Hajduk Bálint

Mérnök Informatikus MSc.

Formális módszerek

Bevezetés

Az automaták egy repülőgép működését szimulálják a reptéren, illetve annak környezetében. Paraméterezésként (rendszer deklarációkon kívül) a deklarációnál meg kell adni a repülőgépek számát, és hogy legfeljebb hány gép állhat a terminálon (bármilyen értéket felvehetnek), illetve a plane automatánál a select-nél a legnagyobb indexet a terminálok tekintetében (r:int[0,i], ahol i a legnagyobb index). Ezek a számok jelen beállításoknál kettő, és egy, hogy ne legyen túl bonyolult, illetve legyen esély várakozásra a terminálon.

Fő automata (plane)

Ez az automata végzi a repülőgép vezérlésest, és követi végig az egyes fázisokon. Az automata időzítővel van ellátva, ami meghatározza az egyes fázisokhoz szükséges időt.

Kezdésként az automata várakozó állapotba kerül, ahol időnként megnézi, hogy szabad-e a kifutópálya. Minden egyes tesztnél (illetve amikor várakozóba kerül) nullázódik az időzítő, így egy guard-dal tesztelhető, hogy a kifutópályát, megfelelő időintervallumon belül elhagyta.

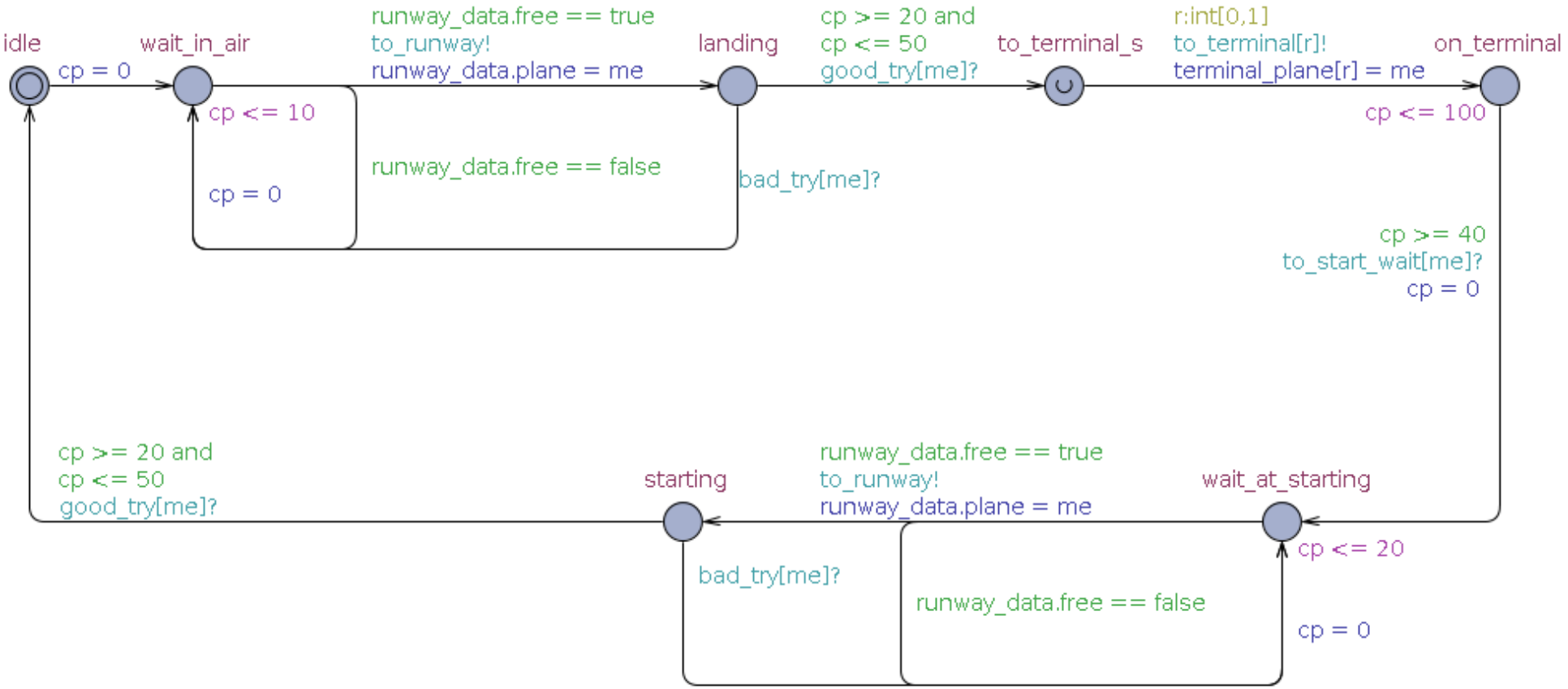
Ha szabad a pálya, akkor megpróbál leszállni, ha ez nem sikerült, akkor visszamegy várakozóba, és ugyanúgy nullázódik az idő (ez külön nincs jelezve, ha egy állapotba tartó éleknél megegyező opciók voltak, akkor egymásra raktam őket, mivel így átláthatóbb, és szerencsére az UPPAAL is támogatja).

