

### Exercise 3.43.

Suppose that the balances in three accounts start out as \$10, \$20, and \$30, and that multiple processes run, exchanging the balances in the accounts. Argue that if the processes are run sequentially, after any number of concurrent exchanges, the account balances should be \$10, \$20, and \$30 in some order. Draw a timing diagram like the one in figure 3.29 to show how this condition can be violated if the exchanges are implemented using the first version of the account-exchange program in this section. On the other hand, argue that even with this **exchange** program, the sum of the balances in the accounts will be preserved. Draw a timing diagram to show how even this condition would be violated if we did not serialize the transactions on individual accounts.

### Answer.


If the processes are run sequentially, we can ensure that no interleaving would occur and just before the moment of every exchange, the balances are still what it thought they were. In other words, every time a process swaps the balances in two bank accounts, it always computes the new balances based on the most advanced information. Additionally, given any two accounts  $a_1$  and  $a_2$  with balances  $m$  and  $n$ , the swap operation changes the balances into an order of  $n$  and  $m$ . This indicates that the account balances should still be \$10, \$20, and \$30 in some order after an exchange and, this principle applies to any number of swaps. Hence, the account balances should be \$10, \$20, and \$30 in some order after any number of concurrent exchanges.

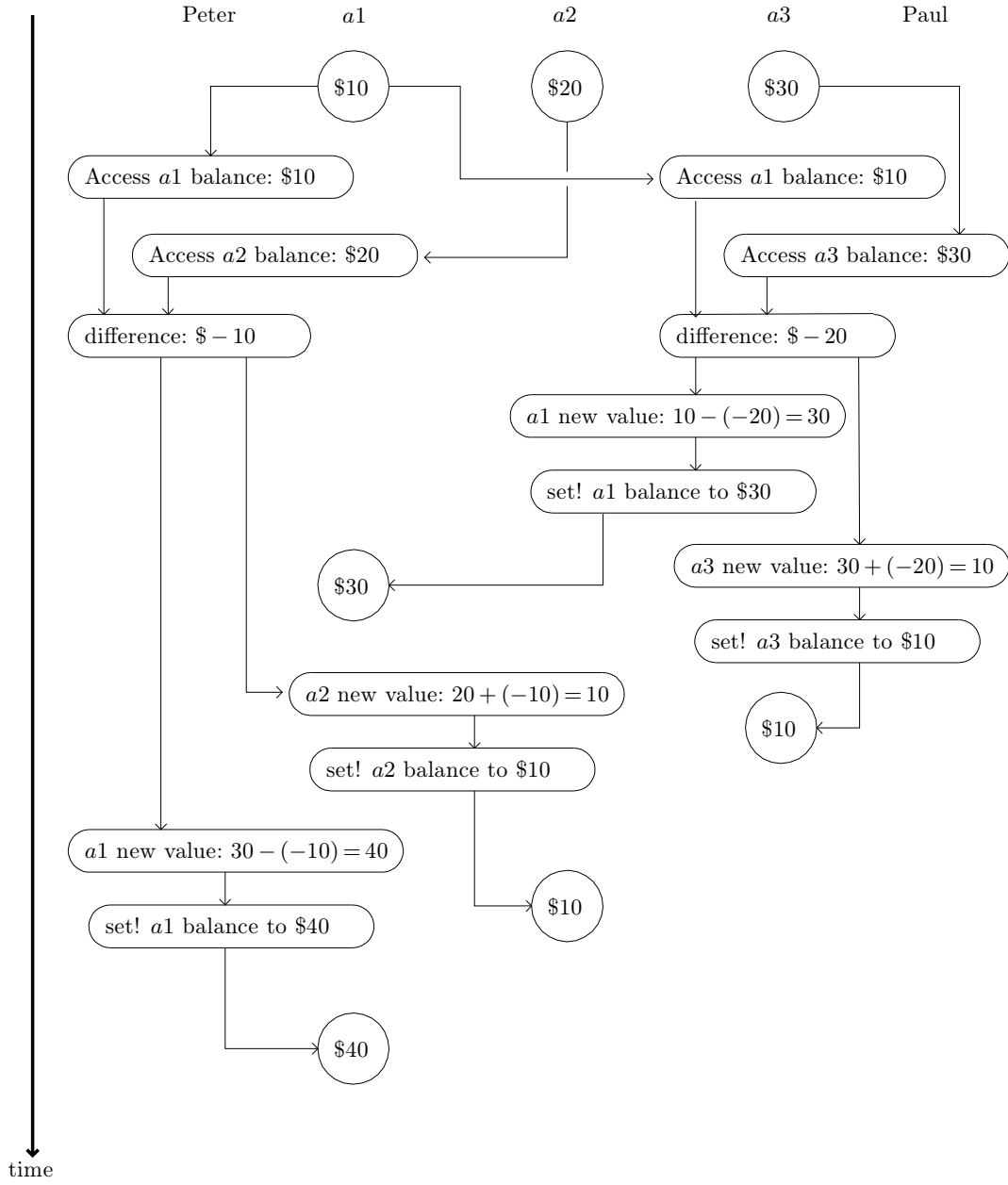
Figure 1 shows how this condition can be violated if the exchanges are implemented using the first version of the account-exchange program in this section. Note that in this version of account-exchange program both **deposit** and **withdraw** are serialized when they are called, this guarantees they change the account balance based on the most advanced account information.

