A Counter-example for "A Simpler Construction for Showing the Intrinsically Exponential Complexity of the Circularity Problem for Attribute Grammars"

Jens M. Dill

87-815

March 1987

Department of Computer Science
Cornell University
Ithaca, New York 14853-7501

A Counter-example for "A Simpler Construction for Showing the Intrinsically Exponential Complexity of the Circularity Problem for Attribute Grammars"

JENS M. DILL

Cornell University, Ithaca, New York

ABSTRACT. Jazayeri [*J. ACM 28*, 4 (Oct. 1981), 715-720] proposes a simpler construction for use in the proof by Jazayeri, Ogden, and Rounds [*Commun. ACM 18*, 12 (Dec. 1975), 697-706] that the circularity problem for attribute grammars has inherent exponential complexity. The simplification introduces a flaw that invalidates the proof. Correcting the flaw leaves the new construction only slightly simpler than the old.

KEY WORDS AND PHRASES: alternating Turing machines, attribute grammars, circularity problem, computational complexity, exponential time

CR CATEGORIES: D.3.4, F.2.2

1. *Introduction*

Jazayeri [3] describes a simpler construction that can be used in the proof [2] that the circularity problem for attribute grammars is of intrinsically exponential complexity. While there is no reason to doubt the correctness of the proof in [2], the new construction suffers from a flaw that prevents it from being directly useful in the proof.

The central idea of the proof is that given a pair consisting of a description of an alternating Turing machine [1,4] and a sample input string, it is possible to construct an attribute grammar that is circular if and only if the given machine

accepts the given input string. We will exhibit an alternating Turing machine and an input string for which the constructed attribute grammar is circular even though the machine does not accept the input string. We construct the counter-example in the same order as the construction outline in [3], using the same notation.

2. *Counterexample*

The given alternating Turing machine is a 6-tuple $M = (Q, \Sigma, \delta, q_0, F, U)$, where

$Q = \{q_0, q_1, q_2, q_3\}$ — set of states

$\Sigma = \{a, b\}$ — input and tape alphabet

$\delta = \{(q_0, a) \mapsto (q_1, a, \mathsf{R}),$ — "next move" relation

$(q_0, a) \mapsto (q_2, a, \mathsf{R}),$

$(q_1, a) \mapsto (q_3, a, \mathsf{L}),$

$(q_2, b) \mapsto (q_3, b, \mathsf{L})\}$

q_0 — start state

$F = \{q_3\}$ — set of accepting states

$U = \{q_0\}$ — set of universal states

and the input is the string aa of length $n=2$. The machine does not accept the given input, as shown by the following configuration sequence:*

$$q_0\,[a]a \rightarrow (q_1\,a[a], q_2\,a[a]) \rightarrow (q_3\,[a]a, \text{reject}) \rightarrow (\text{accept}, \text{reject}) \rightarrow \text{reject}$$

The constructed grammar will have five non-terminals: a start symbol S, and a symbol for each of the machine states, Q_0 through Q_3. Each non-terminal other than S will have nine attributes, partitioned into three cells, $C(1)$, $C(2)$, and $C(3)$. $C(2)$ represents the tape square currently under the tape head, and the other tape square

* The symbol under the tape head is shown in brackets. Note that an alternating Turing machine accepts its input only when all computations reachable from a universal state end in acceptance.

is represented by $C(1)$ or $C(3)$, depending on whether it is to the left or right of the tape head. The three attributes in cell $C(j)$ are named $C(j, a)$, $C(j, b)$, and $C(j, *)$.

The intended purpose of these attributes was to establish attribute dependencies such that if and only if the given machine accepts the given input there will be a cyclic dependency that passes twice through each cell representing a tape square: first downward through either the $C(j, a)$ or $C(j, b)$ attribute and then back upward through the $C(j, *)$ attribute. Each downward leg of the cycle was intended to track the history of the contents of a single tape square, and the corresponding upward leg was to provide a return to the top for connection to the downward leg for the next tape square. The constructed attribution rules* for our counter-example are

$S \rightarrow Q_0$ — reflects initial configuration.

$Q_0.C(2, a) := Q_0.C(3, *)$

$Q_0.C(3, a) := Q_0.C(2, *)$

$Q_0 \rightarrow Q_1 Q_2$ — reflects universal machine move

$Q_1.C(1, a) := Q_0.C(2, a)$ — from (q_0, a) to (q_1, a, R) and (q_2, a, R).

$Q_1.C(2, a) := Q_0.C(3, a)$

$Q_1.C(2, b) := Q_0.C(3, b)$

$Q_2.C(1, a) := Q_1.C(1, *)$

$Q_2.C(2, a) := Q_1.C(2, *)$

$Q_2.C(2, b) := Q_1.C(2, *)$

$Q_2.C(3, a) := Q_1.C(3, *)$

$Q_2.C(3, b) := Q_1.C(3, *)$

$Q_0.C(2, *) := Q_2.C(1, *)$

$Q_0.C(3, *) := Q_2.C(2, *)$

---

* We show here only those attribute equations that contribute edges to the dependency graph. Any attributes not defined here can be assigned an arbitrary constant value.

$Q_1 \rightarrow Q_3$ — reflects existential machine move

$Q_3.C(3, a) := Q_1.C(2, a)$ — from (q_1, a) to (q_3, a, L).

$Q_3.C(2, a) := Q_1.C(1, a)$

$Q_3.C(2, b) := Q_1.C(1, b)$

$Q_1.C(2, *) := Q_3.C(3, *)$

$Q_1.C(1, *) := Q_3.C(2, *)$

$Q_2 \rightarrow Q_3$ — reflects existential machine move

$Q_3.C(3, b) := Q_2.C(2, b)$ — from (q_2, b) to (q_3, b, L).

