# Forenklet, men presis beskrivelse av objekter

Ragnhild Kobro Runde og Olaf Owe
Institutt for informatikk, Universitetet i Oslo

Sammendrag

Vi presenterer her en ny variant av en formalisme for spesifikasjon av parallelle prosesser (objekter). Det viktigste for oss er at formalismen skal være enkel og intuitiv. Vi fokuserer derfor på observerbarhet, det vil si på hva som skjer i kommunikasjonen mellom de ulike objektene. Vi er også spesielt interessert i at ikke-determinisme og parallellitet skal kunne håndteres på en enkel måte.

Formalismen tar utgangspunkt i første-ordens logikk med likhet. I tillegg har vi et begrep om "readiness", det vil si hvilke hendelser det er mulig at et objekt vil kunne ta del i på et gitt tidspunkt. For hvert objekt lagrer vi også en sekvens av de hendelsene det allerede har vært involvert i, slik at vi kan spesifisere neste mulige hendelser ut fra hva som har skjedd tidligere.

Vi har en enkel og intuitiv syntaks, men dermed er det også mulig å skrive meningsløse objektuttrykk. En case-studie viste at det var lite intuitivt å måtte tenke på eksistensen av et meningsløst objekt. Siden et slikt objekt ikke kan forekomme i praksis, er det mest intuitivt å skrive spesifikasjoner som om det meningsløse objektet heller ikke er en teoretisk mulighet. Vi foreslår her en ny semantikk for selve ready-relasjonen slik at vi oppnår nettopp dette.

## 1 Innledning

Vi har som mål å kunne gi presise beskrivelser av objekt-orienterte systemer på en enkel og intuitiv måte, slik at beskrivelsesspråket kan ha praktisk nytte. Vi presenterer her en forenklet formalisme for å kunne spesifisere og resonnere om parallelle objekter på en slik måte. Formalismen er i utgangspunktet som beskrevet i [9], men vi har gjort noen endringer i et forsøk på å gjøre den enda enklere. Disse endringene er motivert ut fra en case-studie hvor vi brukte den opprinnelige formalismen på "The RPC-Memory Specification Problem" [1]. Dette eksempelet er allerede løst av flere andre formalismer, og er derfor et godt utgangspunkt for å sammenligne ulike metoder. I denne artikkelen presenterer vi den endrede formalismen, mens selve case-studien og sammenligningen vil bli beskrevet i [10].

Vi ønsker å holde spesifikasjonene på et relativt abstrakt nivå, med en "black box"-beskrivelse av objekter. Det innebærer blant annet at vi ikke ser på implementasjonsdetaljer, men istedenfor konsentrerer oss om hva som skjer i kommunikasjonen mellom de ulike objektene (og med brukeren).

For hvert objekt ser vi på tids-sekvensen av de observerbare hendelsene det har vært involvert i. Siden vi på et vilkårlig tidspunkt bare kan ha hatt et endelig antall hendelser, er

det naturlig at vi begrenser oss til å se på endelige sekvenser. Ved hjelp av ready-relasjoner kan vi enkelt angi hvilke mulige fortsettelser et objekt kan ha på et gitt tidspunkt. Vi bruker $\leftarrow$ for å betegne umiddelbar "readiness", slik at for eksempel $A \leftarrow x$ angir at objektet $A$ er umiddelbart klar for hendelsen $x$. Siden de lokale variablene i et objekt ikke er observerbare utenfra, kan ikke disse brukes til å beskrive objektets tilstand. Vi bruker istedenfor sekvensen av hendelser $h$ for å gi et abstrakt bilde av tilstanden, slik at vi lar $A/h$ betegne objektet $A$ etter sekvensen $h$. Hva $A$ er klar for på et gitt tidspunkt vil gjerne være (delvis) avhengig av denne sekvensen.

Siden parallellitet gir opphav til ikke-determinisme, er vi også interessert i å omtale ikke-deterministiske objekter, uansett om denne ikke-determinismen skyldes objektet selv eller noe utenfor dette. For ytre ikke-determinisme bruker vi den logiske operatoren *og* ($\wedge$) slik at $A \leftarrow x \wedge A \leftarrow y$ angir at $A$ er klar for både $x$ og $y$. Nøyaktig hvilken av hendelsene $x$ og $y$ som faktisk vil inntreffe avhenger da av omgivelsen til $A$. Tilsvarende bruker vi logisk *eller* ($\vee$), eventuelt *xor*, for å angi indre ikke-determinisme, slik at $A \leftarrow x \vee A \leftarrow y$ angir at $A$ er klar for enten $x$ eller $y$, men akkurat hvilken som vil være aktuell bestemmes av $A$ selv. (Ved bruk av *xor* istedenfor $\vee$ her, sier vi at $A$ ikke kan velge å være klar for begge.)

### *1.1 Eksempel: en sjokoladeautomat (hentet fra [8])*

Først litt notasjon:

- $A \approx B$ angir at $A$ og $B$ har nøyaktig samme mulige observerbare fortsettelser, det vil si at det ikke er eller vil bli mulig å se forskjell på dem bare ved å studere hvilke hendelser de kan akseptere videre.
- Vi lar $\langle x_1, x_2, \ldots, x_n \rangle$ betegne sekvensen av hendelsene $x_1, x_2, \ldots$ og $x_n$.
- $\not\leftarrow$ er den negerte varianten av $\leftarrow$.

