

A Omitted Proofs

Proposition 1. *Unmasked and mixed-masked padded transformers (i.e., uAHATs and mAHATs) can be simulated by causally masked transformers (AHATs). Specifically, for any $d, k \in \mathbb{N}$, the following holds for the corresponding problem classes:*

1. $\text{uAHAT}_k^0 \subseteq \text{mAHAT}_k^0 \subseteq \text{AHAT}_{\max\{k,1\}}^0$
2. $\text{uAHAT}_k^d \subseteq \text{mAHAT}_k^d \subseteq \text{AHAT}_{1+\max\{k,1\}}^d$ for $d \geq 1$
3. $\text{uAHAT}_*^d \subseteq \text{mAHAT}_*^d \subseteq \text{AHAT}_*^d$.

Proof. That unmasked padded transformers can be simulated by mixed-masked transformers of the same depth holds trivially, as the former constitute a special case of the latter. We will next describe how to convert an unmasked padded transformer to a masked one, on a per-head basis. The construction will leave the computation of masked heads unchanged, making the approach suitable for converting both unmasked and mixed-masked padded transformers to masked ones.

Let E be an unmasked encoder in uAHAT_k^d . Then E has depth $\ell = O(\log^d n)$ and operates over $n + O(n^k)$ tokens (including original input and padding tokens). By Lemma 1, E can be simulated by a causally masked decoder D with depth ℓ and with $\ell \cdot (n + O(n^k))$ new padding tokens (after the original $O(n^k)$ padding tokens used by E). Thus, the total number of padding tokens D uses is $p = O(n^k) + \ell \cdot (n + O(n^k))$. Since $\ell = O(\log^d n)$, this simplifies to $p = O(\log^d(n) \cdot n^{k'})$ where $k' = \max\{k, 1\}$. This is $O(n^{k'})$ when $d = 0$, and $O(n^{k'+1})$ when $d \geq 1$. This finishes the proof of the first two parts. For the third part, observe that by definition, $\text{uAHAT}_*^d = \bigcup_{k=0}^{\infty} \text{uAHAT}_k^d$. From the above, this in turn is contained in $\bigcup_{k=0}^{\infty} \text{AHAT}_{k'+1}^d \subseteq \text{AHAT}_*^d$, as desired.

For the mixed-masking case, the construction in Lemma 1 will preserve the original computation of causally masked heads if we add an additional term with large negative weight C (fixed w.r.t. n) that is activated if the index within the block is greater than b_i . We set C large enough to dominate all other terms in the inner product computation at each token. As a result, the head is constrained to only attend to previous tokens within the block. Applying this modified construction on a case-by-case basis per head, we can take Proposition 1 to apply equally to transformers with mixed-masking. \square

Lemma 4. *For any $d \geq 1$, $\text{AHAT}_*^d \subseteq \text{L-uniform TC}^d$.*

Proof. As noted earlier, it is known that $\text{AHAT}^0 \subseteq \text{FO-uniform TC}^0$ (Merrill and Sabharwal, 2023a,c; Chiang, 2025), and this generalizes to transformers with padding tokens (Pfau et al., 2024). We will extend this construction to the case of AHAT_*^d . The main observation is that even though the depth of a $O(\log^d n)$ -depth padded AHAT transformer is input-dependent, the transformer has a looped architecture and thus a fixed (independent of input size) set of parameters. This makes it possible for a Turing machine (which can only store a fixed number of parameters) to construct an equivalent TC^d circuit family.³

Let L be a language in AHAT_*^0 and let T be a looped, padded AHAT transformer that, when unrolled $c \log^d n$ times for large enough n ,⁴ recognizes whether $x \in L$ for any input x with $|x| = n$. Let $\langle A, B, C \rangle$ be the partition of layers of T where A is the set of initial layers, B is the block that's repeated $c \log^d n$ times on inputs of length n , and C is the set of final layers. Each of these itself is a fixed-depth padded transformer; let's call these T_A, T_B , and T_C . By prior results, there are L-uniform TC^0 circuit families $\{C_n^A\}_{n=0}^{\infty}$, $\{C_n^B\}_{n=0}^{\infty}$, and $\{C_n^C\}_{n=0}^{\infty}$ that simulate transformers T_A, T_B , and T_C , respectively. Thus, there is a (single) log-space Turing machine M that, on input $|x|$, can construct each of these three circuit families. By looping the T_B block $c \log^d n$ times, it can therefore also construct a circuit family $\{C_n\}_{n=0}^{\infty}$ that recognizes L . It follows that L is in L-uniform TC^d . \square

³The proof naturally generalizes to the case of \log^d -depth transformers that don't necessarily have identical parameters across blocks of layers but whose parameters in each layer can be computed in an L-uniform manner.

