729G78 Artificiell intelligens
Planering

Robin Keskisärkkä

HCS/IDA

LINKÖPINGS UNIVERSITET

Planering

- Klassisk planering
- Hierarkisk planering
- Icke-deterministisk planering
- Resursplanering

Klassisk planering

- Finna en sekvens av handlingar
  - Diskret, deterministisk, statisk, fullt observerbar omgivning
- Jämför sökning
  - Handlingarna ger möjliga nya tillstånd, tillståndsvektorer
  - Agenten kan testa om målet är uppnått genom att applicera en heuristisk funktion på ett tillstånd
  - Sökningen leder till en obruten sekvens av handlingar

PDDL (Planning Domain Definition Language)

- Mer uttrycksfull representation som låter agenten resonera om tillstånd och handlingar
- Söker inte blint utan kan välja operatorer som för agenten framåt
- Ett tillstånd representeras som predikat med konstanter som argument
  - Closed world assumption
  - Bara positiva predikat i tillstånd, inga negationer
  - Bara konjunktioner
  - Inga kvantifierare

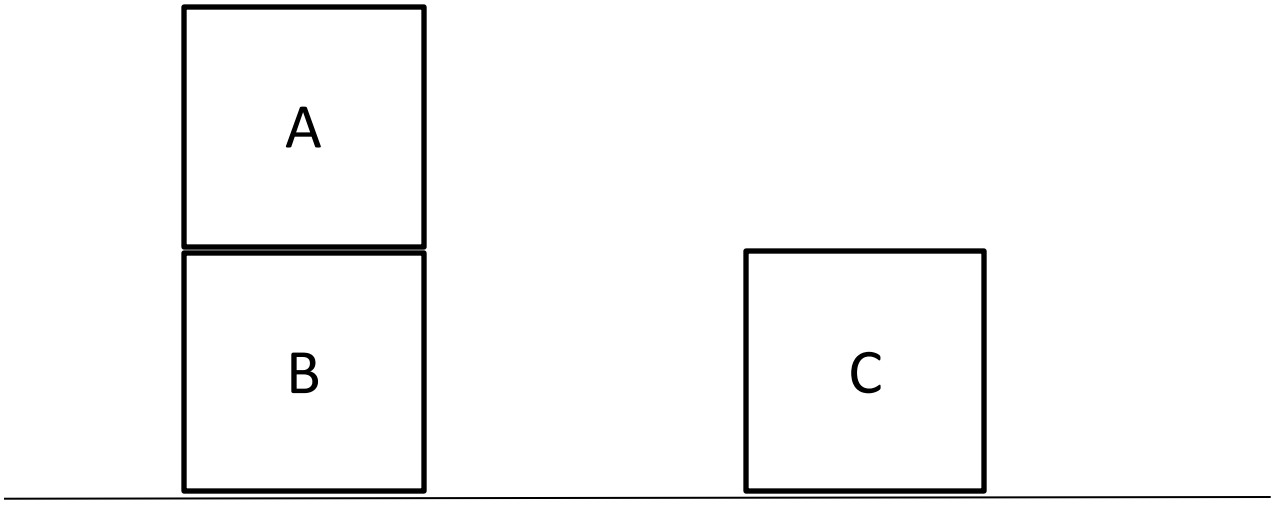
Handlingsrepresentation

- STRIPS
- Villkor (*precondition)*

  Villkor som måste vara uppfyllda för att utföra handlingen
- Effekter (*effect*)

  Effekten av att utföra handlingen. Kan delas upp i en *add*-list (det som ska läggas till) och en *delete*-list (det som ska tas bort)

Exempel, Blocks world



On(A,B) ∧ Clear(A) ∧ Clear(C) ∧ On(B, Table) ∧ On(C, Table)

Exempel, Blocks world

Predikat:

On(x, y)

Clear(x)

Handling:

Action(Move(b, x, y)

Precond: On(b, x) ∧ Clear(b) ∧ Clear(y)

Effect: On(b, y) ∧ Clear(x) ∧ ¬On(b, x) ∧ ¬Clear(y))

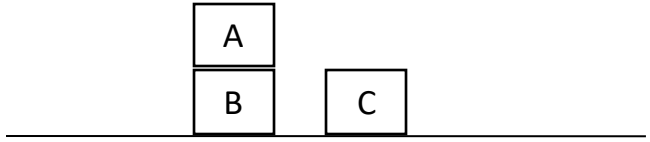
Result(state, action) = (state - DELETE(action)) ∪ ADD(action)

där:

DELETE(action) = On(b, x) ∧ Clear(y)

ADD(action) = On(b, y) ∧ Clear(x)

Exempel

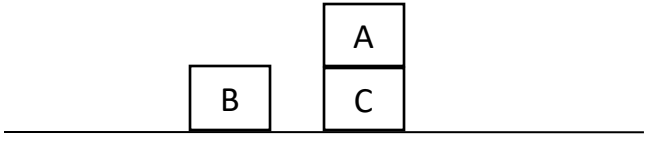


On(A,B) ∧ Clear(A) ∧ Clear(C) ∧ On(B, Table) ∧ On(C, Table)

Move(A, B, C)

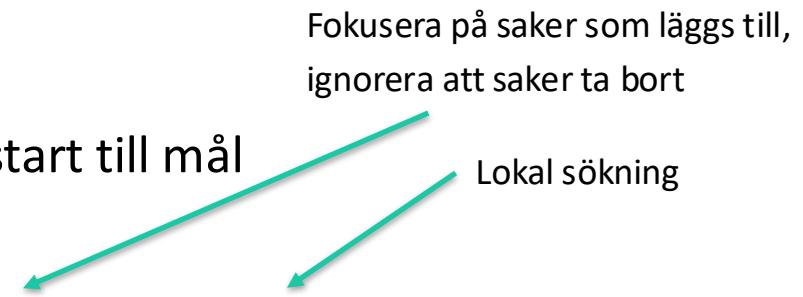
Precondition: On(A, B) ∧ Clear(A) ∧ Clear(C)

