CMPS 203 Homework 2

1. 8-bit ARITH

The new abstract syntax for ARITH is

e ::= n
| e1 + e2
| e1 * e2
| e1 - e2

In syntax above, n denotes integer literals, e1 + e2 denotes sum, e1 * e2 is product and e1 - e2 represents subtraction. According to the assignment, “there is no way to directly refer to OVERFLOW values in a program”, so new definition (syntax) does not include OVERFLOW values.

2. New big-step operational semantic rules for ARITH

Due to the constraints that a variation of ARITH needs to be positive 8 bits number from 0 to 255, additional check for the input variant is added in the following big-step operational semantic rules:

$n \Downarrow n$ if $0 \leq n \leq 255$
$n \Downarrow$ OVERFLOW otherwise

$e1 + e2 \Downarrow$ OVERFLOW if :
one of (or both) $e1, e2 \Downarrow$ OVERFLOW or
$e1 \Downarrow n1$, $e2 \Downarrow n2$, and n (which is n1 + n2) overflows (i.e. $n1 + n2 < 0$ or $n1 + n2 > 255$)
$e1 + e2 \Downarrow n$ if:
$e1 \Downarrow n1$ and $e2 \Downarrow n2$, n is n1 + n2, $0 \leq n \leq 255$

e1 - e2 and e1 * e2 have similar rules:

e1 - e2 $\Downarrow$ OVERFLOW if:

one of (or both) e1, e2 $\Downarrow$ OVERFLOW or

e1 $\Downarrow$ n1, e2 $\Downarrow$ n2, and n (which is n1 - n2) overflows (i.e. n1 - n2 < 0 or n1 - n2 > 255)

e1 - e2 $\Downarrow$ n if

e1 $\Downarrow$ n1 and e2 $\Downarrow$ n2, n is n1 - n2, $0 \leq n \leq 255$

e1 * e2 $\Downarrow$ OVERFLOW if

one of (or both) e1, e2 $\Downarrow$ OVERFLOW or

e1 $\Downarrow$ n1, e2 $\Downarrow$ n2, and n (which is n1 * n2) overflows (i.e. n1 * n2 < 0 or n1 * n2 > 255)

e1 * e2 $\Downarrow$ n if

e1 $\Downarrow$ n1 and e2 $\Downarrow$ n2, n is n1 * n2, $0 \leq n \leq 255$.

The following is the big-step operational semantic rules for this question:

$$\frac{}{n \Downarrow n \quad n < 0 \text{ or } n > 255}$$

$$n \Downarrow \text{OVERFLOW}$$

$$\frac{}{n \Downarrow n \quad 0 \leq n \leq 255}$$

$$n \Downarrow n$$

$$\frac{e1 \Downarrow \text{OVERFLOW}}{e1 + e2 \Downarrow \text{OVERFLOW}}$$

$$\frac{e2 \Downarrow \text{OVERFLOW}}{e1 + e2 \Downarrow \text{OVERFLOW}}$$

$$\frac{}{n \Downarrow n \quad n < 0 \text{ or } n > 255}$$

$$\frac{e1 \Downarrow n1 \quad e2 \Downarrow n2 \quad n \Downarrow \text{OVERFLOW} \quad n \text{ is } n1 + n2}{e1 + e2 \Downarrow \text{OVERFLOW}}$$

$$\frac{}{n \Downarrow n \quad 0 \le n \le 255}$$

$$\frac{e1 \Downarrow n1 \quad e2 \Downarrow n2 \quad n \Downarrow n \quad n \text{ is } n1 + n2}{e1 + e2 \Downarrow n}$$

Evaluation for e1 - e2 and e1 * e2 work similarity with e1 + e2.

$e1 \Downarrow \text{OVERFLOW}$

$e1 - e2 \Downarrow \text{OVERFLOW}$

$e2 \Downarrow \text{OVERFLOW}$

$e1 - e2 \Downarrow \text{OVERFLOW}$

$n \Downarrow n \quad n < 0 \text{ or } n > 255$

$e1 \Downarrow n1 \quad e2 \Downarrow n2 \quad n \Downarrow \text{OVERFLOW} \qquad n \text{ is } n1 - n2$

$e1 - e2 \Downarrow \text{OVERFLOW}$

$$\frac{}{n \Downarrow n \quad 0 \le n \le 255}$$

$$\frac{e1 \Downarrow n1 \quad e2 \Downarrow n2 \quad n \Downarrow n \qquad n \text{ is } n1 - n2}{e1 - e2 \Downarrow n}$$

$$\frac{e1 \Downarrow \text{OVERFLOW}}{e1 * e2 \Downarrow \text{OVERFLOW}}$$

$$\frac{e2 \Downarrow \text{OVERFLOW}}{e1 * e2 \Downarrow \text{OVERFLOW}}$$

$$\frac{}{n \Downarrow n \quad n < 0 \text{ or } n > 255}$$

$$\frac{e1 \Downarrow n1 \quad e2 \Downarrow n2 \quad n \Downarrow \text{OVERFLOW} \qquad n \text{ is } n1 * n2}{e1 * e2 \Downarrow \text{OVERFLOW}}$$

$$\frac{}{n \Downarrow n \quad 0 \le n \le 255}$$

$$\frac{e1 \Downarrow n1 \quad e2 \Downarrow n2 \quad n \Downarrow n \qquad\qquad n \text{ is } n1 * n2}{e1 * e2 \Downarrow n}$$

4. Derivative rules for the repeat c until e construct:

$$\frac{< c, \sigma > \Downarrow \sigma 1 \quad < e, \sigma 1 > \Downarrow \text{true}}{< \text{repeat c until e}, \sigma > \Downarrow \sigma 1}$$

$$\frac{< c , \sigma > \Downarrow \sigma 1 \quad < e, \sigma 1 > \Downarrow \text{false} \quad < \text{repeat c until e}, \sigma 1 > \Downarrow \sigma 2}{< \text{repeat c until e}, \sigma > \Downarrow \sigma 2}$$

Explanation: This question can be solved by analyzing the difference of REPEAT … UNTIL < condition > and WHILE < condition > DO. The latter case is already given. REPEAT… UNTIL <condition> will be executed at least once, even if the condition is true before the execution. However, WHILE < condition > DO will be executed ONLY if condition satisfied when loop is entered. Another difference is the way of exiting loop, the former one needs condition to be true, while the latter one requires condition to be false.

Therefore, repeat c until e will first do c, evaluate e under the new state σ1, if it is true, then exit the loop and remains at state σ1; if it is false, continue next iteration and exit with state σ2.