Complexity Theory More Computability

Charles E. Hughes

COT6410 – Spring 2021 Notes

Constant time:
Not amenable to Rice's

Constant Time

- **CTime = { M | ∃K [M** halts in at most **K** steps independent of its starting configuration **] }**
- **RT** cannot be shown undecidable by Rice's Theorem as it breaks property 2
  - Choose **M1** and **M2** to each Standard Turing Compute (STC) **ZERO**
  - **M1** is **R** (move right to end on a zero)
  - **M2** is $\mathcal{L}\ \mathcal{R}$ **R** (time is dependent on argument)
  - **M1** is in **CTime; M2** is not , but they have same I/O behavior, so **CTime** does not adhere to property 2

Quantifier Analysis

- **CTime = { M | ∃K ∀C [STP(M, C, K)] }**
- This would appear to imply that **CTime** is not even re. However, a TM that only runs for **K** steps can only scan at most **K** distinct tape symbols. Thus, if we use unary notation, **CTime** can be expressed
- **CTime = { M | ∃K** $\forall C_{|C| \leq K}$ **[STP(M, C, K)] }**
- We can dovetail over the set of all TMs, **M**, and all **K**, listing those **M** that halt in constant time.

Mortal Turing Machines

- A TM, ***M***, is **mortal** if it halts on all initial IDs, whether the tape is finitely or infinitely marked.
- A TM is **immortal** if it is not mortal, that is, if there some starting configuration, with the tape either finitely or infinitely marked, on which it does not halt
- The possibility of infinitely marked tapes is essential to the idea of mortality

Complexity of CTime

- Can show it is equivalent to the **Mortality Problem** for TM's with **Infinite Tapes** (not unbounded but truly infinite and potentially infinitely marked)
- This was shown in 1966 to be undecidable*.
- It was also shown to be re, just as we have done so for **CTime**.
- Details Later

*P.K. Hooper, The undecidability of the Turing machine immortality problem, ***J. Symbolic Logic 31*** (1966) 219-234.

Finite Convergence for Concatenation of Context-Free Languages

Relation to Real-Time
(Constant Time) Execution

Powers of CFLs

Let G be a context free grammar.

Consider $L(G)^n$

Question1: Is $L(G) = L(G)^2$?

Question2: Is $L(G)^n = L(G)^{n+1}$, for some finite $n>0$?

These questions are both undecidable.

Think about why question1 is as hard as whether or not L(G) is Σ^*.

Question2 requires much more thought.

$L(G) = L(G)^2$?

- The problem to determine if $L = \Sigma^*$ is Turing reducible to the problem to decide if $L \bullet L \subseteq L$, so long as L is selected from a class of languages C over the alphabet Σ for which we can decide if $\Sigma \cup \{\lambda\} \subseteq L$.
- Corollary 1:
  The problem "is $L \bullet L = L$, for L context free or context sensitive?" is undecidable

$L(G) = L(G)^2$? is undecidable

- Question: Does $L \bullet L$ get us anything new?
  - i.e., Is $L \bullet L = L$?
- Membership in a CFL is decidable.
- Claim is that $L = \Sigma^*$ iff

  (1) $\Sigma \cup \{\lambda\} \subseteq L$; and

  (2) $L \bullet L = L$
- Clearly, if $L = \Sigma^*$ then (1) and (2) trivially hold.
- Conversely, we have $\Sigma^* \subseteq L^* = \cup_{n\geq 0} L^n \subseteq L$
  - first inclusion follows from (1); second from (2)

Finite Power Problem

- **The problem to determine, for an arbitrary context free language L, if there exist a finite n such that $L^n = L^{n+1}$ is undecidable.**
- **Let M be some Turing Machine**
- **$L_1 = \{ C_1\# C_2^R \$ \mid C_1, C_2$ are configurations $\}$,**
- **$L_2 = \{ C_1\#C_2^R\$C_3\#C_4^R \ \ldots \ \$C_{2k-1}\#C_{2k}^R\$ \mid$ where $k \geq 1$ and, for some i, $1 \leq i < 2k$, $C_i \Rightarrow_M C_{i+1}$ is false $\}$,**
- **$L = L_1 \cup L_2 \cup \{\lambda\}$.**

