Complexity Theory
More Computability

Charles E. Hughes

COT6410 – Spring 2023 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** $\forall C_{|C| \le K}$ **[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

Uniform Halting

- A TM, ***M***, **uniformly halts** if it halts when started in any configuration, ***C***.
- Unlike ***HALT***, this does not limit us to initial configurations, e.g., ones that start in the initial state with the arguments to the left.
- Note that this concept is restricted to normal TMs that start with a finitely marked tape.

CTime Again

- A TM, ***M***, **Uniformly Halts in Constant Time** 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 concept is restricted to normal TMs that start with a finitely marked tape but is not limited to those that start in some initial configuration, e.g., the initial state with the arguments to the left.
- Clearly, this class of machines includes those that start in an initial configuration.

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. Thus, for instance, it would lead to the existence of an input such that a search for a zero (a divider between items on the tape) would take an infinite amount of time.

Complexity of CTime

- Wish to 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*.
- Surprisingly, the **Mortality Problem** is re/non-recursive, even though the **mortality problem** for TMs restricted to **finite** initial tape markings (**TOTAL**) is not even re.
- Note that there is an analogy here because **CTime** seems non-re but we have seen it is re using quantification with a bounded "for all" quantifier.

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

CTime is RE

Theorem 1. The set of Turing machines which uniformly halt in constant time (**CTime**) is recursively enumerable

Proof. Any TM which uniformly halts in at most ***K*** steps cannot scan a square more than ***K*** squares from the initial scanned square. Therefore, there exist only a finite number of ID's to check in deciding if a TM halts in at most ***K*** steps, and the TM must be simulated for each such ID for at most ***K*** steps. We can use a dovetailing procedure which will simulate the enumerable set of TM's to generate the subsets which uniformly halt in ***K*** steps, as ***K*** increases to infinity.

Theorem 2. The set of Immortal TMs is re, non-recursive.

Proof. Shown by Hooper.

$T \in$ CTime $\Rightarrow$ $T \in$ Mortal

Theorem 3. Let ***T*** be a TM in **CTime** then ***T*** is **Mortal**.

Proof. This is obvious as **Mortality** includes TMs with finitely or infinitely marked tapes.

$T \in$ Mortal $\Rightarrow T \in$ CTime #1

Theorem 4. Let ***T*** be a TM in **Mortal** then ***T*** is in **CTime**.

Proof: We approach this by contradiction ($T \notin$ **CTime** $\Rightarrow T \notin$ **Mortal**).

Assume $T \notin$ **CTime** then there is either some finite ID that does not lead to a halt or some finite ID for which there is no a priori upper bound on the number of steps taken before halting. If any finite ID does not lead to a halt, then clearly ***T*** is immortal. We need then consider only infinite IDs and unbounded computations.

Let $\mathcal{J}$ be the set of all ID's such that, for each $I \in \mathcal{J}$, 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 $\{q_1, \ldots, q_m\}$ 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 q_j. The direct descendants of q_j are q_j0 and q_j1, representing the shortest IDs involving q_j.

$T \in$ Mortal $\Rightarrow T \in$ CTime #2

If $I_0, I_1 \in \mathcal{J}$, and $\mathbf{q_j}$ is a symbol of I_0 and I_1, and $I_1 = \sigma I_0$ or $I_1 = I_0 \sigma$, where σ is a tape symbol, then I_0 is a parent of I_1 in the **j**-th tree.

Note that when T starts in I_1, the square containing σ is scanned after every other square of I_1 but before any square not in I_1.

Based on prior considerations, we know that T is immortal and that, for every finite ID, T either halts or runs for an unbounded number of steps and eventually halts. In the latter case T cannot stay within a bounded region of the tape as that would result in a loop. Thus, in both cases (non-halting or no bound), at least one of the trees of the forest must have an unbounded number of nodes.

$T \in$ Mortal $\Rightarrow$ $T \in$ CTime #3

Continuing:

As 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 that causes ***T*** to travel an infinite distance on the tape. It follows that ***T*** is immortal.

This shows if ***T*** is not in **CTime** then ***T*** is also not in **Mortal** and hence **Mortal** ⇒ **CTime**.

Complexity of CTime (finally)

Theorem 5. ***T*** **∈ Mortal ⇔** ***T*** **∈ CTime**.

Proof. Follows from Theorems 3 and 4.

Theorem 6. **CTime** is re, non-recursive.

Proof. Follows from Theorems 2 and 5.

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 $\mathbf{L(G)^n}$

Question1: Is $\mathbf{L(G) = L(G)^2}$?

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

These questions are both undecidable.

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

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

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

Finite Power Problem

- The problem to determine, for an arbitrary context free language **L**, if there exist a finite **n** such that $\mathbf{L^n = L^{n+1}}$ 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 $\mathbf{L_1}$ and $\mathbf{L_2}$, which contains $\mathbf{L_2}$ at least once, is $\mathbf{L_2}$. For instance, $\mathbf{L_1 \bullet L_2 = L_2 \bullet L_1 = L_2 \bullet L_2 = L_2}$.
- This shows that $\mathbf{(L_1 \cup L_2)^n = L_1^n \cup L_2}$.
- Thus, $\mathbf{L^n = \{\lambda\} \cup L_1 \cup L_1^2 \ldots \cup L_1^n \cup L_2}$.
- Analyzing $\mathbf{L_1}$ and $\mathbf{L_2}$ we see that $\mathbf{L_1^n \cup L_2 \neq L_2}$ just in case there is a word $\mathbf{C_1 \# C_2^R \$ C_3 \# C_4^R \$ \ldots C_{2n-1} \# C_{2n}^R \$}$ in $\mathbf{L_1^n}$ that is not also in $\mathbf{L_2}$.
- But then there is some valid trace of length **2n**.
- **L** has the finite power property iff ***M*** executes in constant time independent of its starting configuration (is in **CTime**).

