`torch.Tensor.size`

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash E.\texttt{size}() \Rightarrow \mathit{shapeToTuple}(e), c}$$

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ c' = \{(k \geq 1) \land (0 \leq n < k)\}\end{array}}{\sigma \vdash E.\texttt{size}(n) \Rightarrow e[n+1], c \cup c'}$$

`torch.tensor`

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash \texttt{tensor}(E) \Rightarrow e, c}$$

`torch.Tensor.shape`

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash E.\texttt{shape} \Rightarrow \mathit{shapeToTuple}(e), c}$$

`torch.range`

$$\frac{\begin{array}{l}d \neq 0 \\ (e-s)/d > 0\end{array}}{\sigma \vdash \texttt{range}(s, e, d) \Rightarrow (1 + \lfloor (e-s)/d \rfloor), \emptyset} \qquad \text{Default: } s = 0, d = 1$$

`torch.Tensor.item`

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ c' = \{(\forall i = 1, 2, \ldots, k,\ e[i] = 1)\}\end{array}}{\sigma \vdash E.\texttt{item}() \Rightarrow (), c \cup c'}$$

`torch.split`

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = (n)@e[2{:}k] \\ e_2 = (n)@e[2{:}k] \\ \quad \cdots \\ e_{l-1} = (n)@e[2{:}k] \\ e_l = (n')@e[2{:}k] \quad \text{where } e[1] = n(l-1) + n',\ 0 < n' \leq n \\ c' = \{(k \geq 1)\}\end{array}}{\sigma \vdash \texttt{split}(E, n) \Rightarrow (e_1, e_2, \ldots, e_l), c \cup c'} \qquad l\text{-원소 tuple 형태로 반환}$$

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = (n_1)@e[2{:}k] \\ e_2 = (n_2)@e[2{:}k] \\ \quad \cdots \\ e_l = (n_l)@e[2{:}k] \\ c' = \{(k \geq 1) \land (e[1] = n_1 + n_2 + \cdots + n_l)\}\end{array}}{\sigma \vdash \texttt{split}(E, [n_1, n_2, \ldots, n_l]) \Rightarrow (e_1, e_2, \ldots, e_l), c \cup c'}$$

l-원소 tuple 형태로 반환

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = e[1{:}x]@(n)@e[x+2{:}k] \\ e_2 = e[1{:}x]@(n)@e[x+2{:}k] \\ \quad \cdots \\ e_{l-1} = e[1{:}x]@(n)@e[x+2{:}k] \\ e_l = e[1{:}x]@(n')@e[x+2{:}k] \quad \text{where } e[1] = n(l-1) + n',\ 0 < n' \leq n \\ c' = \{(k \geq 1) \land (0 \leq x < k)\}\end{array}}{\sigma \vdash \texttt{split}(E, n, x) \Rightarrow (e_1, e_2, \ldots, e_l), c \cup c'}$$

l-원소 tuple 형태로 반환

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = e[1{:}x]@(n_1)@e[x+2{:}k] \\ e_2 = e[1{:}x]@(n_2)@e[x+2{:}k] \\ \quad \cdots \\ e_l = e[1{:}x]@(n_l)@e[x+2{:}k] \\ c' = \{(k \geq 1) \land (0 \leq x < k) \land (e[x+1] = n_1 + n_2 + \cdots + n_l)\}\end{array}}{\sigma \vdash \texttt{split}(E, [n_1, n_2, \ldots, n_l], x) \Rightarrow (e_1, e_2, \ldots, e_l), c \cup c'}$$

l-원소 tuple 형태로 반환

`torch.zeros`, `torch.rand`, `torch.randn`

$$\forall \texttt{ft} \in \{\texttt{zeros}, \texttt{rand}, \texttt{randn}\}, \qquad \frac{}{\sigma \vdash \texttt{ft}(t_1, t_2, \ldots, t_l) \Rightarrow (t_1, t_2, \ldots, t_l), \emptyset}$$

`torch.mode`

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = \texttt{if } k = 0 \texttt{ then } e \texttt{ else } e[1{:}k-1]\end{array}}{\sigma \vdash \texttt{mode}(E) \Rightarrow (e', e'), c}$$

tuple 형태로 반환

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@e[n+2{:}k] \\ c' = \{(k \geq 1) \land (0 \leq n < k)\}\end{array}}{\sigma \vdash \texttt{mode}(E, n) \Rightarrow (e', e'), c \cup c'}$$

tuple 형태로 반환

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@(1)@e[n+2{:}k] \\ c' = \{(k \geq 1) \land (0 \leq n < k)\}\end{array}}{\sigma \vdash \texttt{mode}(E, n, True) \Rightarrow (e', e'), c \cup c'}$$ tuple 형태로 반환

$$\frac{\sigma \vdash \texttt{mode}(E, n) \Rightarrow (e, e), c}{\sigma \vdash \texttt{mode}(E, n, False) \Rightarrow (e, e), c}$$ tuple 형태로 반환

