A1. The part of my Python regular expression responsible for matching string literals changes from `'[^']*'` to `'(?:\\'|[^'])*'`. That is, the characters within a string literal are either (A) `\'` combinations or (B) non-`'` characters.

A2. I remove the backslashes using a construction like

```
tokens = [re.sub(r"\\'", r"'", token) for token in tokens]
```

B. The basic idea is that, on any given step of computation, the current string stored in the SARF machine encodes the complete configuration of the Turing machine. To be specific, let M be any Turing machine with input alphabet $\Sigma = \{0, 1\}$. A configuration of M consists of a state q, a tape head location $h \in \mathbb{N}$, and the contents t of the tape. Although the tape is infinite, its essential contents at any given time are finite. That is, the tape consists of finitely many characters, followed by infinitely many blanks that do not need to be recorded in t.

The configuration (q, h, t) can be encoded into a bit string $\langle q, h, t\rangle$ according to any of various encoding schemes. Here are a couple of details. Let's agree that the encoding of $\langle q, h, t\rangle$ consists of $\langle q\rangle$, followed by a separator, then $\langle h\rangle$, then a separator, then $\langle t\rangle$. The encoding of t begins with a bit that signals whether the rest of the encoding is the raw contents of the tape, or some non-trivial encoding of those contents. That is, if the essential tape consists entirely of 0s and 1s, so that it is a bit string u, then a valid encoding of t is the bit string $0u$. If the essential tape contains characters other than 0 and 1, then it must be encoded into a bit string u according to some scheme, and $1u$ is a valid encoding of t.

Without loss of generality, we can assume that M, before it accepts or rejects, erases its tape and parks its tape head at the left end of the tape. How? We can alter M to a new Turing machine M' that simulates M, but with special markers wrapped around M's tape. Just before M accepts or rejects, M' scans to the right marker, and then scans back to the left marker, blanking the tape along the way. When the tape head reaches the left marker, M' blanks that space, moves left, and accepts or rejects as M would. Therefore, for any M there exists an equivalent M' that always accepts or rejects with a blank tape and leftward tape head. [We pulled a similar trick when showing that the language of any PDA is context-free.]

So there are unique accepting and rejecting configurations. Also, for any input w, there is a unique starting configuration for M on w. These configurations are encoded using A, R, and S as follows.

1. The string A consists of $\langle q_{\text{acc}}\rangle$, a separator, $\langle 0\rangle$, a separator, and then 0.
2. The string R consists of $\langle q_{\text{rej}}\rangle$, a separator, $\langle 0\rangle$, a separator, and then 0.
3. Let S be the string consisting of $\langle q_0\rangle$, a separator, $\langle 0\rangle$, a separator, and then 0. Therefore Sw encodes the starting configuration of M on input w.

Finally, the function F describes how one configuration changes to the next. For example, if M contains a transition $\delta(q, a) = (r, b, \mathrm{R})$, then for all $h \geq 0$ and all t that contain the symbol a in the hth cell,

$$F(\langle q\rangle, \langle h\rangle, \langle t\rangle) = \langle r\rangle, \langle h+1\rangle, \langle t'\rangle,$$

where t' is t with a replaced by b in the hth cell.

[Here's a subtle question: Does any SARF machine have an equivalent Turing machine?]

C. I will use Rice's theorem, after proving three statements. First, $SMALL_{TM}$ is a property of recognizable languages, because whether $\langle M\rangle$ belongs to $SMALL_{TM}$ depends only on $L(M)$. Second, it is easy to see that $SMALL_{TM}$ is non-empty: For any small Turing machine M, $\langle M\rangle \in SMALL_{TM}$. Third, to see that $SMALL_{TM}$ does not contain all Turing machine encodings $\langle M\rangle$, consider these two facts:

- There are infinitely many recognizable languages. Indeed, there are infinitely many languages consisting of a single bit string each, and each of these languages is finite and hence recognizable.
- There are finitely many small Turing machines (ignoring renamings of the states). For a Turing machine with p states is specified by choices for q_0, q_{acc}, q_{rej}, and $\delta(q, a)$ for each combination of $q \in Q$ and $a \in \Gamma$. Hence there are at most $p^3 \cdot (6p)^{3p}$ Turing machines with p states, and at most this many small Turing machines:

$$\sum_{p=2}^{99} p^3 \cdot (6p)^{3p} \approx 6.33 \cdot 10^{829}.$$

Because there are more recognizable languages than small Turing machines, there must exist a recognizable language $L(M)$ that is not the language of any small Turing machine N, and $SMALL_{TM}$ does not contain $\langle M\rangle$ for this M. Thus $SMALL_{TM}$ is a nontrivial property of recognizable languages, and $SMALL_{TM}$ is undecidable by Rice's theorem.