1. Ein Vertauschen der beiden P-Operationen in der Consumer-Schleife hätte einen möglichen Deadlock zur Folge. Wenn ein Consumer-Prozess den Buffer-Access erhält, beim P(num) aber in der Warteschlange landet, kann kein Product mehr in den Buffer gepusht werden, da ein Producer-Prozess hierfür einen Buffer-Access benötigt, welcher durch den wartenden Consumer aber geblockt wird.

Dies tritt insbesondere dann auf, wenn die ersten n Prozesse in der Warteschlange Consumer sind und n > num gilt. Im Fall von n = num könnte der Deadlock noch vermieden werden, wenn es einen Producer gibt, welcher V(bufferAccess) bereits verarbeitet hat, V(num) aber noch aussteht.

Ein Vertauschen der beiden V-Operationen in der Producer-Schleife dürfte keine Deadlocks hervorrufen. Dies begründet sich vor allem darin, dass die V(num)-Operation keinen Prozess suspendiert und der Buffer-Access somit nicht durch einen wartenden Prozess blockiert werden kann.

Ein Vertauschen würde lediglich den Semaphor ändern, in dessen zugehöriger Warteschleife ein möglicher Consumer-Prozess landen könnte, wenn nur die jeweils erste V-Operation des Producers abgeschlossen wäre:

Angenommen, die beiden V-Operationen wären nicht vertauscht und V(bufferAccess) wäre verarbeitet, V(num) noch nicht. Dann würde ein Consumer-Prozess in der Warteschleife von num landen und auf das Ausführen von V(num) warten.

Unter der Annahme, die beiden V-Operationen wären vertauscht und V(num) wäre verarbeitet, V(bufferAccess) aber noch nicht, würde ein Consumer-Prozess in der Warteschleife von bufferAccess landen und auf das Ausführen von V(bufferAccess) warten.

1. Eine verbesserte (gelockerte) Synchronisation auf den Buffer kann durch die Implementierung von synchronized-Blöcken erreicht werden.

Angenommen, der Semaphor bufferAccess ist nicht vorhanden. Ein synchonized-Block in der Consumer-Schleife um die beiden Codezeilen P(num); und prod = pull(buffer); würde gewährleisten, dass die beiden Operationen immer direkt aufeinanderfolgend ausgeführt werden und die Information über den Zustand von num beim pull(buffer) aktuell ist.

Durch einen weiteren synchronized-Block in der Producer-Schleife um die beiden Zeilen push(newproduct, buffer); und V(num); kann auch auf der Producer-Seite sichergestellt werden, dass diese Operationen direkt aufeinander folgen und der Zustand von num immer der Anzahl von Objekten im Buffer entspricht.

1. Die Verwendung eines beschränkten Puffers kann durch einen weiteren Semaphor zur Verwaltung der verbleibenden Kapazität umgesetzt werden.

Im folgenden Pseudocode-Block sind die nötigen Änderungen beispielhaft implementiert und die neuen / angepassten Zeilen kommentiert:

Buffer buffer **=** **...**

Semaphore num **=** 0**;**

Semaphore bufferAccess **=** 1**;**

int capacity **=** **...;** // Variable zur Festlegung der Kapazität / Beschränkung des Buffers

Semaphore remainingCapacity **=** capacity**;** // Der initiale Wert des neuen Semaphors entspricht der Kapazität

// Producer

**while** **(true)** **{**

newproduct **=** produce**();**

P**(**remainingCapacity**);** // Bevor ein neues Produkt in den Buffer gepusht wird, wird die verbleibende Kapazität vermindert oder der Producer in die Warteschlange aufgenommen

P**(**bufferAccess**);**

push**(**newproduct**,** buffer**);**

V**(**bufferAccess**);**

V**(**num**);**

**}**

// Consumer

**while** **(true)** **{**

P**(**num**);**

P**(**bufferAccess**);**

prod **=** pull**(**buffer**);**

V**(**remainingCapacity**);** // Nach der Entnahme eines Produkts wird die verbleibende Kapazität wieder erhöht oder der erste wartende Producer wird aufgeweckt

V**(**bufferAccess**);**

consume**(**prod**);**

**}**