Effect: On(A, C) ∧ Clear(B) ∧ ¬On(A, B) ∧ ¬Clear(C)



On(A,C) ∧ Clear(A) ∧ Clear(B) ∧ On(B, Table) ∧ On(C, Table)

Planeringsalgoritmer



Generera en sekvens av handlingar från start till mål

- Framåtsökning
  - Blir bra med heuristik, t.ex. ignore-delete-list och hill climbing. Ger en approximativ lösning. FF algoritmen.
- Bakåtsökning
  - Utgå från målet och arbeta bakåt mot starttillståndet
- Partialordningsplanering
  - Vissa handlingar kan utföras parallellt eller i valfri ordning

STRIPS-operatorer

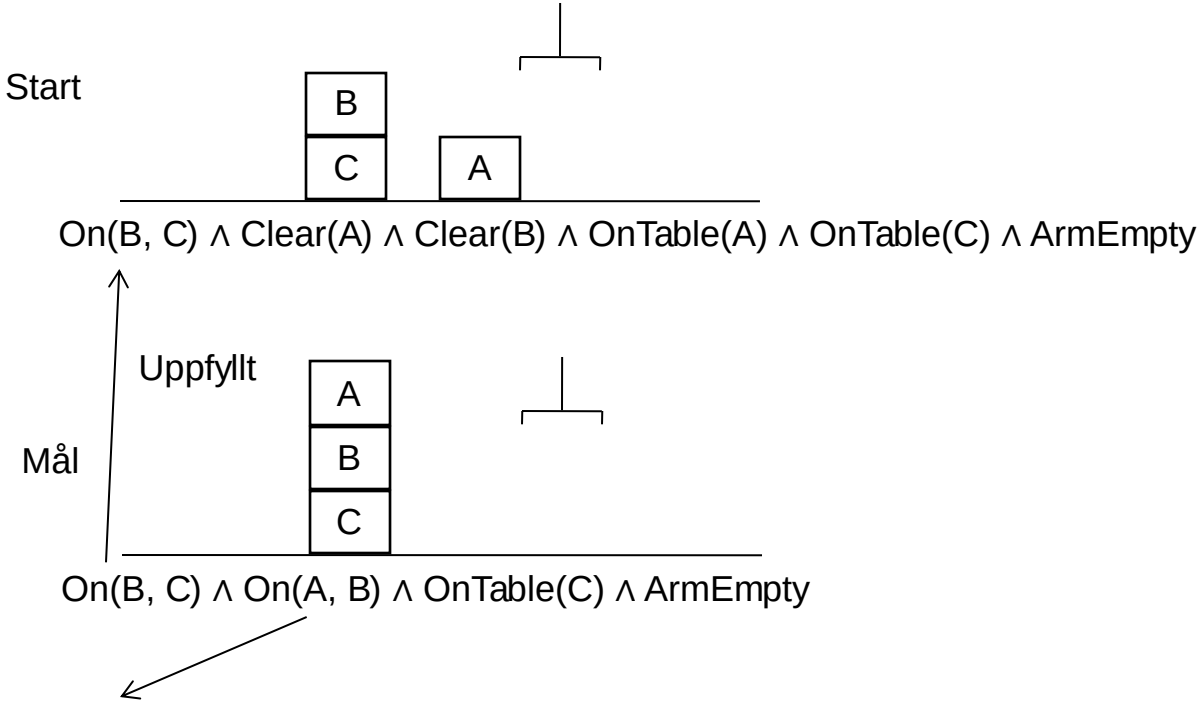
Stack(x, y)
- P: Clear(y) $\wedge$ Holding(x)
- D: Clear(y) $\wedge$ Holding(x)
- A: ArmEmpty $\wedge$ On(x, y)

UnStack(x, y)
- P: On(x,y) $\wedge$ Clear(x) $\wedge$ ArmEmpty
- D: On(x, y) $\wedge$ ArmEmpty
- A: Holding(x) $\wedge$ Clear(y)

PickUp(x)
- P: Clear(x) $\wedge$ OnTable(x) $\wedge$ ArmEmpty
- D: OnTable(x) $\wedge$ ArmEmpty
- A: Holding(x)

PutDown(x)
- P: Holding(x)
- D: Holding(x)
- A: OnTable(x) $\wedge$ ArmEmpty

Exempel



Leta efter operatorer som har On(x, y) på sin ADD-list, dvs Stack(A, B)

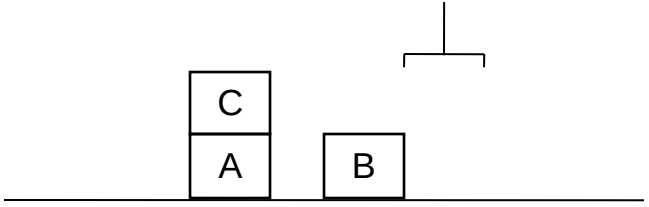
Precondition: Clear(B), Holding(A)