Vi har en maskin $M$ som aksepterer mynter, enten to 5-ere eller en 10-er etter brukerens ønske, og gir en sjokoladeplate ($a$) eller pengene tilbake ($b$) etter maskinens eget valg. Vi kan dermed spesifisere følgende:

- $M \leftarrow 5 \ \wedge \ M \leftarrow 10$
  *Brukeren kan velge å legge på enten en 5-er eller en 10-er. Dette gir opphav til ytre ikke-determinisme.*
- $M\langle 5 \rangle \leftarrow 5$
  *Hvis det er lagt på en 5-er, kan det legges på en til.*
- $M/\langle 5, 5 \rangle \approx M/\langle 10 \rangle$
  *Maskinen skal oppføre seg på samme måte uavhengig av om det er lagt på to 5-ere eller en 10-er.*
- $M/\langle 10 \rangle \leftarrow a \ \vee \ M/\langle 10 \rangle \leftarrow b$
  *Maskinen gir enten en sjokoladeplate eller pengene tilbake som en indre ikke-determinisme.*

- $M \approx M/\langle 10, a\rangle \approx M/\langle 10, b\rangle$
  *Maskinen skal kunne fortsette å ta imot penger og gi tilbake pengene eller en sjokoladeplate i det uendelige.*

Vi ser lett at vi her ikke har sagt noe om hva som skal skje hvis brukeren av automaten først putter på en 5-er og så en 10-er. Vi har her flere fornuftige muligheter:

- Ikke tillate brukeren å legge på en 10-er: $M/\langle 5\rangle \nleftarrow 10$
- Automaten oppfører seg som om den ikke har fått noen 5-er: $M/\langle 5, 10\rangle \approx M/\langle 10\rangle$
- Automaten gir først tilbake 5-eren (hendelsen $c$), og fortsetter så ut fra at en 10-er er lagt på: $M/\langle 5, 10\rangle \leftarrow c, \quad M/\langle 5, 10, c\rangle \approx M/\langle 10\rangle$

Hvis vi ikke sier noe eksplisitt, blir det opp til den som skal implementere spesifikasjonen å bestemme hva som skal gjøres i en slik situasjon. Merk at det da også tillates "ufornuftige" løsninger, for eksempel at maskinen går i stå (deadlock), eller at det gis ti sjokoladeplater.

## 2 Relatert arbeid

Sammenlignet med modelleringsspråk som UML [6] og OCL [12], samt temporal logikk, UNITY [3] og TLA [7], unngår vi å snakke om variabler inne i objektene. Slike detaljer mener vi hører hjemme først på implementasjonsnivå. Ved bare å se på den observerbare oppførselen til objektene, oppnår vi en mer abstrakt, høynivå spesifikasjon.

I motsetning til FOCUS[2] har vi begrenset oss til å se på endelige sekvenser, slik det også er gjort i CSP [5]. Problemet med en slik begrensning er at det blir vanskeligere å snakke om deadlock for ikke-deterministiske objekter. Vi legger imidlertid merke til at når vi kun ser på de observerbare hendelsene, vil det i en enkelt eksekvering ikke være mulig å si om et gitt objekt er deterministisk eller ikke. Vi løser dermed dette problemet ved å se på et ikke-deterministisk objekt som mengden av dets deterministiske instanser.

Når det gjelder tilhørigheten av de ulike observerbare hendelsene, har vi valgt å lagre disse i en sekvens for hvert objekt, også kalt en prosess-trace. Dette er forskjellig fra tilnærmingen i FOCUS, der det er valgt å lagre en sekvens for hver kommunikasjonskanal, en såkalt kanal-trace. I [11] argumenteres det for at prosess-tracer er bedre enn kanal-tracer i den forstand at man da kan oppnå full uttrykkskraft for parallellsammensetning selv uten bruk av eksplisitt timing. Siden alt er i én trace, unngår vi problemer med merge og lignende. [11] viser også at prosess-tracer eller tilsvarende er nødvendig hvis man skal ha håp om å kunne oppnå kompletthet av formalismen.

Eksplisitt timing betyr at sekvensen må inneholde "ticks", eventuelt at man angir tidspunkt i forbindelse med hver hendelse. Begge deler gir mer detaljer og mindre abstrakte beskrivelser. En sekvens gir i seg selv relativ timing, og en spesifikasjon over en sekvens vil naturlig uttrykke partiell ordning i tid. Dette gir enklere spesifikasjoner.

Også CSP ser på prosess-tracer, og er dermed den av formalismene over som ligger nærmest den vi beskriver her. Det er imidlertid en viktig forskjell på de to formalismene: CSP fokuserer på såkalt failure-semantikk, det vil si hvilke hendelser et objekt på et visst tidspunkt ikke kan godta. Vi har den motsatte vinklingen ved at vi har en ready-relasjon som

angir hvilke hendelser et objekt kan godta. Vi mener en slik tankegang er mer intuitiv. CSP er heller ikke objekt-orientert, i den forstand at objekter ikke har identitet og at man kun kan snakke om ett objekt av gangen. For oss er det viktig å kunne snakke om flere objekter i samme formel for eksplisitt å få frem avhengigheter mellom dem.

