Eerste Herdeeltoets Concurrency

3 januari 2014, 11.00 – 13.00, Educ-α.

Motiveer je antwoorden *kort*! Zet je mobiel uit. Stel geen vragen over deze toets; als je een vraag niet duidelijk vindt, schrijf dan op hoe je de vraag interpreteert en beantwoord de vraag zoals je hem begrijpt. Totaal 15pt, uitslag is totaal gedeeld door 1,5.

1. **Beantwoord met Amdahls Regel:** Een programma bestaat voor 80% uit parallelliseerbare code en wordt gedraaid op een quad-core.

   (a) Welke speedup wordt gehaald?
   (b) Om de executie verder te versnellen kunnen we kiezen tussen (1) het programma verbeteren zodat 90% parallelliseerbaar is; (2) meer cores bijschakelen. Hoeveel cores heb je bij optie (2) nodig om dezelfde speedup te halen als bij optie (1)?

2. **Fair executie:** In dit programma kijkt thread 1 steeds naar variabele `i`, die door thread 2 afwisselend op 0 en 1 wordt gezet. Ga uit van read/write atomicity; initieel is `i=0` en `s=True`:

```
Thread 1:          Thread 2:
  t = 0
  while (i==0)       while (s)
  { t = t + 1 }        { i = 1 - i }
  s = False           print t
```

   (a) Beschrijf van dit programma een executie waarin thread 2 de waarde 2 print.
   (b) Beschrijf een oneindige executie die *niet* kan voorkomen als de scheduler fair is.
   (c) Beschrijf een oneindige executie die *wel* mogelijk is onder een fair scheduler.

3. **Safe en Regular Bit:** Bij het implementeren van registers kennen we een onderscheid tussen een *safe* register, een *regular* register en een *atomic* register.

   (a) Leg uit, waarom regular een sterkere eigenschap is dan safe.
   (b) Leg uit, waarom regular een zwakkere eigenschap is dan atomic.
   (c) Laat zien, hoe je met een *safe bit*, een *regular bit* kunt implementeren.

4. **Pepernoten:** Een tutorklas van twaalf studenten is verdeeld in twee introductieprojectgroepjes van zes studenten. Van hun tutor krijgen ze allemaal een pepernoot uit een goed geschudde zak. Helaas blijken vier pepernoten bedorven, waardoor er vier random studenten ziek worden.
   Hoe groot is de kans dat er drie uit één groep ziek zijn en één uit de andere groep?

5. **Atomic Copy:** Een *CopyDog* object heeft als toestandsruimte een rij van waarden, en naast operaties `write(i,x)` (schrijf `x` in locatie `i`) en `read(i)` (geef waarde van locatie `i`), een operatie `copy(i,j)` die de waarde van locatie `i` atomair kopieert naar `j`. Een CopyDog met twee locaties kan 2-Consensus implementeren.

   (a) In de `propose(x)` operatie zal thread `i` (0 of 1) precies eenmaal `Arb.copy(i,1-i)` uitvoeren. Hoe kun je de initiële waarde van `Arb` kiezen om ervoor te zorgen dat alleen de *eerste* `copy`-operatie iets aan de inhoud verandert?
   (b) Hoe kan een thread na het uitvoeren van zijn `copy` zien, welke thread als eerste de `copy` deed?
   (c) Geef een wachtvrij algoritme dat een CopyDog gebruikt en 2-Consensus implementeert.
   (d) Kan een CopyDog wachtvrij worden geïmplementeerd met registers?