Industriautomation

Tentamen SSY 065, lördag 14/4, 08:30-12:30, M

Examinator: Martin Fabian, (772) 3716

Tider för lärarens närvaro: 09:30, 11:30

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 25 poäng. För betygen tre, fyra och fem krävs 10, 15 resp 20 poäng.

Lösningar anslås första vardagen efter tentatillfället på kursens hemsida i Studieportalen.

Tentamensresultaten anslås senast 27/4 på institutionens anslagstavla, kl. 12:30.

Granskning av rättningen får ske 27/4 kl. 12:30 – 13:30 på institutionen.

OBS. Inga hjälpmedel är tillåtna.

Uppgift 1. Olika aktiviteter

I kursen gicks igenom tre olika aktiviteter, flödessimulering, robotsimulering och PLC-programmering.

a. Vad gör man i respektive aktivitet? (1p)

b. Vilken information skapas och används inom respektive aktivitet? (1p)

c. Hur överförs idag denna information mellan aktiviteterna och vad kan det medföra? (1p)

d. Vad är STEP för något och varför vill man använda det? (2p)

Uppgift 2. Flödessimulering

a. Varför utför man flödessimulering? (1p)

b. Varför är det många gånger tidsödande för ett företag att utföra en flödessimuleringsanalys? (1p)

c. Vilka positiva effekter kommer ofta av att man utför ett flödessimuleringsprojekt? (1p)

Uppgift 3. Robotsimulering

Vad är *absolutnoggrannhet* respektive *repeternoggrannhet*? I vilket sammanhang är var och en av dessa viktiga? (2p)

Svar finns i robottext sid 433. (och nedan)

Det största problemet med grafiska CAR-system är att roboten inte intar de OLP-genererade positionerna med erforderlig noggrannhet. Det finns två olika sätt att definiera robotens noggrannhet. Dessa är absolutnoggrannhet, robotens förmåga att inta ett specifikt *xyz*-värde, och repeternoggrannhet, robotens förmåga att repetitivt inta en given punkt (se figur 12.3). Eftersom dagens industrirobotar vanligtvis har mycket

bättre repeternoggrannhet än absolutnoggrannhet är det den sistnämnda som ligger till grund för problemet med noggrannheten i de OLP-genererade positionerna. Typiska repeternoggrannheter är omkring 0.1 mm medan absolutnoggrannheter är i storleksordningen 1 – 3 mm.

Figur 12.3 Beskrivning av repeter- och absolutnoggrannhet.

Detta medför inget problem vid användandet av teach-in metoden, då man med hjälp av det mänskliga ögat kan positionera roboten i rätt läge. Här är det alltså repeternoggrannheten som är mycket viktig för att roboten gång efter gång ska inta den önskade positionen.

Däremot när något off-line system används för att numeriskt generera robotens positioner blir absolutnoggrannheten väldigt viktig. Dessa system kräver att roboten intar en given punkt med tillfredställd noggrannhet för att kunna användas i den utsträckning som det är tänkt. Går inte detta måste ändå roboten tas i anspråk för att omprogrammera alla positioner och off-line programmeringen förlorar sitt syfte.

Uppgift 4. Kommunikation

Förklara OSI modellen för datorkommunikation. (3p)

See the lecture notes.

Uppgift 5. PLC

a) Beskriv de huvudsakliga delarna av en modern PLC. (2p)

I/O-enhet (med in- och utgångsmoduler, plus buffertminne), processor (CPU), programminne (RAM), operativsystemminne (ROM).

