Programmering i svensk skolmatematik

Cecilia Kilhamn[1], Lennart Rolandsson[2] och Kajsa Bråting[2]

[1] Göteborgs Universitet
[2] Uppsala Universitet

När programmering skulle inkorporeras i skolans arbete valde Sverige i sin läroplansrevidering 2017 att skriva in det i matematikämnet, med stark koppling till algebra. Samtliga matematiklärare ställdes då inför utmaningen att undervisa i programmering. Vi undersöker här resultatet av 32 lärargruppers gemensamma arbete med att planera och genomföra lektioner i programmering i matematik i grundskolan. För att få insikt i hur lärare tolkar uppdraget och transponerar läroplananes beskrivning av programmering till klassrumspraktiken analyserar vi det matematiska innehållet i dessa lektioner samt vilken syn på relationen mellan matematik och programmering som framträder i lärarnas beskrivning av syfte, lärandemål, aktivitet och reflektion. Vi finner att programmeringsaktiviteter i 1/3 av lektionerna inte kopplas till något traditionellt matematiskt innehåll. I övriga lektioner är det främst aritmetik eller geometri som utgör det matematiska innehållet. Få explicita kopplingar görs till algebra förutom till begreppet variabler, men då är det främst variabler inom programmering som avses. I materialet framträder fyra olika relationer mellan matematik och programmering: 1) enbart programmering; 2) matematik som en kontext för programmering; 3) programmering som ett verktyg för att effektivisera beräkningar; 4) programmering som ett verktyg för att utforska matematik. Resultaten diskuteras i relation till matematikämnets syfte och innehåll i den svenska läroplanen.

Nyckelord: programmering, matematik, algebra, grundskolan

ARTIKEL

LUMAT General Issue
Vol 9 No 1 (2021), 283–312

Received 11 december 2020
Accepted 25 mars 2021
Publicerad 6 maj 2021

Sidor: 30
Referenser: 41

Kontakt:
cecilia.kilhamn@ped.gu.se

https://doi.org/10.31129/LUMAT.9.2.1457

Programming in Swedish school mathematics

When incorporating programming in the school curricula, Sweden decided to integrate it with mathematics, and specifically within the core content of algebra. As a result, all mathematics teachers at all levels were faced with the challenge of teaching programming. In this study we analyse documentation from 32 lesson studies where groups of teachers have planned, conducted, and revised lessons on programming within the school subject mathematics. To gain insight into how the teachers interpret the new task and thus contribute to the transposition of knowledge from the curriculum level to the classroom level, we analyse the mathematical content in these lessons and the relations between mathematics and programming that emerge in the way the teachers describe the aim, the activites and the learning outcomes of the lessons. We find that 1/3 of the lessons do not connect to any traditional mathematics content, and the rest of the lessons mostly focus on arithmetic or geometry. Few explicit connections are made to algebra, except for the variable concept, but when variables are treated the focus is on variables in the programming sense rather than algebra. Four different relationships between mathematics and programming emerge in the data: 1) programming without connecting to mathematics; 2) mathematics as a context for programming, 3) programming as a tool for efficient calculations; 4) programming as a tool for exploring mathematics. The results are discussed in relation to the transposition of knowledge of mathematics as a school subject.

Keywords: programming, mathematics, school algebra, transposition of knowledge

1 Introduktion

Under de senaste fem åren har flera länder infört programmering i skolundervisningen. I debatten framhålls att programmering numera ses som en nödvändig kunskap för alla medborgare (Mannila m fl., 2014; Nouri m fl., 2020). Programmering brukar även lyftas fram som ett pedagogiskt verktyg för utveckling av elevers datalogiska tänkande.[1] Implementeringen av programmering i skolans läroplaner har gjorts på olika sätt i olika länder, se t ex Australien (Falkner m fl., 2014), England (Brown m fl., 2014), Danmark (Misfeldt m fl., 2020), Finland (Manilla m fl., 2014), Sverige (Kilhamn & Bråting, 2019) och USA (Fisher, 2016). I England och Danmark har programmering blivit en del av ett nytt ämne, medan exempelvis Finland och Sverige har fört in programmering i redan befintliga ämnen, för Sveriges del i matematik och teknik. Något som gör Sverige unikt i sammanhanget är att de grundläggande principerna för programmering skrivits in i kursplanen som en del av det centrala innehållet i algebra (Kilhamn & Bråting, 2019; Skolverket, 2017).

Den forskning som finns kring programmering i matematik har tidigare främst handlat om matematikämnet på gymnasiet eller universitetet (Grover & Pea, 2013). Den pågående världsvida implementeringen av programmering i skolans lägre årskurser har lett till en efterfrågan på kunskap om den roll programmering har eller skulle kunna ha i grundskolans matematikundervisning (Hickmott m fl., 2018; Nouri m fl., 2020). Vi önskar bidra till detta fält genom att analysera dokumentation från lärargrupper som har planerat, genomfört och utvärderat lektioner i programmering i matematik i grundskolan. Lektionerna har arbetats fram inom ramen för så kallade lesson studies, en form av kollegialt fortbildnings- och utvecklingsarbete. Studien är en del av ett mer omfattande forskningsprojekt om den pågående implementeringen av programmering i grundskolans matematik som tar avstamp i Chevallards (2006) teoretiska ramverk om hur kunskap transponeras mellan olika instanser i skolans verksamhet (Bråting m fl., 2021).

Syftet med den studie som rapporteras här är att belysa hur lektioner i programmering som sker inom ramen för matematikämnet interagerar med den traditionella skolmatematiken i det inledande skedet av implementeringen av programmering som ett nytt kunskapsinnehåll i grundskolans matematikämne.

[1] Datalogiskt tänkande, kallas internationellt för ”computational thinking”. Begreppet kan förklaras som en problemlösningsprocess för att beskriva, analysera och lösa problem så att datorer kan hjälpa till, eller som Aho (2012) definierar det ”de tankeprocesser som är involverade i att formulera problem så att dess lösningar går att representera som stegvisa instruktioner eller algoritmer” (p. 832).

Samtliga lektioner i vårt datamaterial har ett givet programmeringsinnehåll helt i linje med läroplanens formulering (se avsnitt 3.1). I analysen ställer vi två frågor:

1. Vilket matematikinnehåll väljer lärare till sina programmeringslektioner i matematik i grundskolan?
2. Hur ser relationen mellan matematik och programmering ut i dessa lektioner?

Forskningsfrågorna fokuserar på de val lärarna gör och den tolkning av relationen mellan programmering och matematik som framkommer av de syften och lärandemål de formulerar relativt programmeringslektionerna.

2 Lärares transposition av kunskap

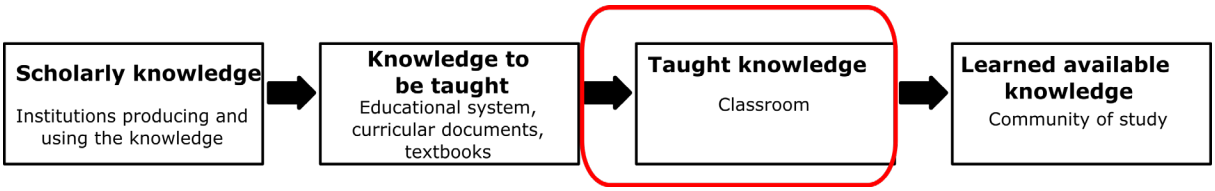
Eftersom vi är intresserade av den kunskap som formuleras av lärare när ett nytt innehåll införs av institutioner utanför skolan är Chevallards (2006) teori om didaktisk transposition av kunskap en lämplig inramning. I avsnitt 2.1 beskrivs den teoretiska inramningen och i 2.2 redogör vi för tidigare forskning om hur svenska lärare ser på programmering i matematik.

2.1 Didaktisk transposition från avsedd till undervisad kunskap

Figur 1 illustrerar hur kunskap formas och förändras när den införlivas i skolans verksamhet. I den första transpositionen behöver den akademiska kunskapen *(scholarly knowledge)* brytas ned i mindre delar och anpassas för att göras undervisningsbar. Resultatet blir vad som kallas *avsedd kunskap (knowledge to be taught)* (Bosch & Gascon, 2006; Kang & Kilpatrick, 1992). Den kunskapen legitimeras i samband med att olika aktörer beslutar vad som ska ingå i skolans läroplaner och i vilket syfte. Exempelvis tas beslut om var i läroplanen i relation till andra ämnen som programmering ska integreras, vilka aktiviteter och begrepp som ska ingå, samt i vilken ordning och i vilka årskurser olika aspekter ska tas upp. I Sverige skedde detta i samband med revideringen av läroplanen 2017, då programmering infördes i kursplanerna för matematik och teknik. Den beskrivning av programmering som nu finns i läroplanen har därmed, enligt Chevallards teori, delvis förändrats i relation till det kunskapsinnehåll som utgör programmering för en verksam programmerare.

Nästa steg i transpositionsprocessen sker när läroplanen tolkas av lärare, som genom sina beslut bidrar till att forma det som faktiskt görs i klassrummet, och som kommer att utgöra så kallad *undervisad kunskap* (*taught knowledge*). Den slutliga

instansen i transpositionsprocessen utgörs av den kunskap som elever får möjlighet att faktiskt lära sig (*learned available knowledge*). I viss mån sker en återkoppling till tidigare nivåer så det som sker i klassrummet kan komma att påverka och förändra det läraren gör osv. För att beskriva kunskap på de olika platserna i utbildningssystemet har Chevallard utvecklat en praxeologi som beskriver dels val av uppgifter och tekniker; kunskapens praxis, dels underliggande argument och teorier som rättfärdigar dessa val; kunskapens logos.



