**Labb 4**

Labb 4.1

Första fallet:

Task som ska utföras är make-delivery(one-ring).

Givet tillstånd:

undelivered(*one-ring*),

destination(*one-ring*, *mount-doom*),

delivery-speed(*one-ring*, *slow*),

at(*hobbit*, *bag-end*),

holding(*hobbit*, *one-ring*)

Det enda som vi kan göra i det givna tillståndet är att använda metoden

make-delivery(pkg, from, to, carrier). Om vi tänker oss en trädstruktur så kommer alltså

walk(carrier, from, to) och deliver(pkg, carrier, to) att vara barn till

make-delivery(pkg, from, to, carrier). Efter att ha utfört den primära tasken walk så befinner vi oss i tillstånd:

undelivered(*one-ring*),

destination(*one-ring*, *mount-doom*),

delivery-speed(*one-ring*, *slow*),

at(*hobbit*, *mount-doom*)

holding(*hobbit*, *one-ring*)

Sedan efter att vi har utfört den primära tasken deliver så befinner vi oss I tillstånd:

destination(*one-ring*, *mount-doom*),

delivery-speed(*one-ring*, *slow*),

at(*hobbit*, *mount-doom*)

Eftersom walk och deliver är primära tasks så är vi färdiga här, trädet består endast av 3 noder.

Andra fallet:

Task som ska utföras är make-delivery(death-star-plans).

Givet tillstånd:

undelivered(*death-star-plans*),

destination(*death-star-plans*, *hidden-rebel-base*),

delivery-speed(*death-star-plans*, *fast*),

at(*farm-boy*, *desert-far-away*),

holding(*farm-boy*,*death-star-plans*),

at(*old-ship*, *desert-far-away*)

Det enda som vi kan göra i det givna tillståndet är att använda metoden

make-delivery(pkg, from, to, carrier, vehicle).

Här finns det nu två olika sätt att binda vehicle, antingen kan vehicle vara farm-boy eller old-ship. Om vehicle skulle bindas till farm-boy så skulle vi först göra:

Primitive-task: enter-vehicle(carrier, veh, loc)

precond: at(veh, loc), at(carrier, loc)

effect: ¬at(carrier, loc), in(carrier, veh)

Detta skulle resultera i att bade vehicle och carrier är farm-boy, detta leder till att farm-boy inte är på plats desert-far-away och att farm-boy är i farm-boy.

Vi skulle sedan försöka göra:

Primitive-task: travel(vehicle, from, to)

precond: at(vehicle, from)

effect: ¬at(vehicle, from), at(vehicle, to)

Här har vi en precondition som sager att farm-boy måste vara på plats desert-far-away, vilket den inte är eftersom det var en av effekterna ifrån enter-vehicle. Så denna kommer misslyckas och vi kommer backtracka till att sätta vehicle till old-ship istället.

Detta kommer nu att fungera hela vägen och vi kommer sluta i en trädstruktur enligt följande:

enter-vehicle, travel, leave-and-deliver blir barn till make-delivery och att leave-vehicle samt deliver blir barn till leave-and-deliver. Vi har nu delat problemet till bara primära tasks. Detta ger att vi har 6 stycken noder totalt i trädet.

Summering av labb 4.2 och labb 4.3

Jag tyckte att det var enklare att bygga en HTN domän när man väl hade kommit in i rätt tänk. Det blev enklare på grund av att man kunde tänkte lite mer som ”funktioner” i ett vanligt programmeringsspråk, som Java, speciellt med rekursionstänktet.

Det som var svårt var just att förstå syntaxen med bland annat ”if else” grenarna i metoderna. Det löste sig dock genom att titta på de givna exemplen på hemsidan samt exempel från föreläsningen. Denna svårighet hade troligtvis blivit enklare om jag hade väntat med labben till efter föreläsningen av Jonas.

JHOP2 är väldigt snabb på att göra beräkningar jämfört med planerarna från första labbarna. Det kändes som att det inte fanns något som gjorde att JHOP2 blev långsam. Gjorde ett ganska stort problem men det gick ändå fort förJHOP2 att ta fram en plan.

Problemen som JHOP2 klarade av är avsevärt mycket större än de problem som gjordes i classical planning och klarar dessutom av att skapa en plan snabbt. Det bror väl dels på att vi tagit bort lådor och personer och istället bara säger hur många ”lådor” som finns i depån och hur många varje plats behöver. Detta för att JHOP2 stödjer aritmetik med siffror. Mycket bra grej!

Kör tiden skalar inte med storleken på problemet, det går nästan lika fort att köra ett litet problem som ett stort.

En brist med min domän i labb 4.3 är att jag inte åker tillbaka med en carrier då den har använts en gång, istället har jag flera carriers som står och väntar i depån. Detta leder till att det finns en risk att det tar slut på carriers i depån och jag får istället köra en content i taget som i labb 4.2. Detta skulle såklart kunna lösas med att lägga till att den ska flyga tillbaka med carriern bara.

En annan brist är att jag inte kan blanda content i en och samma carrier. Detta skulle kunna lösas med att ha en räknare för varje content i en carrier.