Undecidability of $\exists n\ L^n = L^{n+1}$

- **L is context free.**
- **Any product of L_1 and L_2, which contains L_2 at least once, is L_2. For instance, $L_1 \bullet L_2 = L_2 \bullet L_1 = L_2 \bullet L_2 = L_2$.**
- **This shows that $(L_1 \cup L_2)^n = L_1^n \cup L_2$.**
- **Thus, $L^n = \{\lambda\} \cup L_1 \cup L_1^2 \ldots \cup L_1^n \cup L_2$.**
- **Analyzing L_1 and L_2 we see that $L_1^n \cup L_2 \neq L_2$ just in case there is a word $C_1 \# C_2^R \$ C_3 \# C_4^R \ldots \$ C_{2n-1} \# C_{2n}^R \$$ in L_1^n that is not also in L_2.**
- **But then there is some valid trace of length 2n.**
- **L has the finite power property iff M executes in constant time.**

Missing Step

- We have that **CT** (Constant-Time) is many-one reducible to Finite Power Problem (**FPC**) for CFLs
- This means that if **CT** is unsolvable, so is **FPC** for CFLs.
- However, we still lack a proof that **CT** is unsolvable. To achieve that we actually start with the 1966 result* that the mortality problem for TMs with potentially **infinite** initial tape markings is re/non-recursive
  Note that the uniform halting problem for TMs with **finite** initial tape markings is not even re – This is **TOTAL**

*P.K. Hooper, The undecidability of the Turing machine immortality problem, ***J. Symbolic Logic 31*** (1966) 219-234.

Infinite Tape Markings

- If a TM halts for all tape markings, even if the TM’s initial tape is infinitely marked, then there is some fixed maximum amount of the tape that the machine can traverse
- Why is the above so?
- Well, informally, if there was no bound built into the TM’s table then it would be at the mercy of its data to decide when to stop and that would lead a search for a zero (a divider between items on the tape) to take an infinite amount of time

Uniformly Halting

- A TM, ***M***, uniformly halts if there is some **n**, dependent only on ***M***, such that ***M*** halts in at most **n** steps no matter what initial finite input it is given
- Note that this concepts is restricted to normal TMs that start with a finitely marked tape
- Clearly, a TM that uniformly halts runs in constant time

T uniformly halts iff T is mortal

- Let ***T*** be a TM that does not uniformly halt. If any finite ID does not lead to a halt, then clearly ***T*** is immortal.
- Assume then that ***T*** does not uniformly halt but all finite ID's cause it to halt.
- Let $\mathcal{I}$ be the set of all ID's such that, for each $\boldsymbol{I} \in \mathcal{I}$, when ***T*** starts in ***I*** it will eventually scan each square of the tape containing a symbol of ***I*** before it scans a square not containing a symbol of ***I***.
- Let $\{\mathbf{q_1}, \ldots, \mathbf{q_m}\}$ be the states of ***T***. We define a forest of **m** trees, one for each state of ***T***, such that the **j**-th tree has root $\mathbf{q_j}$.
- If $\boldsymbol{I_0}, \boldsymbol{I_1} \in \mathcal{I}$, and $\mathbf{q_j}$ is a symbol of $\boldsymbol{I_0}$ and $\boldsymbol{I_1}$, and $\boldsymbol{I_1} = \boldsymbol{\sigma} \boldsymbol{I_0}$ or $\boldsymbol{I_1} = \boldsymbol{I_0} \boldsymbol{\sigma}$, where $\boldsymbol{\sigma}$ is a tape symbol, then $\boldsymbol{I_0}$ is a parent of $\boldsymbol{I_1}$ in the **j**-th tree.
- Note that when ***T*** starts in $\boldsymbol{I_1}$, the square containing $\boldsymbol{\sigma}$ is scanned after every other square of $\boldsymbol{I_1}$ but before any square not in $\boldsymbol{I_1}$.

T uniformly halts iff T is mortal

- Since ***T*** does not uniformly halt but every finite ID causes it to halt, at least one of the trees of the forest must be infinite.
- The degree of each vertex in each tree is finite (it is bounded by the number of tape symbols). By Koenig's Infinity Lemma, at least one of the trees must have an infinite branch. Therefore, there exists an infinite ID which causes ***T*** to travel an infinite distance on the tape. It follows that ***T*** is immortal.