Figur 1. Chevallards teori om hur kunskap förändras mellan olika instanser, från Bosch & Gascon (2006).

Den studie som presenteras här zoomar in på den didaktiska transpositionen från *avsedd kunskap* till *undervisad kunskap* (markerat med rött i figur 1) i det skede av historien där programmering är ett helt nytt inslag i matematikundervisningen.

2.2 Svenska lärares syn på programmering i skolmatematiken

Tidigare studier om programmering i grundskolans matematik fokuserar ofta elevernas lärande snarare än lärares undervisning. Hittills har endast några få studier genomförts kring svenska lärares initiala reaktioner på införandet av programmering i skolan. Resultatet av dessa visar att lärare i allmänhet har svårt att se ett tydligt samband mellan matematik och programmering men har samtidigt en positiv inställning till inslaget av programmering i matematiken (Kilhamn m fl., 2021; Misfeldt m fl., 2019; Mozelius m fl., 2019). Pörn m fl. (2021) har genom web-baserade enkäter undersökt finlandssvenska grundskollärares spontana syn på kopplingen mellan matematik och programmering. De fann att lärarna i första hand relaterade programmering till att skriva, ge och följa instruktioner samt till olika aspekter av logiskt tänkande. Däremot var det få lärare som kopplade samman programmering med något specifikt område inom matematiken, som exempelvis algebra eller geometri. Vidare visar resultatet av olika intervjustudier att lärare i stor utsträckning relaterar programmering till allmänna förmågor rörande samarbete och kommunikation (Nouri m fl., 2020), samt ser programmering som ett pedagogiskt verktyg som kan öka elevers engagemang på matematiklektionen (Kilhamn m fl.,

2021). Utöver frågan kring vad programmering är och vilken koppling den har till matematiken påpekas att det snabba införandet av programmering har skapat stress, framförallt hos lärare som saknar erfarenhet av programmering (Mozelius m fl., 2019). I den här artikeln vill vi bidra med ytterligare kunskap om lärares syn på programmering i matematik genom att analysera hur lärare som planerat och genomfört programmeringslektioner i matematikämnet beskriver lektionernas syfte och lärandemål.

3 Programmering i svensk skolmatematik

För att förstå den transposition av kunskap som äger rum när programmering inkluderas i matematikämnet ges först en historisk bakgrund och en redogörelse för den avsedda kunskapen så som den är formulerad i nuvarande svenska läroplan.

3.1 Historisk bakgrund till programmering i svensk skolmatematik

Första gången datorer dyker upp i den svenska grundskolans styrdokument är 1969 där ”Orientering om datamaskiner” skrevs in under matematikämnets huvudmoment för högstadiet (Skolöverstyrelsen, 1967, s. 137). Undervisningen handlade då mer om datorer än om programmering, men på olika håll, främst på gymnasiet, gjordes innovativa försök att undervisa programmering. Det var en tid då programmerare använde högnivåspråk som t.ex. Fortran, Basic och Pascal, vilka krävde dyra och kraftfulla datorer. Därför var det vanligare att tillämpa lågnivåspråk som Assembly i skolan och lärare kunde använda bordskalkylatorer som räknehjälpmedel (Rolandsson, 2011). I den typen av datorhybrid matades koden in utan spegling (bildskärmar var ovanliga) och utskriften fanns att hämta på speciella skrivmaskiner (teletypers). Ibland användes modem för att ringa upp en stordator på kvällstid och körresultatet hämtades dagen efter. För en lärare som önskade undervisa programmering fanns det därför flera faktorer som måste hanteras: ekonomi, tid till förberedelser, hårdvarans komplexitet samt programmeringsspråkets syntax och långsamma återkoppling. Idag är förutsättningarna för programmering i skolan betydligt bättre, med snabba datorer och programmeringsspråk som är enklare att hantera. En elev av idag kan relativt snabbt skapa, köra och korrigera sin kod tack vare visuell återkoppling från en bildskärm.

En ny läroplan för grundskolan kom 1980, i vilken datalära som ett nytt innehållsområde skulle undervisas inom ramen för ämnena samhällskunskap och matematik (Skolöverstyrelsen, 1980). Alla skolor hade inte tillgång till datorer, och

bland de grundskolor som hade datorer fanns i snitt endast fem datorer per skola (Söderlund, 2000). Staten gick därför in med ett stimulanspaket, och en detaljerad studieplan för datalära (Skolöverstyrelsen, 1984) där det tydligt framgick att programmering inte skulle ta för mycket tid i anspråk och att fokus skulle vara matematik och inte programmering för att inte riskera att matematikproblem övergick till programmeringsproblem:

> Ytterligare färdighet i datoranvändning får eleverna i ämnet matematik, där problemlösning med datorer ingår som ett moment i undervisningen. I dessa övningar bör färdiga program användas där så är möjligt, och först som en påbyggnad i undervisningen kan problem lösas med programmering (Skolöverstyrelsen, 1984, s. 10).

Datalära fick en trög start, med mycket fokus på hårdvara och lite fokus på undervisning och lärares fortbildning. I väntan på datorer bedrevs undervisningen mestadels med penna och papper, vilket kan verka kontraproduktivt då programmering inte skulle ta för mycket tid i anspråk. Riis (1987) beskrev i en rapport att lärare med egen erfarenhet av programmering i undervisningen också var kritiska till dess innehåll. I Sverige utvecklades under denna tid en egen skoldator, Compis, med grafiska möjligheter och det egna programmeringsspråket Comal (en version av Basic) för problemlösning i matematik.

I samband med implementeringen av 1994 års läroplan försvann datalära och programmering ur kursplanen för grundskolan (Utbildningsdepartementet, 1994). Datorer omnämndes istället i olika ämnen med fokus på datorns möjligheter för beräkningar, datahantering, ordbehandling och informationssökning. År 2017 återinfördes programmering i grundskolan, nu som en aspekt av vad som kallas digital kompetens (Skolverket, 2017).

I slutet av 1960-talet utvecklades programmeringsspråket Logo (Papert, 1980), som fick stor internationell spridning i skolundervisning, men som inte på allvar slog igenom i Sverige (Hedrén, 1990). Programmeringsspråket innehöll bland annat en grafik där en sköldpadda styrdes på skärmen med enkla kommandon. En grundläggande filosofi i Logo var att användaren skulle uppleva matematik inom ramarna för programmets ”mikrovärldar”, och lära sig matematik genom att reflektera över det som skapas i programmet.

De höga förväntningar som fanns på Logo som matematiskt problemlösningsverktyg och medel för att utforska matematik kom delvis på skam i den internationella forskning som bedrevs under 1980-talet (Noss & Hoyles, 1996).

Kritiken handlade ofta om att programmeringen ansågs tidskrävande och svår, och att lärare behövde mer didaktisk kunskap om programmering för att undervisningen skulle utveckla elevers matematiska förmågor. Enligt Hedrén (1990) framhöll vissa forskare att matematikens semantiska innehåll, speciellt i geometri, blev tydligare genom programmeringen, medan andra kritiserade den bakomliggande filosofin som ett önsketänkande och nyttan med Logo som minimal, eftersom språket adresserade ett smalt område i matematik.

Papert såg tidigt att elever kunde skapa mikrovärldar med grafiska figurer, så kallade ”sprites”. Denna idé plockades upp i Scratch, en programmeringsmiljö med grafiska block med en mängd olika funktioner för att hantera färg, form, musik, bild och animeringar som utvecklats under 2000-talet (Resnick m fl., 2009). Scratch har på senare tid fått stor spridning både internationellt och inom den svenska skolan. Programmet skiljer sig från Logo genom att det inte i första hand är utvecklat för att stödja matematiken, utan för att stimulera unga att programmera interaktiva berättelser, animeringar och spel samt att kunna dela dessa med varandra över internet (Resnick m fl., 2009). Ett engelskt forskningsprojekt har visat att det går att undervisa om programmering i Scratch och samtidigt lära sig en del matematik, men att det kräver mycket arbete för att designa aktiviteter som innehåller viktiga matematiska idéer och som bryggar över mellan matematik och programmering (Benton m fl., 2017).

3.2 Programmering i nuvarande svenska läroplan

Den senaste revideringen av svensk läroplan trädde i kraft 2018, då 2011 års läroplan kompletterades med skrivningar om elevers utveckling av digital kompetens (Skolverket, 2017), varvid programmering inkluderades i ämnena matematik och teknik. I kursplanen för matematik står programmering omnämnt i ämnets syftesbeskrivning:

> Genom undervisningen ska eleverna ges förutsättningar att utveckla förtrogenhet med grundläggande matematiska begrepp och metoder och deras användbarhet. *Vidare ska eleverna genom undervisningen ges möjligheter att utveckla kunskaper i att använda digitala verktyg och programmering för att kunna undersöka problemställningar och matematiska begrepp, göra beräkningar och för att presentera och tolka data.*
>
> (Skolverket, 2017, s. 56, vår kursivering)

Vidare finns programmering inskrivet som en del av det centrala innehållet i *algebra* uppdelat efter stadierna. För årskurs 1-3:

> Hur entydiga stegvisa instruktioner kan konstrueras, beskrivas och följas som grund för programmering. Symbolers användning vid stegvisa instruktioner.

