Undecidable Problems for Context-free Grammars

Hendrik Jan Hoogeboom
Universiteit Leiden (NL)

Abstract. We discuss some basic undecidable problems for context-free languages, starting from *Valid and invalid computations of TM's: a tool for proving CFL problems undecidable*, Section 8.6 of the famous "Cinderella Book" by Hopcroft and Ullman [HU79].

As original source for the undecidability results the book refers to Bar-Hillel etal. [B-HPS61] and Ginsburg&Rose [GR63]. The main results relating TM computations to CFG's are from Hartmanis [Ha67]. Before that connection, undecidability results were usually obtained by reduction from PCP, see here Section 2, where the connection to TM computations is hidden. It seems that the PCP is still vary useful when considering undecidability for linear grammars.

Thus in Section 1 we will consider the basic intersection problem for context-free grammars using a direct coding of TM computations. In Section 2 we improve this to linear grammars as an application of the undicidability of PCP. General problems for (linear) context-free grammars are the topic of Section 3 which starts by considering invalid computations.

It is assumed that basic notions of context-free grammars and Turing machines are known. In particular it is should be understood how to construct CFG's for the languages $\{x\#x^R\# \mid x \in \Sigma^*\}^*$ and $\{x\#y^R \mid x, y \in \Sigma^*, x \neq y\}$. Here $\#$ is a symbol not in Σ and $\cdot^R$ denotes the mirror operator.

$L(G_1) \cap L(G_2) = \varnothing$	disjointness	Theorem 3
$L(G) = \Sigma^*$	universality	Theorem 10
$L(G) = R$		"
$L(G_1) = L(G_2)$	equality	"
$L(G)$ is ambiguous	ambiguity	Theorem 8
$L(G)$ is regular	regularity	Theorem 11

1 Coding Turing Machine Computations

The basic tool for proving the various undecidability results for CF languages is a proper coding of TM computations.

A *configuration* of TM $\mathcal{M}$ is a string $w \in \Gamma^* Q \Gamma^*$, where Γ is the tape alphabet of $\mathcal{M}$ and Q is the state set of $\mathcal{M}$. Thus $w = xqy$ with $x, y \in \Gamma^*$ and $q \in Q$, meaning that the tape has left and right parts x and y, and the TM is in state q with its head at the first letter of x. The tape alphabet Γ contains a special symbol `B` to represent blank cells. Clearly we only represent a finite number of `B`'s in a configuration. A single computational step from configuration x to configuration y is denoted as $x \vdash y$.

A *valid computation* (of length n) is represented by string of the form $w_0\#w_1^R\#w_2\#w_3^R\#\ldots\#w_{2k-1}^R\#$ ($n = 2k-1$ is odd) or $w_0\#w_1^R\#w_2\#w_3^R\#\ldots\#w_{2k}\#$ ($n = 2k$ is even) in which the consecutive w_i represent the consecutive configurations from initial to halting. Adding the mirror at alternating positions makes

the coding feasible for context-free grammars, as will be clear next. Thus, the requirements for a valid computation are

- $w_0 = \mathtt{B}q_0x$ is the initial configuration for some input $x \in \Sigma^*$,
- consecutive configurations are obtained by a TM step $w_i \vdash w_{i+1}$ for $0 \leq i < n$, and
- w_n is accepting, i.e., of the form yhz where h is a halting state.

The language of valid computations of TM $\mathcal{M}$ is denoted by valid($\mathcal{M}$).

Theorem 1. *Given a* TM $\mathcal{M}$ *one effectively constructs two* CFG G_1, G_2 *such that* valid($\mathcal{M}$) $= L(G_1) \cap L(G_2)$.

