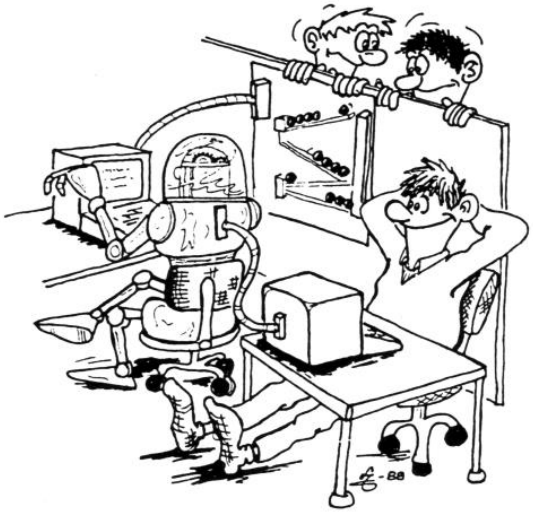
Industriautomation

Tentamen SSY 066, fredag 15/01–2021, fm

Lärare: Kristofer Bengtsson, 0768-979561



Fullständig lösning ska lämnas på samtliga uppgifter. I förekommande fall av tvetydigt formulerade tentamensuppgifter ska den föreslagna lösningen och eventuella antaganden motiveras. Examinator förbehåller sig rätten att godkänna rimligheten i antaganden och motiveringar.

Totalt omfattar tentamen 40 poäng. För betygen tre, fyra och fem krävs 16, 24 resp. 32 poäng.
Lösningar anslås direkt efter tentatillfället på Canvas.

Uppgift 1 - graf

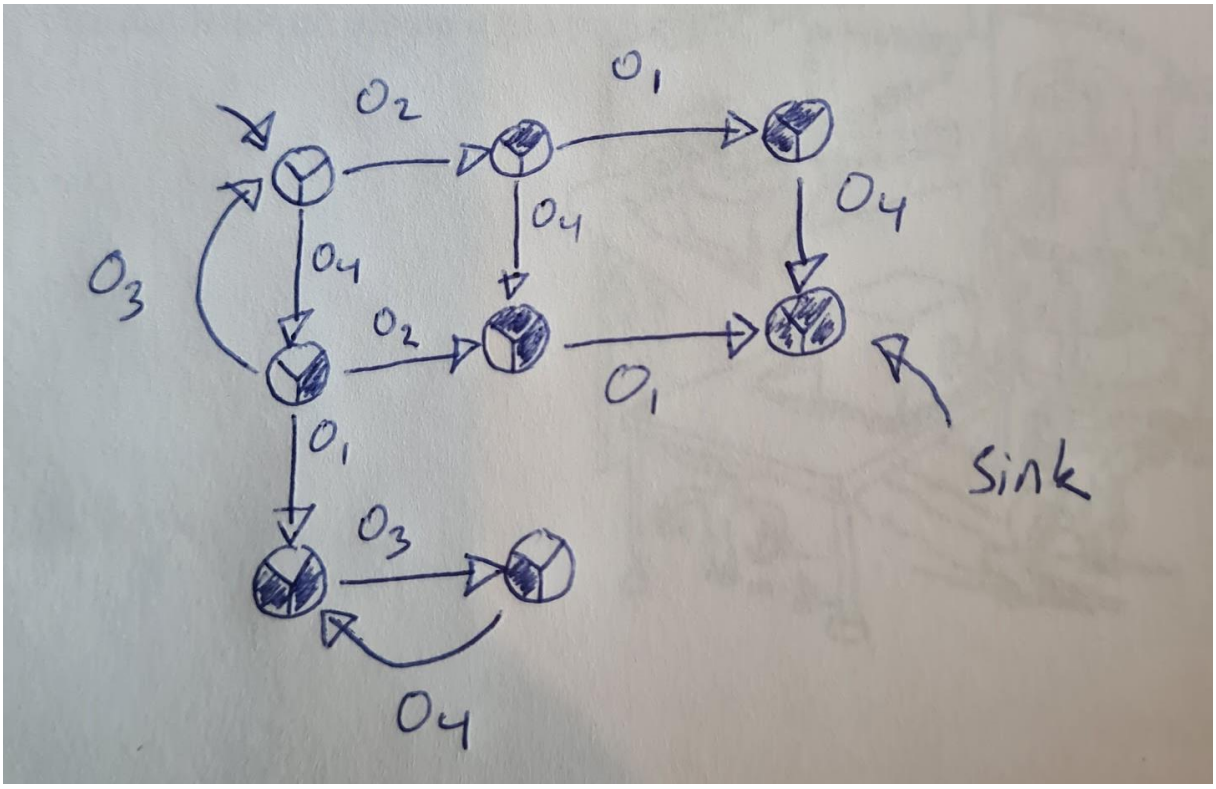
Rita upp grafen som beskriver hur följande operationer ändrar tillståndet. Operationerna tar ingen tid utan är bara guards och actions. Initialtillståndet är $a = b = c = false$.