## 3 Semantikk

Som tidligere forklart ser vi på et ikke-deterministisk objekt som unionen av dets deterministiske instanser. For ready-relasjonen ($\leftarrow$) innebærer dette at den må tolkes som samlingen av de tilsvarende indekserte relasjonene $\underset{i}{\leftarrow}, i : \mathcal{I}$, der $\mathcal{I}$ er et indekssett som fanger inn alle mulige eksekveringer. En formel $p$ skal dermed tolkes semantisk som $\forall i : \mathcal{I} \cdot p_i$,[1] der $p_i$ er lik $p$, men hvor alle forekomster av den syntaktiske $\leftarrow$ er byttet ut med den tilsvarende semantiske $\underset{i}{\leftarrow}$.

For et objekt $A$ skriver vi $[A]_i^*$ for å angi trace-settet til $A$ i eksekveringen $i$, det vil si de mulige sekvenser av hendelser $A$ vil akseptere i denne eksekveringen. Vi krever som vanlig (se for eksempel [5]) at hvert slikt trace-sett er prefiks-lukket og at det ikke inneholder hendelser utenfor alfabetet til $A$.

Vi sier at to objekter $A$ og $B$ med samme alfabet ($\alpha A = \alpha B$) er like hvis de har samme betydning, det vil si at de har nøyaktig samme trace-sett:[2]

$$A = B \triangleq \forall i \cdot [A]_i^* = [B]_i^* \tag{1}$$

Et objekt $A$ forfiner et annet objekt $B$ med samme alfabet hvis hvert mulige trace-sett for $A$ også er mulig for $B$ (men ikke nødvendigvis i samme eksekvering):

$$A \sqsubseteq B \triangleq \forall i \exists j \cdot [A]_i^* = [B]_j^* \tag{2}$$

Vi kan nå definere formelt operatoren $\approx$ fra avsnitt 1.1, som sier at $A$ og $B$ er observerbart like hvis alle mulige fortsettelser for en av dem også er mulige for den andre, og motsatt:

$$A \approx B \triangleq A \sqsubseteq B \wedge B \sqsubseteq A \tag{3}$$

Vi lar $h; h'$ stå for konkateneringen av sekvensene $h$ og $h'$. For objektkontinuasjoner (angitt ved $/$) kan vi da gi følgende regel:

$$h; h' \in [A]_i^* \Leftrightarrow h' \in [A/h]_i^* \tag{4}$$

som sier at hvis $A$ kan godta $h$ etterfulgt av $h'$, så kan $A$ etter $h$ godta $h'$ og omvendt.

[1] Vi bruker en syntaks for kvantorer, $\forall x : T \cdot$ og $\exists x : T \cdot$, som tillater oss å spesifisere typen til de variablene vi kvantoriserer over. Ofte vil typen være underforstått, og vi skriver da bare $\forall x \cdot$ og $\exists x \cdot$.

[2] $\triangleq$ står for "er definert til å være lik".

### *3.1 Det døde og det meningsløse objektet*

Syntaksen i formalismen inkluderer to objektkonstanter:

- $0_s$ - Det døde objektet (med alfabet $s$).
- $\perp_s$ - Det meningsløse objektet (med alfabet $s$).[3]

$0_s$ representerer objekter som er i deadlock, det vil si at de ikke kan komme videre med eksekveringen sin. De har dermed ingen mulige fortsettelser. Vi lar trace-settet bestå av den tomme sekvensen ($\varepsilon$) siden vi for sekvenser har $h;\varepsilon = h$, altså at vi ikke endrer historien til et objekt selv om vi utvider den med $\varepsilon$. Vi får:

$$[0_s]_i^* \triangleq \{\varepsilon\} \tag{5}$$

Det meningsløse objektet forekommer aldri i praksis, men det kan opptre i spesifikasjoner, for eksempel er $A/h \vdash x$ ($A$ etter sekvensen $h$ forlenget med hendelsen $x$) et meningsløst uttrykk hvis $A/h = 0_{\alpha A}$. Vi lar konstanten $\perp_s$ stå for slike meningsløse objekter. Akkurat som det døde objektet, har heller ikke $\perp_s$ noen meningsfulle fortsettelser. Heller ikke $\varepsilon$ vil gi oss noe meningsfylt, og det er dermed naturlig å si at $\perp_s$ har tomt trace-sett:

$$[\perp_s]_i^* \triangleq \emptyset \tag{6}$$

Ved å skille trace-settene til $\perp_s$ og $0_s$ på denne måten, følger $\perp_s \neq 0_s$ fra likhetsdefinisjonen gitt ved (1). Dette er hensiktsmessig da vi ønsker å se på det døde objektet som noe meningsfylt. (Vi kan jo havne i den situasjonen, det vil si i deadlock.)

