# Arithmetic Expression Summary

$$\frac{}{(n, \sigma) \mapsto eval(n)}$$

$$\frac{}{(x, \sigma) \mapsto \sigma(x)}$$

$$\frac{(a_0, \sigma) \mapsto k_0 \qquad (a_1, \sigma) \mapsto k_1}{(a_0 + a_1, \sigma) \mapsto k_0 + k_1}$$

$$\frac{(a_0, \sigma) \mapsto k_0 \qquad (a_1, \sigma) \mapsto k_1}{(a_0 - a_1, \sigma) \mapsto k_0 - k_1}$$

$$\frac{(a_0, \sigma) \mapsto k_0 \qquad (a_1, \sigma) \mapsto k_1}{(a_0 * a_1, \sigma) \mapsto k_0 \times k_1}$$

$$\frac{(a, \sigma) \mapsto k}{((a), \sigma) \mapsto k}$$

with $k$, $k_0$, $k_1 \in \mathbb{I}$, $n$,$x$,$a$,$a_0$, $a_1 \in$ **Aexp**, and $\sigma \in \Sigma$.

# Expression Equivalence

Our notion of semantic value for expressions leads to a natural equivalence relation between arithmetic expressions:

$$a_0 \sim a_1 \text{ iff } \forall \sigma \in \Sigma, \exists n \in \mathbb{I}.\ (a_0, \sigma) \mapsto n \wedge (a_1, \sigma) \mapsto n,$$

where $a_0$, $a_1 \in$ **Aexp**.

> *Two expressions are equivalent if and only if they evaluate to the same semantic value in all possible states.*

(You should convince yourself that this is indeed an equivalence relation, i.e., check that the relation $\sim$ is reflexive, symmetric, and transitive.)

# Expression Equivalence

**Problem:** Let $a_0 = 2 * 3$ and $a_1 = 3 + 3$, with $a_0$, $a_1 \in \textbf{Aexp}$. Show that $a_0 \sim a_1$.

# Expression Equivalence

*Proof*: We need to show that $(2 * 3, \sigma) \mapsto k$ and $(3 + 3, \sigma) \mapsto k$ for all states $\sigma \in \Sigma$ and some $k \in \mathbb{I}$.

Let $\sigma' \in \Sigma$ be any state, then

$$\frac{\overline{(2, \sigma') \mapsto 2} \qquad \overline{(3, \sigma') \mapsto 3}}{(2 * 3, \sigma') \mapsto 6}$$

and

$$\frac{\overline{(3, \sigma') \mapsto 3} \qquad \overline{(3, \sigma') \mapsto 3}}{(3 + 3, \sigma') \mapsto 6}$$

which shows that regardless of the state, the two expressions will always produce the same semantics value, namely the integer 6. This concludes the proof. $\square$

**Problem:** Show that the $+$ operator is commutative.

# Expression Equivalence

*Proof:* We need to show that $a_0 + a_1 \sim a_1 + a_0$ for all $a_0,\ a_1 \in$ **Aexp**. We show this by demonstrating that

$$(a_0 + a_1, \sigma) \mapsto n \wedge (a_1 + a_0, \sigma) \mapsto n$$

for all $\sigma \in \Sigma$ and $n \in \mathbb{I}$.

Assume that

$$\frac{}{(a_0, \sigma') \mapsto k_0}$$

and

$$\frac{}{(a_1, \sigma') \mapsto k_1}$$

for some $\sigma' \in \Sigma$ and $k_0,\ k_1 \in \mathbb{I}$.

# Expression Equivalence

Then we can construct the derivations

$$\frac{\overline{(a_0, \sigma') \mapsto k_0} \qquad \overline{(a_1, \sigma') \mapsto k_1}}{(a_0 + a_1, \sigma') \mapsto k_0 + k_1}$$

and

$$\frac{\overline{(a_1, \sigma') \mapsto k_1} \qquad \overline{(a_0, \sigma') \mapsto k_0}}{(a_1 + a_0, \sigma') \mapsto k_1 + k_0 = k_0 + k_1}$$

This proves the commutativity of $+$. $\square$

**Observation:** Commutativity of the syntactic $+$ operator is provided by the commutativity of the $+$ operator over the set of integers.