Leta efter operator som har Holding(x) på ADD-list: PickUp eller UnStack

Välj PickUp(A). Precondition: ArmEmpty, Clear(A), OnTable(A) uppfyllt

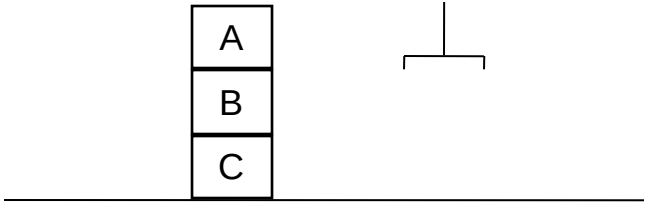
Nytt exempel

Start



On(C, A) ∧ Clear(C) ∧ Clear(B) ∧ OnTable(A) ∧ OnTable(B) ∧ ArmEmpty

Mål



On(A, B) ∧ On(B, C) ∧ OnTable(C) ∧ ArmEmpty

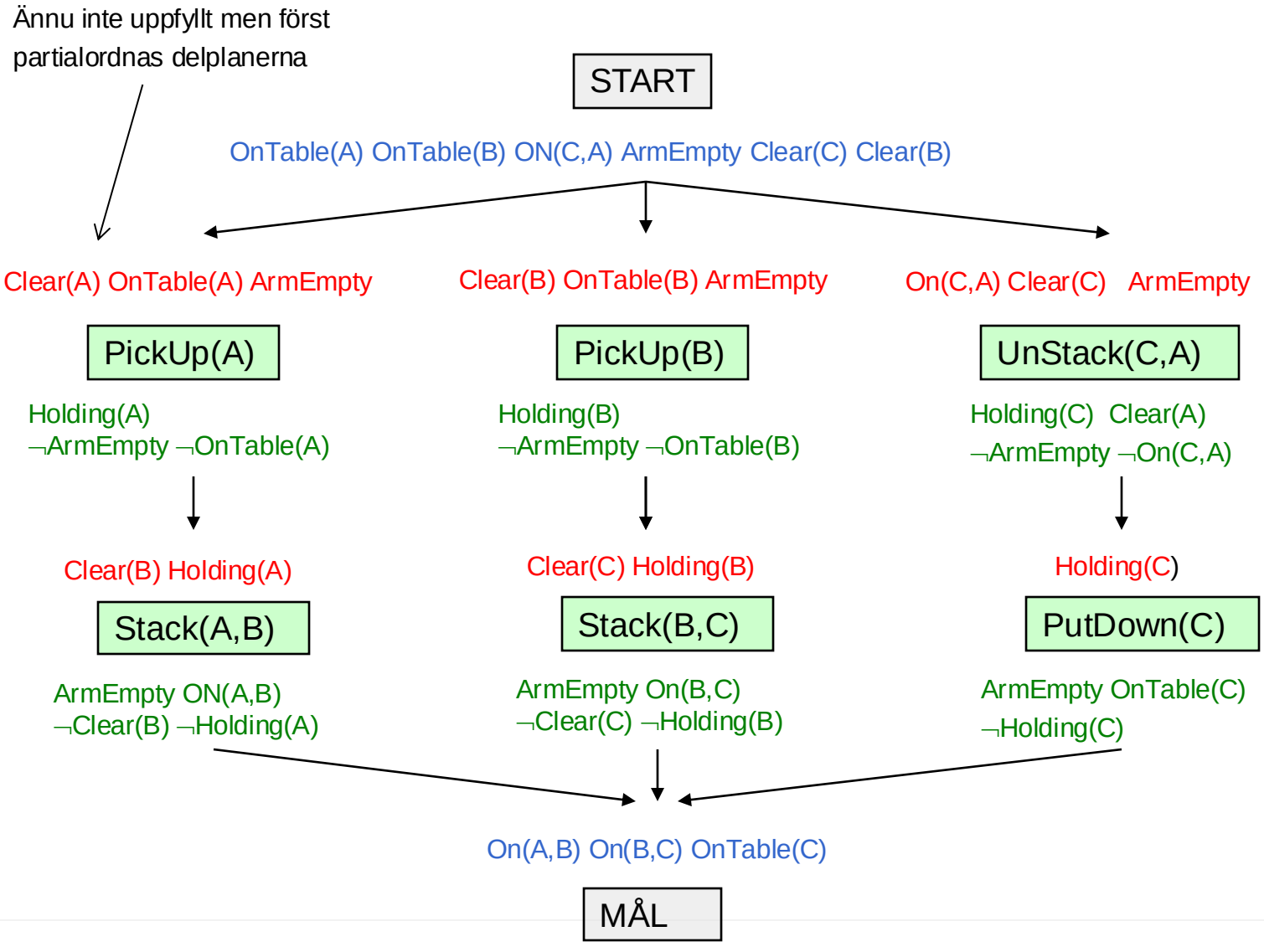
Börja med ett mål, t.ex. On(A, B), ger UnStack(C, A), PutDown(C), PickUp(A), Stack(A, B)

Ta sen nästa mål On(B, C). Ger UnStack(A, B), PutDown(A), PickUp(B), Stack(B, C)

Nu är inte On(A, B) uppfyllt så PickUp(A), Stack(A, B)

Partialordningsplanering

- Ett problem: STRIPS arbetar med ett mål i taget
- Vill kunna avbryta uppfyllandet av ett mål och påbörja nästa och sen fortsätta med det första igen
  - Uppnå On(A, B) ⇒ UnStack(C, A), PutDown(C)
  - Fortsätt med On(B, C) ⇒ PickUp(B), Stack(B, C)
  - Återuppta On(A, B) ⇒ PickUp(A), Stack(A, B)
- Partialordningplanerare skapar partiellt ordnade delplaner enligt least commitment strategy, dvs fatta så få beslut som möjligt