Undecidability of Finite Convergence for Operators on Formal Languages

Relation to Real-Time
(Constant Time) Execution

Simple Operators

- Concatenation
  - $A \bullet B = \{ xy \mid x \in A \ \& \ y \in B \}$

- Insertion
  - $A \triangleright B = \{ xyz \mid y \in A, xz \in B, x, y, z \in \Sigma^* \}$
  - Clearly, since x can be λ, $A \bullet B \subseteq A \triangleright B$

K-insertion

- $A \triangleright^{[k]} B = \{ x_1y_1x_2y_2 \ldots x_ky_kx_{k+1} \mid$
  $y_1y_2 \ldots y_k \in A,$
  $x_1x_2 \ldots x_kx_{k+1} \in B,$
  $x_i, y_j \in \Sigma^*\}$

- Clearly, $B \bullet A \subseteq A \triangleright^{[k]} B$, for all k>0

Iterated Insertion

- $A\ (1) \triangleright^{[n]} B = A \triangleright^{[n]} B$
- $A\ (k+1) \triangleright^{[n]} B = A \triangleright^{[n]} (A\ (k) \triangleright^{[n]} B)$

Shuffle

- Shuffle (product and bounded product)
  - $A \diamond B = \cup_{j \geq 1} A \triangleright^{[j]} B$
  - $A \diamond^{[k]} B = \cup_{1 \leq j \leq k} A \triangleright^{[j]} B = A \triangleright^{[k]} B$

- One is tempted to define shuffle product as $A \diamond B = A \triangleright^{[k]} B$ where
  $k = \mu y [A \triangleright^{[j]} B = A \triangleright^{[j+1]} B]$
  but such a k may not exist – in fact, we will show the undecidability of determining whether or not k exists

More Shuffles

- Iterated shuffle
  - $A \diamond^{0} B = A$
  - $A \diamond^{k+1} B = (A \diamond^{[k]} B) \diamond B$

- Shuffle closure
  - $A \diamond^{*} B = \cup_{k \geq 0} (A \diamond^{[k]} B)$

Crossover

- Unconstrained crossover is defined by
  $A \otimes_u B = \{ wz, yx \mid wx \in A \text{ and } yz \in B \}$

- Constrained crossover is defined by
  $A \otimes_c B = \{ wz, yx \mid wx \in A \text{ and } yz \in B,$
  $|w| = |y|, |x| = |z| \}$

Who Cares?

- People with no real life (me?)
- Insertion and a related deletion operation are used in biomolecular computing and dynamical systems
- Shuffle is used in analyzing concurrency as the arbitrary interleaving of parallel events
- Crossover is used in genetic algorithms

Some Known Results

- Regular languages, A and B
  - A • B is regular
  - A $\triangleright^{[k]}$ B is regular, for all k>0
  - A ◇ B is regular
  - A ◇* B is not necessarily regular
    - Deciding whether or not A ◇* B is regular is an open problem

More Known Stuff

- CFLs, A and B
  - A $\bullet$ B is a CFL
  - A $\triangleright$ B is a CFL
  - A $\triangleright^{[k]}$ B is not necessarily a CFL, for k>1
    - Consider A=a^nb^n; B = c^md^m and k=2
    - Trick is to consider (A $\triangleright^{[2]}$ B) $\cap$ a*c*b*d*
  - A $\diamond$ B is not necessarily a CFL
  - A $\diamond^*$ B is not necessarily a CFL
    - Deciding whether or not A $\diamond^*$ B is a CFL is an open problem

Immediate Convergence

- $L = L^2$?
- $L = L \triangleright L$?
- $L = L \diamond L$?
- $L = L \diamond^* L$?
- $L = L \otimes_c L$?
- $L = L \otimes_u L$?