Proof. We use the fact that TM steps ony locally rewrites the tape contents. Let $x, y \in \Gamma^*$. Then

$$\begin{array}{rlll} x\,ZpX\,y \vdash x\,qZY\,y & \text{if} & (p, X, q, Y, L) \in \delta & \text{'move left'} \\ x\,pX\,y \vdash x\,Yq\,y & \text{if} & (p, X, q, Y, R) \in \delta & \text{'move right'} \\ x\,pX\,y \vdash x\,qY\,y & \text{if} & (p, X, q, Y, S) \in \delta & \text{'stay'} \end{array}$$

If we take note of the mirror operation, consecutive configurations are coded as $\#\,xZpXy\#\,y^RYZqx^R\,\#$, move left for even steps, etc.

Now we define two grammars: G_1 generates repeated occurrences of even steps $w_i\#w_{i+1}^R$, while G_2 takes care of the odd steps $w_i^R\#w_{i+1}$.

Let $L_e = \{u\#v^R\# \mid u, v \in \Gamma^*Q\Gamma^*, u \vdash v\}$ and $L_o = \{u^R\#v\# \mid u, v \in \Gamma^*Q\Gamma^*, u \vdash v\}$. Then G_1 generates $L_e^*\,(\{\lambda\} \cup \Gamma^*H\Gamma^*)$, while similarly G_2 generates $\mathtt{B}q_0(\{\mathtt{B}\} \cup \Sigma^+)\,L_o^*\,(\{\lambda\} \cup \Gamma^*H\Gamma^*)$. Note how the first and last position contain initial and final configurations. Then the intersection of both languages yields computations of $\mathcal{M}$.

Checking whether w_n is final is simple, the form of the possible strings is regular. This restriction can be added to the grammars. □

Now the first undecidability result for CFG's is a simple consequence from the fact that it is undecidable whether the language accepted by a Turing machine is empty. We then immediately apply this to Theorem 1.

Proposition 2. *It is undecidable whether* valid($\mathcal{M}$) *is empty, for* TM $\mathcal{M}$.

Theorem 3 ([B-HPS61, Thm 6.1 a])**.** *It is undecidable whether* $L(G_1) \cap L(G_2) = \varnothing$ *for two* CFG *'s* G_1, G_2.

Given two CFG's G_1, G_2 it is easy to construct a CFG for the concatenation $L(G_1)L(G_2)$ of their languages. Similarly we can consider the language $L(G_1)\,\flat\,L(G_2)\,\flat$ where 'marker' $\flat$ is a new symbol. Note that $L(G_1) \cap L(G_2) \neq \varnothing$ iff $L(G_1)\,\flat\,L(G_2)\,\flat$ contains a square, i.e., a word of the form ww.

Similarly we can observe that $L(G_1) \cap L(G_2) \neq \varnothing$ iff $L(G_1)\,\flat\,L(G_2)^R$ contains a palindrome, i.e., a word w for which $w = w^R$.

Thus we have.

Theorem 4. *For* CFG G *it is undecidable whether*

- $L(G)$ *contains a square.*
- $L(G)$ *contains a palindrome.*

2 Post Correspondence Problem

We can improve Theorem 3 using PCP. It enables us to simplify the proof, and to restrict the type of CFG used to linear grammars.

An instance of *Post Correspondence Problem* [Po46] is given by two lists of equal length, $A = (x_1, x_2, \ldots, x_n)$ and $B = (y_1, y_2, \ldots, x_n)$, of words over an alphabet Σ. It has a solution if there is a sequence of integers $i_1, i_2, \ldots, i_m$, $m > 0$, such that $x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}$.

Also the undecidability of PCP is usually obtained by coding TM computations. (Usually as a first step one considers *modified* PCP where the solutions of PCP are restricted to $i_1 = 1$.)