Ha sikerült leszállni, akkor egy urgent állapoton át átmegy egy véletlenszerűen kiválasztott terminál automatába (ez nem egy terminált, hanem egy helyet jelképez a terminálon). Az urgent állapotra azért van szükség, mert az előtte, és utána lévő éleken lévő szinkronizációkat egyszerre kellene végrehajtani, viszont az UPPAAL ezt hibának veszi, ha egy élen vannak, ami kissé furcsa annak tekintetében, hogy az egyik fogadás, a másik küldés, tehát logikai probléma nem áll fenn.

Ha a terminálon végzett, akkor újra várakozó állapotba kerül, és ha szabad a kifutópálya, akkor a leszálláshoz hasonlóan megpróbál felszállni, ami ha sikerül, akkor visszakerül kezdő állapotba.

A tesztekből kiderül, hogy a rendszer nem kerülhet deadlock-ba, illetve a kifutópályán nem tartózkodhat két gép egyszerre, ellentétben a többi lényegesebb állapottal.

```
A[] not deadlock
E<> (p0.wait_in_air and p1.wait_in_air)
E<> (p0.landing and p1.landing)
E<> (p0.on_terminal and p1.on_terminal)
E<> (p0.wait_at_starting and p1.wait_at_starting)
E<> (p0.starting and p1.starting)
```



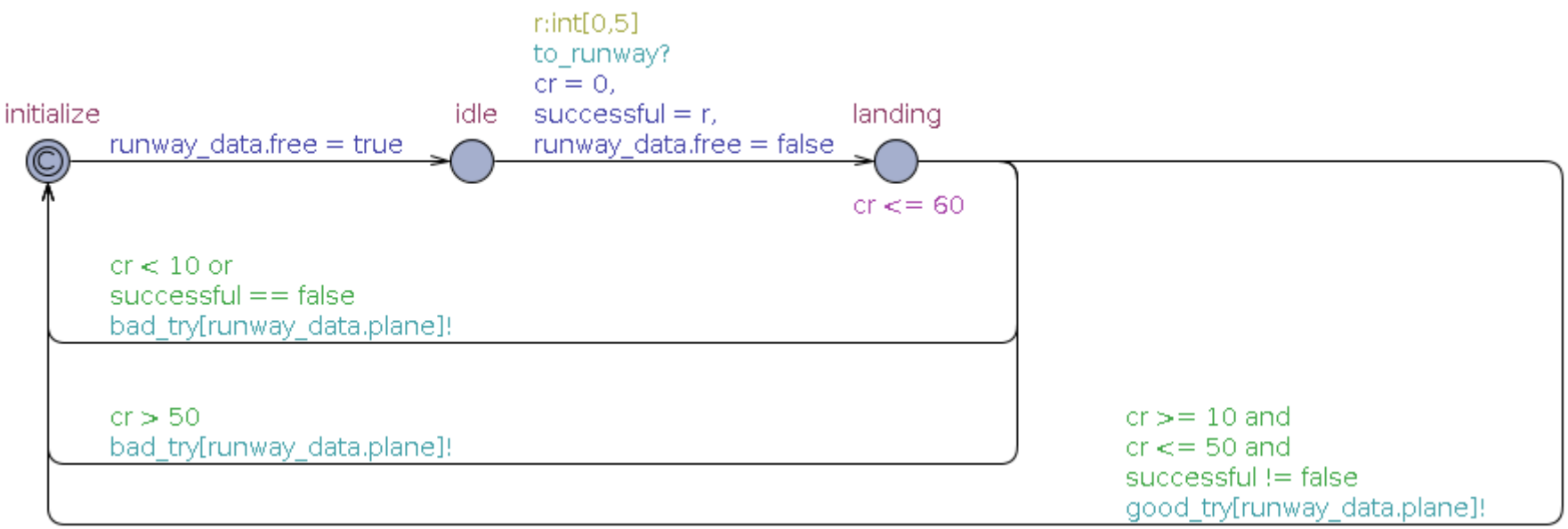
Kifutópálya (runway)

Ez vezérli a gépek le-, és felszállását. Amikor ez az automata aktív, akkor a fő automata, vagy landing, vagy starting állapotban van. A rendszer indításakor, a committed kezdő állapotnak köszönhetően szabadra állítja a pályát. Erre azért van szükség, mert a szabadságot jelképező bool változó alapértéke hamis, és mivel a plane automatában a tesztig nulla idő is eltelhet, így előfordulhatna, hogy hamis foglaltságot jelezne, ha nem ez lépne először.

Az, hogy sikeres-e a le-fel szállás, azt egyrészt egy random szám dönti el, ami ha nulla akkor nem sikkeres. A másik részről adott időintervallumon belül kell megtörténnie a műveletnek, hogy alkalmazkodjon a fő automatához. Azt, hogy nem sikerült (mivel a sikeresnél lévő guard nem elég), két úton kell eldönteni, mivel az UPPAAL nem enged két időkorlátot vagy művelettel megadni, és ezt azonosságokkal sem engedi megkerülni.

