

Theoretical Computer Science Stack Exchange is a question and answer site for theoretical computer scientists and researchers in related fields. It's 100% free, no registration required.

Take the 2-minute tour ×

Membership problem for certain class of unrestricted grammars

Consider an arbitrary context-free grammar G over the alphabet $\{0, 1, \bar{0}, \bar{1}\}$. To the productions of this grammar, add two fixed non-context-free productions P : $\bar{0}\bar{0} \rightarrow \epsilon$ and $\bar{1}\bar{1} \rightarrow \epsilon$. Call the resulting grammar G^P standing for " G augmented with the productions P ".

Is it possible to give an algorithm which takes a grammar G^P and a string s over $\{0, 1, \bar{0}, \bar{1}\}$ and decides whether $s \in \mathcal{L}(G^P)$?

context-free decidability

asked Oct 1 '14 at 12:46

 **Amit. S**
48 3

Interestingly, while the answer seems to be "no", I think that if $L(G)$ is regular, then so is $L(G^P)$. Essentially, an NFA for $L(G)$ can be transformed into one for $L(G^P)$ by iteratively adding ϵ -transitions (s, ϵ, t) whenever you have paths $(s, \bar{0}, p, 0, t)$, $(s, \bar{0}, p, \epsilon, q, 0, t)$, $(s, \bar{1}, p, 1, t)$, $(s, \bar{1}, p, \epsilon, q, 1, t)$ or $(s, \epsilon, p, \epsilon, t)$, and finally performing ϵ -elimination. — **Klaus Draeger** Oct 2 '14 at 12:29

Yes that is true. In fact, the question arose from a problem in program analysis (liveness based garbage collection). We circumvented the problem by approximating the CFG to a strongly regular grammar (Mohri-Nederhoff transformation), and then doing the P simplifications on the resulting NFA in exactly the way Klaus Draeger mentions. — **Amit. S** Oct 2 '14 at 12:59

1 Answer

This class of grammars is undecidable. Here is a rough idea about how to use it to emulate Turing machines.

At each point, the current partially expanded word would look like

[tape to the left][head][tape to the right]

Here:


- [tape to the left], after applying P , only contains characters $\bar{0}$ and $\bar{1}$.
- [tape to the right], after applying P , only contains characters 0 and 1.
- [head] is a single nonterminal, which encodes both the head state and the character at the head position.

Suppose that the head is in state S , and the character under the head is $i \in \{0, 1\}$. Then the head is represented by nonterminal S_i . If it needs to transition to state T , replace the current character with j , and move to the left, there are two transitions $S_i \rightarrow 0T_0j$ and $S_i \rightarrow 1T_1j$. If it needs to move to the right instead, there are two transitions $S_i \rightarrow jT_0\bar{0}$ and $S_i \rightarrow jT_1\bar{1}$. In some sense, the head has to "guess" the character in the direction it is moving to by producing the matching character. If the guess is wrong, the invariant on [tape to the left] or [tape to the right] would be violated, and it would never recover.

When the machine halts, the head should "consume" its tape on both sides by "guessing" and producing matching characters. After that, it should produce empty word. As a result, empty word would be a member of such grammar if and only if the corresponding Turing machine halts.

edited Oct 3 '14 at 10:05

answered Oct 2 '14 at 11:20

 **abacabadabacaba**
216 1 1 8

I am not sure that I understand your reduction. Here is my doubt: If the given Turing machine has N states, then isn't the number of unrestricted productions required to emulate the Turing machine related to N ? But my problem allows just two fixed unrestricted productions. — **Amit. S** Oct 2 '14 at 13:41

@Amit.S I provided some more explanation of transitions in the answer. — **abacabadabacaba** Oct 2 '14 at 17:21