Motivul pentru care algoritmul Bakery compară atât etichetele, cât și label-urile proceselor este pentru a evita fenomenele de Mutual exclusion și starvation.

Mai întâi, vom analiza cazul în care se compară doar indecșii. Să presupunem că avem două procese, A și B, A > B, ambele încercând să intre în secțiunea critică, apelând mai întâi funcția lock. Problema cu acest caz este una foarte clară. Cum A > B, asta inseamnă că threadul B va avea mereu prioritate față de threadul A, indiferent de ordinea în care acestea cer permisiunea de a intra în secțiunea critică, ceea ce poate duce la fenomenul de starvation pentru threadul A. Să presupunem următoarea succesiune de instrucțiun:

write\_A(flag[A]) -> write\_B(flag[B]) -> read\_A(flag[B]) (aici A vede că flag[B] este setat pe true, dar cum label[A] > label[B], așteaptă în continuare) -> read\_B(flag[A]) (aici B intră în secțiunea critică) -> unlock\_B -> write\_B(flag[B]) -> read\_A(flag[B])

Din câte putem vedea, este posibil ca A să rămână blocat în funcția de lock deoarece B are prioritate mereu față de A, având un label mai mic.

Acum, să analizăm cazul în care se compară doar label-urile. Vom presupune cele două procese A și B care încearcă simultan să intre în secțiunea critică.

Să presupunem că se face calculul max(label[0], label[1], …, label[n-1]) simultan în cele două procese iar rezultatul este același și pentru A și pentru B. În alte cuvinte, label[A] = label[B]. Astfel, în momentul în care cele două procese vor ajunge la verificarea condiției din bucla while, vor constata că aceasta returneaza false și vor intra în același timp în secțiunea critică, astfel nerespectând principiul de Mutual Exclusion.

În concluzie, algoritmul Bakery compară tupluri de forma (label, index) pentru a asigura corectitudinea între procese și desfășurarea corectă și completă a algoritmilor la care lucrează procesele.