Ännu inte uppfyllt men först partialordnas delplanerna
START
OnTable(A) OnTable(B) ON(C,A) ArmEmpty Clear(C) Clear(B)
Clear(A) OnTable(A) ArmEmpty
Clear(B) OnTable(B) ArmEmpty
On(C,A) Clear(C) ArmEmpty
PickUp(A)
PickUp(B)
UnStack(C,A)
Holding(A)
¬ArmEmpty ¬OnTable(A)
Holding(B)
¬ArmEmpty ¬OnTable(B)
Holding(C) Clear(A)
¬ArmEmpty ¬On(C,A)
Clear(B) Holding(A)
Clear(C) Holding(B)
Holding(C)
Stack(A,B)
Stack(B,C)
PutDown(C)
ArmEmpty ON(A,B)
¬Clear(B) ¬Holding(A)
ArmEmpty On(B,C)
¬Clear(C) ¬Holding(B)
ArmEmpty OnTable(C)
¬Holding(C)
On(A,B) On(B,C) OnTable(C)
MÅL

Partialordna, 1

Operatorer som lagts till för att uppnå delmål partialordnas

$$PickUp(A) \prec Stack(A,B)$$
$$PickUp(B) \prec Stack(B,C)$$
$$UnStack(C,A) \prec PutDown(C)$$

Vid konflikt ordnas operatorer så att konflikten undviks

- Stack(A,B) är i konflikt med PickUp(B) eftersom Stack tar bort Clear(B)
- Stack(B,C) är på samma sätt i konflikt med UnStack(C,A)

Partialordna:

$$PickUp(B) \prec Stack(A,B)$$
$$UnStack(C,A) \prec Stack(B,C)$$

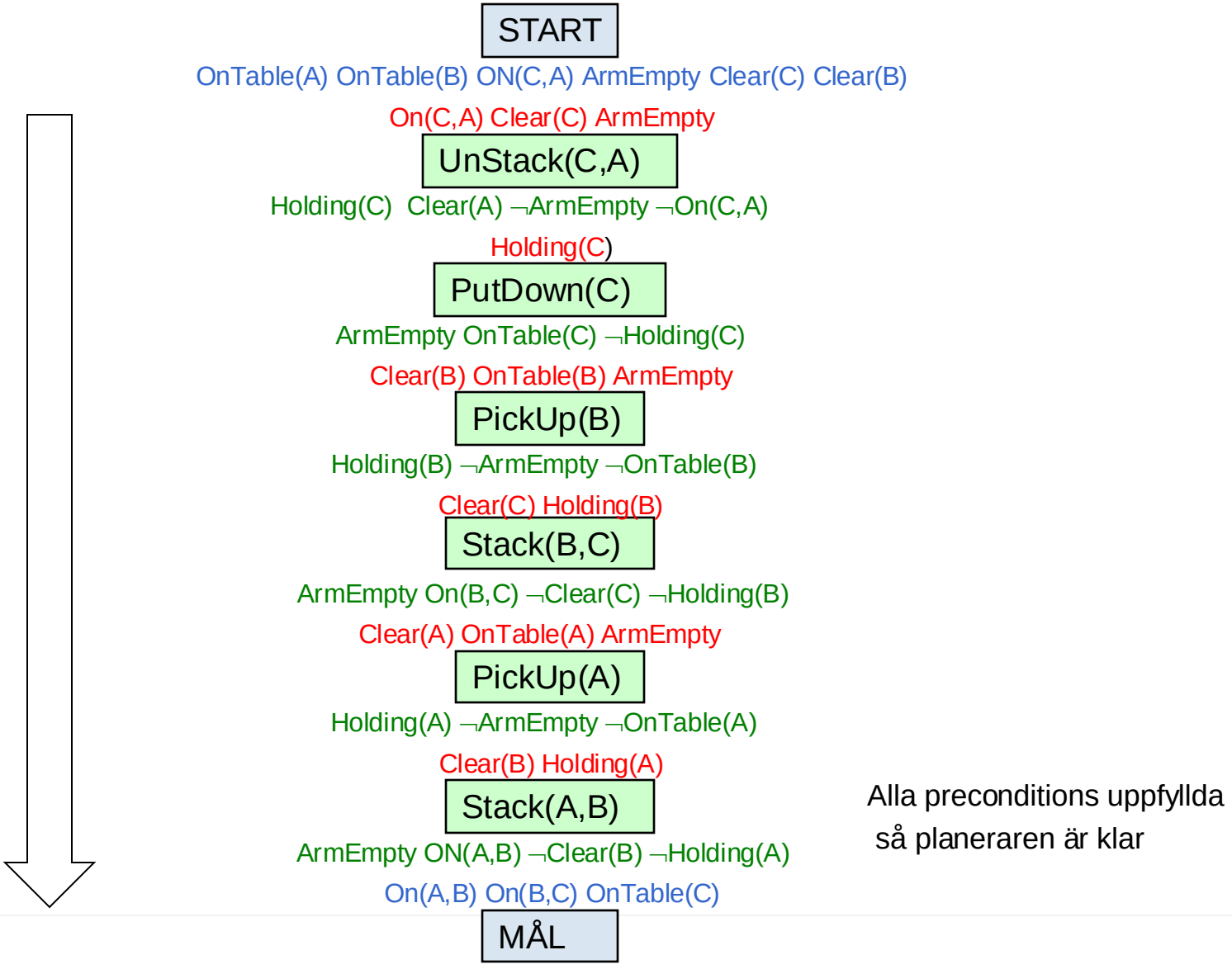
Partialordna, 2

Finns det ytterligare ordning mellan de partiellt ordnade planerna?

