Approach

The following are the steps involved:

1. **Definitions:** Provide definitions for **Views**, **ShapeTracker** and **Mergeability**.

2. **Conditions and Functions:** Read the code thoroughly and attempt to extract/define under which attributes' constraints the class' methods/functions leads to ShapeTracker having a single-view. I intend to provide a rigorous analysis of the code using mathematical notation. I will then use the conditions determined to state **Propositions** that lead to mergeability, and also define **Functions** that either return $\emptyset$ or a **ShapeTracker** based on the conditions/contraints deduced from code analysis. Given a particular proposition and it's antecedents (constraints from code analysis), I intend to use the defined functions to conclude mergeability (which trivially has already been established by the analysis of the code)

3. **Proof methods:**

   - **Induction:**The proof will be based on induction, since a particular ShapeTracker may have a tuple of views applying the simplify function recursively.

4. **Tests:** After proving that a statement/proposition is correct, we need to verify that our conditions on the propositions do indeed return a single view.

Example Test

```
def test_shape_tracker(‘condition1‘, ‘condition2‘, ..., ‘conditionN‘):
    # Assert the conditions of the proposition meet their constraints
    assert condition1
    assert condition2
    ...
    assert conditionN

    ST1 = ShapeTracker
    ST2 = ShapeTracker
    # Perform the addition
    STm = add(ST1, ST2)
    # Construct the ShapeTracker and assert the result
    assert STm is not None
    assert len(STm.views) == len(ST1.views)  # Assuming we expect one view in the re
```

General

Mergeability Definition

Two ShapeTrackers with Views V_1 and V_2 are said to be mergeable if and only if

$$\text{card}(V_m) = \text{card}(V_1)$$

where V_m is the ShapeTracker ST_m View and card(.) denotes the cardinality of a ShapeTracker's View.

Proposition

Here we state the relevant **Propositions** we intend to prove.

- **Proposition 1:**
- **Proposition 2:**

Views

Definition of a View

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ where $s_i \in \mathbb{N}$ be a set of all possible shapes. Let $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ where $t_i \in \mathbb{N}$ be a set of all possible strides. Let $O = \{o \mid o \in \mathcal{C}\}$ be the set of all possible offsets. Define $M = \{m \mid m = (sint_1, sint_2, \ldots, sint_n)\}$ where $sint_i$ is a tuple of tuples: $sint_i = \text{sint}_{i1}, sint_{i2}, \ldots)$, and $sint$ is the set:

$$(\mathbb{N} \backslash (\text{V} \cup MN \cup SN)) \cup (V \backslash (N \cup MN \cup SN)) \cup (MN \backslash (N \cup V \cup SN)) \cup (SN \backslash (N \cup V \cup MN))$$

where $V = Variable$, $MN = MulNode$, $SN = SumNode$

Let $\mathcal{C} = \{c \mid c \in \{0, 1\}\}$ be a set of contiguous attributes.

Define a view $\mathcal{V}$ as a tuple: $\mathcal{V} = (\mathcal{S}, \mathcal{T}, O, M, \mathcal{C})$

where:

- $\mathcal{S}$ is the set of shapes,
- $\mathcal{T}$ is the set of strides,
- O is the set of offsets,
- M is the set of masks,
- $\mathcal{C}$ is the set of contiguous attributes.
- F is the set of functions, for brevity I will include the relevant functions only,

In addition, $\mathcal{V}$ supports the following functions:

function: un1d - To be completed, depnding on the constraints at hand

Define the function: unid : $S \times O \rightarrow \text{Tuple}_{\sin t}$ where the range is the set: $\{sint_1, sint_2, \ldots, sint_n\}$ and $sint$ is defined as above.

function: strides_for_shape - To be made precise

Define the function strides_for_shape : $S \times O \rightarrow \text{Tuple}_{\sin t}$