För årskurs 4-6 och 7-9:

> Hur algoritmer kan skapas och användas vid programmering.

I årskurs 4-6 ska programmering ske i *visuella miljöer* och i årskurs 7-9 i *olika programmeringsmiljöer*. För årskurs 7-9 finns programmering även omnämnt i det centrala innehållsområdet *problemlösning*.

4 Metod

För att besvara forskningsfrågorna har vi samlat skriftligt material från lärare som planerat och genomfört programmeringslektioner i matematikämnet. En systematisk sammanställning och kvalitativ innehållsanalys (Bryman, 2012) av datamaterialet har genomförts. Vi börjar med att redogöra för datamaterialet och beskriver därefter analysmetoden med hjälp av exempel från materialet.

4.1 Empiriskt material

Studien har utförts med empiri från ett nationellt utvecklings- och forskningsprojekt för grundskollärare som undervisar programmering. Skolhuvudmän från fem olika kommuner utspridda i landet anslöt sig till projektet, som kom att involvera ca 135 lärare från 15 olika skolor under tre läsår 2017–2020 (se närmare Jahnke, 2020). Lärarna genomförde så kallade lesson studies (LS) där de i grupper om två till fem lärare tillsammans planerade, genomförde, utvärderade och reviderade en lektion (Takahashi & Yoshida, 2004). De formulerade själva lektionens syfte och lärandemål och hade fria händer att välja programmeringsmiljö. All empiri bygger på den dokumentation som respektive lärargrupp skapat i projektet. I den här studien fokuserar vi på de LS som genomfördes inom ramen för matematikämnet, vilket uppgick till 32 stycken. Urvalet av lärare representerar väl grundskolans samtliga årskurser med 12 LS i årskurs F-3, 9 LS i årskurs 4-6 och 11 LS i årskurs 7-9.

Varje lärargrupp genomförde en LS, där de i två till tre cykler arbetade sig igenom olika steg på vägen mot en gemensam lektionsplanering. Innan lärarna påbörjade arbetet med att utveckla lektionsplaneringar deltog de i en mindre utbildning om LS-metodens olika steg och vikten av dokumentation. Här följer en komprimerad version av den information som lärarna fick i början av projektet.

- *Inventering*: Lärargruppen avgör elevernas förkunskaper och avgränsar ett eller flera lärandemål och kunskapsinnehåll lämpligt för elevgruppens ålder.
- *Planering*: Lärargruppen planerar gemensamt en lektion om programmering.
- *Första lektionen*: En lärare genomför undervisningen i en elevgrupp, medan övriga lärare i gruppen observerar och samlar information.
- *Första reflektionen*: Gjorda observationer delas i gruppen för diskussion om resultat av lektionen. Lektionsplanen revideras.
- De två sista stegen upprepas med nya elevgrupper (och eventuellt en annan undervisande lärare) tills lärarna känner sig nöjda med lektionsplanen.

Lärargruppen fick även en mall för dokumentation av sin arbetsprocess där de på egen hand skulle formulera lektionens syfte och lärandemål. I mallen förväntades lärarna även dokumentera didaktiska metoder och på vilket sätt lektionens innehåll relaterar till kursplanen samt utvärdera lektionens resultat. Omfattningen och detaljnivån på dokumentationen varierade mellan lärargrupperna, där somliga endast bidrog med en sammanfattande powerpoint-presentation enligt mallen, medan andra även lämnade planeringsunderlag och exempel på programkod.

De samlade skriftliga dokumentationerna av planering, genomförande och utvärdering av lektioner från dessa 32 LS utgör således det empiriska materialet för den innevarande studien, där varje LS genererat en lektion som under processens gång utprövats i flera klasser. Dokumentationen kring varje lektion har givits ett unikt namn bestående av två bokstäver följt av en siffra som anger årskursen där lektionen utprövades, exempelvis SJ_1.

4.2 Analysmetod

En sammanställning av de 32 dokumenterade lektionerna gjordes med fokus på det innehåll som på olika sätt behandlade matematiklärande. I den här studien definierar vi syfte och lärandemål relaterat till programmering och datalogiskt tänkande och särskiljer det från matematik, eftersom vi är intresserade av relationen mellan dessa två kunskapsfält. Vi använder således termen *matematik* i meningen centralt innehåll i skolmatematiken så som det beskrivits i den svenska läroplanen före införandet av programmering (Skolverket, 2011). Lärandemål relaterat till sociala och praktiska dimensioner av lektionen har analyserats separat (Jahnke, 2020; Sjöberg m fl., 2019; Zhang m fl., 2020). Följande delar av dokumentationen utgjorde underlaget för analysen i den här artikeln:

- matematikinnehållet i det syfte och lärandemål som lärarna formulerat
- det explicita matematikinnehållet i respektive aktivitet
- lärarnas utvärdering av lektionen avseende matematiklärande
- den programmeringsmiljö som användes.

Analysen företogs i flera steg. I första steget identifierandes citat ur materialet där matematikinnehållet tydligt framgick. I en iterativ process reviderades dessa så att de citat som föreföll innehållsligt lika kom att formuleras på samma sätt i en komprimerad sammanställning. Varje gång en formulering ändrades validerades den mot dokumentationen för att säkerställa att ändringen bibehållit innebörden i originaldokumentet. Resultatet av denna process gav en översikt över de 32 lektionerna där olika trender kunde utläsas och analyskategorier formuleras.

Matematikinnehållet i lektionerna (forskningsfråga 1) kategoriserades enligt läroplanens beskrivning av det centrala innehållet i kursplanen innan programmering skrevs in (Skolverket, 2011). Inledningsvis testade vi att analysera innehållet efter kategorier som växte fram ur materialet, men fick problem med att kategorierna blev tvetydiga och beroende av definitionen av matematik. Exempelvis var beskrivningar av lärandemål avseende resonemang, kommunikation och problemlösning svåra att kategorisera. När vi utgick från läroplanens indelning av det centrala innehållet försvann tvetydigheten och en deskriptiv sammanställning kunde genomföras med få inslag av tolkning. Resonemang och kommunikation betraktades därvidlag som förmågor relaterade till det matematiska innehållet som angivits. I de fall där flera matematiska områden aktualiserades i lektionen valdes det som var mest explicit uttalat i lärarnas formuleringar. Exempelvis klassades ett innehåll som geometri om syfte och lärandemål relaterar till geometri trots att algebraiska formler för beräkning av area och omkrets är en bärande del av innehållet. I många lektioner förekommer variabler men utan explicita formuleringar kring variabelbegreppet i relation till algebraiska uttryck. När dokumentationen endast innehöll skrivningar kring hantering av variabler i programmeringsspråket snarare än hantering av variabler i algebra klassades det inte som en lektion i algebra. Denna aspekt återkommer vi till i diskussionen. När det gäller problemlösning, som i läroplanen anges både som förmåga och centralt innehåll, har vi endast klassificerat lektionen som problemlösning om den explicit handlat om matematiska problem eller matematiska lösningsstrategier.

Vad gäller *relationen mellan matematik och programmering* (forskningsfråga 2) var processen att finna kategorierna längre och mer omfattande. Analysen

genomfördes först i flera iterationer av författare 1. Därefter validerades tolkningen av övriga författare, och justerades tills vi tillsammans enades om formuleringen av fyra kategorier som kunde återfinnas i materialet. Därefter återvände vi till materialet och kategoriserade ånyo samtliga lektioner för att säkerställa en enhetlig tolkning av kategorierna. Vi fann ett visst överlapp mellan kategorierna på så sätt att kategori 2 även ingår i kategori 3 och 4 eftersom matematiken utgör en kontext för programmeringen även i båda dessa kategorier. De fyra kategorierna beskrivs i följande stycke, med exempel på hur klassificeringen gått till.

1. *Enbart programmering*. Dessa lektioner handlar enbart om programmering.
2. *Matematik som kontext för programmering*. I dessa lektioner utgör matematiken huvudsakligen en kontext för att lära sig programmering. Inget nytt matematiklärande åsyftas, men programmering kan utnyttjas för att repetera eller befästa matematikkunskaper.
3. *Programmering som ett verktyg för att effektivisera beräkningar inom matematiken*. I dessa lektioner finns ett tydligt matematikinnehåll. En dator programmeras och används som ett effektivt sätt att utföra beräkningar.
4. *Programmering som ett verktyg för att utforska matematik*. I dessa lektioner används programmering för att utforska matematiska begrepp eller samband. Programmeringen tillför nya sätt att närma sig matematiken och uppges på så sätt kunna fördjupa den matematiska förståelsen på en nivå som är relevant för den angivna årskursen.