Finite Convergence

- $\exists k{>}0\ L^k = L^{k+1}$
- $\exists k{\geq}0\ L\ (k) \rhd L = L\ (k{+}1) \rhd L$
- $\exists k{\geq}0\ L \rhd^{[\,k\,]} L = L \rhd^{[\,k+1\,]} L$
- $\exists k{\geq}0\ L \Diamond^{k} L = L \Diamond^{k+1} L$
- $\exists k{\geq}0\ L\ (k) \otimes_c L = L\ (k{+}1) \otimes_c L$
- $\exists k{\geq}0\ L\ (k) \otimes_u L = L\ (k{+}1) \otimes_u L$

- $\exists k{\geq}0\ A\ (k) \rhd B = A\ (k{+}1) \rhd B$
- $\exists k{\geq}0\ A \rhd^{[\,k\,]} B = A \rhd^{[\,k+1\,]} B$
- $\exists k{\geq}0\ A \Diamond^{k} B = A \Diamond^{k+1} B$
- $\exists k{\geq}0\ A\ (k) \otimes_c B = A\ (k{+}1) \otimes_c B$
- $\exists k{\geq}0\ A\ (k) \otimes_u B = A\ (k{+}1) \otimes_u L$

Finite Power of CFG

- Let G be a context free grammar.
- Consider $L(G)^n$
- Question1: Is $L(G) = L(G)^2$?
- Question2: Is $L(G)^n = L(G)^{n+1}$, for some finite $n>0$?
- These questions are both undecidable.
- We showed that question1 is as hard as whether or not $L(G)$ is Σ^*.
- Question2 required more work.

1981 Results

- Theorem 1:
  The problem to determine if $L = \Sigma^*$ is Turing reducible to the problem to decide if
  $L \bullet L \subseteq L$, so long as L is selected from a class of languages C over the alphabet Σ for which we can decide if $\Sigma \cup \{\lambda\} \subseteq L$.
- Corollary 1:
  The problem "is $L \bullet L = L$, for L context free or context sensitive?" is undecidable

Proof #1

- Question: Does $L \bullet L$ get us anything new?
  - i.e., Is $L \bullet L = L$?
- Membership in a CSL is decidable.
- Claim is that $L = \Sigma^*$ iff

  (1) $\Sigma \cup \{\lambda\} \subseteq L$; and

  (2) $L \bullet L = L$
- Clearly, if $L = \Sigma^*$ then (1) and (2) trivially hold.
- Conversely, we have $\Sigma^* \subseteq L^* = \cup_{n\geq 0} L^n \subseteq L$
  - first inclusion follows from (1); second from (2)

Subsuming $\bullet$

- Let $\oplus$ be any operation that subsumes concatenation, that is $A \bullet B \subseteq A \oplus B$.
- Simple insertion is such an operation, since $A \bullet B \subseteq A \triangleright B$.
- Unconstrained crossover also subsumes $\bullet$,
  $A \otimes_c B = \{ wz, yx \mid wx \in A \text{ and } yz \in B\}$

$L = L \oplus L$?

- Theorem 2:
  The problem to determine if $L = \Sigma^*$ is Turing reducible to the problem to decide if $L \oplus L \subseteq L$, so long as $L \bullet L \subseteq L \oplus L$ and L is selected from a class of languages C over Σ for which we can decide if $\Sigma \cup \{\lambda\} \subseteq L$.

Proof #2

- Question: Does $L \oplus L$ get us anything new?
  – i.e., Is $L \oplus L = L$?
- Membership in a CSL is decidable.
- Claim is that $L = \Sigma^*$ iff
  (1) $\Sigma \cup \{\lambda\} \subseteq L$; and
  (2) $L \oplus L = L$
- Clearly, if $L = \Sigma^*$ then (1) and (2) trivially hold.
- Conversely, we have $\Sigma^* \subseteq L^* = \cup_{n\geq 0} L^n \subseteq L$
  – first inclusion follows from (1); second from (1), (2) and the fact that $L \bullet L \subseteq L \oplus L$

Quotients of CFLs

Quotients of CFLs (Trace-Like Sequences)

Let L1 = L(G1) = { $ # Y_0 # Y_1 # Y_2 # Y_3 # … # Y_{2j} # Y_{2j+1} # }
where $Y_{2i} \Rightarrow Y_{2i+1}$, $0 \le i \le j$.
This checks the even/odd steps of an even length computation.

