**ABSTRACT**

Base-Level Activation (Anderson et al. 2004)
- Core to ACT-R declarative module
- Models historical retrieval bias and errors
- Used to model forgetting

How to efficiently remove items from memory while preserving model fidelity?

**EFFICIENT IMPLEMENTATION**

On activation, predict decay ($t_d$)
1. On access, fast via approximation
2. If needed, in future, exact via binary parameter search $\sim O(\log t_d)$

Maintain decay map: $M[t] = \{m_1, m_2, ...\}$
- Key: time step
- Value: set of elements predicted to decay at that time step

Memory events:
- **New**: insert into map at $t_d$
- **Remove**: find in map at $t_d$, remove
- **Activate**: find in map at $t_d$, old, remove, insert into map at $t_d$, new

At time step $t$, check elements at map[$t$]:
- If decayed, remove
- Else, predict via #2

**SYNTHETIC EVALUATION**

50,000 memories, valid @ $t=1000$
- $n \sim U(1, 10)$
- # activations $\sim U(1,10)$
- $t_d \sim U(1,999)$, one @ $t=999$

Experimental conditions: $d = 0.8$, $\theta = -1.6$
- Largest possible $t_d = 3332$

**PROBLEM FORMULATION**

Base-Level Activation

$$B(t, d) = \ln \left( \sum_{i=1}^{n} (t - t_i)^{-d} \right)$$

Activation at time $t$ is the logarithm of an exponential decay over usage history, with respect to rate $d$.

**EFFICIENT AND CORRECT**

Decay Problem

$$B(t_d, d) < \theta$$

When will activation fall below threshold?

**NAÏVE APPROACH**

Compute the activation of each memory at each point in time: $O(\# \text{memories})$

**QUALITY ANALYSIS**

- 60%
- 20%

**COMPLEXITY COMPARISON**

**COMPUTATION COMPARISON**

Maximum decision time = 6 msec.

**WORKING-MEMORY FORGETTING**

- Experimental conditions: $d = 0.8$, $\theta = -1.6$
- Largest possible $t_d = 3332$