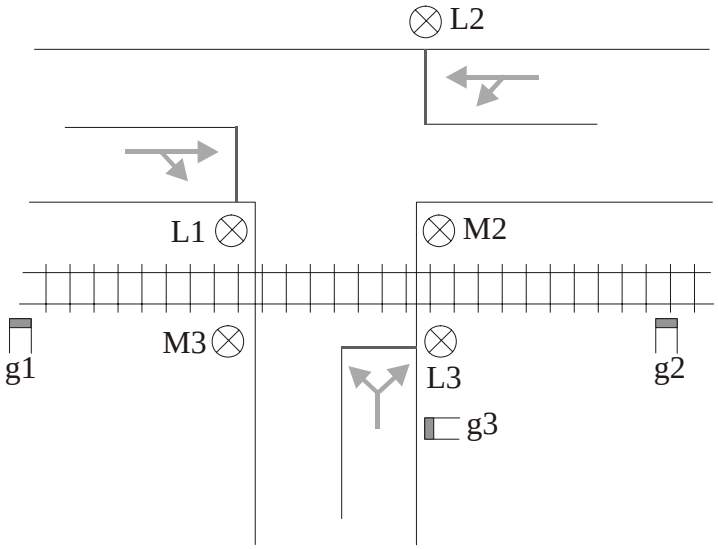
b) Förklara den principiella idén bakom *scan-cykel* konceptet som (så gott som) alla moderna PLCer använder? (2p)

Alla insignalers värde kopieras till buffertminnet, hela programmet exekveras med insignalernas sparade värde och utsignalernas nya värde sparas i utsignalsbufferten. Därefter kopieras utsignalbuffertens innehåll till de egentliga utsignalerna. Denna läs-exekvera-skriv-cykel upprepas om och om igen så länge PLC:n är påslagen, och är det som kallas för *scan-cykel*. Detta görs för att simulera det parallella beteende som reläkopplingar och fast trådad logik uppvisar. Från en utomstående betraktare verkar det som om alla utsignaler ändrar sina värden beroende på insignalerna (och interna minnen), samtidigt.

c) Get exempel på problem som kan inträffa om scan-cykeln inte är kort i förhållande till den styrda processens tidskonstanter. (2p)

Under exekvera-skriv-delen av scan-cykeln registrerar PLC:n inga insignalsändringar. Det kan medföra att de genererade utsignalerna är felaktiga i förhållande till det aktuella tillståndet (vilket alltså då är skiljt från det tillstånd PLC:n registrerat) hos den styrda processen.

Uppgift 6. SFC programmering



En T-korsning med ett korsande spårvagnsspår visas ovan. I korsningen finns fem trafikljus, där L1, L2 och L3 styr biltrafiken, medan M2 och M3 styr spårvagnstrafiken. Bilar kommande från höger kan fortsätta rakt fram eller svänga (vänster) söderut. Likaså kan bilar från vänster fortsätta rakt fram eller svänga (höger) söderut. Fordon från söder kan svänga antingen vänster eller höger. Spårvagnen kan komma från antingen vänster eller höger, och givarna g1 och g2 ger signal (logisk etta) när det finns vagnar vid dem. Givaren g3 känner av när en bil finns i närheten av L3. I övrigt ger givarna logiska nollor.

I normalfallet visar L1 och L2 grönt ljus, medan L3, M2 och M3 visar rött. Om g3 ger signal att det kommer en bil söderifrån, ska L1 och L2 växla till rött ljus. Därefter ska L3 växla till grönt och sedan visa grönt i minst 30 sekunder och max i 60 sekunder. Grönt ljus ska visas i 10 sekunder efter det att inga bilar längre kommer söderifrån, vilket indikeras av g3. Notera dock att det maximalt ska vara grönt i 60 sekunder. Därefter visar L1 och L2 grönt i minst 60 sekunder. Notera också att vid växling ska alla ljusen visa rött samtidigt under en kort stund.

