Verifying Programs with Arrays

[redacted slides]

Adam Chlipala
MIT

[based on Armando Solar-Lezama's slides]

6.820
October 23, 2013
Recap: Weakest Preconditions

Weakest predicate $P$ such that $\{P\} \implies \{A\}$.
- $P$ weaker than $Q$ iff $Q \implies P$

$wp(\text{skip}, A) = A$

$wp(x := e, A) = A[e/x]$

$wp(c_1 ; c_2, A) = wp(c_1, wp(c_2, A))$

$wp(\text{if } b \text{ then } c_1 \text{ else } c_2, A) = (b \land wp(c_1, A)) \lor (\neg b \land wp(c_2, A))$
Recap: Weakest Preconditions

“while” is tricky!

Let \( W = \text{wp}(\text{while } e \text{ do } c, A) \)
Then: \( W \leftrightarrow (e \Rightarrow \text{wp}(c, W) \land \neg e \Rightarrow A) \)

[A recursive equation, where it isn't obvious a solution exists!]

*Pragmatic solution:* ask programmers to annotate loops with *loop invariants.*

\[
c := x := e | c; c | \text{if } b \text{ then } c \text{ else } c | \{I\} \text{ while } b \text{ do } c
\]
Adding the “while” case

\[ wp(x := e, A) = A[e/x] \]

\[ wp(c_1 ; c_2, A) = wp(c_1, wp(c_2, A)) \]

\[ wp(\text{if } b \text{ then } c_1 \text{ else } c_2 , A) = (b \land wp(c_1, A)) \lor (\neg b \land wp(c_2, A)) \]

\[ wp(\{I\} \text{ while } b \text{ do } c, A) = I \]
\[ \land \ \text{written}(c) = \{x_1, \ldots, x_n\} \]
\[ \land (\forall x_1, \ldots, x_n . I \land b \Rightarrow wp(c, I)) \]
\[ \land (\forall x_1, \ldots, x_n . I \land \neg b \Rightarrow A) \]
The problem with arrays

\{true\}
a[k]=1;
a[j]=2;
x=a[k]+a[j];
{x=3}\ {true}\ a[k]=1;
a[j]=2;
{a[k]+a[j]=3} x=a[k]+a[j];
{x=3} Now what?
Can we use the standard rule for assignment?
wp(x := e, A) = A[e/x]
Theory of arrays

Let $a$ be a metavariable standing for an array.

$a_{i \rightarrow e}$ is a new array where:

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

It's possible to expand any quantifier-free formula involving arrays into a set of implications that don't use array operators. [SMT solvers do this, as you may see in PS4!]

**Example:** Where $a_0$ is the array where all cells hold 0:

$$a_0_{i \rightarrow 5}{j \rightarrow 7}[k] = 5 \iff \ldots$$
Assignment rule with arrays

{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}
Arrays and loops

Consider the following program:

\[
\begin{align*}
j &= i + 1; \\
\text{while } j < n \text{ do} & \quad a[i] = a[i] + a[j]; \\
& \quad j = j + 1
\end{align*}
\]