A tesztből látszik, hogy van olyan eshetőség, amikor a megfelelő időintervallumon belül sikkeres a leszállás. Tehát az automatából ki tud lépni a rendszer, és ne okoz livelock-ot.

```
E<> ((runway.cr >= 20 and runway.cr <= 50) and runway.successful > 0)
```

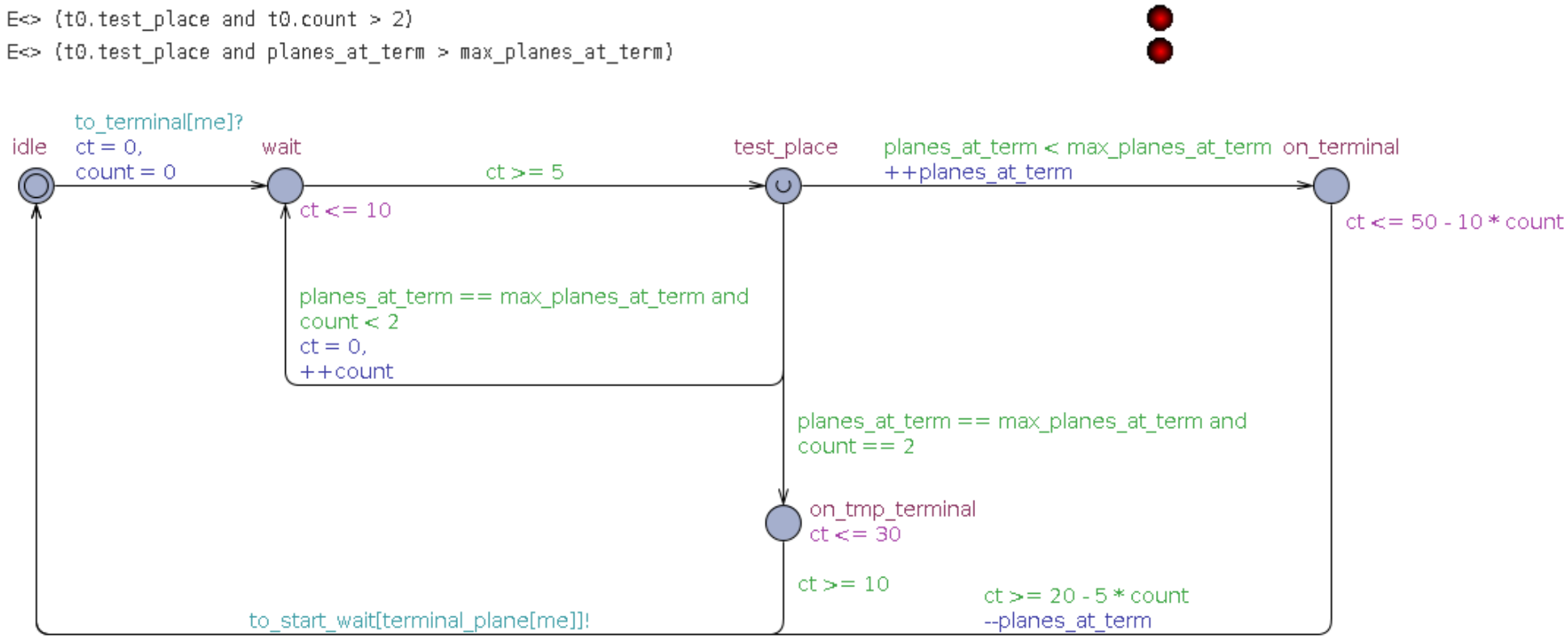


Terminál (terminal)

Amikor ez az automata aktív, akkor az adott gép a plane automata on_terminal állapotában van.

Mivel a helyek száma fix, és ez lehet kevesebb mint a gépek száma, ezért, hogy a gép tartsa magát a megadott időhöz, ezért két teszt után (urgent, hogy a fő automatához igazodjon), beáll az ideiglenes terminálra (amikor a pályán szállítják le az utasokat).

A tesztek megmutatják, hogy kettőnél nem tesztelhet többször az automata, illetve nem lehet több a beálló gép, mint a hely. Ezzel elkerülve livelockot, és deadlockot.



További tesztek

Ezekből a tesztekből kiderül, hogy ha a fő automata a kifutópályán van, akkor a kifutópálya automata aktív, illetve nem lehet várakozóban a kifutópálya automata, ha fő automata ott van. Továbbá a termnál sem lehet várakozóban, ha vannak ott gépek.

```
E<> (not(runway.idle) and (p0.landing or p1.landing or p0.starting or p1.starting))
E<> (runway.idle and (p0.landing or p1.landing or p0.starting or p1.starting))
E<> (not(t0.idle or t1.idle) and not(p0.on_terminal or p1.on_terminal))
```