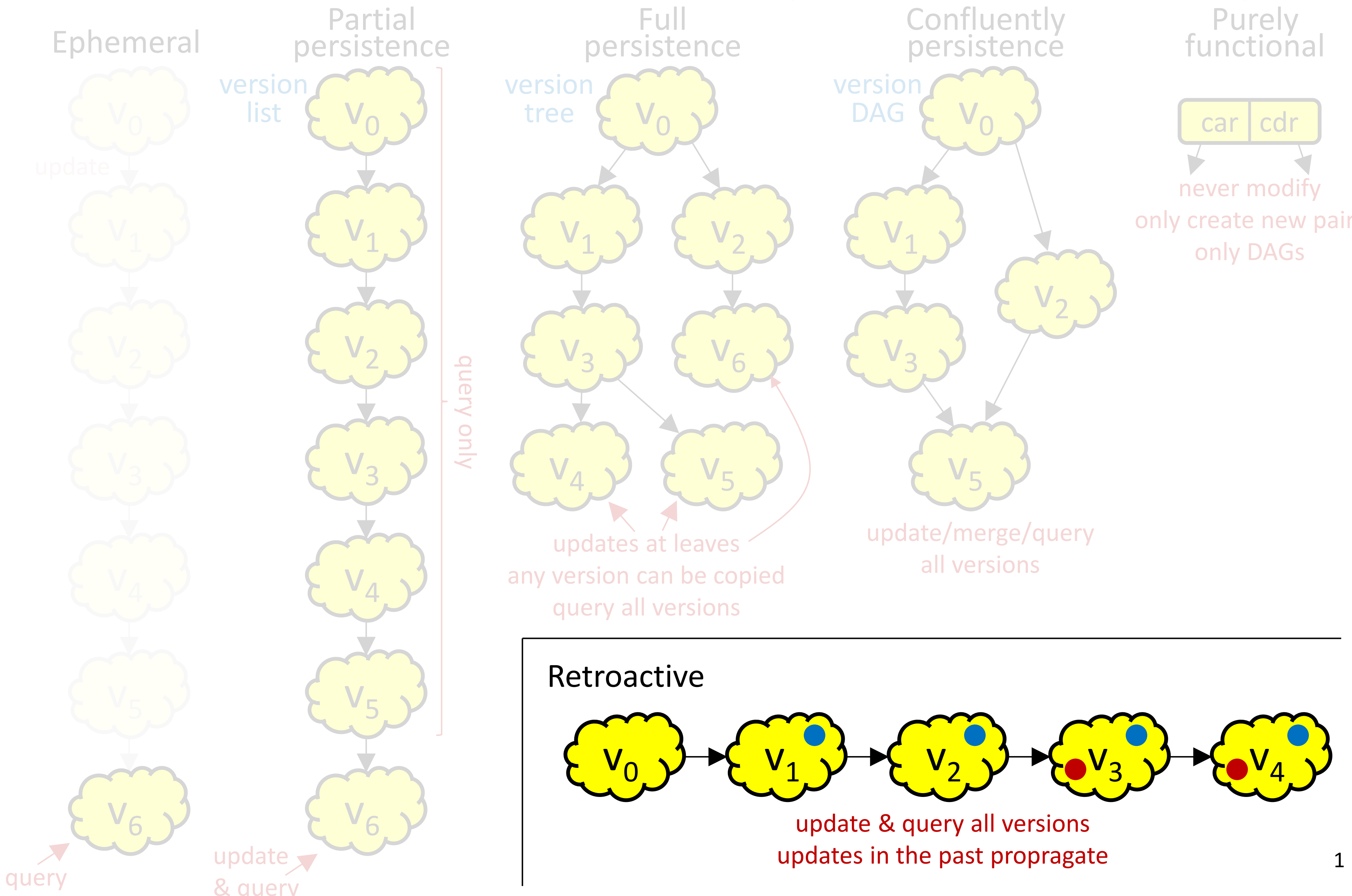
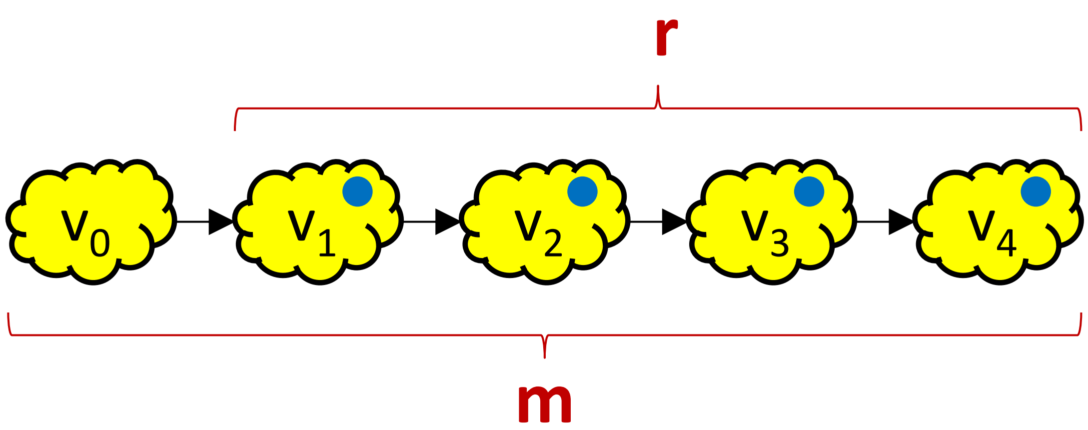
Persistent Data Structures (Version Control)