$O_1: (a \vee b) \wedge \neg c \ / \ c \coloneqq true$
$O_2: \neg c \wedge \neg a \ / \ a \coloneqq true$
$O_3: b \wedge \neg a \ / \ b \coloneqq false$
$O_4: \neg b \ / \ b \coloneqq true$

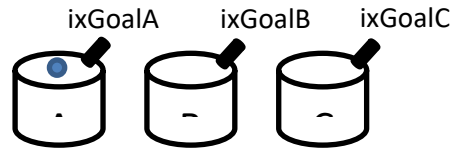
Vilka tillstånd är ”Sink States”?

(5 poäng)

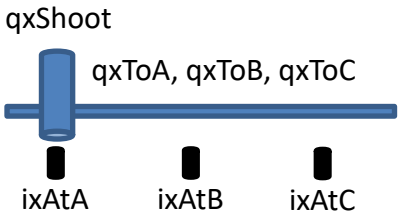
Möjlig lösning uppgift 1



Uppgift 2 - SFC



I denna uppgift skall du implementera styrningen av en bollskjutarmaskin, som skjuter 3 bollar, 1 i respektive korg, innan den startar om och skjuter i tre nya. Problemet är att den inte alltid prickar, utan ibland behöver försöka igen.



Bollskjutaren styrs med qxToA, qxToB, qxToC, vilka kör skjutaren till respektive position när de är sanna och qxShoot, som skjuter en boll (det räcker att den är sann under en scan-cykel). Det finns tre sensorer, ixAtA, ixAtB och ixAtC som mäter om skjutaren är på plats vid respektive position.

Skjutaren är riktigt snabb och hinner åka och skjuta flera bollar innan den första bollen landar i korgen. Om bollen går i ”mål” blir respektive sensor, ixGoalA, ixGoalB och ixGoalC sanna under en kort stund. Om inte respektive sensor blir sann inom 1s så har bollen missat korgen och då skall skjutaren skjuta igen.