Proposition 5. PCP *is undecidable.*

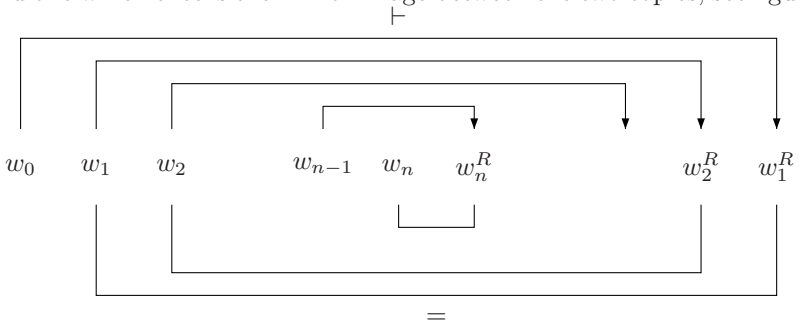
A CFG is called linear if it only has productions of the form $A \to xBy$ and $A \to x$ (with A, B nonterminal, and x, y terminal).

Consider a list $A = (x_1, x_2, \ldots, x_n)$ over an alphabet Σ. It is quite easy to encode the strings of words generated by A together with their indexes using a linear grammar G_A. Its productions are $A \to x_i A i, A \to x_i \# i$, $i = 1, \ldots n$. The language generated is $x_{i_1} x_{i_2} \ldots x_{i_m} \# i_m \ldots i_2 i_1$, $1 \leq i_1, i_2, \ldots, i_m \leq n$, over the alphabet $\Sigma \cup \{1, 2, \ldots, n\} \cup \{\#\}$.

Given two lists A, B obviously $L(G_A) \cap L(G_B)$ codes solutions of the corresponding PCP. Hence we have.

Theorem 6. *It is undecidable whether* $L(G_1) \cap L(G_2) = \varnothing$ *for two* linear *grammars* G_1, G_2.

Alternatively, we can restrict ourselves to linear grammars in Theorem 1 if we change the representation for a computation $w_0, w_1, \ldots, w_n$ to $w_0 \# w_1 \# w_2 \# \ldots \# w_n \# w_n^R \# \ldots \# w_2^R \# w_1^R \#$. This representation can be obtained as the intersection of two linear languages, one which checks the steps, and one which checks the mirror image between the two copies, see figure below.



This coding was used in [BB74] to obtain a characterization for recursively enumerable languages. Just code the first string w_0 in a different alphabet from the other strings.

Theorem 7. *Every recursively enumerable language can be expressed as the homomorphic image of the intersection of two linear context-free languages.*

The same technique can be used to obtain recursively enumerable languages as the *quotient* of two linear languages.

A context-free grammar G is *ambiguous* if there exists a string x such that there are two different derivation trees for x in G. For a list of words A the grammar G_A as given above is unambiguous: each string has a unique derivation (tree). Given two lists we may assume that the grammars G_A and G_B have different nonterminals. In particular that means they have no derivations in common. Thus the grammar with productions $S \to A, S \to B$ together with the productions for G_A and G_B is unambiguous, unless there is a string that is in both $L(G_A)$ and $L(G_B)$. The latter is undecidable. Hence

Theorem 8 ([Ca62, Fl62, CS63])**.** *It is undecidable whether G is ambiguous for linear grammar G.*

By the way, note that it is very easy to code solutions of PCP directly as palindromes. Consider the linear grammar $G_{A,B}$ with productions are $A \to x_i A y_i^R$, $S \to x_i \# y_i^R$, $i = 1, \ldots n$. Then its language contains a *palindrome* iff the original PCP has a solution. Thus we directly see that it is undecidable whether $L(G)$ contains a palindrome (see Theorem 4), even for linear grammars. To show that the same for containing a *square* use productions $S \to A\#$, $A \to x_i A i \mid x_i B i$, $B \to j B y_j \mid j \# y_j$. A string of the form $x_{i_1} \ldots x_{i_k} j_\ell \ldots j_1 \# y_{j_1} \ldots y_{j_\ell} i_k \ldots i_1 \#$ is a square iff it codes a solution for PCP.

3 Coding Invalid Computations

We can obtain an even more striking result by focussing on the strings that do code invalid computations (which include both non-accepting computations and strings that do not code a computation at all).