Retroactive Data Structures

[E.D. Demaine, J. Iacono, S. Langerman, *Retroactive Data Structures*, Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms, 274-283, 2004]

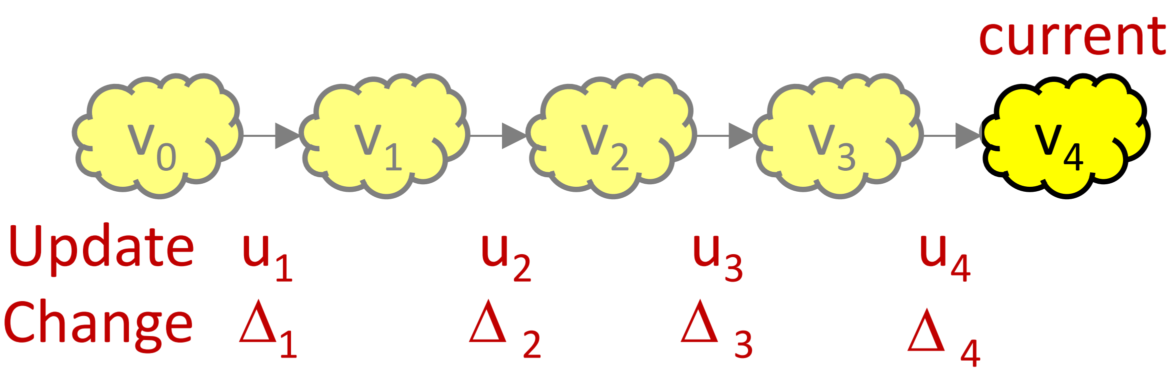


m Total number of updates/versions
r Distance from current time
n Maximal data structure size at any time

Partial retroactive Update all versions & **query current**
Full retroactive Update & query all versions

Rollback → Full Retroactivity

THEOREM 3.1. *Given any data structure that performs a collection of operations each in worst case $T(n)$ time, there is a corresponding retroactive data structure that supports the same operations in $O(T(n))$ time, and supports retroactive versions of those operations in $O(rT(n))$ time.*



+ Generic, can always be applied, space efficient

- Slow retroactive operations

Lower bounds for Retroactivity

THEOREM 3.2. *There exists a data structure in the straight-line-program model, supporting $O(1)$ time update operations, but the (partially) retroactive insertions of those operations require $\Omega(r)$ time, worst case or amortized.*

$$a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

(requires Ω(n) multiplications given *x* by Motzkin's theorem)