⁴As before, for small n , we assume T uses a lookup table.

835 This argument does not work for FO-uniform TC^d since it is not clearly that case that the circuit that
 836 *repeats* a layer $\log n$ times can be defined in FO just because the one that simulates one layer can
 837 (Merrill and Sabharwal, 2023c).

838 B TC^d Circuit Evaluation Problem

839 To formalize the circuit evaluation problem, we will use the following serialized format for repre-
 840 senting a circuit. This format is a simplification of the one used by Merrill and Sabharwal (2023a),
 841 with two main differences: (a) instead of a single threshold gate, we use AND, OR, NOT, and MAJ (the
 842 majority gate), which will simplify the description of the construction; and (b) we do not require the
 843 gates in the serialization to be sorted in any particular order, which makes it easier (and perhaps even
 844 possible) to have a log-space reduction from any L-uniform TC^d language to the circuit evaluation
 845 problem in this specific serialization format. The syntax of this circuit format is governed by the
 846 following grammar:

$$\begin{aligned}\text{Circuit} &\rightarrow \text{Gate}^* \\ \text{Gate} &\rightarrow \text{X} \mid \text{Op Arg}^* \\ \text{Op} &\rightarrow \text{AND} \mid \text{OR} \mid \text{NOT} \mid \text{MAJ} \\ \text{Arg} &\rightarrow \&1^*\end{aligned}$$

847 Semantically, we take the k -th X gate to return the k -th input bit. Other gates retrieve the values of
 848 the gates referred to by their argument pointers and apply the associated logical function. We take
 849 the final gate in the serialization to be the output gate of the circuit. Note that not all strings in this
 850 grammar represent a well-formed circuit, but any valid circuit can be serialized in this format.

As an example, the threshold circuit $\text{Majority}(x_1, x_2 \vee x_3, \neg x_3)$ stating that at least two of $x_1, x_2 \vee x_3$,
 and $\neg x_3$ should be true, would be represented as follows:

$$\underbrace{\text{X X X}}_{\text{input}} \underbrace{\text{MAJ } \&1 \&11111 \&111111}_{\text{Majority gate}} \underbrace{\text{OR } \&11 \&111}_{\text{Or gate}} \underbrace{\text{NOT } \&111}_{\text{Not gate}}$$

Note that the non-input gates need not be serialized in this particular order. The following is also an
 equally valid serialization of the same formula:

$$\underbrace{\text{X X X}}_{\text{input}} \underbrace{\text{OR } \&11 \&111}_{\text{Or gate}} \underbrace{\text{NOT } \&111}_{\text{Not gate}} \underbrace{\text{MAJ } \&1 \&1111 \&11111}_{\text{Majority gate}}$$

851 We now formalize the circuit evaluation problem, for any class of circuit families, such as the class
 852 TC^d of \log^d -depth circuit families:

853 **Definition 11** (C circuit evaluation). Let C be a class of (potentially non-uniform) circuit families.
 854 The C circuit evaluation problem is defined as follows:

- 855 • Input: $(x, \langle C \rangle)$ where $x \in \{0, 1\}^*$ is a string and $\langle C \rangle$ is the serialization of a circuit C such
 856 that $\langle C \rangle = C_{|x|}$ for some circuit family $\{C_n\}_{n=0}^\infty \in C$.
- 857 • Output: The value $C(x)$.

858 We refer to the special case where $C = \text{TC}^d$ as the TC^d *circuit evaluation problem*. This corresponds
 859 to threshold circuit evaluation where the depth of the circuit is constrained to be $O(\log^d |x|)$.