Här följer två exempel från årskurs 1 som visar hur innehållet har klassificerats. SJ_1 beskriver syfte och lärandemål som fokuserar på att styra en BeeBot på en matta med hjälp av pilar. Lärarna skriver:

> Hur entydiga stegvisa instruktioner kan konstrueras, beskrivas och följas som grund för ***programmering***. Symbolers användning vid stegvisa instruktioner. Målet är att eleverna ska förstå hur stegvisa instruktioner kan konstrueras och läsas av genom att lägga egen kod till en BeeBot samt tolka andras kod. [SJ_1]

Lektionen klassificerades i enlighet med kategori 1 ovan, det vill säga som en lektion som innehöll enbart programmering eftersom ingen matematik utöver programmering explicit omnämndes. Inledningen av citatet är direkt taget ur läroplanens skrivningar om programmering, något som förekom i så gott som alla dokumentationer. Citat ur läroplanen gällande övrigt matematikinnehåll fanns inte i

den här lektionen. Även när sådana fanns var de inte alltid uppenbart relaterade till lektionens faktiska innehåll, varför störst vikt lades vid lärarnas egenformulerade syften och lärandemål, samt de lärandemål ur kursplanen som även kommenterades i utvärderingen av lektionen.

AS_1 beskriver en lektion i analog programmering som går ut på att beskriva ett händelseförlopp med hjälp av numrerade instruktioner. Lärarna skriver:

> Målet med matematiklektionen är att eleverna ska förstå hur entydiga stegvisa instruktioner kan konstrueras, beskrivas och följas som grund för programmering. Symbolers användning vid stegvisa instruktioner. Naturliga tal och hur de kan användas för att ange ordning. / … / Förutsättningar och förkunskapskrav: Känna till siffrorna 1-8 och kunna ange ordningsföljden. [AS_1]

Lektionen skiljer sig från SJ_1 genom att naturliga tal som ordningstal anges som en viktig del av lektionen, samtidigt som ordningstalen 1-8 betraktades som en förkunskap. Aktiviteten går ut på att eleverna ska ”rita sin morgon” i 8 rutor numrerade från 1 till 8. Lärarna menar att det är en lektion i analog programmering. Lektionen bedömdes innehålla matematik tillhörande det centrala innehållet taluppfattning och tals användning, och klassificerades i enlighet med kategori 2 ovan, där matematiken har rollen av en kontext i vilken (analog) programmering äger rum men utan att det finns ambitioner för något nytt matematiklärande.

5 Resultat

Först sammanfattas resultatet av analysen i tabell 1, där lektionernas matematikinnehåll (forskningsfråga 1) visas på den vertikala axeln och relationen mellan matematik och programmering enligt våra fyra kategorier (forskningsfråga 2) på den horisontella axeln. Tabellen visar också hur lektionerna fördelar sig över olika programmeringsmiljöer enligt följande kategorier: analog programmering helt utan dator eller digitala verktyg (n=4); robotprogrammering med exempelvis BeeBot (n=5); blockprogrammering som vanligtvis innebar Scratch eller Microbit (n=14); textprogrammering som främst innebar programmering i Python (n=8); samt Excel (n=1).

Table 1. Matematikinnehållet relaterat till relationen mellan matematik och programmering.

	Relationen mellan matematik och programmering			
Matematik-innehåll	1.Enbart programmering	2. Matematik som kontext för programmering	3. Programmering som verktyg för att effektivisera beräkningar	4. Programmering som verktyg för att utforska matematik
Enbart programmering n=10	**Analog**: HH_2, TO_4a **Robot**: SJ_1, TB_2, SE_3, TO_4b **Block**: AS_2, AS_9 MA_2, VT_3			
Taluppfattning och tals användning n=8		**Analog**: AS_1 **Block**: TO_3a, ST_5, HG_6 **Text**: SH_7, NB_7, ST_7	**Text**: BB_7	
Geometri n=8		**Analog**: TO_3b **Robot**: TB_1 **Block**: BF_3, MA_5 **Text**: NB_8	**Text**: KV_7	**Block**: AS_4, TO_6
Statistik och sannolikhet n=3			**Block**: TO_9 **Text**: ST_9 **Excel**: AS_7	
Algebra n=0				
Problemlösning n=1			**Text**: BB_8	
Samband och förändring n=2				**Block**: SJ_5, ST_6
n=32	n=10 (31%)	n=12 (38%)	n=6 (18%)	n=4 (13%)

5.1 Matematikinnehållet

Vi ser i tabell 1 att ganska olika matematik utgör innehållet när lärarna ska planera programmeringslektioner i matematik, men att 31% saknar matematikinnehåll. Taluppfattning och geometri är ofta förekommande. Algebra, som är det centrala innehåll läroplanen har placerat programmering inom, saknas helt som explicit matematikinnehåll i dessa lektioner. I följande avsnitt presenterar vi beskrivningar av och exempel på de lektioner som klassificerats i de fyra analyskategorierna med utgångspunkt i de kluster som framträder i tabellen relativt de olika matematiska innehållen.

5.2 Enbart programmering

Nästan en tredjedel (31%) av lektionerna handlar enbart om programmering. Lärandemål gällande programmering är klart och tydligt beskrivna men inga övriga lärandemål finns för matematik. Aktiviteterna går ofta ut på att programmera en robot att röra sig över golvet [SJ_1, TB_2, SE_2], att programmera en person (analogt) eller en virtuell robot i exempelvis ett dansprogram, eller att skapa en presentation i Scratch [AS_2, AS_9]. Det finns ingenting i dessa lektioner som markerar att de är matematiklektioner. Flera av lektionerna i den här gruppen är planerade som ämnesövergripande samarbeten där programmeringen utgör matematikdelen. Ett exempel är en lektion som genomfördes i årskurs 4 som ett samarbete mellan matematik och geografi. Lärarna skriver i sin planering av lektionen:

> Syftet med lektionen är att eleverna får öva på stegvisa instruktioner. Även möta nya begrepp som om (if), villkorssats. Testa felsöka. /…/ Introducera Sverigekartan med några stora städer utmärkt samt “hinder” som finns på ett rutsystem. /…/ De ska lära sig formuleringen: om hinder “hoppa över”. Förstå vilka instruktioner de ska använda. [TO_4b]

I lärarnas utvärdering ser vi att det är geografin som utgör kontext för programmering och att någon annan matematik än programmeringsinnehållet inte aktualiseras:

> Eleverna diskuterade engagerat geografi och hittade flera lösningar hur de kunde gå. Arbetet skapade kreativitet och lust hos de flesta eleverna.
> Många frågade om de fick gå nordöst eller om de fick simma hela vägen. Flera kom också på att man kunde skriva tre steg norr istället för att upprepa ett steg norr tre gånger. [TO_4b]

I den här kategorin återfinns exempel på lektioner där lärare och elever arbetar med stegvisa instruktioner och upprepning, två vanliga konstruktioner i programmering. I övriga kategorier kommer vi att se hur dessa och liknande konstruktioner används i mer matematiska sammanhang.

5.3 Matematik som kontext för programmering

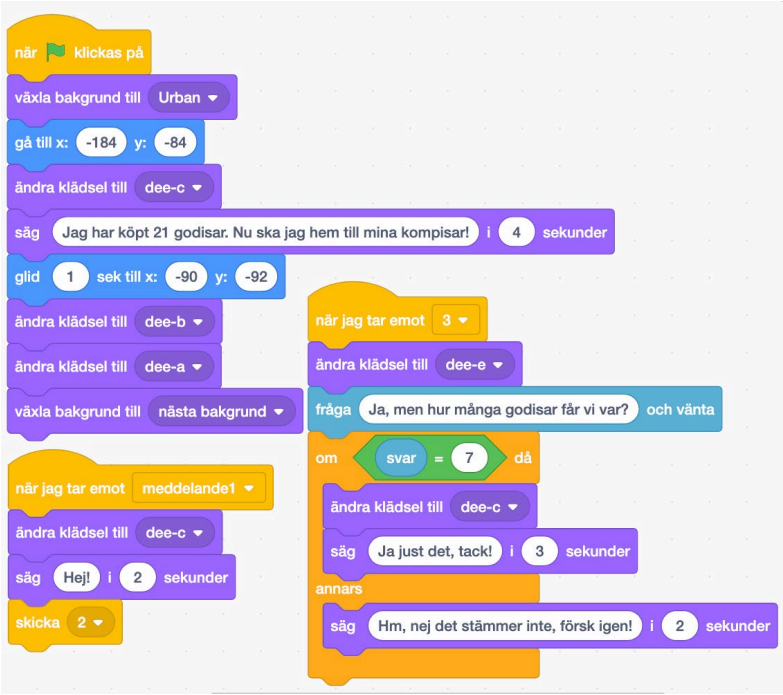
Den största gruppen (38%) utgörs av lektioner där matematiken fungerar som en kontext för programmering. Utöver mål relaterade till programmering finns i dessa lektioner ett tydligt matematiskt innehåll relaterat till taluppfattning eller till geometri, med lärandemål som handlar om att repetera och befästa kunskaper, eller

om att tillämpa gamla kunskaper i nya sammanhang. Exempelvis beskriver lärarna i ST_5 sitt lärandemål på följande sätt: ”Kunna använda tidigare kunskaper i matematik och programmering för att skapa ett program för att öva en huvudräkningsstrategi”, men konstaterar i utvärderingen att det blev ”Lite prat om matematik och mycket om kod”.