Define strides to be a sequence $(\Pi_{i=1}^{d} o_i, \Pi_{i=1}^{d} o_2, \ldots, \Pi_{i=1}^{d} o_n)$ where shape is given by: $o = (o_1, o_2, \ldots, o_n)$. Thus, strides_for_shape is:

$$\text{strides\_for\_shape} = \begin{cases} \emptyset & \text{if shape } \notin \mathcal{C} \\ \text{canonicalise\_strides} & \text{(as defined above)} \end{cases}$$

function: canonicalise_strides - To be made precise

Define the function canonicalise_strides:

$$\text{canonicalise\_strides} : O \times S \rightarrow \{0, 1\}^n$$

It returns a sequence $(t_1, t_2, \ldots, t_n)$ where:

$$t_i = \begin{cases} 0 & \text{if } o_i = 1 \\ s_i & \text{if } o_i \neq 1 \quad \forall c \in \{1, 2, \ldots, n\} \end{cases}$$

function: __add__ - To be completed, however I used case 1 on the Induction proof

Given two views V_1 and V_2, the function add is defined as follows:

$$\text{\_\_add\_\_} : V_s \times V_s \rightarrow V_s, \text{where } V_s \text{ is defined as the set of all possible Views.}$$

where V_s is the set of all views V such that:

1. If V_2.contiguous $= 1$, then $\text{add}_{C(V_1,V_2)} = V_s$. Here we introduce the dot (.) notation to access the attributes of $\mathcal{V}$.
2. If V_1.contiguous $= 1$ and V_1.shape $= V_2$.shape, then $\text{add}_{C(V_1,V_2)} = V_s$.
3. If V_1.contiguous $= 1$ and $V_1.\text{size}(C) = V_2.\text{size}(C)$, then $\text{add}_{C(V_1,V_2)} = \text{ret}$, where the functions size and reshape are defined.
4. If V_1.mask $\in \emptyset$, and the following conditions are also true:

   Let origin $= \text{unid}(V_2.\text{shape}, V_1.\text{offset})$ for a given shape and offset.

   $\forall d_1 \in \{0, 1, \ldots, |S| - 1\}, \quad \forall st \in S, \text{ if } st \neq 0, \text{ then } \forall d_2 \in \{0, 1, \ldots, |O| - 1\}, \quad \forall (o, s_1) \in \text{zip}(O, U), \text{ let } s_1' = s_1 - o, \text{ if } s_1' \neq 0, \text{ then}$

Definition of a ShapeTracker

Define a shapetracker $\mathcal{ST}$ as a tuple: $\mathcal{ST} = (\mathcal{V}, \mathcal{F})$

where:

- $\mathcal{V}$ is the set of sequences of views,
- F is the set of functions, for brevity I will include the relevant functions only,

In addition, $\mathcal{ST}$ supports the following functions:

Function `__add__`

The computation of the merged ShapeTracker ST_m, denoted as $ST_m = ST_1 + ST_2$, with views V_m, V_1, and V_2, given ST_1 and ST_2, is determined by the function add defined as follows:

$$\text{add} : ST_s \times ST_s \to ST_s,$$

where ST_s is defined as the set of all possible ShapeTrackers. For each View in ST_1 and ST_2, we have specific conditions and rules for combining them into ST_m.

Proof of Mergeability by Induction

Given: ShapeTrackers ST_1^{initial} with a Views attribute $V^{\text{initial}} = \{\text{ v}_1^{\text{initial}}, v_2^{\text{initial}}, \ldots, v_n^{\text{initial}}\} := \text{ST}_1^{\text{initial}}.\text{views}$ and ST_2^{initial} with a Views attribute $W^{\text{initial}} = \{w_1^{\text{initial}}, w_2^{\text{initial}}, \ldots, w_n^{\text{initial}}\} := ST_2^{\text{initial}}.\text{views}$ V^{initial} and W^{initial} may be $\emptyset$.

Let ST_n^{append} be a sequence of appended ShapeTrackers with the Views $V^{\text{append}} = \{v_1^{\text{append}}, v_2^{\text{append}}, \ldots, v_n^{\text{append}}\} :=$ $\text{ST}^{\text{append}}.\text{views}$ $(\forall n \in |W_2^{\text{initial}}|)$, and $v_i^{\text{append}} = ST_1^{\text{initial}}.\text{views}{+}ST_2^{\text{initial}}.\text{views}_i$.