$$PickUp(A) \prec Stack(A,B)$$
$$PickUp(B) \prec Stack(B,C)$$
$$UnStack(C,A) \prec PutDown(C)$$
$$PickUp(B) \prec Stack(A,B)$$
$$UnStack(C,A) \prec Stack(B,C)$$

PickUp(B) (och därmed också Stack(B,C)) före Stack(A,B) och UnStack(C,A) före Stack(B,C)

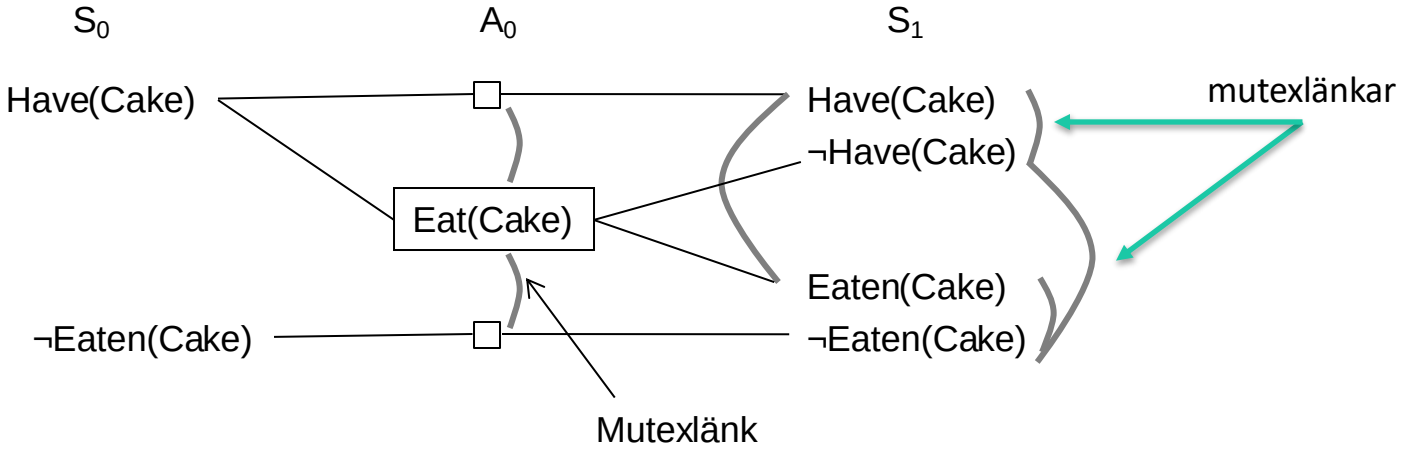
$$UnStack(C,A) \prec PutDown(C) \prec PickUp(B) \prec Stack(B,C) \prec PickUp(A) \prec Stack(A,B)$$



Alla preconditions uppfyllda
så planeraren är klar

Planeringsgraf

- Graf med en sekvens av nivåer som svarar mot temporala steg i planen.
- Representerar handlingar och ”icke-handlingar”
- Ex