`torch.randint`

$$\frac{}{\sigma \vdash \texttt{randint}(low, high, e_s) \Rightarrow e_s, \emptyset}$$

$$\frac{}{\sigma \vdash \texttt{randint}(high, e_s) \Rightarrow e_s, \emptyset}$$

`torch.max`, `torch.min`

$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\sigma \vdash E \Rightarrow \_, c}{\sigma \vdash \texttt{ft}(E) \Rightarrow (), c}$$

$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@e[n+2{:}k] \\ c' = \{(k \geq 1) \land (0 \leq n < k)\}\end{array}}{\sigma \vdash \texttt{ft}(E, n) \Rightarrow (e', e'), c \cup c'}$$ tuple 형태로 반환

$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@(1)@e[n+2{:}k] \\ c' = \{(k \geq 1) \land (0 \leq n < k)\}\end{array}}{\sigma \vdash \texttt{ft}(E, n, True) \Rightarrow (e', e'), c \cup c'}$$ tuple 형태로 반환

$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\sigma \vdash \texttt{ft}(E, n) \Rightarrow (e, e), c}{\sigma \vdash \texttt{ft}(E, n, False) \Rightarrow (e, e), c}$$ tuple 형태로 반환

$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\begin{array}{c}\sigma \vdash E_1 \Rightarrow e_1, c_1 \\ \sigma \vdash E_2 \Rightarrow e_2, c_2\end{array}}{\sigma \vdash \texttt{ft}(E_1, E_2) \Rightarrow broadcast(e_1, e_2), c_1 \cup c_2 \cup broadcastable(e_1, e_2)}$$

`torch.sum`, `torch.mean`

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\sigma \vdash E \Rightarrow \_, c}{\sigma \vdash \texttt{ft}(E) \Rightarrow (), c}$$

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@e[n+2{:}k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \end{array}}{\sigma \vdash \texttt{ft}(E, n) \Rightarrow e', c \cup c'}$$

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = \texttt{if } 1 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } () \texttt{ else } (e[1]) \\ e_2 = \texttt{if } 2 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } () \texttt{ else } (e[2]) \\ \quad \cdots \\ e_k = \texttt{if } k \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } () \texttt{ else } (e[k]) \\ e' = e_1 @ e_2 @ \cdots @ e_k \\ c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \ldots, r,\ 0 \leq n_i < k)\} \end{array}}{\sigma \vdash \texttt{ft}(E, (n_1, n_2, \ldots, n_r)) \Rightarrow e', c \cup c'}$$

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@(1)@e[n+2{:}k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \end{array}}{\sigma \vdash \texttt{ft}(E, n, True) \Rightarrow e', c \cup c'}$$

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = \texttt{if } 1 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } (1) \texttt{ else } (e[1]) \\ e_2 = \texttt{if } 2 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } (1) \texttt{ else } (e[2]) \\ \quad \cdots \\ e_k = \texttt{if } k \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } (1) \texttt{ else } (e[k]) \\ e' = e_1 @ e_2 @ \cdots @ e_k \\ c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \ldots, r,\ 0 \leq n_i < k)\} \end{array}}{\sigma \vdash \texttt{ft}(E, (n_1, n_2, \ldots, n_r), True) \Rightarrow e', c \cup c'}$$

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\sigma \vdash \texttt{ft}(E, X) \Rightarrow e, c}{\sigma \vdash \texttt{ft}(E, X, False) \Rightarrow e, c}$$

$$\frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ w = \left\lfloor \frac{e[3]+2\times padding[0]-dilation[0]\times(kernel\_size[0]-1)-1}{stride[0]} \right\rfloor + 1 \\ h = \left\lfloor \frac{e[4]+2\times padding[1]-dilation[1]\times(kernel\_size[1]-1)-1}{stride[1]} \right\rfloor + 1 \\ e' = (e[1], out, w, h) \\ c_{dim} = \{(k=4)\} \\ c_w = \{(kernel\_size[0] \leq e[3] + 2 \times padding[0])\} \\ c_h = \{(kernel\_size[1] \leq e[4] + 2 \times padding[1])\} \\ c_{group} = \{(in\%groups = 0) \wedge (out\%groups = 0)\} \end{array}}{\sigma \vdash \texttt{Conv2d}(in, out, kernel\_size, stride, padding, dilation, groups)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h \cup c_{group}}$$

default values(when omitted): $stride = 1, padding = 0, dilation = 1, groups = 1$

$kernel_size, stride, padding, dilation$는 가로-세로별 2-tuple로도 들어갈 수 있음

이 경우를 위해 $stride[0], stride[1]$으로 표기함

만일 $stride$가 튜플이 아닌 스칼라라면 $stride[0]$ 또는 [1]은 $stride$ 값 자체를 의미