A DEFINITION FOR OTHER OPERATIONS

A.0.1 Multiplication. Tensor Multiplication, in the form of $C = A \times B$, in general does not induce affine equality relations among the resulting tensor C and the input tensors A, B. Hence, we do not maintain any elementwise linear relations for the resulting tensor C and apply array smashing to C. In other words, we abstract the whole C into a scalar summary variable, i.e., $\alpha(C) = c$, and maintain the interval range over c. If $A \in \mathbb{R}^{n\times m} \wedge B \in \mathbb{R}^{m\times r}$, the interval range for tensor C can be computed by:

$$\sigma(c) = \sigma(a) \times \sigma(b) \times m$$

where $\sigma(a) = \cup_{1\le i\le \mathcal{N}_A}\sigma(a_i)$, $\sigma(b) = \cup_{1\le i\le \mathcal{N}_B}\sigma(b_i)$.

For example, suppose that both the input two-dimensional tensors $A \in \mathbb{R}^{5\times3}$ and $B \in \mathbb{R}^{3\times6}$ are partitioned into 2 partitions, and

$$\alpha(A_1) = a_1 \wedge \sigma(a) = 1 \wedge \alpha(A_2) = a_2 \wedge \sigma(a_2) = [2,3]\wedge$$
$$\alpha(B_1) = b_1 \wedge \sigma(b_1) = 4 \wedge \alpha(B_2) = b_2 \wedge \sigma(b_2) = [5,6]$$

Then, after $C = A \times B$, for C, we have:

$$\mathbb{I}_C = [0..4] \times [0..5] \wedge \alpha(C) = c\wedge$$
$$\sigma(c) = (\sigma(a_1) \cup \sigma(a_2)) \times (\sigma(b_1) \cup \sigma(b_2)) \times 3 = [12, 54]$$

A.0.2 ReLU. *ReLU* is a commonly used linear activation function in DNN implementations: $ReLU(A) = \max(0, A)$. For an assignment statement $B = ReLU(A)$ in the DL implementation, first we reuse the set of partitioning positions of A as that for the output tensor B. And for each partition A_i, we introduce A_i^{ReLU} to denote $ReLU(A_i)$. After that, for each partition B_i, we calculate its interval range by $\sigma(b_i) = \sigma(a_i) \cap [0, +\infty)$, and maintain the elementwise affine equality relation between B_i and $ReLU(A_i)$, i.e.,

$$b_i = a_i^{ReLU} \qquad i = \{1, \ldots, \mathcal{N}_B\}$$

which means that

$$\forall j \in \mathbb{I}_{(B_i.shape)}.B_i[j] = A_i^{ReLU}[j] \qquad i = \{1, \ldots, \mathcal{N}_B\}$$

For example, suppose that the input one-dimensional tensor A is partitioned into 2 partitions, and

$$\mathbb{I}_{A_1} = [0..2] \wedge \mathbb{I}_{A_2} = [3..9]$$
$$\alpha(A_1) = a_1 \wedge \sigma(a_1) = 1 \wedge \alpha(A_2) = a_2 \wedge \sigma(a_2) = [-2, 3]$$

Then, after $B = ReLU(A)$, for B, we have:

$$\mathbb{I}_{B_1} = [0..2] \wedge \mathbb{I}_{B_2} = [3..9]\wedge$$
$$\alpha(B_1) = b_1 \wedge \sigma(b_1) = 1 \wedge \alpha(B_2) = b_2 \wedge \sigma(b_2) = [0, 3]\wedge$$
$$b_i = a_i^{ReLU} \quad i = \{1, 2\}$$

A.0.3 Exp. The exponential function $Exp(A)$, used to compute e^A, is non-linear. Thus, for an assignment statement $B = Exp(A)$, we do not maintain affine equality relations between the output tensor B and the input tensor A. However, we reuse the set of partitioning positions of A as that for the output tensor B. After that, for each partition B_i, we calculate its interval range by

$$\sigma(b_i) = e^{\sigma(a_i)} \qquad i = \{1, \ldots, \mathcal{N}_B\}$$

For example, suppose that tensor A is the same one in Sect. A.0.2. After $B = Exp(A)$, for B, we have

$$\mathbb{I}_{B_1} = [0..2] \wedge \mathbb{I}_{B_2} = [3..9]\wedge$$
$$\alpha(B_1) = b_1 \wedge \sigma(b_1) = e \wedge \alpha(B_2) = b_2 \wedge \sigma(b_2) = [e^{-2}, e^3]$$

A.0.4 Softmax. $Softmax$ is a function that takes as input a tensor A, and normalizes it into a probability distribution B, where

$$B[i] = \frac{e^{A[i]}}{\sum_j e^{A[j]}}$$

For $Softmax$ function, we reuse the set of partitioning positions of A as that for the output tensor B. Let n be the size of A, and $\alpha(A_i) = a_i$ (for $1 \le i \le \mathcal{N}_A$). Then, for each partition B_i, we calculate its interval range of its summary variable $\alpha(B_i) = b_i$, by

$$\sigma(b_i) = \frac{e^{\sigma(a_i)}}{n * e^{\sigma(a)}}$$

where $\sigma(a) = \cup_{1\le j\le \mathcal{N}_A}\sigma(a_j)$ denotes the interval range of the whole A. Since *Softmax* is not linear, we do not maintain the elementwise affine equality relations between B_i and A_i.

For example, suppose that tensor A is the same one in Sect. A.0.2. After $B = Softmax(A)$, for B, we have

$$\mathbb{I}_{B_1} = [0..2] \wedge \mathbb{I}_{B_2} = [3..9]\wedge$$
$$\alpha(B_1) = b_1 \wedge \sigma(b_1) = \frac{e^1}{10 * e^{[-2,3]}} = [\frac{e^1}{10 * e^3}, \frac{e^1}{10 * e^{-2}}]\wedge$$
$$\alpha(B_2) = b_2 \wedge \sigma(b_2) = \frac{e^{[-2,3]}}{10 * e^{[-2,3]}} = [\frac{e^{-2}}{10 * e^3}, \frac{e^3}{10 * e^{-2}}]$$

A.0.5 While loop. RNN architectures contain while loops in the computation graph. These architectures always use non-linear activation functions such as gates in the loops, and the output of the loops is non-linear with respect to the input. Hence, we do not maintain any elementwise affine equality relations for loops. Moreover, we use array smashing to deal with all tensors appearing in the loop. Every tensor in the loop is abstracted by one scalar summary variable with its interval range. To cope with the loop, we use the classic Kleene iteration together with the widening operator [4] in the interval abstract domain to compute the interval range of each scalar variable after the loop.