### *3.2 Ready-relasjonen — Ny semantikk*

Vi er nå klare til å definere den ønskede ready-relasjonen. [9] har definert denne slik at $A \underset{i}{\leftarrow} x$ betyr det samme som $\langle x \rangle \in [A]_i^*$, og dermed at $A/h \underset{i}{\leftarrow} x$ betyr $h \vdash x \in [A]_i^*$. I arbeidet med RPC-eksempelet, så vi imidlertid at vi endte opp med veldig mange spesifikasjoner på formen

$$\forall h \cdot A/h \neq \perp \Rightarrow A/h \leftarrow x \tag{7}$$

I tillegg var en tilbakevendende feil underveis i arbeidet at vi intuitivt ikke tenkte på muligheten for $\perp$, og dermed istedenfor det korrekte gitt ved (7) bare skrev:

$$\forall h \cdot A/h \leftarrow x \tag{8}$$

Dette er ikke så rart, i og med at det vi spesifiserer er et virkelig system, og i praksis vil selvfølgelig det meningsløse objektet aldri kunne forekomme.

Jo mindre vi trenger å tenke på $\perp$ mens vi spesifiserer, jo bedre er det. Vi ønsket derfor å undersøke muligheten for å kunne lage spesifikasjonene etter mønster fra (8) ved å justere semantikken til $\leftarrow$. For at disse spesifikasjonene skulle være gyldige, måtte vi da la dem

[3] Vi utelater gjerne å angi det aktuelle alfabetet hvis dette enten er underforstått eller ikke relevant i den aktuelle sammenhengen.

automatisk være oppfylt for $A/h = \bot$. En positiv bieffekt er at spesifikasjonsformlene blir kortere, og dermed mer oversiktlige.

Selve ready-relasjonen ($\underset{i}{\leftarrow}$) defineres dermed semantisk ved:

$$A \underset{i}{\leftarrow} h \triangleq A \neq \bot \Rightarrow h \in [A]_i^* \tag{9}$$

For $\bot_s$ følger da

$$\forall h \cdot \bot_s \underset{i}{\leftarrow} h \tag{10}$$

Siden vi har definert $\underset{i}{\leftarrow}$ til å ta en sekvens på høyresiden, innfører vi også notasjonen $A \underset{i}{\leftarrow} x$ som en forkortelse for $A \underset{i}{\leftarrow} \langle x \rangle$.

I [9] er $\underset{i}{\leftarrow}$ kun definert til å ta enkelthendelser på høyresiden. Vi har imidlertid funnet det hensiktsmessig å kunne ha en hel sekvens der. Dette gjelder særlig for deterministiske sider ved objektene, hvor det er en fordel å kunne angi hele den påtvungne hendelsesrekkefølgen i samme formel. Ved å kunne angi en hel slik sekvens samtidig oppnår vi at spesifikasjonen som helhet blir kortere (færre enkeltformler), og den samlede lesbarheten blir også bedre. Hver enkelt formel kan bli lengre, men vil fortsatt være grei å lese. Hvis vi ikke hadde ønsket å endre profilen til $\leftarrow$, kunne vi innført sekvenser på høyresiden ved at $A \leftarrow \langle x_1, ..., x_n \rangle$ er det samme som

$$A \leftarrow x_1 \ \wedge \ A/\langle x_1 \rangle \leftarrow x_2 \ \wedge \ \ldots \ \wedge \ A/\langle x_1, x_2, \ldots, x_{n-1} \rangle \leftarrow x_n \tag{11}$$

I [9] er $\underset{i}{\leftarrow}$ definert både binært (som her) og unært, det vil si med tom høyreside. Semantikken til den unære var da gitt ved at $A/h \underset{i}{\leftarrow}$ skulle bety det samme som $h \in [A]_i^*$. Dermed betydde også $A \underset{i}{\leftarrow}$ at $A$ var et meningsfylt uttrykk. Korrespondansen mellom de to variantene av $\underset{i}{\leftarrow}$ var gitt ved at $A/\langle x \rangle \underset{i}{\leftarrow}$ betydde det samme som $A \underset{i}{\leftarrow} x$. Vi måtte dermed hele tiden velge hvilken av disse vi skulle bruke, uten å ha gitt noen retningslinjer for hvordan dette valget burde foretas. I den reviderte formalismen har vi derfor valgt å kutte ut den unære $\underset{i}{\leftarrow}$.

Vi kommer imidlertid veldig nær, ved at vi tillater $A \leftarrow \varepsilon$, som betyr $A \neq \bot \Rightarrow \varepsilon \in [A]_i^*$ som er triviell sann. Det hadde dermed ikke vært noe i veien for å tillate $A \leftarrow$, da med betydningen *true*, men det har vi ikke særlig nytte av.