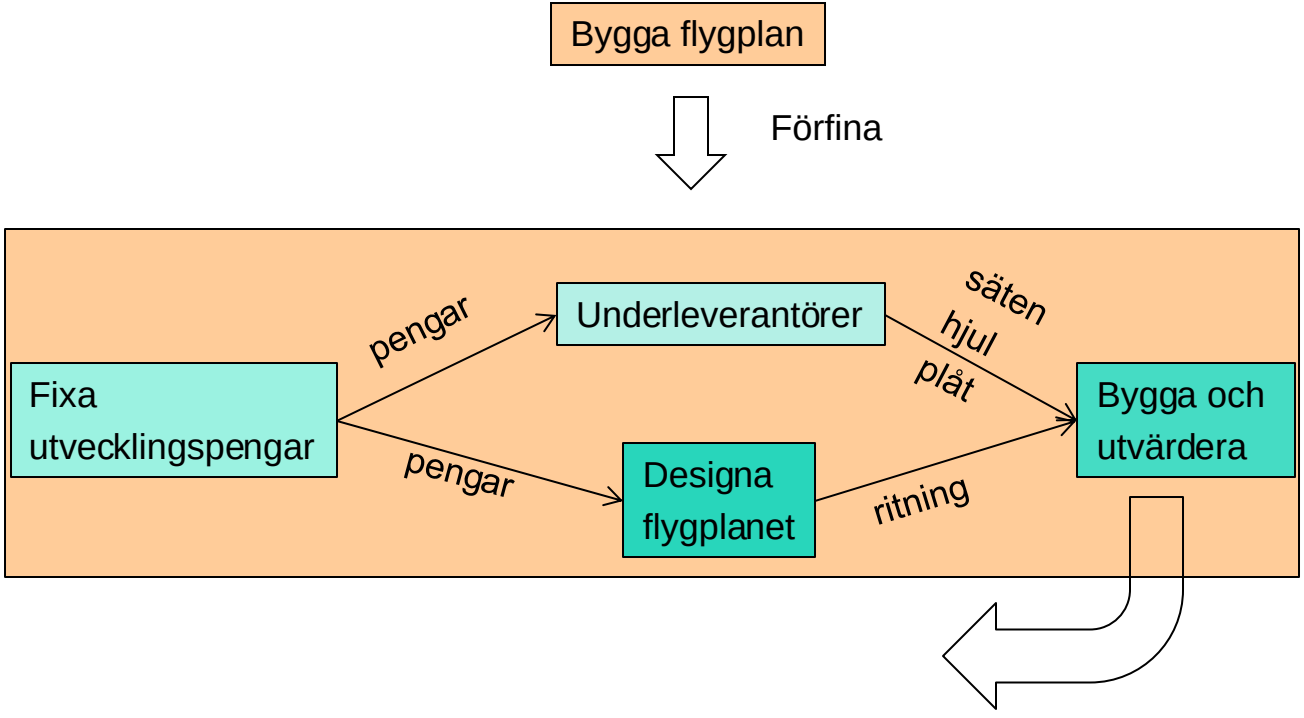


Hierarkisk planering

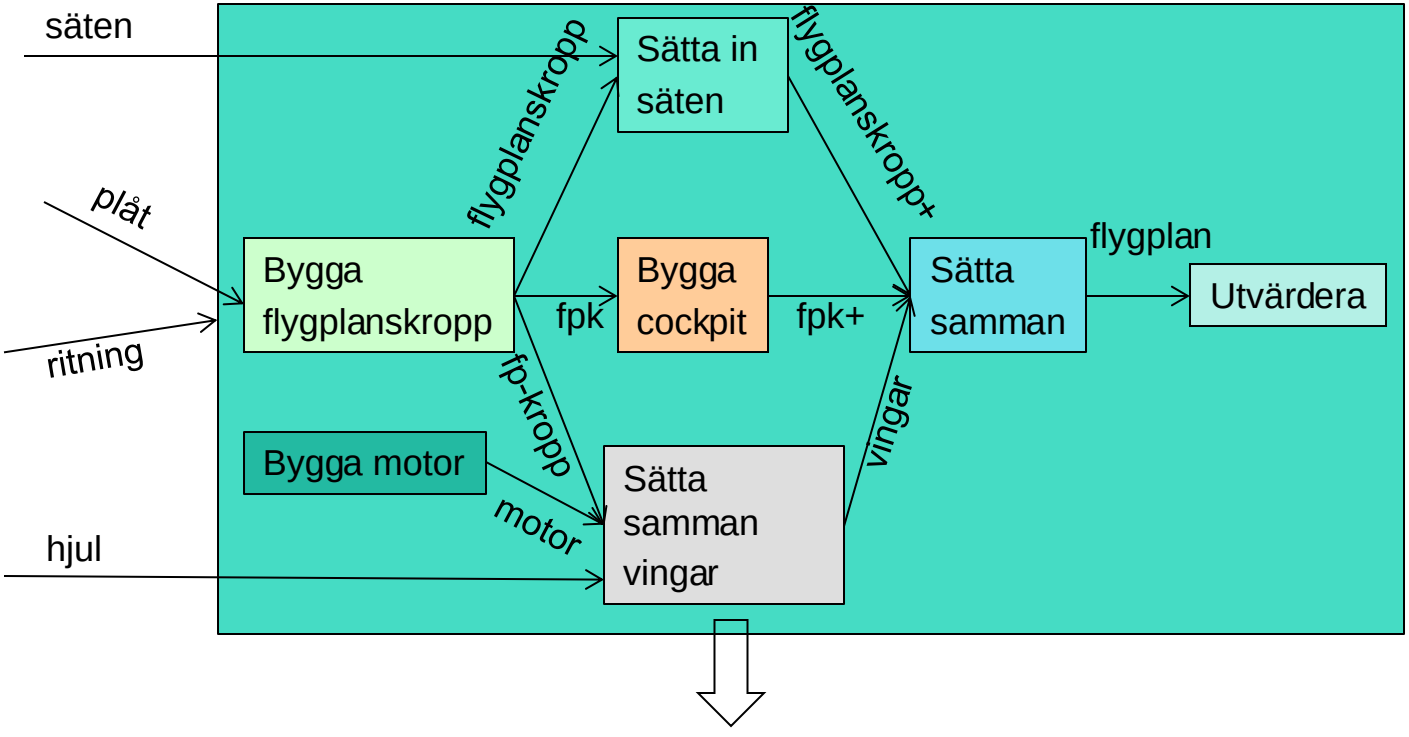
Hierarkisk planering

- HLA (high level action) abstrakta operatorer som håller steg i planen
  - Kan bestå av nya HLA eller primitiva operatorer
  - Kan innehålla flera möjliga lösningsvägar
- Förfinas till dess att planeraren bara har primitiva operatorer

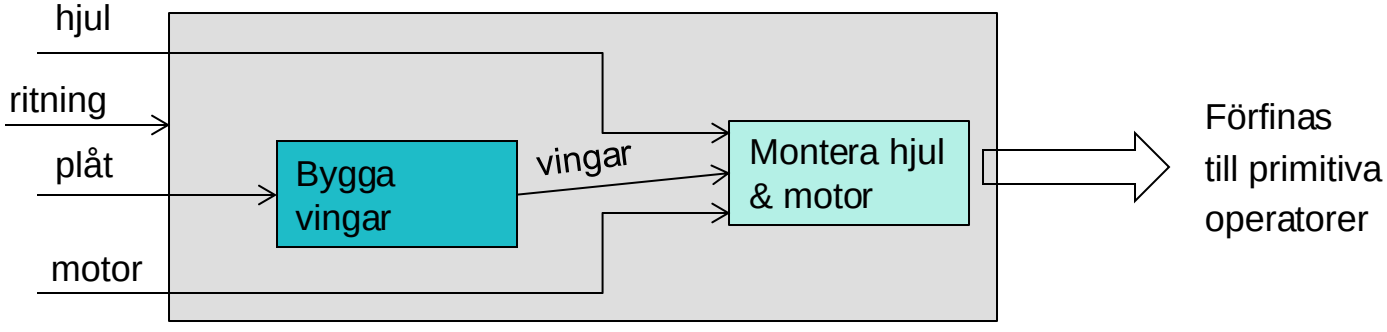
Exempel, bygga flygplan



Bygga och utvärdera förfinad



Sätta samman vingar förfinad



Planering i icke-deterministiska domäner

Planering i icke-deterministiska domäner

- Hittills har världen varit full observerbar, statisk och deterministisk
- Handlingar korrekta och kompletta
- Två problem:
  - Världen är inte fullt observerbar, kan t.ex. inte se allt som finns
  - Världen är inte ”korrekt”. Någon kan t.ex. flytta objekt eller handlingar misslyckas