THEOREM 3.3. *In the cell-probe model, there exists a data structure supporting partially retroactive updates in $O(1)$ time, but fully retroactive queries of the past require $\Omega(\log n/$ ~~$\log\log n$~~$)$* time.*

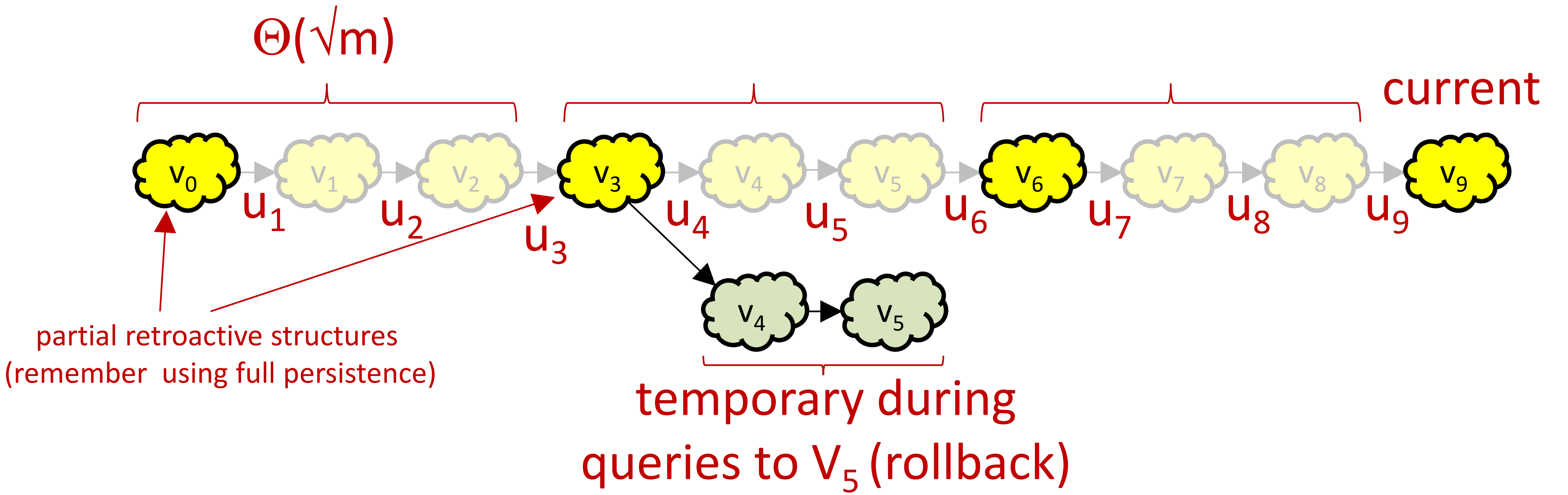
x_0	x_1	x_2	$\cdots$	x_i	$\cdots$	x_n

(prefix sum queries require Ω(log *n*))

* [M. Patrascu, E.D. Demaine, *Logarithmic Lower Bounds in the Cell-Probe Model*, SIAM J. of Computing 35(4): 932-963, 2006]

Partial → Full Retroactivity

THEOREM 3.4. *Any partially retroactive data structure in the pointer-machine model with constant indegree, supporting $T(m)$-time retroactive updates and $Q(m)$-time queries about the present can be transformed into a fully retroactive data structure with amortized $O(\sqrt{m}\,T(m))$-time retroactive updates and $O(\sqrt{m}\,T(m) + Q(m))$-time fully retroactive queries using $O(mT(m))$ space.*



Partial Retroactive Commutative Data Structures

LEMMA 4.1. *Any data structure supporting a commutative set of operations allows the retroactive insertion of operations in the past (and queries in the present) at no additional asymptotic cost.*