Fra en spesifikasjon $A \leftarrow h$ kan man nå ikke utlede noe om at $A$ er et meningsfylt uttrykk. Hvis $A$ ikke er meningsfylt sier egentlig ikke spesifikasjonen noe som helst. For et gitt konstantuttrykk $K$ er det derfor ønskelig å kunne kreve/forutsette $K \neq \bot$, det vil si at $K$ er et meningsfylt uttrykk. Fra en slik antagelse vil vi kunne utlede at også andre objektuttrykk er meningsfulle. Har vi for eksempel konstantuttrykket $K$ (som per antagelse er meningsfylt) sammen med spesifikasjonen $K \leftarrow x$, vet vi da at $K/\langle x \rangle$ må være meningsfylt. (En slik forutsetning var tidligere ikke så interessant, da vi hadde $K \leftarrow x \Rightarrow K \neq \bot$ uansett.) For objektkonstanter krever vi derfor at de har et ikke-tomt trace-sett, dvs. $\varepsilon \in [K]_i^*$ for konstant $K$. Av dette følger $K \neq \bot$ som ønsket.

## 4 Aksiomer og bevisregler

For å kunne bevise egenskaper om objekter som beskrevet i forrige avsnitt, definerer vi her en logikk som er en utvidelse av første-ordens logikk med likhet. Utvidelsen består av aksiomene (B1)–(B9) og en bevisregel for likhet. Disse omtaler objekt-kontinuasjoner

og selve ready-relasjonen. I tillegg trenger vi aksiomer for sekvens-operatorer. (Se for eksempel [4].)

Aksiomer:

- (B1) $\vdash A/\varepsilon = A$
- (B2) $\vdash A/h/h' = A/h;h'$
- (B3) $\vdash A \leftarrow h \Rightarrow h \in (\alpha A)^*$
- (B4) $\vdash A \leftarrow h;h' \Leftrightarrow A \leftarrow h \ \wedge \ A/h \leftarrow h'$
- (B5) $\vdash A \leftarrow h \Rightarrow A = \bot \vee A/h \neq \bot$
- (B6) $\vdash \bot_s \leftarrow h$
- (B7) $\vdash \bot/h = \bot$
- (B8) $\vdash \forall x \cdot 0_s \nleftarrow x$
- (B9) $\vdash \forall x \cdot 0/\langle x \rangle \leftarrow h$

I bevisregelen for likhet må vi passe på at ingen av de involverte objektuttrykkene kan være lik det meningsløse objektet. For to objekter A og B med samme alfabet ($\alpha A = \alpha B$) får vi da:

$$\frac{\vdash \forall h \cdot A \leftarrow h \Leftrightarrow B \leftarrow h \quad \vdash A \neq \bot_{\alpha A} \quad \vdash B \neq \bot_{\alpha B}}{\vdash A = B} \, R=$$

### *4.1 Sunnhet*

Et bevissystem er *sunt* dersom alt som kan utledes i systemet er gyldig. [9] definerer gyldighet ved:

> En sekvent $\vdash p$ fra spesifikasjonsspråket er *gyldig i en modell* hvis $\forall i$ : $\mathcal{I} \cdot p_i$ er tilfredsstilt for alle verdier av frie variable, hvor $p_i$ er $p$ med alle forekomster av $\leftarrow$ erstattet med $\underset{i}{\leftarrow}$. En sekvent $\vdash p$ er *gyldig*, skrevet $\models p$, hvis den er gyldig i alle modeller for $p$.

Sunnhet vises generelt ved induksjon over lengden av utledninger. I basissteget må det vises at alle aksiomer er gyldige. For hver bevisregel må vi så vise at hvis premissen(e) er gyldig(e), så er også konklusjonen gyldig. Hvis vi i induksjonsskrittet antar at premissene kan utledes, gir induksjonshypotesen at de da også er gyldige.

*Sunnhet av aksiomene*

Generelt vises sunnhet av aksiomene ved å vise at de følger fra den underliggende semantikken (se avsnitt 3). Vi vil her gi en skisse av beviset for noen utvalgte aksiomer, de andre vises i [10].

Aksiom (B6): $\vdash \bot_s \leftarrow h$
Semantisk tolkes dette som $\vdash \forall i : \mathcal{I} \cdot \bot_s \leftarrow_i h$. Dette er (10) direkte.

Aksiom (B8): $\vdash \forall x \cdot 0_s \nleftarrow x$
Vi har at $0_s \neq \bot_s$ følger fra likhetsdefinisjonen (1) fordi de har forskjellige trace-sett. Da er $0_s \nleftarrow_i x$ ekvivalent med $x \notin [0_s]_i^*$ (ved (9)), som igjen følger fra definisjon (5).

Aksiom (B5): $\vdash A \leftarrow h \Rightarrow A = \bot \vee A/h \neq \bot$
Semantisk blir dette til: $\forall i : \mathcal{I} \cdot A \underset{i}{\leftarrow} h \Rightarrow A = \bot \vee A/h \neq \bot$.
Ved (9) har vi at $A \underset{i}{\leftarrow} h$ er det samme som $A \neq \bot \Rightarrow h \in [A]_i^*$. Denne implikasjonen kan vi på vanlig måte gjøre om til $A = \bot \ \vee \ h \in [A]_i^*$. For $A = \bot$ holder da høyresiden i aksiomet automatisk. Det gjenstår å se på tilfellet $A \neq \bot$: Ved (4) får vi at $h \in [A]_i^*$ gir $\varepsilon \in [A/h]_i^*$ siden $h$ også kan skrives $h;\varepsilon$. Siden vi også har $\varepsilon \notin [\bot]_i^*$ fra (6), kan vi dermed konkludere $A/h \neq \bot$ ved (1).