Now, let L2=L(G2)=
{X_0 $ # X_0 # X_1 # X_2 # X_3 # X_4 # … # X_{2k-1} # X_{2k}# Z_0 #}
where $X_{2i-1} \Rightarrow X_{2i}$, $1 \le i \le k$ and Z_0 is a unique halting configuration.

This checks the odd/steps of an even length computation and includes an extra copy of the starting number prior to its $.

If a Turing Machine Trace

Let L1 = L(G1) = { $ # Y_0^R # Y_1 # Y_2^R # Y_3 # … # Y_{2j}^R # Y_{2j+1} # }
where $Y_{2i} \Rightarrow Y_{2i+1}$, $0 \le i \le j$.
This checks the even/odd steps of an even length computation.

Now, let L2=L(G2)=
{X_0 $ # X_0^R # X_1 # X_2^R # X_3 # X_4^R # … # X_{2k-1} # X_{2k}^R# Z_0 #}
where $X_{2i-1} \Rightarrow X_{2i}$, $1 \le i \le k$ and Z_0 is a unique halting configuration.

This checks the odd/steps of an even length computation and includes an extra copy of the starting number prior to its $.

Quotients of CFLs (results)

L1 = $\{ \$ \# Y_0 \# Y_1 \# Y_2 \# Y_3 \# Y_4 \# \ldots \# Y_{2k-1} \# Y_{2j} \# Y_{2j+1} \# \}$
L2 = $\{X0 \$ \# X_0 \# X_1 \# X_2 \# X_3 \# X_4 \# \ldots \# X_{2k-1} \# X_{2k} \# Z_0 \#\}$
Now, consider the quotient of L2 / L1 . The only way a member of L1 can match a final substring in L2 is to line up the \$ signs. But then they serve to check out the validity and termination of the computation. Moreover, the quotient leaves only the starting point (the one on which the machine halts.) Thus,

L2 / L1 = $\{ X_0 \mid$ the system being traced halts$\}$.

Since deciding the members of an re set is in general undecidable, we have shown that membership in the quotient of two CFLs is also undecidable.
Note: Intersection of two CFLs is a CSL but quotient of two CFLs is an re set and, in fact, all re sets can be specified by such quotients.

Quotients of CFLs (precise)

- Let (n, ((a1,b1,c1,d1) , … ,(ak,bk,ck,dk)) be some factor replacement system with residues. Define grammars G1 and G2 by using the 4k+4 rules

  G : $F_i \rightarrow 1^{ai}F_i1^{ci} \mid 1^{ai+bi}\#1^{ci+di}$ $\quad 1 \le i \le k$

  $T_1 \rightarrow \# F_i T_1 \mid \# F_i \#$ $\quad 1 \le i \le k$

  $A \rightarrow 1 A 1 \mid \$ \#$

  $S_1 \rightarrow \$T_1$

  $S_2 \rightarrow A T_1 \# 1^{z0} \#$ $\quad Z_0$ is 0 for us

  G1 starts with S_1 and G2 with S_2
- Thus, using the notation of writing Y in place of 1^Y,

  L1 = L(G1) = $\{ \$ \#Y_0 \# Y_1 \# Y_2 \# Y_3 \# \ldots \# Y_{2j} \# Y_{2j+1} \# \}$

  where $Y_{2i} \Rightarrow Y_{2i+1}$, $0 \le i \le j$.

  This checks the even/odd steps of an even length computation.

  But, L2 = L(G2) = $\{X_0 \$ \#X_0 \# X_1 \# X_2 \# X_3 \# X_4 \# \ldots \# X_{2k-1} \# X_{2k}\# Z_0 \# \}$

  where $X_{2i-1} \Rightarrow X_{2i}$, $1 \le i \le k$ and $X = X_0$

  This checks the odd/steps of an even length computation, and includes an extra copy of the starting number prior to its $.

Summarizing Quotient

Now, consider the quotient **L2 / L1** where **L1** and **L2** are the CFLs on prior slide. The only way a member of **L1** can match a final substring in **L2** is to line up the **$** signs. But then they serve to check out the validity and termination of the computation. Moreover, the quotient leaves only the starting number (the one on which the machine halts.) Thus,

L2 / L1 = { X | the system F halts on zero }.

Since deciding the members of an re set is in general undecidable, we have shown that membership in the quotient of two CFLs is also undecidable.