Define the sequence ST_n^{merged} to be sequnece of merged ShapeTrackers with the Views given by $V^{\text{merged}} = \{v_1^{\text{merged}}, v_2^{\text{merged}}, \ldots, v_n^{\text{merged}}\} := ST^{\text{merged}}.\text{views}$ such that $ST_n^{\text{merged}} = ST_1^{\text{initial}}$ and $ST_n^{\text{append}} = \text{simplify}(ST_n^{\text{append}})$ for $(0 \leq n < |W_2^{\text{initial}}|)$

Define the function $\_\_\text{init}\_\_(ST_n^{\text{append}}) = (ST_{n+1}^{\text{append}})$, this function returns a ShapeTracker class,

Define $\text{simplify}(ST_n^{\text{append}})$ as:

$$\text{simplify}(\text{ST}_n^{\text{append}}) = \begin{cases} \text{simplify}(ST_n^{\text{append}}) \text{ for n } (0 \leq n < |W_2^{\text{initial}}|) \text{ and merged} \neq \emptyset \\ ST_n^{\text{merged}} \text{ otherwise} \end{cases}$$

Define $\_\_\text{add}\_\_(ST_1^{\text{initial}}, n) = ST_n^{\text{merged}}$, this function returns the nth term of ST_n^{merged}.

The recursion is given by:

$$\text{\_\_add\_\_}(\text{ST}_1^{\text{initial}}, n) = \begin{cases} \text{simplify}(\text{\_\_init\_\_}(ST_n^{\text{append}}.\text{views})) \text{ for } (0 \leq n < |W_2^{\text{initial}}|) \\ ST_1^{\text{initial}} \text{ if } n = 1 \end{cases}$$

Proposition: If ST_1^{initial}.views.contiguous = 1. Define the function __add__(ST_1^{initial}, n) as above. Assume both ST_1^{initial}.views and ST_2^{initial}.views are not $\emptyset$.
Then ST_1^{initial} and ST_2^{initial} are mergeable. Assume $|ST_1^{\text{initial}}.\text{views}| = w$.

Proof: Base case: for $\emptyset$ if $n = 1$

$$\text{ST}_1^{\text{merged}} = \text{\_\_add\_\_}(ST_1^{\text{initial}}, 1) = ST_1^{\text{initial}}, \text{ hence } |ST_1^{\text{merged}}| = |ST_1^{\text{initial}}|$$

Inductive step: for $n = k$

Assume for some $k \geq 2$, $\Big|ST_k^{\text{merged}}| = |\text{\_\_add\_\_}(ST_1^{\text{initial}}, k)| = |ST_1^{\text{initial}}|$

We need to show that: $\Big|ST_{k+1}^{\text{merged}}| = |\text{\_\_add\_\_}(ST_1^{\text{initial}}, k+1)| = |ST_1^{\text{initial}}|$

By definition of __add__, $\text{ST}_{k+1}^{\text{merged}} = \text{simplify}(\text{\_\_init\_\_}(ST_{k+1}^{\text{append}}.\text{views}))$
the function __init__ creates a new ShapeTracker by appending a view of ST_2^{initial} to ST_1^{initial}.views, hence the ShapeTracker created will have $ST^{append}.\text{views} = |ST^{\text{initial}}.\text{views}| + 1$.

The function simplify computes: merged $= ST^{\text{appended}}.\text{views}_{n-1} + ST^{append}.\text{views}_n$ as one of its conditions. merged is the sum of two views and then added according to the function __add__ belonging to the class View object defined above.

Change notation $n = w$, so we may index ST^{append}.views using w.
Since ST_1^{initial}.views.contiguous = 1, by the definition of __add__ from above defined View class object we can conclude that merged $\notin \emptyset$, i.e., it will be a particular view object. Hence, by the definition of simplify function a new ShapeTracker will be initialized by __init__, and it will have a view = merged + $ST^{\text{appended}}.\text{views}(1 \leq w < w - 1)$, since the cardinality of a finite site is its number of elements we can conlude that this new view will be of size $|ST_1^{\text{initial}}|$.

When simplify is again called recursively on the above returned ShapeTracker, it will just return merged ShapeTracker, which is the equal to the appended ShapeTracker because merged will return $\emptyset$, since we are at the initial size of ST_1^{initial} and its views are not mergeable.
The inductive step holds, hence $\forall \mathbf{N}$