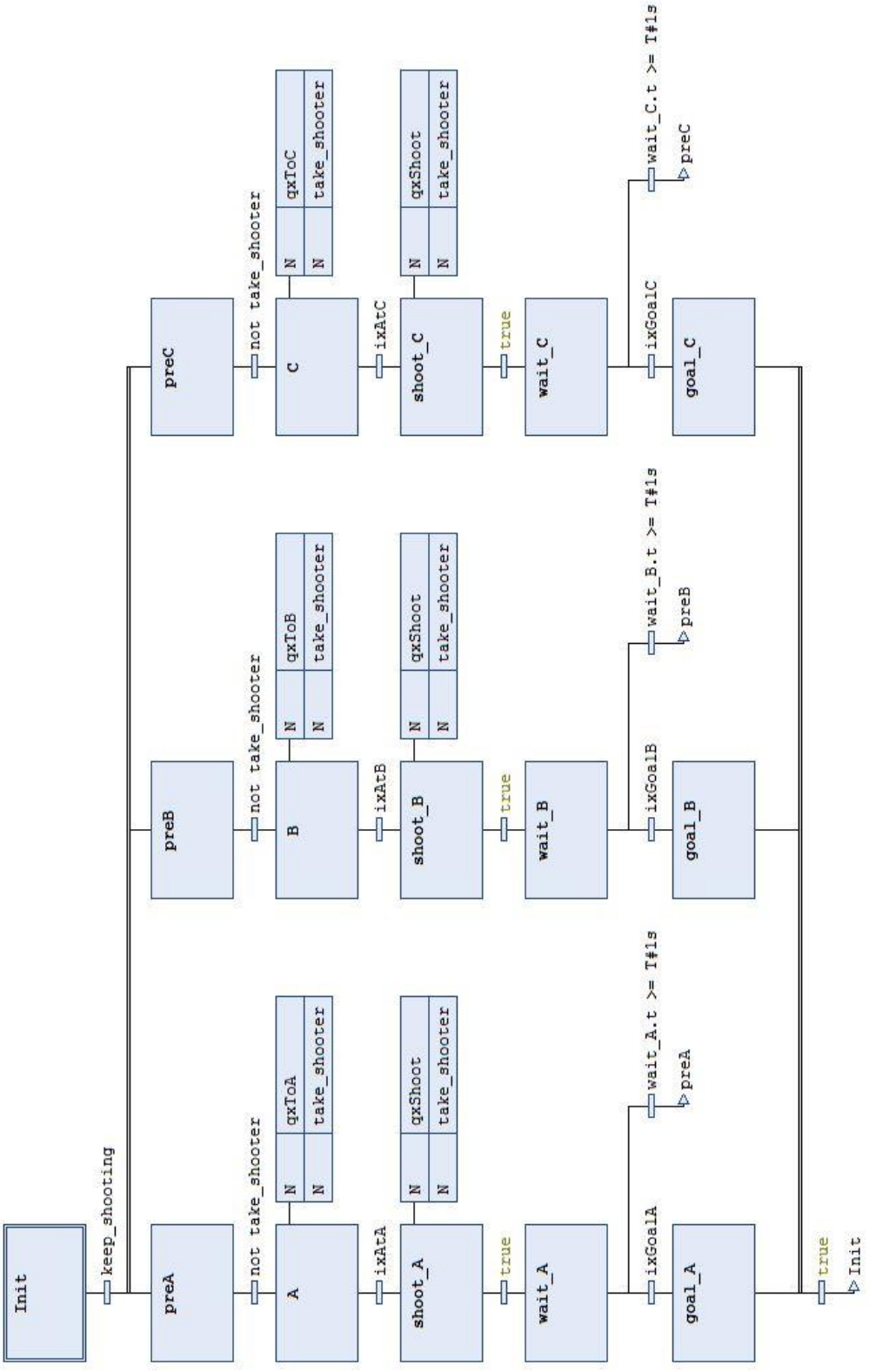
Du skall implementera styrningen av skjutaren med hjälp av en eller flera SFCer. Skjutaren skall så fort den kan skjuta de tre bollarna i respektive korg. Eftersom skjutaren är snabb skall din kod åka vidare och försöka skjuta i nästa korg innan du vet om den första bollen gått i korgen. Koden får alltså inte vänta på att en boll gått i mål om det finns andra korgar att skjuta bollar i. Vilken ordning som skjutaren skjuter i korgarna spelar ingen roll. När alla tre korgarna har fått en boll i sig, så skall din kod börja om och skjuta i tre nya bollar.

Om du vill använda ST-koden i en action association (tex för att kunna använda annat än boolska variabeltyper), skriv den direkt i rutan. Om du använder codesys, klipp ut ST-koden och klistra in en bild så den hamnar i SFCn.

(8 poäng)

Möjlig lösning uppgift 2

Den viktiga delen är att dela upp de tre korgarna antingen i tre SFCer eller som nedan i en parallell. Dessa kan inte styra armen samtidigt så därför måste de blockera de andra sekvenserna när de kör.