860 We further define **wide-TC** $^d \subseteq \text{TC}^d$ as the class of circuit families $\{C_n\}_{n=0}^\infty$ such that there exists
 861 some c such that, for large n , the depth of C_n is at most $c \log^d n$ and, crucially, the size is *at least* n^c .
 862 That is, the size (and hence the width) of the circuit is large relative to its depth.

863 We define the corresponding **wide-TC** d **circuit evaluation problem** as a variant of circuit evaluation
 864 problem with $C = \text{wide-TC}^d$.

865 B.1 Hardness of TC^d Circuit Evaluation

866 **Lemma 5.** For $d \geq 1$, TC^d circuit evaluation is hard for L-uniform TC^d under L-reductions.

867 *Proof.* Given any $L \in \text{L-uniform TC}^d$, there exists a log-space Turing machine T_L that constructs
868 a circuit family $\{C_n\}_{n=0}^\infty$ that recognizes L . We can construct an L-reduction from L to the TC^d
869 circuit evaluation problem as follows. Given an input x whose membership in L we would like to
870 check, the reduction first copies x to the output. It then uses the log-space Turing machine T_L to
871 build the circuit $C_{|x|}$ and output it in the above serialized format. We thus have a log-space reduction
872 from x to $(x, \langle C_{|x|} \rangle)$. We conclude that TC^d circuit evaluation is hard for L-uniform TC^d under
873 L-reductions. \square

874 **Lemma 6.** For $d \geq 1$, wide- TC^d circuit evaluation is hard for L-uniform TC^d under L-reductions.

875 *Proof.* As in the proof of Lemma 5, we are given L such that there exists a log-space Turing machine
876 T_L that constructs $\{C_n\}_{n=0}^\infty$ recognizing L , where the depth of each C_n is at most $c \log^d n$ for some c
877 and large enough n . We can create a modified log-space Turing machine T'_L that builds $\{C'_n\}_{n=0}^\infty$ that
878 still recognizes L in depth $c \log^d n$ but is padded with dummy gates so that it has size at least n^c : we
879 do this by keeping a counter for the number of gates and wires produced and outputting dummy ones
880 until n^c is exceeded. We then follow the rest of the proof of Lemma 5 with T'_L instead of T_L . \square

881 B.2 Solving TC^d Circuit Evaluation with Transformers

882 **Lemma 7.** There is a mixed-masked looped transformer T that, on input $(x, \langle C \rangle)$ where $x \in \{0, 1\}^*$
883 and $\langle C \rangle$ is the serialization of a depth ℓ circuit with $|x|$ inputs, computes $C(x)$ when unrolled ℓ times.

884 *Proof.* We adapt the proof of Merrill and Sabharwal (2023a, Theorem 3), which sketches how
885 log-depth (unlooped) transformers can implement the TC^0 circuit evaluation problem. We construct
886 a looped transformer that will “attempt” to evaluate every gate: if its arguments have already been
887 computed, the gate will return $\{0, 1\}$, and, if not, it will return undefined (\perp).

888 Let i be a token index. We say the token w_i is a gate token if it is X, AND, OR, NOT, MAJ. We will use
889 gate token i to store the value $v_i \in \{0, 1, \perp\}$ for the gate it represents as a one-hot encoded vector. In
890 the base case (embedding layer), we initialize $v_i = \perp$ for every gate token. Using a looped block of
891 2 layers, we will proceed in a way that propagates the computation of v_i at later layers in terms of
892 previously computed values.

893 X Gates. In the setup layers at an X token i , we use a causally masked uniform attention head with
894 value $\mathbb{1}[w_j = \text{X}]$. Thus, this head computes r_i/i , where r_i number of X gates before and including i .
895 We compute $\phi(r_i/i, 1/i) = \phi(r_i)$ and store it in the residual stream.

896 In the looped layers, we define an attention head with query $\phi(r_i)$, key $\phi(j)$, and value w_j . This head
897 thus retrieves input token w_{r_j} . We update the gate value to $v_i \leftarrow w_{r_j}$ (viewing both as vectors in the
898 same space).