Merk at vi ikke har implikasjon mot venstre, siden $A/h \neq \bot$ bare sier at det finnes minst en eksekvering hvor $A/h$ har et ikke-tomt trace-sett, mens $A \leftarrow h$ krever at $h$ er med i trace-settet til $A$ (og dermed $\varepsilon$ med i trace-settet tl $A/h$) i alle mulige eksekveringer.

*Sunnhet av bevisregelen for likhet*

For å vise sunnhet av en bevisregel, må vi generelt vise at hvis premissen(e) er gyldig(e), så er også konklusjonen gyldig.

Altså antar vi at vi har et bevis for hver av premissene, det vil si at vi har et bevis for $\vdash \forall h \cdot A \leftarrow h \Leftrightarrow B \leftarrow h$, et bevis for $\vdash A \neq \bot_{\alpha A}$ og et bevis for $\vdash B \neq \bot_{\alpha B}$.
Ved induksjon antar vi at alle disse tre sekventene er gyldige, det vil si at
$\forall i : \mathcal{I}, h \cdot A \underset{i}{\leftarrow} h \Leftrightarrow B \underset{i}{\leftarrow} h$, $\forall i : \mathcal{I} \cdot A \neq \bot_{\alpha A}$ og $\forall i : \mathcal{I} \cdot B \neq \bot_{\alpha B}$ er tilfredsstilt for alle verdier av frie variable.

Ved (9) har vi dermed $\forall i : \mathcal{I}, h \cdot h \in [A]_i^* \Leftrightarrow h \in [B]_i^*$.
Siden denne gjelder for alle mulige $h$ er betingelsen for likhet gitt ved (1) oppfylt, og vi kan konkludere med at $\vdash A = B$ er gyldig.

## 5 Parallellsammensetning

Motivasjon: Vi ønsker at parallellsammensetning i størst mulig grad skal kunne tilsvare bruk av $\wedge$, blant annet ved at følgende aksiom holder for $A$ og $B$ med samme alfabet:

$$(A \parallel B) \leftarrow x \Leftrightarrow A \leftarrow x \ \wedge \ B \leftarrow x \tag{12}$$

For parallellsammensetning med $\bot$ har vi i utgangspunktet to muligheter:

- $(\bot \parallel B) = \bot$

  Intuitivt: Har vi først introdusert det meningsløse blir alt meningsløst uansett hva vi gjør.

  Dette er den varianten som er valgt i [9], og vi kan dermed bruke de samme semantiske definisjonene av $[A \parallel B]_i^*$ og $[A + s]_i^*$. Det viser seg at vi ikke kan beholde aksiom (12) slik det er, men at vi må kreve $(A \parallel B) \neq \bot$ som en tilleggsbetingelse:

  $$(A \parallel B) \neq \bot \Rightarrow ((A \parallel B) \leftarrow x \Leftrightarrow A \leftarrow x \ \wedge \ B \leftarrow x)$$

- $(\perp \parallel B) = B$

  Intuitivt: Vi glemmer det meningsløse, da vi ser på denne som uinteressant og uvesentlig i forhold til resten. Dette innebærer også at en parallellsammensetning er meningsløs hvis og bare hvis alle komponentene er meningsløse.

  Siden vi ønsker at aksiom (12) skal holde, må vi definere $[A \parallel B]_i^*$ slik at aksiomet holder på semantisk nivå. Semantisk har vi at venstresiden tilsvarer:

  $$(A \parallel B) \neq \perp \Rightarrow x \in [A \parallel B]_i^*$$

  mens høyresiden tilsvarer:

  $$(A \neq \perp \Rightarrow x \in [A]_i^*) \ \wedge \ (B \neq \perp \Rightarrow x \in [B]_i^*)$$

  Ved å se litt på dette, ser vi at for $A \neq \perp$ og $B \neq \perp$ ønsker vi dermed at $x \in [A \parallel B]_i^*$ skal være det samme som $x \in [A]_i^* \wedge x \in [B]_i^*$, det vil si $[A \parallel B]_i^* = [A]_i^* \cap [B]_i^*$. For $A = \perp$ eller $B = \perp$ ønsker vi derimot å beholde trace-settet til den komponenten som ikke er $\perp$. Siden $\perp$ har tomt trace-sett, får vi dermed $[A \parallel B]_i^* = [A]_i^* \cup [B]_i^*$. Denne kan også gjelde i tilfellet $A = B = \perp$, da vi her ønsker å ha $(A \parallel B) = \perp$ og dermed $[A \parallel B]_i^* = \emptyset$.

