Es kann nach wie vor zwei Deadlock-Situationen geben:

1. Es können alle Philosophen zeitgleich den linken Stick nehmen, auf den rechten nicht mehr zugreifen, den linken zurücklegen und wieder von vorne anfangen. Wenn immer alle Philosophen gleichzeitig essen wollen, dann wird nie einer zum Zuge kommen, aber alle werden ordnungsgemäß arbeiten (livelock).

Um dies zu beheben müsste es einen Monitor geben, der überwacht, ob ausreichend Ressourcen zur Verfügung stehen, um einen weiteren Prozesszugriff zuzulassen.

Konkret: Der Monitor müsste wissen, dass es nur fünf Sticks gibt und dass ein Philosoph immer zwei Sticks benötigt. Wenn nicht mehr genügend Ressourcen verfügbar sind, werden weitere Ressourcenanforderungen abgewiesen / in eine Warteschlange aufgenommen. Beispiel: Philosoph P1 nimmt stickl. In diesem Augenblick blockt der Monitor P1.stickr. Jetzt können sich zwar noch P2, P3 und P4 gegenseitig blockieren und P5 muss warten, aber für P1 besteht eine Garantie, ausreichend Ressourcen zu haben, Essen zu können, und die Ressourcen freigeben zu können. Im Anschluss hätte dann P2 ausreichend Ressourcen. Der livelock lässt sich jetzt reihum auflösen.

2. Es besteht die Möglichkeit, dass sich zwei Paare aus Philosophen immer gegenseitig die Sticks geben, während der letzte Philosoph nie zum Zuge kommt (deadlock).

Im vorliegenden Beispiel wäre das der Fall, wenn Philosophen P1 und P3 gleichzeitig essen, dann P2 und P4 gleichzeitig, dann wieder P1 und P3, dann wieder P2 und P4. P5 würde in diesem Szenario nie essen können. Dieser Prozess hängt in einem Deadlock.

Eine Lösung wäre, einen Monitor zu implementieren der sich merkt, wann welcher Thread zuletzt Ressourcen erhielt, und hiernach zu priorisieren. So wäre beispielsweise folgender Ablauf denkbar:

Initial: Am längsten Wartend sind die Philosophen P1, P2, P3, P4, P5.

Ressourcenzuweisung an P1 und P3 (P1 blockiert P2). Am längsten Wartend sind P2, P4, P5.

Ressourcenfreigabe und Zuweisung an P2 und P4. Am längsten Wartend sind P5, P1, P3.

Ressourcenfreigabe und Zuweisung an P5 und P3 (P5 blockiert P1). Am längsten Wartend sind P1, P2, P4.

Ressourcenfreigabe und Zuweisung an P1 und P4 (P1 blockiert P2). Am längsten Wartend sind P2, P3, P5.

So gäbe es weder livelocks noch deadlocks, alle Prozesse kämen irgendwann garantiert zum Zuge.