Thomas's Write Rule

(Modified Basic TO algorithm but does not guarantee conflict serializability)

Each active database item will have the following two timestamps:

read_TS(X): the sequence number of the youngest transaction that read item X.
write_TS(X): the sequence number of the youngest transaction that wrote item X.

Every transaction is assigned with a unique integer sequence number.
TS(T_6) is the timestamp of T_6 which is the sequence number of the transaction = 6

Consider two transactions T_5 and T_8, where T_8 has started after T_5.
Hence, T_8 is younger to T_5. Also, TS(T_8) > TS(T_5).

Note that this rule is applicable **only** if a younger transaction has performed write operation on the same database item by <u>assigning a constant value</u> to the item.

T_6 requests write_item(X):

```
If (read_TS(X) > TS($T_6$))
{
     $T_6$ will abort; //some younger trans. has read X
}
Else if (write_TS(X) > TS($T_6$))
  {
     $T_6$ will skip the write_item(X) operation since $T_6$ value
      to X would have been overwritten by $T_{write\_TS(X)}$
     $T_6$ will simply continue with its next operation
  }
  Else
  {
     $T_6$ performs write_item(X);
       write_TS(X) = TS($T_6$);
  }
```

T_6 requests read_item(X): (same as in Basic TO algorithm)

```
If (write_TS(X) > TS($T_6$))
{
     $T_6$ will abort; //some younger trans. has written X
}
Else
{
     $T_6$ performs read_item(X);
     read_TS(X) = max(read_TS(X),TS($T_6$));
}
```