Uppgift 3 - ladder

I denna uppgift skall du implementera en stack i ladder. Du har följande variabler:

stack: ARRAY[1..20] of INT;
input: INT;
output: INT;
ixPush: BOOL;
ixPull: BOOL;
qxEmpty: BOOL;

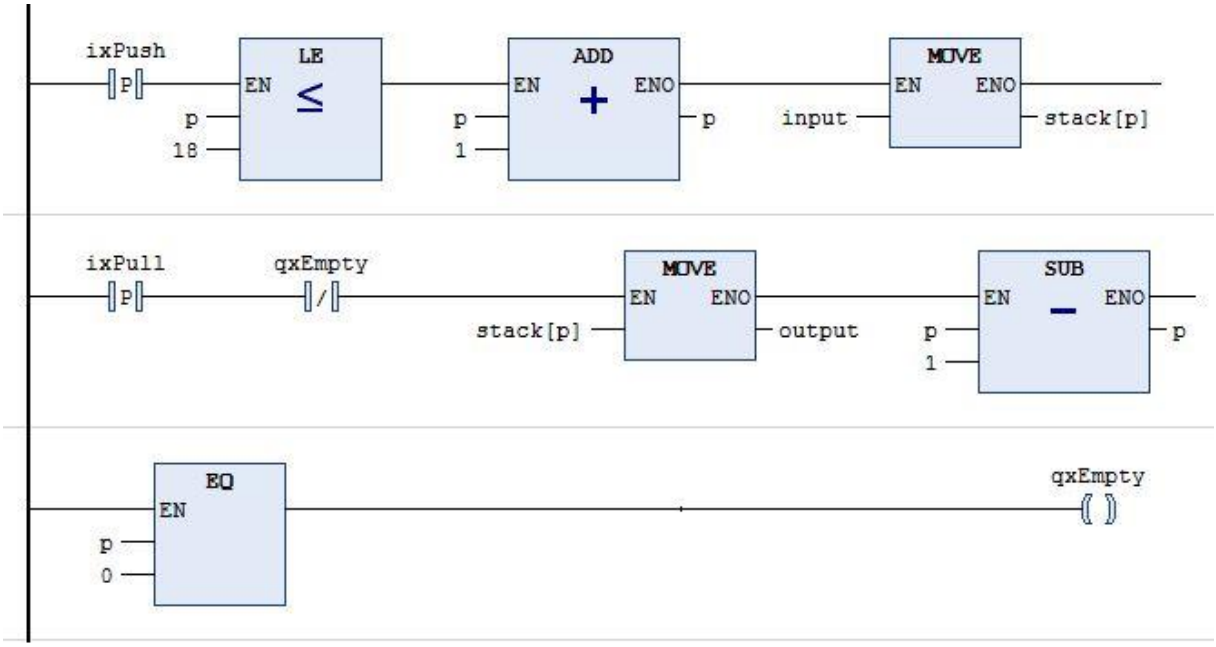
Implementera ladderkod för en stack, alltså en Last-In-First-Out kö, så att det går att mata in värdet som ligger i input, när ixPush trycks in, och mata ut ett värde från stacken till output, när ixPull, trycks in. Om stacken är tom skall qxEmpty vara sann och det skall inte gå att lägga in fler värden om stacken är full.

Om du behöver, skapa egna variabler.

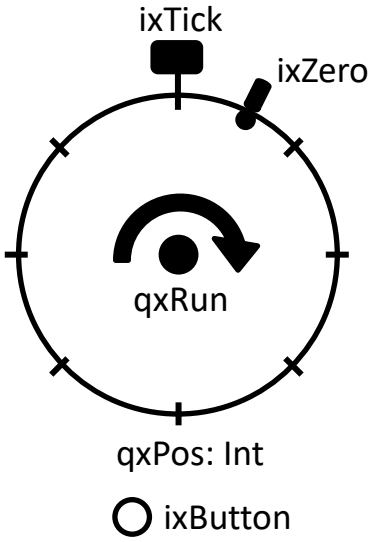
(7 poäng)

Möjlig lösning uppgift 3:

Intern variabel: p: int := 0;



Uppgift 4 - ladder



I denna uppgift skall du implementera styrningen av ett lotterihjul i ladder. Hjulet startas och stoppas genom att trycka in knappen ixButton och dess nuvarande position visas på en display som är kopplad till qxPos som är en INT. Övriga in och utgångar är BOOL. Hjulets har en sensor som är sann när det är i ”noll”-position, där qxPos skall vara 0. Sedan finns det en liten arm som böjs när små piggar åker förbi. När armen böjs blir ixTick sann och när en pigg passerat, åker armen tillbaka och ixTick blir falsk. När armen har passerat en pigg, skall positionen räknas upp med 1.

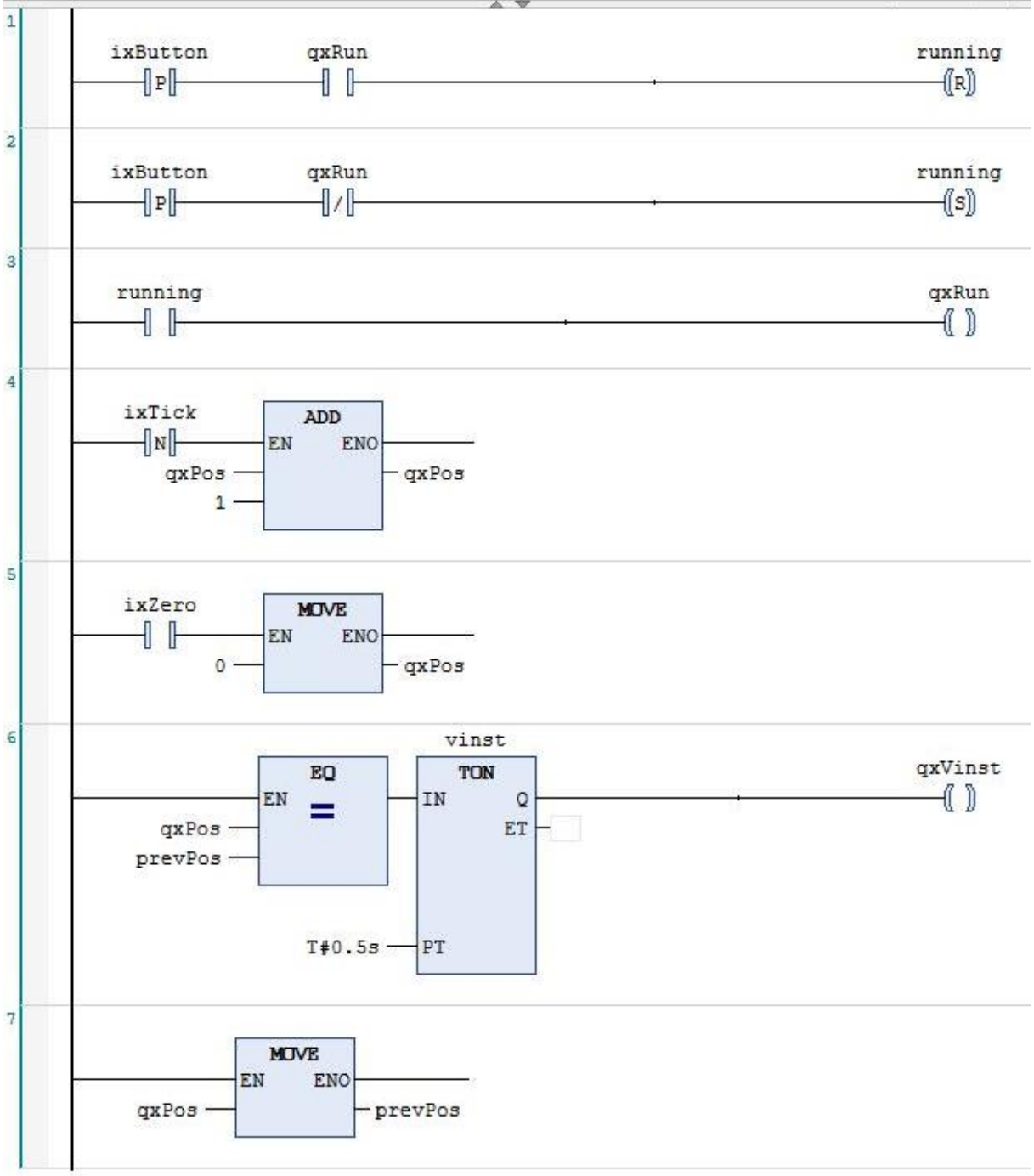
Din uppgift är att implementera så att hjulet kan startas och stoppas med ixButton. Signalen ixButton är sann när knappen är nertryckt. När hjulet skall snurra skall qxRun vara sann. Hjulet kan aldrig snurra åt fel håll. Hjulet skall snurra tills ixButton trycks ner igen. Du skall också implementera räknaren för vilken position hjulet har. När sensorn ixZero är sann är hjulet på position 0 och för varje gång armen passerar en pigg, skall din kod räkna upp.