899 Other Gates. In the setup layers, each argument token i attends with causally masked uniform
900 attention with value $\mathbb{1}[w_j = \&]$ to compute a_i , the number of arguments to its left (including it).
901 Each $\&$ token attends with query $\phi(a_i)$, key $\phi(a_j)$, and value $\mathbb{1}[w_j = \&]$, which returns $1/(1 + z_i)$,
902 where z_i is the number of 1’s following $\&$ token i . We compute and store $\phi(z_i + 1)$ in the residual
903 stream. We compute g_i similarly to a_i , the number of gate tokens to left of token i (also inclusive).

904 In the first looped layer, each $\&$ token i attends with query $\phi(z_i + 1)$, key $\phi(g_j)$, and value v_j . Thus,
905 the argument token i retrieves v_{z_i+1} , the value at gate $z_i + 1$. In the second looped layer, gate token
906 i attends with query $\phi(g_i)$, key $\phi(g_j)$, and value v_{z_j+1} . This, it returns the vector $\frac{1}{|A_i|} \sum_{j \in A_i} v_j$,
907 where A_i is the set of gates that are arguments of gate i . From this vector, we can apply projections
908 to recover T , the fraction of $j \in A_j$ with $v_j = 1$, as well as U , the fraction of $j \in A_i$ with $v_j = \perp$.
909 If $U > 0$, we set $v_i \leftarrow \perp$. Otherwise, we set v_i by thresholding T against some threshold k
910 based on the gate type ($T \geq 1$ for AND, $T > 0$ for OR, and $T \geq 1/2$ for MAJ). In effect, this sets
911 $v_i \leftarrow G(\{v_j\}_{j \in A_j})$, where G is the gate type.

912 Thus, the looped layers either keep v_i as \perp or correctly update it to its true value. Furthermore, the
913 number of looping steps until v_i is updated is exactly the depth of node i . Thus, a circuit C of depth ℓ
914 can be fully evaluated by looping the depth-2 block ℓ times. \square

915 **Corollary 7.1.** *For any $d \geq 0$, the wide-TC^d circuit evaluation problem is in mAHAT₀^d, and hence*
 916 *in AHAT₁^d.*

917 *Proof.* We are given input $(x, \langle C_n \rangle)$, where C_n comes from some TC^d circuit family $\{C_n\}_{n=0}^\infty$ with
 918 depth at most $c \log^d n$ and size at least n^c , where c is a constant specific to the circuit family. If we
 919 unroll the transformer from Lemma 7 to depth $c \log^d n$, we can solve wide-TC^d circuit evaluation
 920 problem for large enough n (w.l.o.g. we can solve small- n examples via table lookup).

921 We next justify that a mixed-masked mAHAT^d transformer will unroll at least $c \log^d n$ times for large
 922 enough n . This transformer will unroll exactly $\log^d N$ times, where $N = n + |\langle C_n \rangle|$ is the total
 923 input length for a circuit evaluation instance. Since the size of C_n is at least n^c , we have that $N \geq n^c$.
 924 Thus, our mAHAT₀^d transformer unrolls the following number of times:

$$\log^d N \geq \log^d n^c = c^d \log^d n \geq c \log^d n.$$

925 Thus, our transformer will unroll a sufficient number of times to solve the wide-TC^d circuit evaluation
 926 problem. It follows that this problem is in mAHAT₀^d.

927 Finally, we conclude via Proposition 1 that this mixed-masked transformer can be converted to a
 928 corresponding causally masked padded transformer, placing the problem also in AHAT₁^d. \square

929 C Uniformity Ceiling: Sanity Checking Our Results

930 We recount some simple but counterintuitive facts obtained while verifying the consistency of our
 931 results. In general, for classes A, B such that $A \subseteq B$, B-uniformity often leads to larger classes of
 932 languages than A-uniformity, as we have more resources to construct a circuit. However, as the
 933 following proposition shows, this is not always the case.

934 For the purposes of this result, we assume that A-uniform XC means there exists $f \in A$ such that f
 935 maps 1^n to an XC circuit that operates over inputs of size n . This is not satisfied by some stronger
 936 notions of uniformity like DLOGTIME uniformity, but it is satisfied for L and NL uniformity.

937 **Proposition 2** (Uniformity ceiling). *Consider classes A, B and a circuit class XC such that:*

- 938 1. $A \subseteq B \subseteq \text{A-uniform XC}$;
- 939 2. XC circuit evaluation is in A-uniform XC.

