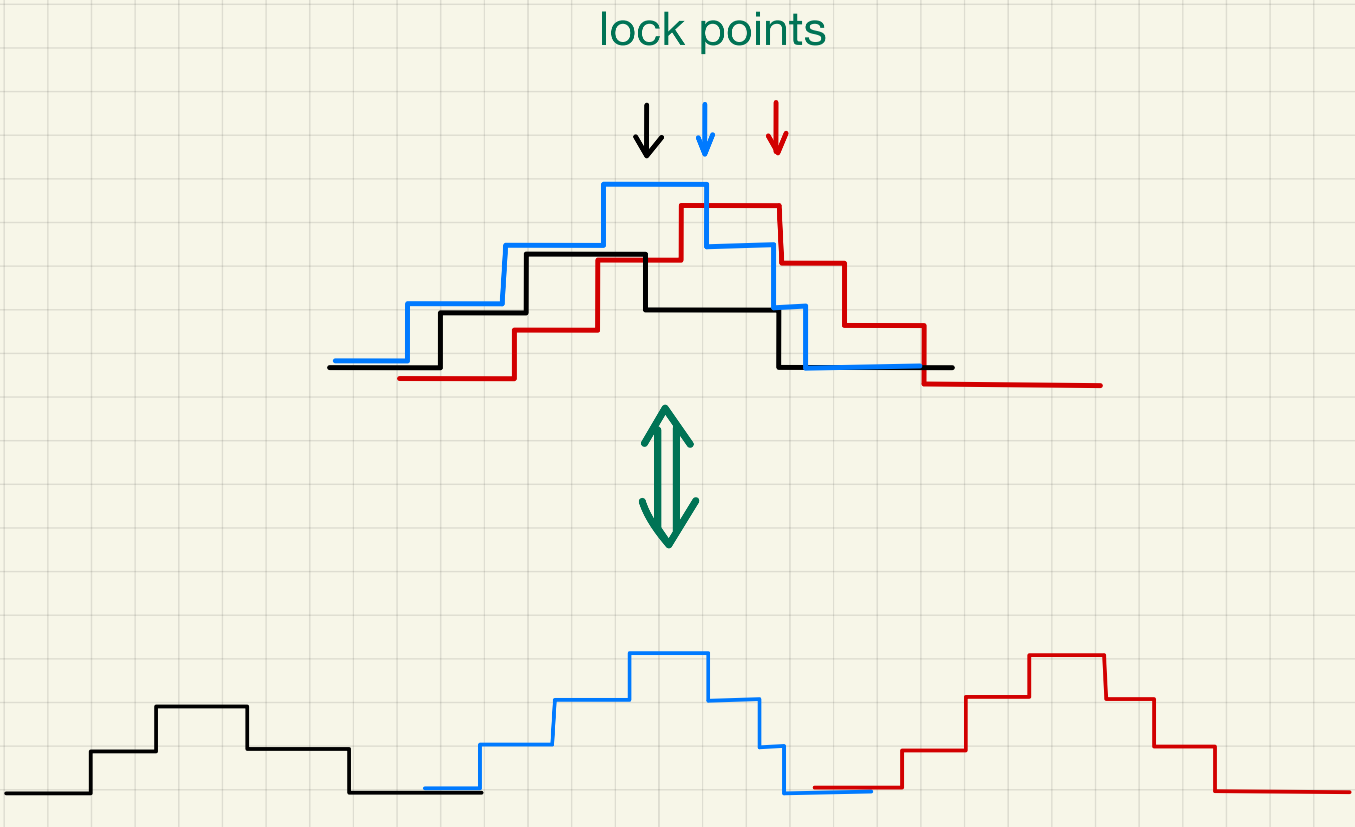
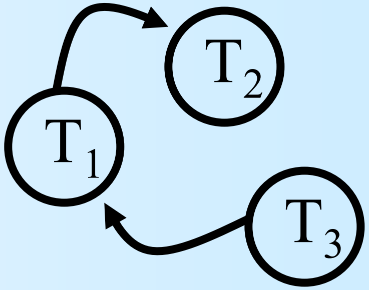
Toute exécution 2PL peut être sérialisée en une succession où les trx apparaissent dans l'ordre de leurs "lock points".



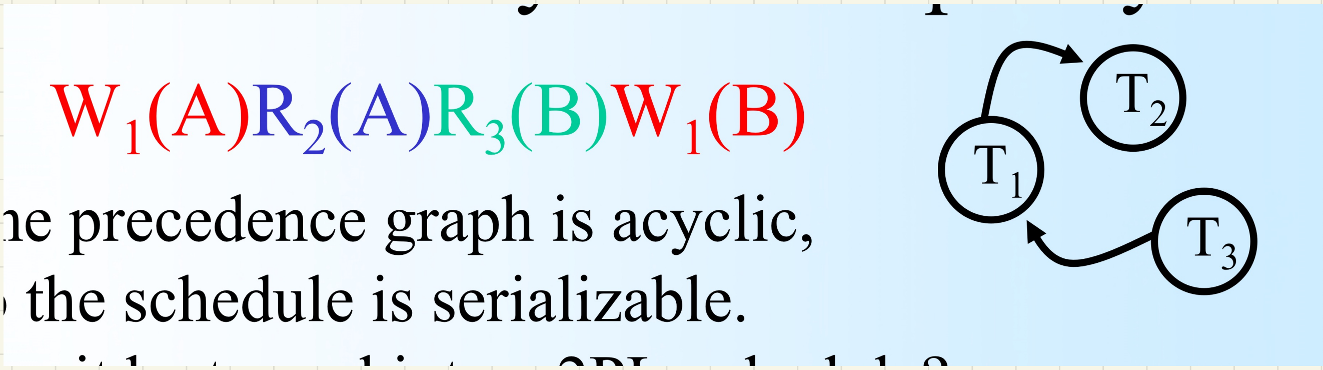
The Price to Pay For 'Simplicity'…

$W_1(A)R_2(A)R_3(B)W_1(B)$



The precedence graph is acyclic,
so the schedule is serializable.
Can it be turned into a 2PL-schedule?

- By rules *L1* and *L3*, T_1 must issue $U_1(A)$ prior to $R_2(A)$.
- Because of $R_3(B)W_1(B)$, the first (and only) unlock $U_3(B)$ of T_3 must precede the first unlock of T_1 (cf. lemma).
- It follows that $U_3(B)$ must precede $R_2(A)$.
- But then $T_{\underset{3}{2}}$ cannot satisfy rules *L1* and *L2*…
- To conclude, in 2PL, the reads and writes cannot occur in exactly the order shown.



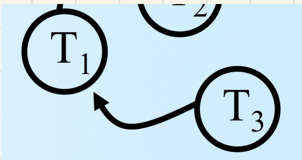
$W_1(A)R_2(A)R_3(B)W_1(B)$

he precedence graph is acyclic,

the schedule is serializable.

A cause de L_1 et L_3 :

$W_1(A) U_1(A) S_2(A) R_2(A) R_3(B) W_1(B)$

A cause de  et Lemma,

$W_1(A) U_1(A) S_2(A) R_2(A) R_3(B) W_1(B)$

lock point de T_1

lock point de T_3

⇓

$U_3(B) \ldots U_1(A) S_2(A) R_2(A) R_3(B) W_1(B)$

MAIS alors, $R_3(B)$ ne peut pas apparaître entre $S_3(B)$ et $U_3(B)$.