# Boolean Expressions

Recall our production for boolean expressions:

$$\mathsf{B} ::= \mathbf{true} \mid \mathbf{false} \mid \mathsf{A} = \mathsf{A} \mid \mathsf{A} \leq \mathsf{A} \mid !\mathsf{B} \mid \mathsf{B}\&\&\mathsf{B} \mid \mathsf{B}||\mathsf{B} \mid (\mathsf{B})$$

To compute the semantic value of boolean expressions we define an evaluation function '$\mapsto$', [1]

$$\mapsto : \ \mathbf{Bexp} \times \Sigma \to \mathbb{B},$$

and write

$$(be, \sigma) \mapsto \mathbf{t},$$

with $be \in \mathbf{Bexp}$, $\sigma \in \Sigma$, and $\mathbf{t} \in \mathbb{B}$.

As in the case of the arithmetic expressions we introduce an *eval* function in order to map the syntactic representations of boolean constants in $\mathbf{T}$ into the semantic concepts of the constant in $\mathbb{B}$,

$$eval : \ \mathbf{T} \to \mathbb{B}$$

[1] What does the inductive definition of **Bexp** look like?

# Boolean Expressions

$$\frac{}{(\textbf{true}, \sigma) \mapsto \mathit{eval}(\textbf{true})}$$

$$\frac{}{(\textbf{false}, \sigma) \mapsto \mathit{eval}(\textbf{false})}$$

$$\frac{(a_0, \sigma) \mapsto n \qquad (a_1, \sigma) \mapsto m}{(a_0 = a_1, \sigma) \mapsto \mathit{true}} \quad \text{if } n \text{ and } m \text{ are equal}$$

$$\frac{(a_0, \sigma) \mapsto n \qquad (a_1, \sigma) \mapsto m}{(a_0 = a_1, \sigma) \mapsto \mathit{false}} \quad \text{if } n \text{ and } m \text{ are not equal}$$

$$\frac{(a_0, \sigma) \mapsto n \qquad (a_1, \sigma) \mapsto m}{(a_0 \leq a_1, \sigma) \mapsto \mathit{true}} \quad \text{if } n \text{ is less than or equal to } m$$

$$\frac{(a_0, \sigma) \mapsto n \qquad (a_1, \sigma) \mapsto m}{(a_0 \leq a_1, \sigma) \mapsto \mathit{false}} \quad \text{if } n \text{ is not less than or equal to } m$$

with $\textbf{true}, \textbf{false} \in \mathbf{T}$, $a_0, a_1 \in \textbf{Aexp}$, $\sigma \in \Sigma$, and $m, n \in \mathbb{I}$.

# Boolean Expressions

$$\frac{(b, \sigma) \mapsto \mathit{true}}{(!b, \sigma) \mapsto \mathit{false}} \qquad \frac{(b, \sigma) \mapsto \mathit{false}}{(!b, \sigma) \mapsto \mathit{true}}$$

$$\frac{(b_0, \sigma) \mapsto t_0 \qquad (b_1, \sigma) \mapsto t_1}{(b_0 \&\& b_1, \sigma) \mapsto t}$$

where $t$ is *true* if $t_0 = \mathit{true}$ and $t_1 = \mathit{true}$, and *false* otherwise.

$$\frac{(b_0, \sigma) \mapsto t_0 \qquad (b_1, \sigma) \mapsto t_1}{(b_0 || b_1, \sigma) \mapsto t}$$

where $t$ is *true* if $t_0 = \mathit{true}$ or $t_1 = \mathit{true}$, and *false* otherwise.

Here $b, b_0, b_1 \in \mathbf{Bexp}$, $t_0, t_1, t \in \mathbb{B}$, and $\sigma \in \Sigma$.

# Expression Equivalence

As in the case of **Aexp**, our notion of semantic value for expressions leads to an equivalence relation between boolean expressions:

$$b_0 \sim b_1 \text{ iff } \forall \sigma \in \Sigma, \exists t \in \mathbb{B}.\ (b_0, \sigma) \mapsto t \wedge (b_1, \sigma) \mapsto t,$$

where $b_0,\ b_1 \in$ **Bexp**.