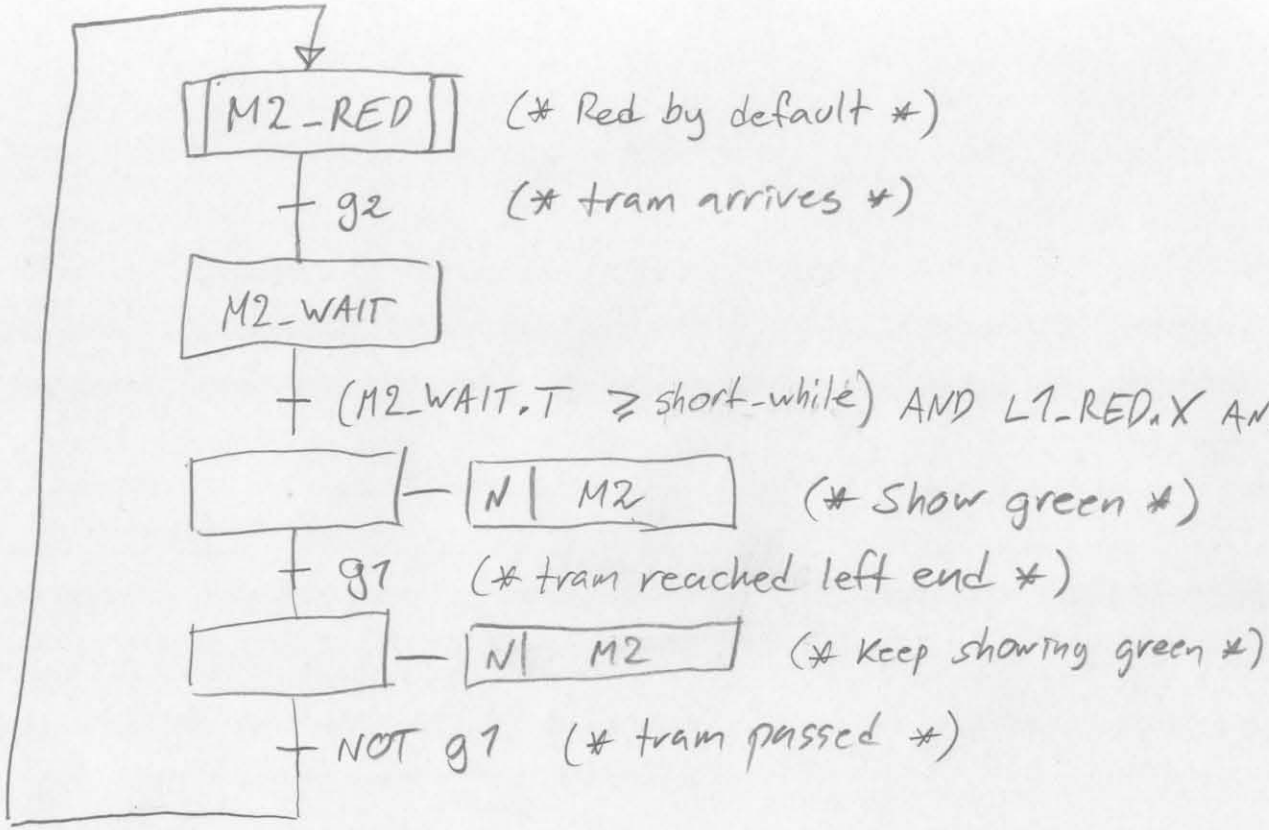
Spårvagnen komplicerar till det ytterligare eftersom den ska ha prioritet. När spårvagnen kommer ska L1, L2 och L3 alla växla till rött och M2 eller M3, beroende på varifrån vagnen kommer, ska växla till grönt. Detta tillstånd ska råda tills alla vagnarna har passerat givaren på motsatta sidan (i vagnens färdriktning). Därefter ska den normala ljusväxlingen ta vid.