Undecidability of Finite Convergence for Operators on Formal Languages

Relation to Real-Time
(Constant Time) Execution

Simple Operators

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

Subsuming $\bullet$

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

$L = L \oplus L$?

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

Proof #2

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

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

  (2) $\mathbf{L} \oplus \mathbf{L} = \mathbf{L}$ (use notation $\mathbf{L}_\oplus^{\mathbf{n}} = \mathbf{L}_\oplus^{\mathbf{n-1}} \oplus \mathbf{L}$, $\mathbf{n>0}$, $\mathbf{L}_\oplus^{\mathbf{0}} = \{\lambda\}$
- Clearly, if $\mathbf{L} = \Sigma^*$ then (1) and (2) trivially hold.
- Conversely, we have $\Sigma^* \subseteq \mathbf{L}^* = \cup_{\mathbf{n \geq 0}} \mathbf{L}_\oplus^{\mathbf{n}} \subseteq \mathbf{L}$
  - first inclusion follows from (1); second from (1), (2) and the fact that $\mathbf{L} \bullet \mathbf{L} \subseteq \mathbf{L} \oplus \mathbf{L}$
  - equality is by definition of the * operator

Who Cares?

- People with no real life (me?)
- Insertion and a related deletion operation are used in biomolecular computing and dynamical systems
- Shuffle (shown at end of these notes but not discussed in class) is used in analyzing concurrency as the arbitrary interleaving of parallel events
- Crossover is used in genetic algorithms

Quotients of CFLs

Quotients of CFLs (Trace-Like Sequences)

Let **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.

Now, let **L2 = L(G2) =**

$$\{X_0 \ \$ \# X_0 \# X_1 \# X_2 \# X_3 \# \ldots \# X_{2k} \# \ Z_0 \ \#\}$$

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 terminating computation and includes an extra copy of the starting number prior to its **$**.

If a Turing Machine Trace

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

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

This checks the odd/steps of an even length halting 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 # … #Y_{2k-1} # Y_{2j} # Y_{2j+1} # }
L2 = {X0 \$ # X_0 # X_1 # X_2 # X_3 # X_4 # … # 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 | 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: The Intersection of two CFLs is a CSL but the quotient of two CFLs is an re set and, in fact, all re sets can be specified by such quotients.

Details of Traces as CSL

- Easiest starting point is not Turing Machines but rather FRS's with Residue
- Rules are of form
  $ax + b \rightarrow cx + d$
  a, b, c, d are natural numbers,
  $1 \le b < a$; $1 \le d < c$
- Can show that these systems do not require order as do FRS's
- Residues can check for non-divisibility

Traces of FRS with Residues

- I have chosen, once again to use the Factor Replacement Systems, but this time, Factor Systems with Residues.
  The rules are unordered and each is of the form
  $a\,x + b \rightarrow c\,x + d$
- These systems need to overcome the lack of ordering when simulating Register Machines. This is done by
  j. $INC_r[i]$ $\quad p_{n+j}\ x \rightarrow p_{n+i}\ p_r\ x$
  j. $DEC_r[s, f]$ $\quad p_{n+j}\ p_r\ x \rightarrow p_{n+s}\ x$
  $\quad p_{n+j}\ p_r\ x + k\ p_{n+j} \rightarrow p_{n+f}\ p_r\ x + k\ p_{n+f}\ ,\ 1 \le k < p_r$
  We also add the halting rule associated with m+1 of
  $$p_{n+m+1}\ x \rightarrow 0$$
- Thus, halting is equivalent to producing 0. We can also add one more rule that guarantees we can reach 0 on both odd and even numbers of moves
  $$0 \rightarrow 0$$

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	**→**	**$1^{ai}F_i1^{ci}$ | $1^{ai+bi}\#1^{ci+di}$**	**$1 \le i \le k$**
T_1	**→**	**# F_i T_1 | # F_i #**	**$1 \le i \le k$**
A	**→**	**1 A 1 | \$ #**	
S_1	**→**	**\$$T_1$**	
S_2	**→**	**A T_1 # 1^{z0} #**	**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 # … # 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 # … # 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.

Undecidability of Finite Convergence for Lots of Other Operators

Relation to Real-Time
(Constant Time) Execution

K-insertion

- $\mathbf{A \triangleright^{[\,k\,]} B = \{ x_1 y_1 x_2 y_2 \ldots x_k y_k x_{k+1} \mid}$
  $\mathbf{y_1 y_2 \ldots y_k \in A,}$
  $\mathbf{x_1 x_2 \ldots x_k x_{k+1} \in B,}$
  $\mathbf{x_i, y_j \in \Sigma^* \}}$

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

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**)
  - $\mathbf{A} \diamond \mathbf{B} = \cup_{j \geq 1} \mathbf{A} \triangleright^{[j]} \mathbf{B}$
  - $\mathbf{A} \diamond^{[k]} \mathbf{B} = \cup_{1 \leq j \leq k} \mathbf{A} \triangleright^{[j]} \mathbf{B} = \mathbf{A} \triangleright^{[k]} \mathbf{B}$

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

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

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** ▷ $^{[k]}$ **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 • B** is a CFL
  - **A** ▷ **B** is a CFL
  - **A** ▷ $^{[k]}$ **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** ▷ $^{[2]}$ **B)** ∩ **a*c*b*d***
  - **A** ◇ **B** is not necessarily a CFL
  - **A** ◇* **B** is not necessarily a CFL
    - Deciding whether or not **A** ◇* **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

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

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