I vissa lektioner försvinner det matematiska innehållet nästan helt genom att andra aspekter av programmeringen hamnar i fokus. Ett sådant exempel är lektion TO_3a där följande aktivitet introduceras:

> Du ska göra en räknesaga i Scratch. Sagan ska innehålla följande:
> ★ Två eller flera sprajter som omväxlande gör saker, t.ex. rör sig eller pratar.
> ★ En mattefråga som den som kör ditt program kan svara på. Programmet ska berätta om man svarat rätt eller fel. [TO_3a]

Eleverna får en exempelkod att utgå från, se figur 2, och arbetar sedan med att först rita sin ”storyboard” och därefter programmera i Scratch. I utvärderingen skriver lärarna ingenting om de räknesagor som eleverna hittade på. Istället är fokus helt på elevernas svårigheter att förstå, välja, hämta och använda olika koder i Scratch. Matematiken blir i sammanhanget underordnad programmeringen.



Figur 2. Exempel på räknesaga i Scratch [TO_3a].

Ett annat exempel är lektion ST_7 som handlar om att lära sig skilja mellan de olika inmatningskommandona `input()` för text, `int()` för heltal och `float()` för decimaltal. Övningsuppgifterna innehåller en del algebraiska formler och matematiska beräkningar, men syftet med lektionen är att lära sig programspråkets struktur, inte att träna matematik eller lösa matematiska problem. Variabelbegreppet tas upp som problematiskt i utvärderingen:

> Eleverna blandade ihop två olika sätt att skapa variabler. Många använde "input" när de skulle skapa en variabel av typen `a="jonas"`. Utifrån detta tänker vi förtydliga skillnaden mellan dessa två sätt genom att ge eleverna en uppgift där de ska jämföra: `namn="Robert" , print(namn*5)` med `namn=input("vad heter du?") , print(namn*5)`. [ST_7]

Det är uppenbart att det här handlar om vad en variabel är i koden och hur programmet tolkar det som skrivs, snarare än variabelbegreppet i algebraisk mening.

Hälften av lektionerna i den här kategorin har geometri som matematiskt innehåll. Ett typexempel är lektion BF_3 där eleverna ska programmera en blue-bot med hjälp av pilar så att den tar sig fram i ett rutsystem. Lärarna beskriver aktiviteten:

> Pedagogen som håller i lektionen ger varje grupp en instruktion. T.ex "placera blue-boten på kvadraten med ögonen mot cirkeln och ta dig sedan till cylindern. /.../ För att öka svårighetsgraden kommer vi även att ge instruktioner som t.ex. "ställ blue-boten på den figuren som inte har några hörn med ögonen mot pyramiden. Ta dig sedan till den figuren som har 2 korta och 2 långa sidor. [BF_3]

Trots att lärandemålet uppges vara att "befästa de geometriska formerna och dess egenskaper" nämner lärarna i utvärderingen ingenting om elevernas förståelse av de geometriska begreppen utan fokuserar helt på deras svårigheter att relatera pilarna till roboten. Matematik fungerar således i första hand som en kontext för programmering där programmeringen används för att befästa matematikkunskaper.

En av geometrilektionerna, NB_8, arbetar med textprogrammering. Liksom i lektion ST_7 används övningar som går ut på att lära sig programspråket Python. Matematiken i NB_8 utgörs av att skriva ett program som beräknar area och omkrets av rektanglar. Eftersom lektionen hålls i årskurs 8 är sambanden mellan sidorna i en rektangel och dess omkrets respektive area inget nytt. Det eleverna ombeds reflektera över är hur koden kommer att se ut och vilket resultat en viss kod kommer att få, inte vilka generella matematiska slutsatser de kan dra om rektanglar. Det är känd matematik som används som kontext för att lära sig begrepp och syntax i ett

programmeringsspråk. Det som kommenteras av lärarna är åter igen begreppet variabel.

En sak som flera lärargrupper återkommer till i sina utvärderingar är att det tar lång tid för eleverna att lära sig programmera. Om matematiken endast utgör en kontext för programmering kan eleverna därför uppleva att de inte lär sig matematik på matematiklektionerna. Lärarna som genomfört lektion NB_7 noterar i sin utvärdering att ”Några elever var oroliga för att programmeringstiden tas från tiden för matematikundervisningen”.

5.4 Programmering som verktyg för att effektivisera beräkningar

I tabell 1 återfinns programmering som ett verktyg för att utföra effektiva beräkningar i matematik i sex lektioner med stor variation av matematiskt innehåll. KV_7 är en lektion i geometri, inte helt olik NB_8, men med programmeringsspråket Javascript. I KV_7 beskriver lärarna de matematiska målen för elevernas lärande i termer av att ”Stärka sin förmåga att använda formler som en metod att lösa enkla geometriuppgifter” samt ”stärka sin förståelse för begrepp kopplade till geometrin”. Eleverna får följande instruktioner:

> Nu ska ni få träna er i att använda datorerna som hjälpmedel när ni gör beräkningar med formler. Som vi har gått igenom tidigare är ju formler ett verktyg för att underlätta vid komplicerade beräkningar. Ibland är formlerna ganska enkla: Rektangelns area = basen · höjden (A=b·h), men ofta är de mer komplicerade: Klotets volym = 4 · 3,14 · radien · radien · radien/3 ($V=4\cdot\pi\cdot r^3/3$). [KV_7]

Skillnaden mot NB_8 är att dessa lärare på ett explicit sätt talar om för eleverna vad det är datorn tillför, att det är när beräkningarna är komplicerade som datorn är ett användbart verktyg.

Hälften av lektionerna i denna kategori handlar om statistik och sannolikhet. Två av dem är lektioner i årskurs 9 där datorn användes för att simulera tärningskast vid sannolikhetsexperiment [ST_9; TO_9]. Den tredje är AS_7, som handlar om att föra in värden i ett kalkylblad och sedan beräkna medelvärde, median och typvärde. Den klassificeras därför som en lektion i statistik. Lärarna citerar elevernas utvärdering:

> “Jag har lärt mig hur Variabel, Funktion och om vilkorssats. Jag har även lärt mig hur kalkylark fungerar”. “Jag har lärt hur kan man räkna medelvärde och median och typvärde utan ansträngning”. [AS_7]

Begreppen variabel och funktion relaterar i detta sammanhang mer till hur begreppen används i kalkylark än till deras innebörd inom områdena algebra respektive samband och förändring. Lärarna reflekterar kring just begreppet funktion:

> Vi upptäckte att lärare la olika vikt vid de olika begreppen, vilket senare ledde till en diskussion om begreppens innebörd. Ex: Funktion kan dels beskrivas som "två variabler som är beroende av varandra" och dels som ett sätt att förklara för datorn "vad den ska göra". Under LS förklarades detta begrepp på olika grunder vilket kan ha ursprung i olika erfarenhet och skolning i excel och matematik. [AS_7]

Vi noterar i den här lektionen dels att begreppet programmering sträcker sig utanför block- och textprogrammering till att även omfatta arbete med kalkylblad, dels att ett matematiskt begrepp som funktion kan ha en annan innebörd i programmering.

5.5 Programmering som verktyg för att utforska matematik

I två av geometrilektionerna ska eleverna använda programmering i syfte att utforska geometriska begrepp eller samband. I AS_4 får eleverna utforska begreppen vinkel och rotation samt fundera på sambandet mellan vinkelns gradtal och andel av cirkel som hel, halv, fjärdedel och tredjedel. Lektionen utförs i både årskurs 4 och årskurs 6. Lärarna beskriver syftet med lektionen på följande sätt:

> Vi har tagit fram den här lektionen för att tydliggöra kopplingen mellan programmering och matematik. Vi vill se om eleverna kan konstruera geometriska objekt med hjälp av sina kunskaper i programmering. Vi vill se om eleverna får mer fördjupade kunskaper i matematiska begrepp med hjälp av programmering t.ex. se sambandet mellan gradantalet i en cirkel och gradantalet i en halv cirkel. Vi vill att de ska visa detta genom att programmera dessa objekt. [AS_4]

Eleverna får en färdig kod som ritar en cirkel, se figur 3. Läraren frågar:

> Var tror ni att jag ska ändra någonstans för att det ska bli en halvcirkel, fjärdedelar och tredjedelar? Testa själv! / … / Hur gör ni för att få halva cirkeln röd och den andra halva blå? [AS_4]

Figur 3. Kod som ritar en cirkel.

I TO_6 får eleverna utforska sambandet mellan arean på en rektangel och en triangel med samma bas och höjd. Exempelkod i Scratch ges för rektangel och ska modifieras för triangel. Lektionen utmärker sig av att lärarna uppger ett utforskande syfte, eleverna utmanas matematisk. De skriver:

> Vi valde området geometri och area då vi kunde se att det skulle vara en lagom utmaning att förstå sambandet mellan arean av en rektangel och arean av en triangel. [TO_6]

Eftersom programmeringsövningen bygger på att eleverna redan känner till sambandet är aktiviteten endast svagt utforskande. Den klassificeras som utforskande eftersom lärarna förväntar sig att programmeringsaktiviteten ska bidra till att skapa förståelse för sambandet. Lärarna skriver i utvärderingen:

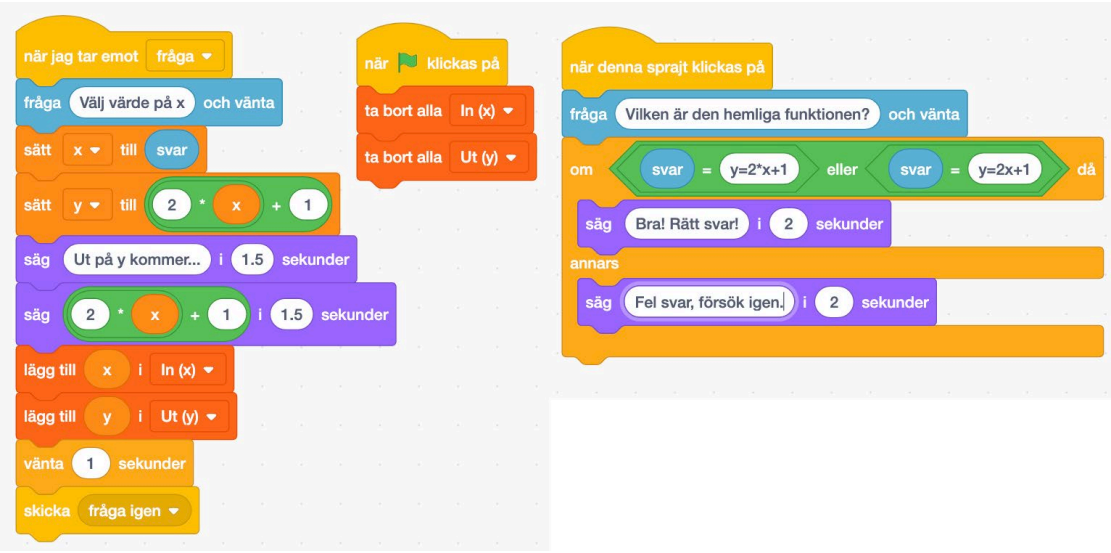
> Eleverna befäste och fick förståelse för areabegreppet. En del kunde sedan innan men en hel del var tvungna att be om hjälp för att förstå. De kunde inte gå vidare med utmaningen att modifiera koden om de inte förstod vad de skulle

> göra. /.../ Lektionens syfte och mål nåddes i hög grad, eleverna förstod area begreppet och kunde modifiera en befintlig kod. [TO_6]

Vidare uppmärksammar lärarna att eleverna ”inte förstod att de kunde stoppa in bas*höjden som en variabel”.

En del av det centrala innehållet som i kursplanen kallas samband och förändring utgörs i årskurs 4-6 av koordinatsystemet, något som utforskas i SJ_5. Lektionen handlar om stegvisa instruktioner där olika figurer ritas i ett koordinatsystem med hjälp av kommandon i Scratch. Det som skiljer den här aktiviteten från lektionerna om stegvisa instruktioner i analyskategori 1 är att instruktionerna ges med hjälp av ett matematiskt representationssystem istället för med ord och ikoniska bilder. I den här aktiviteten används programmering som ett sätt att utforska begreppen koordinater, *x*-led, *y*-led och origo.

Även funktioner och räta linjens ekvation återfinns i kursplanens samband och förändring (dock endast för årskurs 7-9), något som här utgör innehållet i ST_6. Lärarna uppger att lektionen bygger på en tidigare lektion där en analog ”funktionsmaskin” skulle programmeras. Här görs det i Scratch med hjälp av en delvis färdig kod som eleverna ska remixa, se figur 4.



Figur 4. Exempelkod med frågan: På vilket sätt kan koden ändras?

Syftet med lektionen uppges vara att eleverna ska ”fördjupa sig i funktionsbegreppet genom att använda tidigare kunskaper i en ny situation”. De beskriver lärandemålet på följande sätt:

> Målet är att alla ska kunna remixa en exempelkod och skapa en egen funktion som gör att programmet kan räkna ut y-värdet om användaren anger ett x-värde. Funktionen i exempelkoden är räta linjens ekvation, dvs y = k x + m där både k och m är positiva heltal. Ytterligare ett mål är att elever som vill ha mer utmaning ska kunna göra andra typer av funktioner där t.ex. k<1 och m<0, vilket skulle kunna bli en funktion av typen y = x/2 – 4. [ST_6]

Resultatet av att använda programmering som ett verktyg i syfte att utforska begreppet närmare var enligt lärarna blandat. I utvärderingen skriver de:

> Vår uppfattning är att vissa elever fördjupade sina kunskaper om funktioner, medan andra mest härmade koden och inte riktigt förstod hur det hängde ihop med det vi arbetat med tidigare kring funktioner. Något som indikerar att elever fördjupade sina kunskaper om funktionsbegreppet var att flera valde att göra annat än räta linjens ekvation, vilket de inte arbetat med tidigare. I klass nr 2 har 14% av eleverna gjort ickelinjära funktioner. [ST_6]

6 Diskussion

Studiens resultat kommer nu att diskuteras i relation till tidigare forskning samt olika faktorer som är relevanta för att belysa transpositionen av kunskap från den *avsedda kunskapen* i termer av kursplanens syftesbeskrivning och centrala innehåll till *undervisad kunskap* så som den framstår i det empiriska materialet. Den första forskningsfrågan fokuserar på de innehållsliga val lärarna gör när de ska programmera på matematiklektionen och den andra frågan handlar om den relation mellan matematik och programmering som framträder i dessa lektioner. Det går inte att avgöra om valet av innehåll är ett resultat av lärarens syn på relationen mellan de två kunskapsfälten eller om det förhåller sig tvärtom. Men det är tydligt att *vad och hur* samverkar med *varför*. Nedan diskuterar vi denna samverkan i lektioner i det inledande skedet av implementeringen av programmering som ett nytt kunskapsinnehåll i grundskolans matematikämne.

6.1 Syftet med programmering i matematik

Vi har i analysen av lärarnas dokumentation av de lektioner de planerat och genomfört upptäckt en viss otydlighet gällande relationen mellan programmering och matematik. Drygt 30% av de studerade lektionerna handlade uteslutande om programmering utan annan koppling till matematik. Ytterligare 38% av lektionerna handlade om programmering som är inlagd i en matematisk kontext utan att ha som ambition att tillföra matematiklärandet något nytt. Matematiken utgör i dessa lektioner en ram kring programmeringen för att legitimera dess existens som en del

av matematikämnet utan att programmeringen tas i anspråk som en resurs för matematiken.

Det går att tolka situationen som att transpositionen av matematiken medför en epistemologisk förändring av matematikämnet. Om programmering finns med som ett innehåll i matematikämnet kommer lärarna att betrakta den som en del av matematiken. Att många av lektionerna vi studerat fokuserar programmering i sig självt snarare än matematiken är inte förvånande då man ser på tidigare försök med programmering i matematik (se avsnitt 3). Det finns här vissa likheter med ämnet statistik. Ursprungligen var statistik ett samhällsvetenskapligt ämne där matematik användes för att förstå samhälleliga problem. Så småningom utvecklades speciella statistiska metoder och representationer som kom att inlemmas som en del av matematiken. Det vi nu ser är hur programmering går från att vara ett eget fält som utvecklats med hjälp av matematik till att inlemmas som en del av matematiken. Därmed förändras innebörden av skolmatematiken så att ämnet får en ökad inriktning mot tillämpad matematik.

En annan tolkning av resultatet är att programmering i det inledande skedet av implementeringen, när programmering även är nytt för lärarna, av nödvändighet handlar mest om programmeringens vad och hur. Senare, när lärarna utvecklat en säkerhet i programmering och eleverna möter det varje skolår, kommer kanske större fokus att riktas mot programmering som verktyg för problemlösning, beräkningar och utforskande av matematik. En sådan argumentation fanns bland en del lärare som intervjuats i en tidigare studie (Kilhamn m fl., 2021). I sammanhanget kan man fråga sig varför det i så fall är just matematikämnet som ska lägga tid på att lära eleverna programmera och varför det är just matematiklärare som ska undervisa om det.

I motsats till kategori 1 och 2 som främst fokuserar programmering som innehåll, relaterar lektionerna i kategori 3 och 4 tydligare till läroplanens beskrivning av syftet med programmering. I kategori 3 används programmeringen som ett verktyg för att utföra beräkningar. Programmering blir en matematisk metod bland många andra med vissa unika fördelar. Alla sex lektionerna i denna kategori genomfördes i årskurs 7-9, vilket skulle kunna tyda på att det är först då som beräkningarna är så komplicerade eller omfattande att programmering verkligen är en hjälp. Att ta in programmering som beräkningsmetod där det är enklare med huvudräkning eller miniräknare är som att använda grävmaskin där det bara behövs en spade. Detta fenomen såg vi i en del av lektionerna där matematiken utgjorde en kontext för

programmering, där övningarna inte gick ut på att så smidigt som möjligt utföra beräkningen, utan att lära sig utföra den med hjälp av programmering.

I lektionerna i kategori 4 används programmering för att utforska matematiska begrepp och samband mellan begrepp. De fyra lektionerna i den här kategorin genomfördes alla i årskurs 4-6, i en blockprogrammeringsmiljö. I en kritisk granskning av dessa fyra lektioner framstår inte programmeringen som det bästa verktyget för det undersökande arbetssättet. Om lärarna inte samtidigt haft som syfte att skapa en lektion i programmering skulle de med stor sannolikhet ha nått bättre resultat med hjälp av andra digitala verktyg, som exempelvis GeoGebra. Det kräver stor kunskap om programmet för att åstadkomma bra matematiska visualiseringar i exempelvis Scratch, eftersom programmet inte i första hand är tänkt som ett matematikverktyg (Resnick m fl., 2009).