940 *Then B-uniformity does not strengthen A-uniformity, i.e., A-uniform XC = B-uniform XC.*

941 *Proof.* We will show that any language $L \in \text{B-uniform XC}$ is also in A-uniform XC. By definition
 942 of B-uniform XC, there exists a procedure $f \in B$ that maps any input 1^n to an XC circuit C_n^f that
 943 checks membership in L over all strings w of size n , i.e., $f(1^n) = C_n^f$. By the first precondition of
 944 the proposition, f is also in A-uniform XC. This means there exists a procedure $g \in A$ that maps
 945 any input 1^n to an XC circuit C_n^g that does what f does, i.e., on input 1^n constructs the XC circuit
 946 C_n^f . Thus, we have $g(1^n) = C_n^g$ and $C_n^g(1^n) = C_n^f$. Furthermore, by the second precondition of
 947 the proposition, there exists a procedure $h \in A$ that maps any input $(1^n, 1^m)$ to an XC circuit $C_{n,m}^h$
 948 that evaluates any XC circuit C_n of size m over any input w of size n , i.e., $h(1^n, 1^m) = C_{n,m}^h$ and
 949 $C_{n,m}^h(C_n, w) = C_n(w)$.

950 Now we are ready to construct an A-uniform XC circuit for recognizing L . Using currying notation
 951 from functional programming,⁵ this circuit is $C_L = h(1^n, 1^m)(g(1^n)(1^n))$ where $n = |w|$ and
 952 $m = |g(1^n)|$. Here we use currying notation to specify that the first input of the two-input circuit $C_{n,m}^h$
 953 generated by h is fixed as $g(1^n)(1^n)$, turning it into a single-input function. Similarly, $g(1^n)(1^n)$
 954 denotes first applying g to 1^n in order to construct a circuit $g(1^n) = C_n^g$, and then passing 1^n as input
 955 to this constructed circuit C_n^g .

⁵Currying is the technique of translating a function that takes multiple arguments into a sequence of families of functions, each taking a single argument, cf. <https://en.wikipedia.org/wiki/Currying>. As an example, currying turns a two-argument function $f : (x, y) \mapsto z$ into a family of single-argument functions $g : x \mapsto (y \mapsto z)$, i.e., $g(x)$ is a single-argument function that maps y to z (in a manner that depends on x).

956 By design, C_L takes as input a string w and computes $C_L(w) = h(1^n, 1^m)(g(1^n)(1^n), w)$.

957 It is easy to see that C_L can be constructed by a procedure in A —after all, g and h are in A , 1^n can
 958 be easily computed from w by converting all 0s to 1s, and similarly 1^m can be computed from the
 959 output $g(1^n)$ of applying procedure g (which is in A) on 1^n . Further, C_L is a circuit in XC as it is a
 960 fixed composition of the XC circuits that functions h and g output. All that remains to be shown is
 961 that C_L recognizes L . To this end, we note:

$$\begin{aligned} C_L(w) &= h(1^n, 1^m)(g(1^n)(1^n), w) \\ &= C_{n,m}^h(g(1^n)(1^n), w) \\ &= C_{n,m}^h(C_n^g(1^n), w) \\ &= C_{n,m}^h(C_n^f, w) \\ &= C_n^f(w). \end{aligned}$$

962 By design, $C_f(w) = 1$ if and only if $w \in L$. Thus, C_L recognizes L . □

963 Applying this result in the case of L and NL as classes A and B , respectively, we obtain:

964 **Corollary 3.1.** *For any circuit class XC containing NL , $L\text{-uniform } XC = NL\text{-uniform } XC$.*

965 This may appear surprising on the surface level, but it has a quite intuitive interpretation: weaker
 966 uniformity provably cannot increase a circuit class that is sufficiently expressive. For example,
 967 applying Proposition 2 with $A = L$, $B = P$, and $XC = P$ (poly-size circuits), we can derive
 968 the unsurprising result that relaxing the uniformity condition for polynomial-size circuits does not
 969 increase their expressive power beyond P :

970 **Corollary 3.2.** $L\text{-uniform } P = P\text{-uniform } P$.