Strategier för att hantera motsägelser och icke-komplett information

- Sensorlös planering
  - Skapar plan utan sensordata
- Villkorlig planering
  - Skapar plan för olika utfall
- Omplanerande system
  - Inspekterar världen och planen innan handling

Exempel, 1

Måla stol och bord i samma färg.

Init(Objekt(Bord) $\wedge$ Objekt(Stol) $\wedge$ Burk(B_1) $\wedge$ Burk(B_2) $\wedge$ ISynfältet(Bord))
Mål(Färg(Stol, c) $\wedge$ Färg(Bord, c))

Action(TaAvLock(b),
    Precond: Burk(b)
    Effect: Öppen(b))

Action(Måla(x, b),
    Precond: Burk(b) $\wedge$ Objekt(x) $\wedge$ Färg(b, f) $\wedge$ Öppen(b)
    Effect: Färg(x, f))

Exempel, 2

Varseblivningsschema

Percept(Färg(x, c),
    Precond: Objekt(x) $\wedge$ ISynfältet (x))
Agenten vet färgen, c, på objektet, x.

Percept(Färg(b, c),
    Precond: Burk(b) $\wedge$ ISynfältet (b) $\wedge$ Öppen(b))

Action(TittaPå(x),
    Precond: ISynfältet (y) $\wedge$ x $\neq$ y
    Effect: ISynfältet (x) $\wedge$ $\neg$ISynfältet (y))

Sensorlös planering, 1

- Ser inget och kan därmed bara måla både bord och stol i en av de två färgerna.
- Inga varseblivningsschema
  Init(Objekt(Bord) $\wedge$ Objekt(Stol) $\wedge$ Burk(B_1) $\wedge$ Burk(B_2))
- Vet också att objekt kan ha en färg
  $\forall x \exists y$ Färg(x, y) vilket efter skolemisering ger Färg(x, C(x))

Sensorlös planering, 2

- Applicera handlingen:
  Action(TaAvLock(b), Precond: Burk(b), Effect: Öppen(b))
  med $\{b/B_1\}$ ger:
  Objekt(Bord) $\wedge$ Objekt(Stol) $\wedge$ Burk(B_1) $\wedge$ Burk(B_2) $\wedge$ Öppen(B_1) $\wedge$ Färg(x, C(x))
- Med $\{x/B_1$ och $f/C(B_1)\}$ kan vi applicera handlingen:

  Action(Måla(x, b),
  Precond: Burk(b) $\wedge$ Objekt(x) $\wedge$ Färg(b, f) $\wedge$ Öppen(b)
  Effect: Färg(x, f))

  på t.ex. Stol och får:

  Action(Måla(Stol, B_1),
  Precond: Burk(B_1) $\wedge$ Objekt(Stol) $\wedge$ Färg(B_1, C(B_1)) $\wedge$ Öppen(B_1)
  Effect: Färg(Stol, C(B_1))

Sensorlös planering, 3

- Samma handling på Bord ger:

  Färg(Bord, $C(B_1)$)
- och måltillståndet:

  Färg(Stol, $C(B_1)$) $\wedge$ Färg(Bord, $C(B_1)$)

Villkorlig planering

Skapa en plan som inspekterar världen och handlar därefter

[TittaPå(Bord), TittaPå(Stol),

if Färg(Bord, c) $\wedge$ Färg(Stol, c) **then** NoOp

else [TaAvLock($Burk_1$), TittaPå($Burk_1$), TaAvLock($Burk_2$), TittaPå($Burk_2$),

if Färg(Bord, c) $\wedge$ Färg(burk, c) **then** Måla(Stol, burk)

else if Färg(Stol, c) $\wedge$ Färg(burk, c) **then** Måla(Bord, burk)

else [Måla(Stol, $Burk_1$), Måla(Bord, $Burk_1$)]]]

Omplanerande system

- Action monitoring
  - Säkerställ att precond fortfarande håller
- Plan monitoring
  - Säkerställ att planen fortfarande håller
- Goal monitoring
  - Se efter om det nu finns bättre mål att uppnå

Resursplanering

Resursplanering

- Handlingar tar en viss tid att utföras
- Resurser kan vara begränsade

Resursplanering, exempel

Init(Vinge(v_1) $\wedge$ Vinge(v_2) $\wedge$ Motor(m_1, v_1, 40) $\wedge$
Motor(m_2, v_2, 30) $\wedge$ Hjul(h_1, v_1, 20) $\wedge$ Hjul(h_2, v_2, 15))

Goal(Klar(v_1) $\wedge$ Klar(v_2))

Action(AddMotor(m, v, d),

Precond : Motor(m, v, d) $\wedge$ Vinge(v) $\wedge \neg$MotorOn(v),

Effect: MotorOn(v) $\wedge$ Duration(d))

Action(AddHjul(h, v, d),

Precond : Hjul(h, v, d) $\wedge$ Vinge(v) $\wedge$ MotorOn(v) $\wedge \neg$HjulOn(v),