Genom att läroplanen tar upp digitala verktyg och programmering i samma syftesformulering blir det upp till läraren att avgöra när programmering eller andra digitala verktyg fungerar bäst i relation till syftet. Frågan är också om programmering är att betrakta som ett bland andra digitala verktyg eller om programmering är något i grunden annorlunda. Att en lektion handlar om programmering i kalkylark antyder att det inte för lärarna är uppenbart var gränsen går mellan programmering och andra digitala verktyg.

6.2 Kursplanens centrala innehåll i matematik

Bortsett från de lektioner som klassificerades som endast programmering var det innehållsområdena taluppfattning och tals användning respektive geometri som var vanligast förekommande i den här studien. I majoriteten av dessa lektioner användes dock matematiken enbart som en kontext för att lära sig programmera, vilket ofta ledde till att det matematiska innehållet hamnade i skymundan av programmeringen eller till och med försvann. Detta resultat, och att en tredjedel av lektionerna inte visar någon koppling till matematik, indikerar att flera av lärarna hade svårigheter att hitta meningsfulla samband mellan programmering och matematik. Denna slutsats ligger i linje med vad som hittills framkommit i studier om lärares initiala respons på programmeringens införande i skolmatematiken (Kilhamn m fl., 2021; Misfeldt m fl., 2019; Pörn m fl., 2021). I vår studie framkom dessutom en oro hos flera lärare och elever över att programmeringen tar lång tid att lära sig och därmed stjäl tid från matematikinnehållet. Detta är en oro som bör tas i beaktande, speciellt med tanke på att det redan påvisats att det snabba införandet av programmering i svensk

skolundervisning har skapat stress hos lärare, framförallt hos de lärare som saknar erfarenhet av programmering (se Mozelius m fl., 2019).

Några av de lektioner som ingick i den här studien klassificerades som statistik och sannolikhet respektive samband och förändring. Inte helt oväntat användes programmeringen som ett verktyg för att utföra beräkningar under lektionerna i statistik och sannolikhet. I de lektioner som behandlade samband och förändring användes programmeringen som ett verktyg för att utforska matematik, till exempel fördjupade eleverna sina kunskaper om funktionsbegreppet. I en programmeringskontext gav just detta begrepp anledning till en fördjupad diskussion bland lärare, då de insåg att begreppet tolkades olika: under vissa lektioner definierades funktioner utifrån ett matematiskt perspektiv som ”samband mellan variabler” och under andra lektioner utifrån en programmeringskontext som ”instruktioner för att lösa problem”. I ett lärandeperspektiv är denna diskussion mycket relevant med tanke på att man inom programmering och matematik ofta använder samma begrepp och symboler utan att dessa alltid har samma innebörd (se Bråting & Kilhamn, 2020). Funktionsbegreppet är ett typiskt exempel på detta då en funktion inom matematiken alltid innebär en relation mellan två mängder medan en funktion inom programmering ibland likställs med en procedur. För matematiklärare är dessa begreppsliga skillnader av stor vikt att känna till för att inte skapa begreppsförvirring hos eleverna. Studier om tidigt algebraiskt tänkande visar att funktionsbegreppet kan vara problematiskt för många elever även utan att programmeringen förs in i skolmatematiken (Kieran, 2018). Vi menar därför att det är viktigt att matematiklärare inte bara fortbildas i programmering utan också i hur införandet av programmering i matematiken kan påverka elevers matematiska begreppsbildning.

De innehållsområden som förekom minst i vårt empiriska material var problemlösning respektive algebra. Endast en lektion klassificerades som problemlösning och ingen som algebra. Detta resultat är något överraskande med tanke på att programmering har införts som en del av det centrala innehållet i algebra (Skolverket, 2017). Det är dock viktigt att påpeka att variabler förekom i flera lektioner som inte klassats som algebra, exempelvis när eleverna använde formler för att beräkna areor av geometriska objekt. På motsvarande sätt som funktionsbegreppet diskuterade lärarna att det finns olika slags variabler i matematik jämfört med programmering. Lärarna syftade här på att inom programmering kan en variabel innehålla text, vilket skiljer sig från matematiken där variabler vanligtvis innehåller

tal (se Bråting & Kilhamn, 2020). Ur ett lärandeperspektiv hävdar vi att det är viktigt att matematiklärare får tid och möjlighet att fördjupa sig i variabelbegreppets olika innebörder inom såväl programmering som matematik.

I den här studien har vi definierat algebra utifrån det centrala innehållsområdet i läroplanen. Efter revideringen 2017 inkluderar innehållsområdet algebra även stegvisa instruktioner, algoritmer och programmering i visuella och textuella programspråk. Det vi kan konstatera är att de mer traditionella områdena inom skolalgebra som till exempel ekvationslösning och mönster inte finns representerade i vårt empiriska material. Algebra är ett område där svenska elever i internationella tester inte brukar prestera så bra, och också ett område som i våra läroplaner haft en svag inramning och otydlig beskrivning (Prytz, 2020). Tydliga exempel på detta är att stora delar av det som nu klassificeras som samband och förändring i tidigare kursplaner har tillhört innehållsområdet algebra samt att man i revideringen år 2017 förde in programmering som en del av algebrainnehållet. I samband med detta har dessutom begreppet algoritm, som tidigare kopplats till aritmetik, flyttats till algebra. Detta visar att ämnet genomgår en transposition som knappast stärker aspekter som traditionellt kopplats till algebra, såsom fokus på generalisering, samband och struktur, och som utgör fokus för internationell forskning kring algebra i grundskolan (Kieran, 2018). Man kan spekulera kring vad som i framtiden kommer att innefattas av skolalgebran, speciellt med tanke på att många lärare kopplar samman programmering med mer allmänna förmågor som att kunna samarbeta, kommunicera och följa instruktioner (Nouri m. fl., 2020; Pörn m fl., 2021), och utveckling av datalogiskt tänkande (Jahnke, 2020).

6.3 Val av programmeringsmiljö och programmeringsspråk

Bland de lektioner som endast handlar om programmering arbetar elever i årskurs 1-4 antingen analogt, med robotar eller med blockprogrammering. I dessa programmeringsmiljöer kan elever enkelt skapa kod och testa den kod de själva skapar. För eleverna blir det ett arbetssätt med många och snabba körningar, där syntaktiska fel uppenbaras om de har skrivit fel i koden och logiska fel om algoritmen inte är korrekt. Ett bra exempel på detta är analog programmering (dans) där elever måste reflektera över ”robotens” handlingar i relation till det önskade rörelsemönstret. Lektionerna i denna kategori visar att lärare har följt kursplanens centrala innehåll genom att undervisa om stegvisa instruktioner. Innehållet har sitt ursprung i programmering som vetenskap snarare än i matematik som vetenskap.

Lärandemålen i dessa lektioner är väl i linje med mål för datalogiskt tänkande som beskrivs av exempelvis Brennan och Resnick (2012). Några uttalade mål kring datalogiskt tänkande finns dock inte i kursplanen för matematikämnet.

I de lektioner där matematiken görs till kontext för programmering är spridningen stor över årskurser och programmeringsmiljöer. Val av programmeringsmiljö följer den progression som kursplanen beskriver, med blockprogrammering i årskurs 4-6 och textprogrammering i årskurs 7-9. I vissa textbaserade språk, som t.ex. Python, kan datorn inte avgöra om ett inmatat tecken är ett värde eller ett tecken. För att arbeta med heltal behöver man använda en instruktion `int()` som omvandlar tecknet till ett värde, vilket eleverna i lektion ST_7 tränar på. Just denna instruktion kan dock bli obegriplig om läraren inte har kunskaper om programmeringens historia, och matematiken riskerar att antingen blandas ihop med, eller hamna i skymundan av, programmeringen.

I de fyra lektioner där programmering används som verktyg för att utforska matematik, används blockprogrammering. I dessa lektioner beskriver lärarna elevers svårigheter att hantera kod och syntax, vilket begränsar utforskandet av matematiska begrepp. En annan begränsande faktor skulle kunna vara lärares bristande kännedom om vilka möjligheter som erbjuds och vilka svårigheter som finns, exempelvis med distraktorer som för fokus bort från matematiken (Benton m fl., 2017; Bråting m fl., 2021). Att få elever att ta steget från att använda program till att skapa program, vilket beskrivs som en av målsättningarna med Scratch (Resnick m fl., 2009), ställer stora krav även på lärares kunskaper och förmåga att hantera programmeringsmiljön som ett undervisningsverktyg.

I en intervjustudie lyfter lärare fram att blockprogrammering främjar elevers kreativitet och engagemang (Kilhamn m fl., 2021). Att gå vidare från ökat engagemang till ökat lärande kräver dock en viss struktur med tydliga målsättningar så att eleverna arbetar mot avsedda kunskapsmål. I de utforskande lektionerna (kategori 4) ser vi att lärarna försöker skapa en sådan struktur i aktiviteterna. Samtliga har valt att arbeta med färdig kod där eleverna förväntas reflektera över matematiska konsekvenser då koden ändras (remixing), ett arbetssätt som Brennan och Resnick (2012) lyfter fram som en viktig datalogisk praktik.

6.4 Avslutande reflektion

Genom införandet av programmering i matematikämnet har Skolverket gjort ett val som påverkar synen på matematik. Många av lärargrupperna citerar följande ur läroplanens syftesformulering.

> Genom undervisningen ska eleverna ges förutsättningar att utveckla förtrogenhet med grundläggande matematiska begrepp och metoder och deras användbarhet (Skolverket, 2017).

