute of Advanced Industrial Science and Technology (AIST)
3-11-46 Nakouji, Amagasaki, Hyogo, 661-0974, Japan
e-mail: informatics- inquiry@m.aist.go.jp