Det tar en liten stund för hjulet att stanna efter att det stängts av. Så när positionen på hjulet inte har ändrat sig under 0.5s, skall utgången qxVinst sättas, vilket tänder en lampa. Denna släcks när hjulet startas igen.

(10 poäng)

Möjlig lösning uppgift 4

```
    running: BOOL;
    prevPos: INT;
    vinst: TON;
END_VAR
```



Uppgift 5 - Modellering

I denna uppgift skall du modellera ett robotproblem med operationer. I problemet måste en robot flytta tre vita och tre svarta bollar från bord A till bord B. Roboten kan som högst bära två bollar åt gången, och när roboten plockar bollar så tar den alltid en eller två bollar direkt och när den lämnar bollar så lämnar den alltid alla bollar den bär samtidigt.

Syftet med din modell är att använda den i en sökalgoritm för att hitta sekvensen för att flytta alla 6 bollar från A till B. Men för att göra det lite roligare så finns följande begränsningar:

- Det får aldrig finnas fler svarta än vita bollar på respektive bord, förutom om det bara finns svarta bollar.
- Roboten får aldrig röra sig mellan borden utan att bära minst en boll.

Din uppgift är att definiera lämpliga variabler som kan beskriva tillståndet och sedan skapa ett antal operationer med guards och actions som beskriver robotens agerande. Det är viktigt att ditt system aldrig kan bryta mot kraven när operationer exekverar så att det råkar finnas fler svarta än vita bollar på ett av borden eller att roboten kör mellan borden utan en boll. Fundera också på vilka operationer som du faktiskt behöver och hur tillståndet av systemet skall representeras för att lösa problemet utan att det blir en massa operationer eller tillstånd.

Tänk på att syftet med modellen är att klura ut hur bollarna skall flyttas fram och tillbaka för att få över dem all till bord B utan att bryta mot kraven.

Så beskriv dina variabler och operationer med guards och actions.
Svara också på frågan: Hur skulle en möjlig sekvens kunna se ut för att flytta alla bollar? Om det blir många steg i din sekvens, ta med några steg i början och i slutet av lösningen.

Tips: Tänk på att variabler som beskriver tillstånd inte alltid behöver vara boolean.

(10 poäng)

Möjlig lösning uppgift 5:

Eftersom roboten alltid kommer att plocka upp på ena bordet och sedan lämna bollar på det andra, räcker det att operationerna direkt flyttar bollar mellan respektive bord. Antal bollar är alltid samma i hela systemet och de finns antingen på ena eller andra bordet, vilket gör att det räcker med tre variabler för att beskriva hela tillståndet:

v: 0–3. // antal vita bollar på bord A. Om tex v = 1 så är det 1 vit boll på bord A och 2 på bord B.
s: 0–3. // antal svarta bollar på bord A. och så klart indirekt antal på bord B
r: T, F. // Om roboten är vid bord A, True, eller vid bord B, False

Kravet på att det aldrig får vara fler svarta bollar än vita på något av borden kan efter lite funderande uttryckas som:
$v = 3 \vee v = 0 \vee v = s$

Vi har ett antal operationer för att flytta svarta och vita bollar mellan de två borden. Deras guard skall hålla koll på att roboten är vid rätt bord, att det finns bollar att flytta och att inte vi hamnar i ett förbjudet tillstånd efter flytten. Dock kan vi tänka att vi inte befinner oss i ett förbjudet tillstånd när operationen startar.