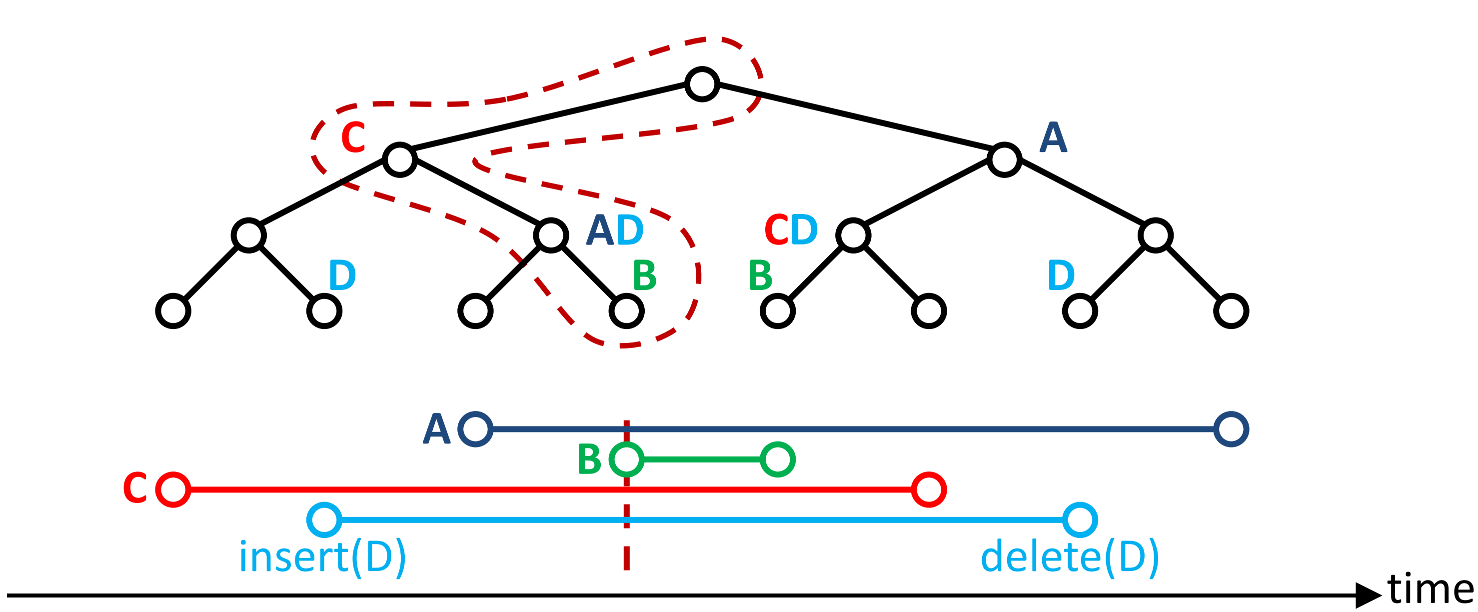
LEMMA 4.2. *Any data structure supporting a commutative and invertible set of operations can be made partially retroactive at no additional asymptotic cost.*

LEMMA 4.3. *Any data structure for a searching problem can be made partially retroactive at no additional asymptotic cost.*

commutative = state independent of order of operations

Decomposable Search Problems

THEOREM 4.1. *Any data structure for a decomposable searching problem supporting insertions, deletions, and queries in time $T(n)$ and space $S(n)$ can be transformed into a fully retroactive data structure with all operations taking time* ~~$O(T(m))$ *if* $T(m) = \Omega(n^\epsilon)$ *for some* $\epsilon > 0$, *or*~~ $O(T(\boldsymbol{n}) \log m)$ ~~*otherwise.*~~ *The space used is* $O(S(m) \log m)$.



Specific Retroactive Data Structures

Data Structure	Partially Retroactive	Fully Retroactive
Dictionary (exact)	$O(\log m)$	$O(\log m)$
Dictionary (successor)	$O(\log m)$	$O(\log^2 m)$
Queue	$O(1)$ **?**	$O(\log m)$
Stack	$O(\log m)$	$O(\log m)$
DEQUE	$O(\log m)$	$O(\log m)$
Union/Find *	$O(\log m)$	$O(\log m)$
Priority Queue	$O(\log m)$	$O(\sqrt{m} \log m)$

* [D.D. Sleator, R.E. Tarjan, *A Data Structure for Dynamic Trees*, Proc. 13th Annual ACM Symposium on Theory of Computing, 114-122, 1981]