  Ut fra dette kan vi nå definere:
  $[A \parallel B]_i^* \triangleq$
  $\{h \mid A \neq \perp \ \wedge \ B \neq \perp \ \wedge \ h \in [A]_i^* \ \cap \ [B]_i^*\} \cup$
  $\{h \mid (A = \perp \vee B = \perp) \ \wedge \ h \in [A]_i^* \ \cup \ [B]_i^*\}$

Siden det andre alternativet er det som gir det ønskede aksiomet, samtidig som det virker greit intuitivt, er det foreløpig denne varianten vi har valgt å bruke. I [10] ser vi nærmere på disse to alternativene. Der ser vi også på hvordan det blir i det generelle tilfellet når $A$ og $B$ har ulikt alfabet.

## 6 Eksempel: En spilleautomat

Vi lar en spilleautomat $A$ ha de to hendelsene:

- $p(v)$ - spilleren putter på en mynt med verdi $v$ kroner.
- $g(v)$ - spilleren får en gevinst på $v$ kroner.

Ved hjelp av induktive definisjoner (se for eksempel [4]) kan vi spesifisere to hjelpefunksjoner som teller hvor mye penger som har gått inn og ut av automaten:

| | |
|---|---|
| $Inn(\varepsilon) == 0$ | $Ut(\varepsilon) == 0$ |
| $Inn(h \vdash p(v)) == Inn(h) + v$ | $Ut(h \vdash p(v)) == Ut(h)$ |
| $Inn(h \vdash g(v)) == Inn(h)$ | $Ut(h \vdash g(v) == Ut(h) + v$ |

Spilleautomaten kan da spesifiseres ved:

- $\forall v : Nat, h : seq \cdot A/h \leftarrow p(v)$
  *Automaten må alltid kunne akseptere å få mer penger.*

- $\forall h : seq, v : Nat\ \exists v' : Nat \cdot A/h \vdash p(v) \leftarrow g(v')$
  *Etter at spilleren har lagt på penger, må automaten alltid kunne gi melding om gevinst - men denne gevinsten kan være null.*

- $\forall v : Nat, h : seq \cdot v > Inn(h) - Ut(h) \Rightarrow A/h \nleftarrow g(v)$
  *Automaten kan ikke gi ut gevinster som er større enn nåværende overskudd.*

Her ser vi for eksempel at det første aksiomet ville vært usant med den opprinnelige semantikken, siden $A/\langle g(5)\rangle$ er et eksempel på et meningsløst objekt ($A$ kunne ikke gi ut noen ekte gevinst før det var puttet på noen penger). Med den nye semantikken er ikke dette lenger noe problem.

### *6.1 En spillehall*

I en spillehall har vi $n$ forskjellige spilleautomater, indeksert fra 1 til $n$. Hver automat $A_i$ har to hendelser, å motta penger ($p_i(v)$) og å gi gevinst ($g_i(v)$). Vi spesifiserer $A_i$ ved:

- $\forall v, h \cdot A_i/h \leftarrow p_i(v)$
  *Automaten må alltid kunne akseptere å få mer penger.*

- $\forall v, h \cdot A_i/h \leftarrow g_i(0)$
  *Automaten må alltid kunne gi melding om null gevinst.*

- $\forall v, h \cdot A_i/h \vdash p_i(v) \leftarrow g_i(2 * v)$
  *Etter å ha mottatt penger, skal automaten kunne gi det dobbelte som gevinst.*

I tillegg til alle automatene, spesifiserer vi en ansvarlig sjef $S$ som har ansvaret for at spillehallen som helhet går med minst 20% overskudd ($Inn$ og $Ut$ er som før, bare tilpasset $i$-indeksen ):

- $\forall v, h, i \cdot S/h \leftarrow p_i(v)$
  *Sjefen aksepterer at det puttes på penger på hvilken som helst maskin.*

- $\forall v, h, i \cdot S/h \leftarrow g_i(0) \vee S/h \leftarrow g_i(v)$
  *For alle mulige gevinstbeløp $v$, må sjefen enten kunne akseptere $v$ eller $0$ som gevinst.*

- $\forall v, h, i \cdot v > In(h) * 0.8 - Ut(h) \Rightarrow S/h \nleftarrow g_i(v)$
  *Gevinster er ikke tillatt hvis overskuddet til nå blir på mindre enn 20%.*

### *6.2 Spillehallen er en spilleautomat*

Vi kan nå vise at spillehallen med $n$ automater $A_i$ og en sjef $S$, implementerer den opprinnelige spilleautomaten $A$ (hvor aksiomene antas å stå for samlingen av de tilsvarende $i$ aksiomene). Denne implementasjonen kan vises for vilkårlig mange automater i spillehallen, men for enkelthets skyld viser vi det for en liten hall med bare en enkelt automat. Spillehallen kan da skrives $A_1 \parallel S$. Å vise at $A_1 \parallel S$ implementerer $A$ vil si å vise at $A_1 \parallel S$ tilfredsstiller de gitte $A$-aksiomene.