Avstånden och hastigheterna är sådana att när spårvagnen syns vid g1 eller g2 så hinner ljusen slås om. Notera dock att spårvagnens trafikljus och bilarnas trafikljus *aldrig* får visa grönt samtidigt. Notera också att du inget vet om längden på vagnen, dvs huruvida både g1 och g2 nånsin ger signal samtidigt.

a) Ge fem st, ett för varje ljus, parallellt exekverande funktionsdiagram (IEC 61131, SFC) som styr trafikljusen. Ange vilken exekveringsmodell (immediate/deferred transit/action) du antar. (4p)

b) Om spårvagnarna kommer för ofta kan det hända att bilar söderifrån blir stående oacceptabelt länge. Utöka din lösning i a) till att vara sådan att eventuella bilar söderifrån garanterat släpps ut på vägen mellan varannan spårvagn. (2p)

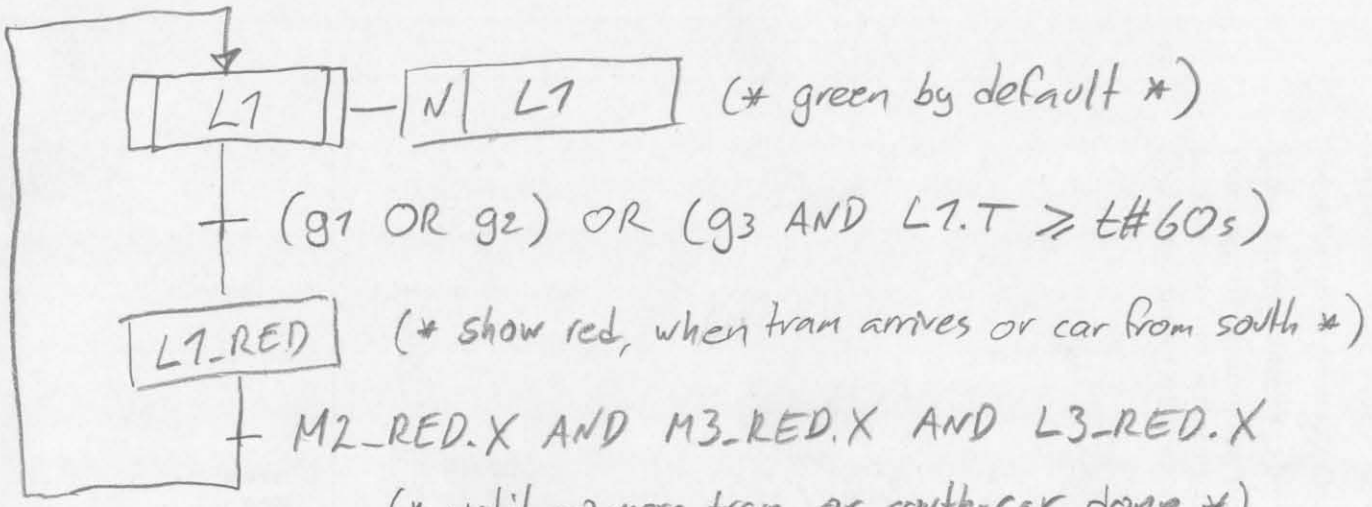
Du kan bortse från att ljusen visar gult och växla direkt från rött till grönt och tvärtom.



Note that "NOT g1" is not enough here, since the tram can be shorter than the distance between g1 and g2.

