Verifying Programs with Arrays

Computer Science and Artificial Intelligence Laboratory
MIT

Nov 1, 2011
Recap: Weakest Preconditions

Weakest predicate $P$ such that $\models \{P\} c \{A\}$
- $P$ weaker than $Q$ iff $Q \Rightarrow P$

$P = wpc(c, A)$

$wpc(skip \{Q\}) = Q$

$wpc(x = e\{Q\}) = Q[e/x]$  

$wpc(C1; C2\{Q\}) = wpc(C1\{wpc(C2\{Q\})\})$

$wpc(if \ B \ then \ C1 \ else \ C2\{Q\}) = \ (B \ and \ wpc(C1\{Q\})) \ or \ (not \ B \ and \ wpc(C2\{Q\}))$
Recap: Weakest Precondition

While-loop is tricky

- Let $W = wpc(\text{while } e \text{ do } c, B)$
- then,

$$W = e \Rightarrow wpc(c, W) \land \neg e \Rightarrow B$$
Recap: Verification Condition

Stronger than the weakest precondition

Can be computed by using an invariant

\[ VC(\text{while}_l \ e \ \text{do} \ c, B) = \]
\[ I \land \forall x_1, \ldots, x_n \ I \Rightarrow (e \Rightarrow VC(c, I) \land \neg e \Rightarrow B) \]
- Where \( x_i \) are variables modified in \( c \).
The problem with arrays

Now what?
Can we use the standard rule for assignment?

\[ wpc(x := e, C) = C[x \rightarrow e] \]
The problem with arrays

{true}
a[k]=1;
a[j]=2;
x=a[k]+a[j];
{x=3}

What if k=j?
Theory of arrays

Let $a$ be an array

$a\{i \rightarrow e\}$ is a new array whose $i^{th}$ entry has value $e$

- $a\{i \rightarrow e\}[k] = \begin{cases} a[k] & \text{if } k \neq i \\ e & \text{if } k = i \end{cases}$

A formula involving TOA can be expanded into a set of implications.
- Ex. Assume Zero is the zeroed out array

- $Zero\{i \rightarrow 5\}{j \rightarrow 7\}[k] = 5 \iff \ldots$
Assignment rule with theory of arrays

\[ \vdash \{ P[a \rightarrow a[i \rightarrow e]] \} \quad a[i] = e \quad \{ P \} \]

\{true\}
a[k]=1;
a[j]=2;
{a[k]+a[j]=3}
x=a[k]+a[j];
{x=3}

\{true\}
\{a\{k->1\}{j->2}[k]+a\{k->1\}{j->2}[j]=3\}
a[k]=1;
\{a\{j->2\}[k]+a\{j->2\}[j]=3\}
a[j]=2;
{a[k]+a[j]=3}
x=a[k]+a[j];
{x=3}
Assignment rule with theory of arrays

\[ \vdash \{ P[a \rightarrow a[i \rightarrow e]] \} \quad a[i] = e \quad \{ P \} \]

\{true\}
a[k]=1;
a[j]=2;
\{a[k]+a[j]=3\}
x=a[k]+a[j];
\{x=3\}

\{k \neq j\}
a[k->1]{j->2}[k]+a[k->1]{j->2}[j]=3
\{a[k]=1;\}
a[k->1]{j->2}[k]+a[k->1]{j->2}[j]=3
\{a[j]=2;\}
a[k]+a[j]=3
x=a[k]+a[j];
\{x=3\}
Arrays and loops

Consider the following program:

\[ \{0 \leq i < n\} \]
\[ j = i+1 \]
\[ \text{while } j < n \text{ do} \]
\[ \quad a[i] = a[i] + a[j]; \]
\[ \quad j = j+1; \]

\[ \{a[i] = \sum_{i \leq k < n} a_0[k]\} \]

A reasonable loop invariant: \( a[i] = \sum_{i \leq k < j} a_0[k] \)
Arrays and loops

Let’s try to verify our candidate loop invariant

\[ \{ a[i] = \sum_{i \leq k < j} a_0[k] \text{ and } j < n \} \]

\[ a[i] = a[i] + a[j]; \]

\[ j = j + 1 \]

\[ \{ a[i] = \sum_{i \leq k < j} a_0[k] \} \]

We can’t quite prove this implication!

We don’t know that \( a[j] = a_0[j] \)
A better loop invariant

\{ \forall i \leq k < n a[k] = a_0[k] \text{ and } 0 \leq i < n \}\)
\[ j = i + 1; \]
\{ a[i] = \sum_{i \leq k < j} a_0[k] \text{ and } \forall j \leq k < n a[k] = a_0[k] \}\)
while \( j < n \) do
\[ a[i] = a[i] + a[j]; j = j + 1 \]
\{ a[i] = \sum_{i \leq k < n} a_0[k] \}\)
Checking the loop invariant

\{ a[i] = \sum_{i \leq k < j} a_0[k] \text{ and } \forall j \leq k < n a[k] = a_0[k] \}
and \ j < n \}
\{ a\{ i \rightarrow a[i] + a[j]\}\}[i] = \sum_{i \leq k < j+1} a_0[k]
and \ \forall j+1 \leq k < n a\{ i \rightarrow a[i] + a[j]\}\}[k] = a_0[k] \}\}
a[i] = a[i] + a[j];
\{ a[i] = \sum_{i \leq k < j+1} a_0[k] \text{ and } \forall j+1 \leq k < n a[k] = a_0[k] \}\}
j = j + 1
\{ a[i] = \sum_{i \leq k < j} a_0[k] \text{ and } \forall j \leq k < n a[k] = a_0[k] \}\}
An even better invariant

\{ \forall_{i \leq k < n} a[k] = a_0[k] \text{ and } 0 \leq i < n \}

\begin{align*}
& j = i + 1; \\
& \{ a[i] = \sum_{i \leq k < j} a_0[k] \text{ and } \forall_{j \leq k < n} a[k] = a_0[k] \text{ and } i < j \}
\end{align*}

while \( j < n \) do
\begin{align*}
& a[i] = a[i] + a[j]; j = j + 1 \\
& \{ a[i] = \sum_{i \leq k < n} a_0[k] \}
\end{align*}