$SS_{toB}: r \wedge s \geq 2 \wedge v \neq 2 \;/\; s \coloneqq s - 2, r \coloneqq F$ // flytta 2 svarta från A till B

// $r \wedge s \geq 2$ är krav på att roboten är vid A och att det finns svarta bollar att flytta från A.
// $v \neq 2$ garanterar att det alltid är minst lika många vita bollar som svarta på de båda borden.

$SV_{toB}: r \wedge v = s \neq 0 \;/\; s \coloneqq s - 1, v \coloneqq v - 1, r \coloneqq F$ // flytta 1 svart och 1 vit till B

$VV_{toB}: r \wedge v = 3 \wedge s = 1 \;/\; v \coloneqq v - 2, r \coloneqq F$ // flytta 2 vita till B.

$S_{toB}: r \wedge s \geq 1 \wedge (v = 3 \vee v = 0) \;/\; s \coloneqq s - 1, r \coloneqq F$ // flytta 1 svart till B.

$V_{toB}: r \wedge (v = 3 \vee s = 2) \;/\; v \coloneqq v - 1, r \coloneqq F$ // flytta 1 vit till B.

$SS_{toA}: \neg r \wedge s \leq 1 \wedge v \neq 1 \;/\; s \coloneqq s + 2, r \coloneqq T$ // flytta 2 svarta till A

$SV_{toA}: \neg r \wedge v = s \neq 3 \;/\; s \coloneqq s + 1, v \coloneqq v + 1, r \coloneqq T$ // flytta 1 svart och 1 vit till A

$VV_{toA}: \neg r \wedge v = 0 \wedge s = 2 \;/\; v \coloneqq v + 2, r \coloneqq T$ // flytta 2 vita till A.

$S_{toA}: \neg r \wedge s \leq 2 \wedge (v = 3 \vee v = 0) \;/\; s \coloneqq s + 1, r \coloneqq T$ // flytta 1 svart till A.

$V_{toA}: \neg r \wedge (v = 0 \vee s = 1) \;/\; v \coloneqq v + 1, r \coloneqq T$ // flytta 1 vit till A.

Möjlig sekvens: $SV_{toB}, V_{toA}, SS_{toB}, S_{toA}, VV_{toB}, SV_{toA}, VV_{toB}, S_{toA}, SS_{toB}, S_{toA}, SS_{toB}$

Problemet bygger på det kända problemet
https://en.wikipedia.org/wiki/Missionaries_and_cannibals_problem

En annan variant är att ha olika variabler för borden:

v_A, s_A, v_B, s_B, r

$v_A \geq s_A, v_B \geq s_B$

$SS_{toB}: r \wedge s_A \geq 2 \wedge v_B \geq s_B + 2 \; / \; s_A \coloneqq s_A - 2, s_B = s_B - 2, r \coloneqq F$

$SV_{toB}: r \wedge v_A - 1 \geq s_A - 1 \wedge v_B + 1 \geq s_B + 1 \; /$

$$s_A \coloneqq s_A - 1, v_A \coloneqq v_A - 1, s_B + 1, v_B \coloneqq v_B + 1 \; r \coloneqq F$$

Men guarden kan ju förenklas:

$r \wedge v_A - 1 \geq s_A - 1 \wedge v_B + 1 \geq s_B + 1$ => $r \wedge v_A \geq s_A \wedge v_B \geq s_B$ =>

Vi har också $3 - v_A = v_B, \; 3 - s_A = s_B$ så => $r \wedge v_A = s_A \neq 0$

Osv. Men som ni ser så blir det inte enklare med fler variabler.