Kölcsönös kizárási algoritmusok vizsgálata

A kölcsönös kizárás a konkurens programozás és az elosztott rendszerek fejlesztésének klasszikus problémája. A lényege a következő: Processzek egy csoportjának egy megosztott erőforrást (pl. nyomtató) kell elérnie, amelyet egyidejűleg legfeljebb egy processz használhat. A megoldásnak a következő tulajdonságokkal kell rendelkeznie.

- *Kölcsönös kizárás*: Minden processznek végre kell hajtania egy kritikus szakasznak nevezett kódszegmenst úgy, hogy bármely adott időpillanatban legfeljebb egy processz hajthatja azt végre (azaz egy processz van a kritikus szakaszban).
- *Kiéheztetésmentesség*: A kritikus szakaszba belépni kívánó processz valamikor végül sikerrel jár, feltéve, hogy egy processz sem tartózkodik a kritikus szakaszban örökké.

A feladat keretein belül olyan elosztott algoritmusokat vizsgálunk, amelyek csak közönséges megosztott változókat használnak, tehát a kölcsönös kizárást a megosztott változókon végrehajtott írás illetve olvasás műveletekkel kell megvalósítani.

Feltesszük, hogy egy processz programja az alábbi szakaszokra bomlik:

- Belépő (próbálkozó): a kritikus szakaszba való belépés előkészítése.
- Kritikus: az egyidejű végrehajtástól megvédendő szakasz.
- Kilépő: a kritikus szakasz elhagyásakor végrehajtott kód.
- Fennmaradó: a kód többi része.

Egy-egy processz rendre a Belépő – Kritikus szakasz – Kilépő – Fennmaradó kódrészleteket hajtja végre ciklikusan. Feltesszük, hogy nincs olyan processz, ami állandóan a kritikus szakaszban tartózkodik.

A következőkben az a célunk, hogy a megadott algoritmusokat modellezzük az UPPAAL eszköz által támogatott formalizmust használva, majd megpróbáljuk a fenti elvárt tulajdonságok (mint követelmények) teljesülését az UPPAAL modellellenőrző komponense segítségével igazolni. Ha egy követelmény nem teljesül, akkor pedig elemezzük a modellellenőrző által adott ellenpéldát.

Hyman algoritmusa

Harris Hyman 1966-ban javasolta a következő algoritmust: Legyen két processz P1 és P2, a használt megosztott változók pedig a következők:

- *blocked0*: Az első processz (P1) be akar lépni;
- *blocked1*: Második processz (P2) be akar lépni;
- *turn*: Ki következik belépni (0 esetén P1, 1 esetén P2).

Az algoritmusok a két processz esetén a következők:

P1 processz	**P2 processz**

while (true) {
blocked0 = true;
while (turn!=0) {
while (blocked1==true) {
skip;
}
turn=0;
}
// Critical section here
blocked0 = false;
// Other things
}
	while (true) {
	blocked1 = true;
	while (turn!=1) {
	while (blocked0==true) {
	skip;
	}
	turn=1;
	}
	// Critical section here
	blocked1 = false;
	// Other things
	}

Vizsgáljuk meg, hogy az algoritmus teljesíti-e a következő követelményeket:

- Holtpontmenteség.
- Kölcsönös kizárás.
- Kiéheztetésmentesség.

Ha valamelyik követelmény nem teljesül, próbáljuk meg elemezni, hogy miért (milyen példa adható a követelmény megsértésére)!

Peterson algoritmusa

Peterson 1981-ben adott egy módosított algoritmust a következők szerint:

P1 processz	**P2 processz**

while (true) {
blocked0 = true;
turn=1;
while (blocked1==true &&
turn!=0) {
skip;
}
 	
// Critical section here
blocked0 = false;
// Other things
}
	while (true) {
	blocked1 = true;
	turn=0;
	while (blocked0==true &&
	turn!=1) {
	skip;
	}
	
	// Critical section here
	blocked1 = false;
	// Other things
	}

A modellezés során ügyelnünk kell arra, hogy a belső *while* ciklus feltételvizsgálata nem atomi jellegű.

Vizsgáljuk meg, hogy az algoritmus teljesíti-e a következő követelményeket:

- Holtpontmenteség.
- Kölcsönös kizárás.
- Kiéheztetésmentesség.