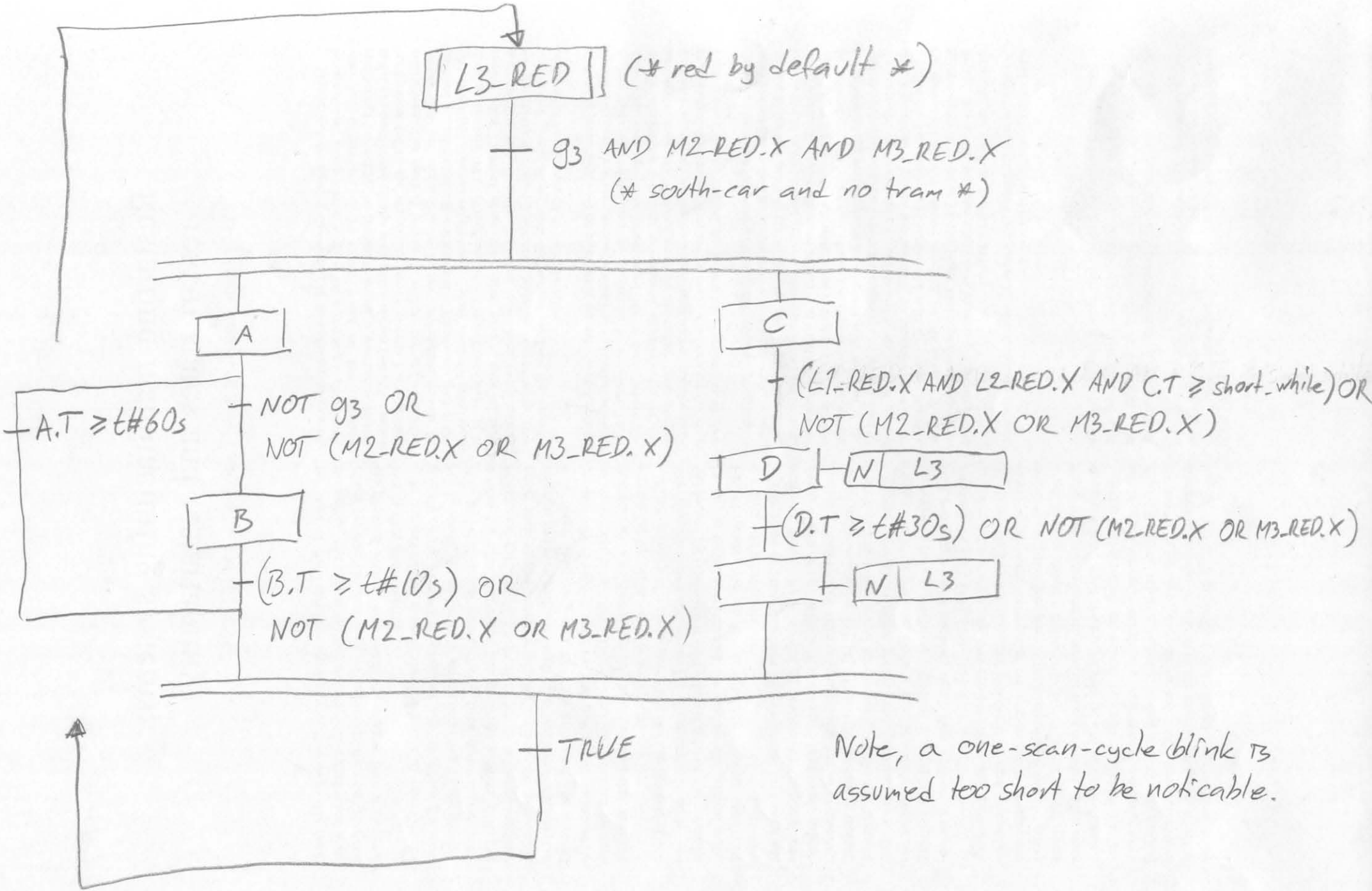
We assume that the trams do not arrive too frequently from the same direction.