När detta syfte relateras till en lektion som helt ägnas åt programmeringsbegrepp och kodning utan att andra grundläggande matematisk begrepp och metoder tas upp kan vi dra slutsatsen att lärarna anser att programmering utgör en del av matematiken och att programmeringsbegrepp nu anses vara matematiska begrepp. Detta resultat skiljer sig från resultatet av vår tidigare intervjustudie, där lärarna sällan uppfattade programmering i sig som en matematisk aktivitet (Kilhamn m fl., 2021).

När lektioner i programmering saknar koppling till traditionell matematik kan det diskuteras om programmering bör förläggas till just matematikämnet. Med tanke på vilket matematikinnehåll lärarna i den här studien sett som lämpligt för arbete med programmering, där ingen lektion klassificerades som en lektion i algebra, kan det också diskuteras om programmeringen verkligen ska anses utgöra en del av det centrala innehållet i just algebra. Om den svenska modellen med programmering som en del av algebran får internationellt genomslag återstår att se.

I den transpositionsprocess från avsedd kunskap till undervisad kunskap som vi beskrivit här ser vi att större vikt läggs vid det centrala innehållet än vid läroplanens syftesbeskrivning som legitimerar kunskapsstoffet och talar om vad det ska användas till. I Chevallards (2006) teori skulle detta kunna uttryckas i termer av större fokus på praxis (vad och hur) än på logos (varför). Att lära sig vad programmering är och hur det går till är mer framträdande i vårt material än frågan om vad programmering kan användas till och varför.

Tillkännagivanden

Det forskningsprojekt som denna artikel baseras på är finansierat av Vetenskapsrådet [Bidragsnummer 2018-03865]. Vi vill tacka IFOUS-gruppen vid Stockholms universitet för värdefullt samarbete.

Referenser

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: some findings of design research in England. *Digital Experiences in Mathematics Education 3*(2), 115–138. DOI https://www.doi.org/10.1007/s40751-017-0028-x

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association* (s. 1-25), Vancouver, Canada.

Bosch, M., & Gascón, J. (2006). Twenty-five years of the didactic transposition. *ICMI Bulletin, 58*, 51-65.

Brown, N., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education, 14*(2), 1–22.

Bryman, A. (2012). *Social research methods*. Oxford University Press.

Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning*. *23*(2), 170-185.

Bråting, K., Kilhamn, C., & Rolandsson, L. (2021). Integrating programming in Swedish school mathematics: description of a research project. I Y. Liljekvist, L. Björklund Boistrup, J. Häggström, L. Mattsson, O. Olande, H. Palmér (Red.), *Sustainable mathematics education in a digitalized world. Proceedings of MADIF12*. (s. 101–110). NCM & SMDF

Chevallard, Y. (2006). Steps towards a new epistemology in mathematics education. I M. Bosch (Red.), *Proceedings of the Fourth Congress of the European Society for Research in Mathematics Education, CERME 4* (s. 21–30). FUNDEMI IQS-Universitat Ramon Llull.

Falkner, K., Vivian, R., & Falkner, N. (2014). The Australian digital technologies curriculum: Challenge and opportunity. I J. Whalley & D. D'Souza (Red.), *Proceedings of the Sixteenth Australasian Computing Education conference* (s. 3–12). Australian Computer Society.

Fisher, L. (2016). A decade of ACM efforts contribute to computer science for all. *Communications of the ACM, 59*(4), 25–27.

Grover & Pea. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher, 42*(1), 38–43.

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education, 4*(1), 48–69.

Hedrén, R. (1990). *Logoprogrammering på mellanstadiet : en studie av fördelar och nackdelar med användning av Logo i matematikundervisningen under årskurserna 5 och 6 i grundskolan*. Linköpings universitet.

Jahnke, A. (Red). (2020). *Programmering i skolan. Var, när, hur och varför?* Slutrapport från FoU-programmet Programmering i ämnesundervisningen. Ifous rapportserie 2020:5

Kang, W., & Kilpatrick, J. (1992). Didactic transposition in mathematics textbooks. *For the Learning of Mathematics*, *12*(1), 2–7.

Kieran, C. (Red.) (2018). *Teaching and learning algebraic thinking with 5-12-year-olds*. Springer.

Kilhamn, C., & Bråting, K. (2019). Algebraic thinking in the shadow of programming. I U. T. Jankvist, M. van den Heuvel-Panhuizen, & M. Veldhuis (Red.), *Proceedings of the Eleventh Congress of the European Society for Research in Mathematics Education, CERME11* (s. 566-573). Freudenthal Group & Freudenthal Institute, Utrecht University and ERME.

Kilhamn, C., Bråting, K., & Rolandsson, L. (2021). Teachers' arguments for including programming in mathematics education. I G.A. Nortvedt, N.F. Buchholtz, J. Fauskanger, F. Hreinsdóttir, M. Hähkiöniemi, B. E. Jessen, ..., A. Werneberg (Red.) *Bringing Nordic mathematics education into the future. Papers from NORMA 20. Preceedings of the Ninth Nordic Conference on Mathematics Education*. (s. 169–176). Oslo, June 1 - 4, 2021. NCM & SMDF.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in K-9 education. I A. Clear & R. Lister (Red.), *Proceedings of Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference* (s. 1–29). ACM.

Misfeldt, M., Jankvist, U.T., Geraniou, E., & Bråting, K. (2020). Relations between mathematics and programming in school: Juxtaposing three different cases. I A. Donevska-Todorova, E. Faggiano, J. Trgalova, Z. Lavicza, R. Weinhandl, A. Clark-Wilson & H-G. Weigand (Red.), *Proceedings of the 10th ERME topic conference on mathematics education in the digital era (MEDA 2020),* (s. 255-262). Johannes Kepler University.

Misfeldt, M., Szabo, A., & Helenius, O. (2019). Surveying teachers' conception of programming as a mathematical topic following the implementation of a new mathematics curriculum. I U.T Jankvist, M. Van den Heuvel-Panhuizen, & M. Veldhuis (Red.), *Proceedings of the Eleventh Congress of the European Society for Research in Mathematics Education, CERME11*, (s. 2713-2720). Freudenthal Group & Freudenthal Institute, Utrecht University and ERME.

Mozelius, P., Ulfenborg, M., & Persson, N. (2019). Teacher attitudes towards the integration of programming in middle school mathematics. I *INTED 2019* (s. 701-706). The International Academy of Technology, Education and Development.

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers* (Vol. 17). Springer Science & Business Media.

Nouri, J., Zhang, L., Mannila, L., & Norén. E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1-17.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Prytz, J. (2020), Algebra in Swedish mathematics curricula (1930–2000). In E. Barbin, K. Bjarnadóttir, F. Furinghetti, A. Karp, G. Moussard , J. Prytz, & G. Schubring (Red.), *"Dig where you stand" 6. Proceedings of the sixth International Conference on the History of Mathematics Education*. WTM-Verlag

Pörn, R., Hemmi, K., & Kallio-Kujala, P. (2021). "Programming is a new way of thinking" – teacher views on programming as a part of the new mathematics curriculum in Finland. I Y. Liljekvist, L. Björklund Boistrup, J. Häggström, L. Mattsson, O. Olande, H. Palmér (Red.), *Sustainable mathematics education in a digitalized world. Proceedings of MADIF12*. SMDF.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67.

Riis, U. (1987). *Datalära på grundskolans högstadium – Utvärdering av en treårssatsning*. Linköpings Universitet.

Rolandsson L. (2011). Teacher pioneers in the introduction of computing technology in the Swedish upper secondary school. I J. Impagliazzo P. Lundin B. Wangler (Red.), *History of Nordic Computing 3. HiNC 2010. IFIP Advances in Information and Communication Technology, vol 350*. Springer.

Sjöberg, C., Risberg, T., Nouri, J., Norén E. & Zhang, L. (2019). A lesson study on programming as an instrument to learn mathematics and social science in primary school. *13th annual*

International Technology, Education and Development Conference. Valencia, Spain. 11-13 March,

Skolverket. (2017). *Läroplan för grundskolan, förskoleklassen och fritidshemmet, Lgr11*. Fritzes.

Skolverket. (2011). *Läroplan för grundskolan, förskoleklassen och fritidshemmet, Lgr11*. Fritzes.

Skolöverstyrelsen. (1967). *Läroplan för grundskolan, Lgr69*. Liber UtbildningsFörlaget.

Skolöverstyrelsen (1980). *DIS. Datorn i skolan*. Slutrapport SÖ-projekt 628.

Skolöverstyrelsen (1984). *Datalära i grundskolan*. Studieplan. Liber UtbildningsFörlaget.

Söderlund, A. (2000). *Det långa mötet - IT och skolan: om spridning och anammande av IT i den svenska skolan*. Luleå Tekniska Universitet.

Takahashi, A., & Yoshida, M. (2004). Lesson-study communities. *Teaching children mathematics*, *10*(9), 436-437.

Utbildningsdepartementet. (1994) *Läroplan för det obligatoriska skolväsendet, förskoleklassen och fritidshemmet*, *Lpo 94*. Fritzes.

Zhang, L., Nouri, J. & Rolandsson, L. (2020). Progression of computational thinking skills in Swedish compulsory schools with blockbased programming. I *Proceedings of 22nd Australasian Computing Education Conference* (ACE'2020). ACM, Melbourne, Australia. https://doi.org/10.1145/3373165.3373173