Thus we define $\text{invalid}(\mathcal{M}) = (\Gamma \cup Q \cup \{\#\})^* - \text{valid}(\mathcal{M})$. It is easy to see that a string of the form $x_1 \# x_2 \# \ldots x_m \#$ is invalid if one or more of the following cases hold.

- x_i not a configuration
- x_1 not initial
- x_n not final
- not $x_i \vdash x_{i+1}^R$ for even i
- not $x_i^R \vdash x_{i+1}$ for odd i

Hence we have the following result.

Theorem 9. *Given a* TM $\mathcal{M}$ *one effectively constructs a* CFG G *such that* $\text{invalid}(\mathcal{M}) = L(G)$.

In fact the grammar G can be taken to be linear [reference?]

Proof. Configurations (and initial or final configurations) form a regular language, so also their complements are regular, and hence the strings described by the first three cases.

Recall from Theorem 1 and its proof that TM steps ony locally rewrite part of the tape. Pairs of configurations $x_i \# x_{i+1}$ such that $x_i \vdash x_{i+1}^R$ are of one (or more) of the three forms $x\,ZpX\,y\#y^R\,YZq\,x^R$, $x\,pX\,y\#y^R\,qY\,x^R$, or $x\,pX\,y\#y^R\,Yq\,x^R$ ($x, y \in \Gamma^*$, $p, q \in Q$, $X, Y \in \Gamma$) depending on the instructions of the TM.

It is a little tedious, but the language for the fourth case can be seen to be linear in the same way as the language $\{x\#y^R \mid x, y \in \Sigma^*, x \neq y\}$. The fifth and last case is symmetric. □

A TM accepts the empty language iff all its computations are invalid. Consequently for context-free grammars we have the following undecidability results (which also hold when restricted to linear grammars, as proved in [BB74] using PCP).

Theorem 10 ([B-HPS61, Thm 6.2 ab, Thm 6.3 d] [BB74, Lemma 1])**.**

1. *It is undecidable whether $L(G) = \Sigma^*$ for a* CFG *G over Σ.*
2. *It is undecidable whether $L(G) = R$ for* CFG *G and regular language R.*
3. *It is undecidable whether $L(G_1) \subseteq L(G_2)$ for* CFG*'s G_1, G_2.*
4. *It is undecidable whether $L(G_1) = L(G_2)$ for* CFG*'s G_1, G_2.*

Proof. Undicidability of (1) by the coding result above and the undecidability of the emptiness of TM languages. Then (2) follows by taking $R = \Sigma^*$, and (3,4) by choosing G_1 such that $L(G_1) = \Sigma^*$. □

In the second item we assume both G and R are part of the input of the problem. For fixed R the problem may be undecidable (as for $R = \Sigma^*$) or decidable (as for $R = \varnothing$).

Note that $R \subseteq L(G)$ is undecidable, by having $R = \Sigma^*$. The question $L(G) \subseteq R$ on the other hand *is* decidable, it holds iff $L(G) \cap (\Sigma^* \setminus R) = \varnothing$. This is decidable as emptiness of context-free languages is decidable (and CFL are effectively closed under intersection with regular languages).

Note that for any family of languages effectively closed under union the last two items are equivalent. On the one hand $K \subseteq L$ iff $K \cup L = L$, while on the other $K = L$ iff both $K \subseteq L$ and $L \subseteq K$.

Not only we cannot decide whether $L(G) = R$ for a given regular R, but we cannot decide whether $L(G)$ is regular at all.

Theorem 11 ([B-HPS61, Thm 6.3 c])**.** *It is undecidable whether $L(G)$ is regular for* CFG *G.*

Proof. Start with a fixed nonregular context-free language $L_0 \subseteq \Sigma^*$. Let $\#$ be a symbol not in Σ.

Now for given G consider $L_1 = L_0 \# \Sigma^* \cup \Sigma^* \# L(G)$. L_1 is context-free. We argue that L_1 is regular iff $L(G) = \Sigma^*$.