One way to look at this is that boolean expressions behave analogous to arithmetic expression except that the base has changed.

# Command Evaluation

Recall our grammar production for commands[2]:

C ::= **skip** | V := A | C ; C | **if** B **then** C **else** C **end** | **while** B **do** C **end**

In order to design a semantics for commands we have to answer the following questions:

1. What is the semantic domain for commands?
2. What does the evaluation function look like?

[2] Inductive definition of the syntactic domain **Com**?

# Command Evaluation

In our simple imperative language commands modify the state of the computation, that is, **commands map one state into another**. Therefore we define our evaluation function '$\mapsto$' as,

$$\mapsto : \ \mathbf{Com} \times \Sigma \to {\color{red}\Sigma}$$

and we write, given a command $c \in \mathbf{Com}$ and some state $\sigma \in \Sigma$ ,

$$(c, \sigma) \mapsto \sigma'$$

where $\sigma' \in \Sigma$ is the state after command $c$ has *fully executed*.

# Command Evaluation

Before we can give the full natural semantics for commands we need some more machinery. Consider,

$$(x := 5, \sigma) \mapsto \sigma'$$

where $x \in \mathbf{Loc}$, $5 \in \mathbf{I}$, and $\sigma$, $\sigma' \in \Sigma$.

Here, $\sigma'$ is the state $\sigma$ updated to have the value 5 in location $x$. We write,

$$\sigma' = \sigma[5/x].$$

# Command Evaluation

More formally, let $\sigma \in \Sigma$, $m \in \mathbb{I}$, and $x$, $y \in \mathbf{Loc}$. We write $\sigma[m/x]$ for the state obtained from $\sigma$ by replacing the contents in $x$ with $m$. We can define this functionally,

$$\sigma[m/x](y) = \begin{cases} m & \text{if } y = x \\ \sigma(y) & \text{if } y \neq x \end{cases}$$

$\Rightarrow$ States are "lookup tables" for values associated with locations.

Note that $\sigma[m/x] : \mathbf{Loc} \to \mathbb{I}$ is still considered a function from locations into the integers.

# Command Evaluation

**Exercises:** Let $\sigma' = \sigma_0[3/q]$ with $3 \in \mathbb{I}$ and $q \in \mathbf{Loc}$,

- Compute the value of $\sigma'(q)$.
- Compute the value of $\sigma'(k)$ with $k \in \mathbf{Loc}$ and $k \neq q$.

# Command Evaluation

Assume that all metavariables range over their appropriate domains and $\sigma$, $\sigma'$, and $\sigma'' \in \Sigma$.

$$\frac{}{(\mathbf{skip}, \sigma) \mapsto \sigma}$$

$$\frac{(a, \sigma) \mapsto m}{(x := a, \sigma) \mapsto \sigma[m/x]}$$

$$\frac{(b, \sigma) \mapsto \mathit{true} \qquad (c_0, \sigma) \mapsto \sigma'}{(\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1\ \mathbf{end}, \sigma) \mapsto \sigma'}$$

$$\frac{(b, \sigma) \mapsto \mathit{false} \qquad (c_1, \sigma) \mapsto \sigma'}{(\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1\ \mathbf{end}, \sigma) \mapsto \sigma'}$$

# Command Evaluation

$$\frac{(c_0, \sigma) \mapsto \sigma'' \qquad (c_1, \sigma'') \mapsto \sigma'}{(c_0; c_1, \sigma) \mapsto \sigma'}$$

$$\frac{(b, \sigma) \mapsto \mathit{false}}{(\textbf{while } b \textbf{ do } c \textbf{ end}, \sigma) \mapsto \sigma}$$

$$\frac{(b, \sigma) \mapsto \mathit{true} \qquad (c, \sigma) \mapsto \sigma'' \qquad (\textbf{while } b \textbf{ do } c \textbf{ end}, \sigma'') \mapsto \sigma'}{(\textbf{while } b \textbf{ do } c \textbf{ end}, \sigma) \mapsto \sigma'}$$