- Vis: $\forall v, h \cdot (A_1 \parallel S)/h \leftarrow p_1(v)$.
  Ved å bruke aksiomet for parallellsammensetning (12), får vi å vise:
  a) $\forall v, h \cdot A_1/h \leftarrow p_1(v)$ (trivielt)
  b) $\forall v, h \cdot S/h \leftarrow p_1(v)$ (trivielt)

- Vis: $\forall h, v\ \exists v' \cdot (A_1 \parallel S)/h \vdash p_1(v) \leftarrow g_1(v')$
  1) Spesifikasjonen av $A_1$ gir $A_1/h \vdash p_1(v) \leftarrow g_1(0)$ og $A_1/h \vdash p_1(v) \leftarrow g_1(2 * v)$, altså er $0$ og $2 * v$ kandidater til $v'$.
  2) Spesifikasjonen av $S$ gir enten $S/h \vdash p_1(v) \leftarrow g_1(0)$ eller
  $S/h \vdash p_1(v) \leftarrow g_1(2 * v)$, og sikrer dermed at $v'$ kan være enten $0$ eller $2 * v$.

- Vis: $\forall v, h \cdot v > Inn(h) - Ut(h) \Rightarrow (A_1 \parallel S)/h \nleftarrow g_i(v)$
  For å vise at en parallellsammensetning *ikke* er klar for en hendelse, er det nok å vise at hendelsen ikke er mulig for en av komponentene. For $v > Inn(h) - Ut(h)$, har vi opplagt også $v > Inn(h) * 0.8 - Ut(h)$, og spesifikasjonen av $S$ gir da $S/h \nleftarrow g_1(v)$.

Med flere automater i spillehallen får vi essensielt det samme beviset repetert for hver automat siden ingen hendelse er felles for mer enn to komponenter (sjefen og den aktuelle automaten).

## 7 Oppsummering

Vi har tatt utgangspunkt i formalismen som beskrevet i [9]. Ved en justering av semantikken har vi fått både kortere og mer intuitive spesifikasjoner. Denne justeringen har delvis gått på bekostning av noen av aksiomene/bevisreglene, da vi i større grad enn tidligere har måttet spesialbehandle det meningsløse objektet ($\perp$). Regelen for parallellsammensetning er imidlertid like pen som før, men da på bekostning av en litt mer komplisert semantikk for parallellsammensetning.

I en del tilfeller fører dette til noe lengre bevis, både på semantisk og syntaktisk nivå, men tilfellene med $\perp$ blir stort sett veldig greie. Etter vår mening veier forenklingen av selve spesifikasjonene opp for merarbeidet.

Mange av problemene har oppstått fordi vi behandler $\perp$ ulikt på semantisk og syntaktisk nivå. Egentlig sier trace-sett og ready-relasjonen det samme, nemlig hvilke mulige fortsettelser et gitt objekt har. For $\perp$ endret vi derimot dette slik at trace-settet anga ingen mulig fortsettelse, mens ready-relasjonen tillot alle mulige fortsettelser. Et alternativ kunne vært også å endre trace-settet til $\perp$, slik at denne ble universalmengden istedenfor den tomme mengden. Vi ønsket ikke en slik variant, da det ville innebære likhet mellom det meningsløse objektet og et meningsfylt objekt som alltid var klart for alle hendelsene i alfabetet.

Siden endringen av semantikken er motivert ut fra arbeidet med et konkret eksempel, vil det bli interessant å se om endringen viser seg å være like nyttig i andre eksempler.

Vi har i denne artikkelen ikke gått inn på aspekter som for eksempel liveness. I øyeblikket ser vi ingen grunner til at dette skulle være spesielt vanskelig, og det skulle kunne fanges inn ved “mange-stegs readiness” på samme måte som i [9]. Foreløpig har vi heller ikke sett noe på (relativ) kompletthet av bevissystemet.

## Referanser


[1] Manfred Broy og Leslie Lamport. The RPC-memory specification problem - problem statement. *Lecture Notes in Computer Science*, 1169:1–4, 1996.

[2] Manfred Broy og Ketil Stølen. Specification and development of interactive systems; FOCUS on streams, interfaces, and refinement. Planlagt utgitt på Springer, utkast februar 2000.

[3] K. Mani Chandy og Jayadev Misra. *Parallel Program Design : a Foundation*. Addison-Wesley, 1988.

[4] Ole-Johan Dahl. *Verifiable Programming*. C.A.R. Hoare Series. Prentice Hall International, 1992.

[5] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.

[6] Martin Fowler with Kendall Scott. *UML Distilled : a Brief Guide to the Standard Object Modeling Language*. Object Technology Series. Addison-Wesley, 1999.

[7] Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

[8] Olaf Owe. Specifying concurrency by generalized ready relations. Research Report 176, Department of Informatics, University of Oslo, Norway, 1993.

[9] Olaf Owe. A specification formalism for interacting objects. *Proceedings of the Estonian Academy of Sciences: Physics, Mathematics*, 47(3):198–215, 1998.

[10] Ragnhild Kobro Runde. Hovedoppgave. Institutt for informatikk, Universitetet i Oslo. Under utarbeidelse.

[11] Neelam Soundarajan. Communication traces in the verification of distributed programs. I D. Duke og A. Evans, redaktører, *BCS-FACS Northern Formal Methods Workshop (NFM-97)*, Ilkley, UK, september 1997.

[12] Jos B. Warmer og Anneke G. Kleppe. *The Object Constraint Language : Precise Modeling with UML*. Object Technology Series. Addison-Wesley, 1999.