$Q_3.C(2, a) := Q_2.C(1, a)$

$Q_3.C(2, b) := Q_2.C(1, b)$

$Q_2.C(2, *) := Q_3.C(3, *)$

$Q_2.C(1, *) := Q_3.C(2, *)$

$Q_3 \rightarrow t$ — reflects machine final state.

$Q_3.C(1, *) := Q_3.C(1, a) + Q_3.C(1, b)$

$Q_3.C(2, *) := Q_3.C(2, a) + Q_3.C(2, b)$

$Q_3.C(3, *) := Q_3.C(3, a) + Q_3.C(3, b)$

Figure 1 illustrates an attributed tree for this grammar in which the dependency graph contains a cycle. The cycle exists because the attribution rules for productions representing universal machine moves do not carry enough information from the left subtree to the right. Specifically, for cells that were not under the tape head at the start of the universal move, both the a and the b attributes on the right depend on the $*$ attribute from the left. Thus, the right subtree has lost the information that would tell it what symbols are contained in those cells. In our example, the cycle is completed because the right subtree is able to assume that the second symbol on the tape is a b, when in fact it is an a.

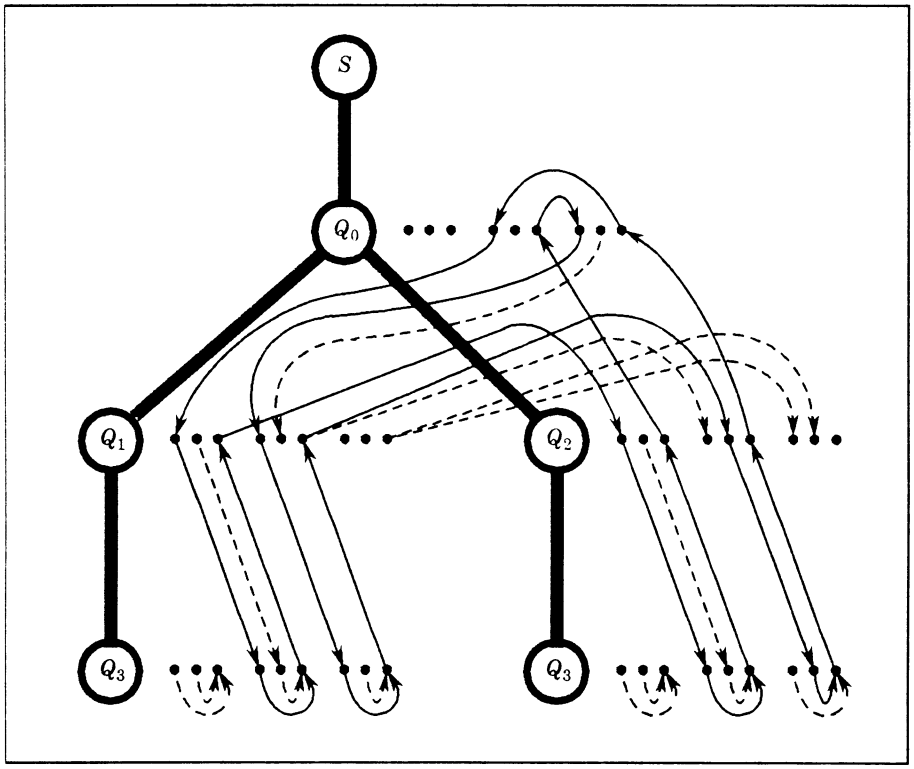

Figure 1. The cycle in the attribute dependency graph is shown by the solid arrows. Dependencies not in the cycle are shown as dashed arrows. The attribute instances for each tree node are grouped into cells, with $C(1)$ on the left and $C(3)$ on the right. Within each cell, the attributes are a, b, and $*$, again from left to right.

3. *Conclusion*

The flaw in the construction in [3] was introduced in the process of simplifying the construction used in the original proof [2]. The original proof has twice as many attributes per non-terminal, with the C group carrying information down the tree and a mirror-image P group carrying the same information back up. The connections at the leaves link $C(j, a)$ and $C(j, b)$ to $P(j, a)$ and $P(j, b)$, respectively, and $P(j, *)$ to $C(j, *)$ for the return. At the top, the C group is connected as shown here, and the P group connections are analagous to the connections at the leaves in this construction. In the analogue of the universal-move case, the $P(j, a)$ and $P(j, b)$

attributes of the root of the left subtree are connected to the $C(j, a)$ and $C(j, b)$ attributes of the root of the right subtree. Thus, there is no possibility of cross-linking the paths that represent a and b.

Thus, we can correct the flaw by re-introducing the P attributes and redefining the dependencies to be analagous to those in [2]. This in turn undoes much of the simplification promised by [3]. What simplification remains is counterbalanced by the fact that more people seem to understand the pushdown automata used in [2] than understand the alternating Turing machines used in [3].

REFERENCES

1. CHANDRA, A, AND STOCKMEYER, L.J. Alternation. Proc. 17th IEEE Symp. on Foundations of Computer Science, Houston, Texas, 1976, pp. 98-108.
2. JAZAYERI, M., OGDEN, W.F., AND ROUNDS, W.C. The intrinsically exponential complexity of the circularity problem for attribute grammars. *Commun. ACM 18*, 12 (Dec. 1975), pp. 697-706.
3. JAZAYERI, M. A simpler construction for showing the intrinsically exponential complexity of the circularity problem for attribute grammars. *J.ACM 28*, 4 (Oct. 1981), pp. 715-720.
4. KOZEN, D. On parallelism in Turing machines. Proc. 17th IEEE Symp. on Foundations of Computer Science, Houston, Texas, 1976, pp. 89-97.