On the other hand, this **exchange** program always takes the difference from one account and add it to the other one. Hence, the sum of the balances in the accounts will be preserved.

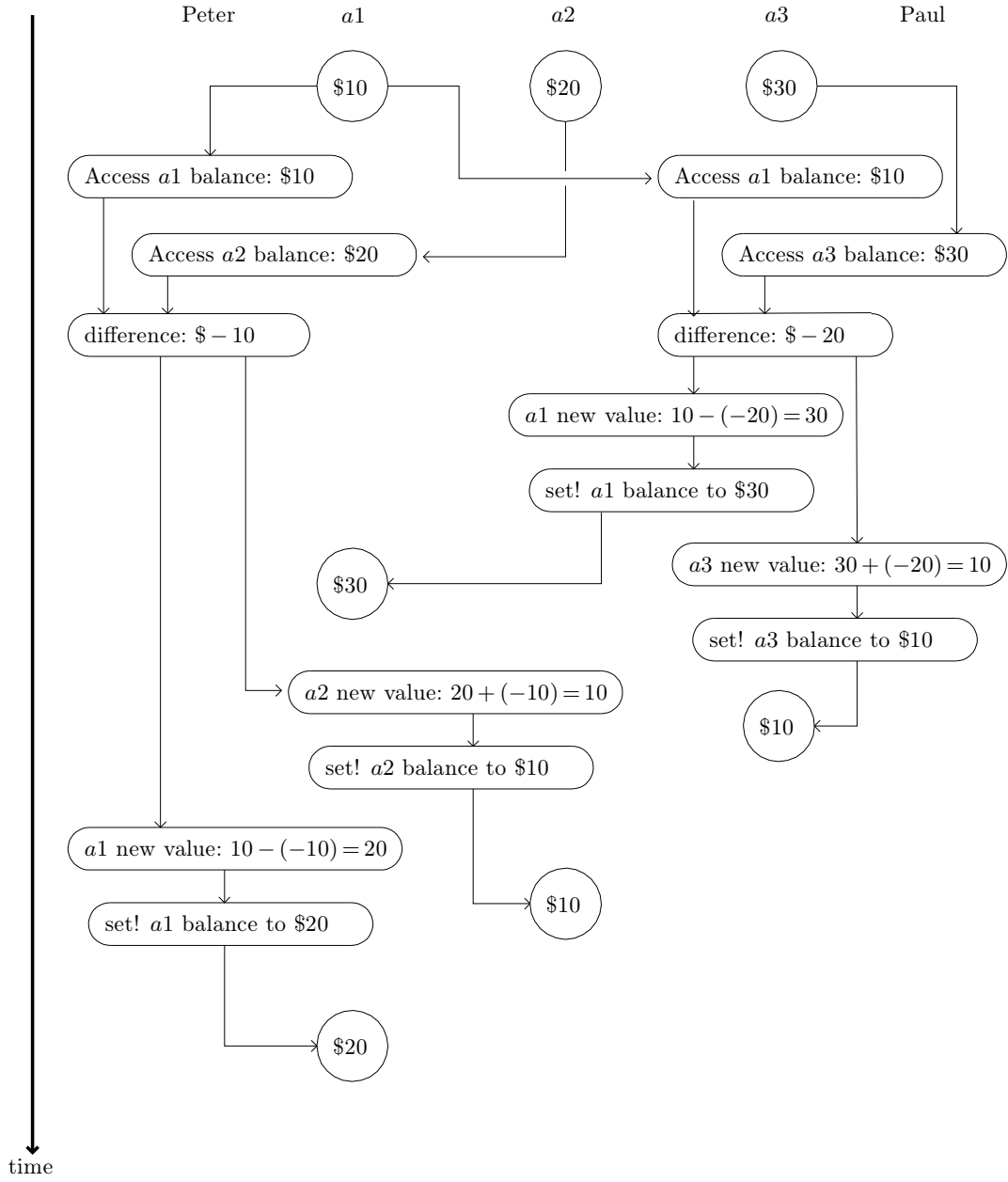
If we did not serialize the transaction on individual accounts, then the two concurrent operations might interleave the order in which they access **balance**. This indicates that they would probably make modification on the account based on the outdated information and thus yield undesired result. Figure 2 presents a particular example illustrating how the transaction of Peter interrupted by that of Paul can lead to the inconsistency of the sum in the accounts after concurrent exchanges.

---

\*. Creative Commons  2013, Lawrence X. Amlord (颜世敏, aka 颜序).



**Figure 1.** Timing daigram showing how the first version of the account-exchange program in this section can violate the condition that account balances would always be \$10, \$20 and \$30 in some order after some number of concurrent exchanges if the processes are run sequentially.



**Figure 2.** Timing daigram showing how the condition of conserved sum on balances would be violated if we did not serialize the transactions on individual accounts.