Assume we find a string $w \notin L(G)$ then $L_1 \cap (\Sigma^* \# w) = L_0 \# w$. As L_0 is nonregular, also $L_0 \# w$ is nonregular. Context-free languages are closed under intersection with regular languages so L_1 cannot be regular.

On the other hand, when $L(G) = \Sigma^*$ then $L_1 = \Sigma^* \# \Sigma^*$, which is regular.

So deciding regularity of L_1 would be equivalent to deciding whether $L(G) = \Sigma^*$, which is impossible. □

The above proof technique works for larger classes of grammars, and is called *Greibach's Theorem* [Gr68].

Theorem 12. *Let $\mathcal{C}$ be a family of languages effectively closed under union and concatenation with regular sets, and for which equality to Σ^* is undecidable (for sufficiently large Σ). Let $\mathcal{P}$ be a proper subset of $\mathcal{C}$, which contains all regular languages, and is closed under $/a$ (single letter quotient). Then membership of $\mathcal{P}$ is undecidable for $\mathcal{C}$.*

Even when it is known that $L(G)$ regular, finding a FSA A with $L(A) = L(G)$ is not effective. This is seen as follows. Let $\Delta = \Gamma \cup Q \cup \{\#\}$, and consider $L_0 = \text{invalid}(\mathcal{M}) \cup q_0 \, \Sigma^+ \, \# \Delta^*$. Apart from all invalid computations, L_0 also contains all computations on non-empty words. L_0 is effectively context-free.

We consider two cases. If $\lambda \in L(M)$, then $L_0 = \Delta^* - \{w\}$, where w codes the accepting computation for λ. Otherwise, if $\lambda \notin L(M)$, then $L_0 = \Delta^*$.

The language L_0 is regular in both cases, but because of the empty tape halting problem it is impossible to decide which of the two forms it has.

Acknowledgements

Parts of this overview were inspired by answers on stackexchange, like for instance by users reinierpost and Sylvain.

References

[BB74] B.S. Baker, R.V. Book: Reversal-bounded multipushdown machines. Journal of Computer and System Sciences 8 (1974) 315–332. DOI 10.1016/S0022-0000(74)80027-9

[B-HPS61] Y. Bar-Hillel, M. Perles, E. Shamir: On formal properties of simple phrase structure grammars. Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung 14 (1961) 143–172. DOI 10.1524/stuf.1961.14.14.143

[Ca62] D.G. Cantor: On The Ambiguity Problem of Backus Systems. Journal of the ACM 9 (Oct. 1962) 477–479. DOI 10.1145/321138.321145

[CS63] N. Chomsky, M.P. Schützenberger: The Algebraic Theory of Context-Free Languages. Computer Programming and Formal Systems (P. Braffort and D. Hirschberg, eds.) Studies in Logic and the Foundations of Mathematics 35 (1963) 118–161. DOI 10.1016/S0049-237X(08)72023-8

[Fl62] R.W. Floyd: On ambiguity in phrase structure languages. Communications of the ACM 5 (Oct. 1962) 526. DOI 10.1145/368959.368993

[Gr68] S.A. Greibach: A Note on Undecidable Properties of Formal Languages. Mathematical Systems Theory 2 (1968) 1–6. DOI 10.1007/BF01691341

[GR63] S. Ginsburg, G.F. Rose: Some Recursively Unsolvable Problems in ALGOL-Like Languages. Journal of the ACM 10 (January 1963) 29–47. DOI 10.1145/321150.321153

[Ha67] J. Hartmanis: Context-free Languages and Turing Machine Computations. Mathematical Aspects of Computer Science, Proceedings of Symposia in Applied Mathematics, vol. 19, American Mathematical Society, 1967, pages 42–51.

[HU79] J.E. Hopcroft, J.D. Ullman: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, 1979.

[Po46] E. L. Post: A variant of a recursively unsolvable problem. Bulletin American Mathematical Society 52 (1946) 264–268. DOI 10.1090/S0002-9904-1946-08555-9

May 2015 (updated February 2, 2017)