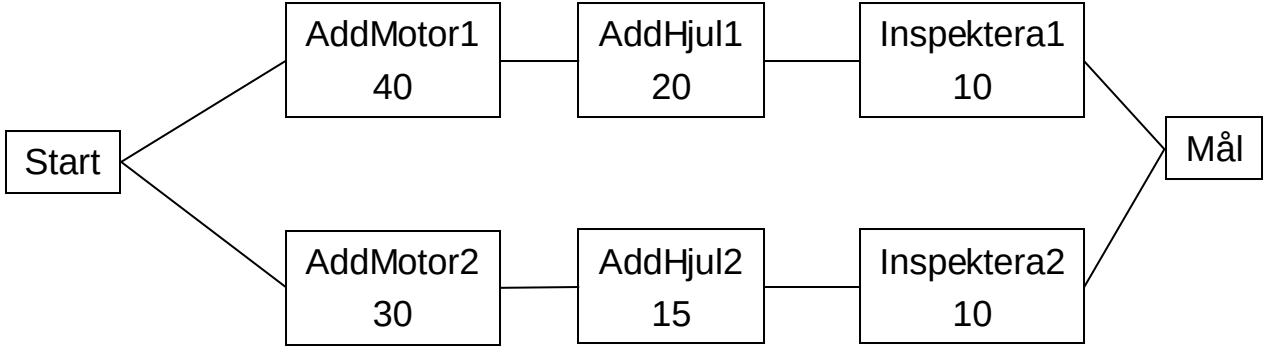
Effect: HjulOn(h) $\wedge$ Duration(d))

Action(Inspektera(v),

Precond : MotorOn(v) $\wedge$ HjulOn(v) $\wedge$ Vinge(v),

Effect: Klar(v) $\wedge$ Duration(10))

Vanlig partialordningsplanerare



Resursplanering

Måste veta när handlingar börjar och slutar

- ES: tidigaste möjliga starttid
- LS: senast möjliga starttid
- Slack = LS-ES

Exempel

ES

AddMotor1 0 AddHjul1 40 Inspektera1 60

AddMotor2 0 AddHjul2 30 Inspektera2 45

LS

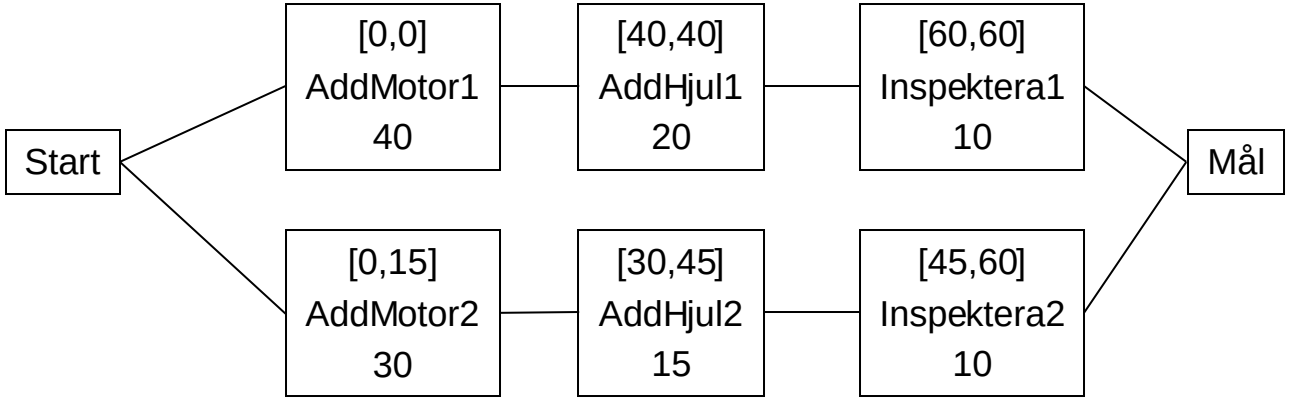
$LS(OP_{i-1})=LS(OP_i)-Duration(OP_{i-1})$, LS(Mål)=ES(Mål)=70

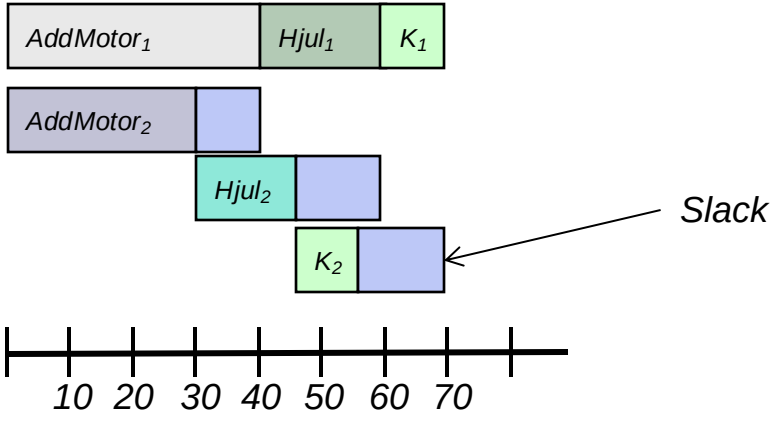
Inspektera2=70-10, AddHjul2=60-15, AddMotor2=45-30

AddMotor2 0 15 AddHjul2 30 45 Inspektera2 45 60

AddMotor1 0 0 AddHjul1 40 40 Inspektera1 60 60

Resursplanerare





Planering med resursbegränsning

- Antag begränsade resurser. Resource anger begränsningen
- Svårare problem eftersom planeraren måste välja vilken handling som skall utföras först
- Ofta används minimum slack, dvs ta i varje steg den plan som har minst slack, Greedy.