The SFC for M3 is essentially the same as for M2, with obvious renaming

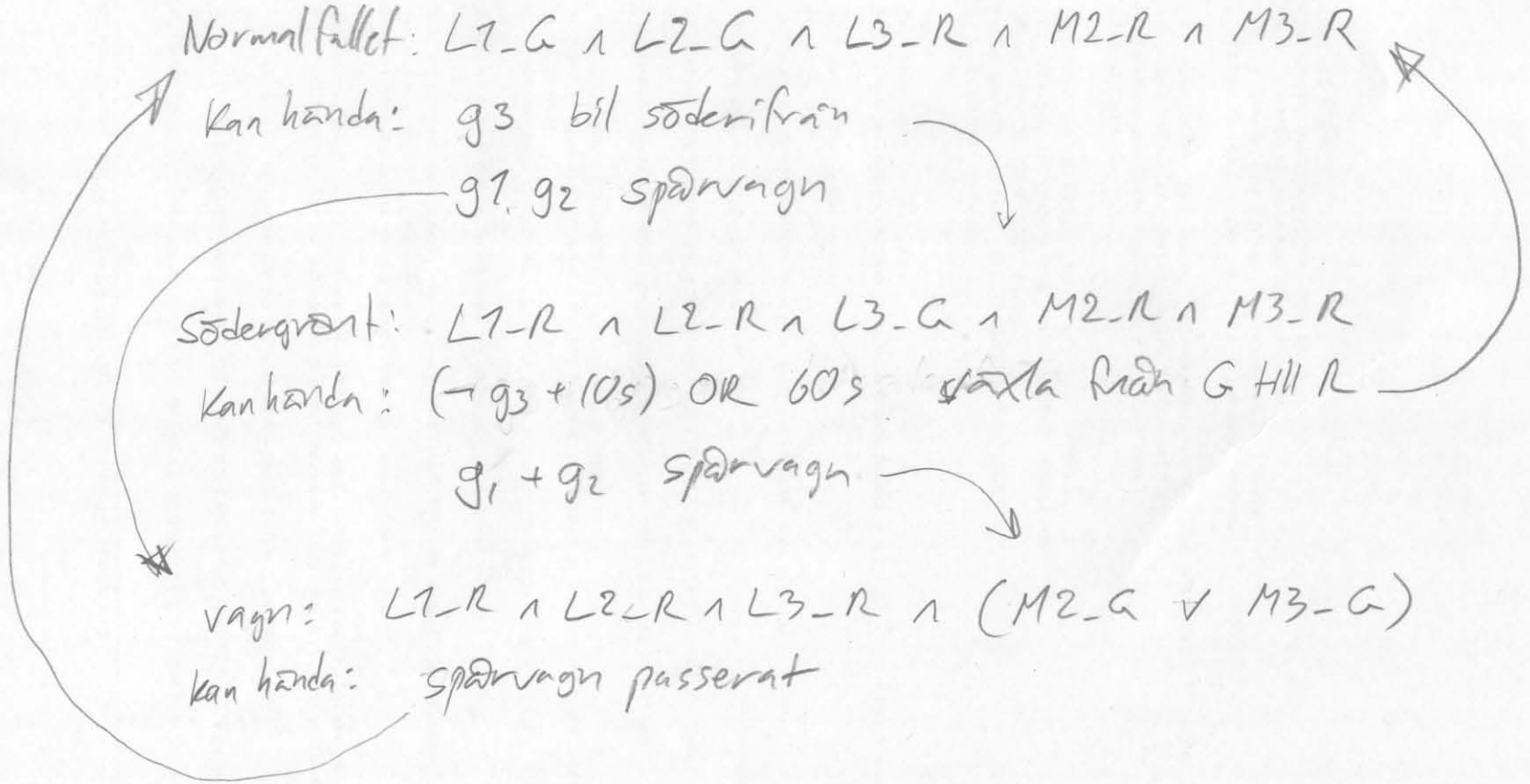


Note that "NOT g_1 AND NOT g_2" is not ok here, since the tram can be shorter than the distance between g_1 and g_2.

The SFC for L2 is essentially the same as for L1, with obvious renaming.


L3_RED
(* red by default *)
g3 AND M2_RED.X AND M3_RED.X
(* south-car and no tram *)
A
A.T ≥ t#60s
NOT g3 OR
NOT (M2_RED.X OR M3_RED.X)
B
(B.T ≥ t#10s) OR
NOT (M2_RED.X OR M3_RED.X)
C
(L1_RED.X AND L2_RED.X AND C.T ≥ short_while) OR
NOT (M2_RED.X OR M3_RED.X)
D
N L3
(D.T ≥ t#30s) OR NOT (M2_RED.X OR M3_RED.X)
N L3
TRUE
Note, a one-scan-cycle blink is
assumed too short to be noticable.

Olika "lägen", alla måste funka



Antaganden: Immediate transit, immediate actions