Ajastettu hajautettujen jaettujen olioiden eheys

Jan Lindström jplindst@cs.helsinki.fi

20. helmikuuta 2001

Perustuu artikkeliin: Torres-Rojas, F. J., Ahamad, M.ja Raynal, M., Timed Consistency for Shared Distributed Objects, Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing, May 3-6, 1999, Atlanta, Georgia, USA. ACM, 1999.

1 Tapahtumat ja tietokannan eheys

- Tietokanta on joukko nimettyjä tietoalkioita, joilla jokaisella on arvo. Tietokannan tila on tietoalkioiden arvot tiettynä hetkenä. Tietoalkio voi olla esim relaatio, monikko tai sivu.
- Tietokanta on *ehyt* jos tietokannan jokainen eheysehto on tosi. Yleinen oletus: tietokannan alkutila on ehyt ja yksittäiset tapahtumat säilyttävät tietokannan eheyden.
- Tapahtuma on tietokantaa käsittelevän ohjelman suoritus.
- Kun kahta tai useampaa tapahtumaa suoritetaan samanaikaisesti voidaan minkä tahansa tapahtuman kahden operaation välissä suorittaa toisen tapahtuman operaatioita.

- Suoritus on siis lomittunut. Kaikki tapahtumien operaatioiden lomittumiset eivät säilytä tietokannan eheyttä.
- Samanaikaisuudenhallinta varmistaa ettei samanaikaiset tapahtuman suoritukset häiritse toisiaan.
- Toipumisessa varmistetaan että tapahtuman suorituksen tulee olla atominen.
- *Toipuvassa suorituksessa* (recoverable execution) tapahtuma ei voi sitoutua ennenkuin tapahtumat, joiden kirjoittaman arvon se on lukenut, ovat sitoutuneet.
- Suoritus on *ketjuperuutukset estävä*, jos tapahtumat lukevat vain sitoutuneiden tapahtumien kirjoittamia arvoja.
- Suoritus on *alkuarvoketjut estävä*, jos tapahtumat kirjoittavat uudelleen vain sitoutuneiden tapahtumien kirjoittamia arvoja.
- Suoritusta sanotaan *striktiksi* (strict), jos se on ketjuperuutukset ja alkuarvoketjut estävä.
- Suoritusta sanotaan *sarjalliseksi* (serial), jos jokaiselle tapahtumaparille toisen tapahtuman kaikki operaatiot suoritetaan ennen toisen tapahtuman yhtäkään operaatiota. Sarjallinen suoritus säilyttää tietokannan eheyden, koska jokainen tapahtuma oletetaan oikeaksi ja sarjallisessa suorituksessa tapahtumat eivät voi lomittua virheellisesti.
- Sarjallistuvien suoritusten perusoletus on että yksittäiset tapahtumat ovat virheettömiä. Tällöin tapahtumien perättäiset eli sarjalliset suoritukset ovat virheettömiä (tapahtumat eivät häiritse toinen toisiaan). Lisäksi suoritukset, joilla on sama

vaikutus kuin sarjallisilla suorituksilla (= sarjallistuvat suoritukset) ovat virheettömiä.

- *Ajoittaja* on ohjelma, joka kontrolloi samanaikaisten tapahtumien suoritusta määräämällä järjestyksen, jossa tiedohallitsija suoritaa tietokantaoperaatiota (begin, read, write, commit ja abort). Ajoittaja myös varmistaa, että suoritus on sarjallistuva ja toipuva. Ajoitukseen voi vaikuttaa:

    1. Suorittamalla vastaanotettu operaatio välittömästi.
    2. Hylkäämällä operaatio, joka aiheuttaa vastaavan tapahtuman keskeytyksen.
    3. Viivästyttää operaation suoritusta.

- Ajoittaja ei tiedä mitään tapahtumien semantiikasta. Se näkee tapahtuman vain begin, read, write, commit, ja abort operaatioiden jonona.

- Sarjallistuvuusteoria tarjoaa matemaattisen välineistön ajoittajien oikeellisuuden tutkimiseen.

- Sarjallistuvuusteoriassa tapahtumien samanaikaiset suoritukset esitetään historian avulla.

- Historia on sarjallistuva jos se kuvaa sarjallistuvaa suoritusta.

- Sarajallistuvuusteoriassa määritellään ehdot, joiden vallitessa historia on sarjallistuva. Oletetaan, että mikään tapahtuma ei lue eikä kirjoita samaa tietoalkiota kuin yhden kerran.

- Tapahtuma voidaa esittää osittain järjestettynä joukkona. Yleisesti osittainen järjestys määritellään parina $(\Sigma, \prec)$, missä Σ on joukko ja $\prec$ on transitiivinen binäärirelaatio joukossa Σ.

- Osittainen järjestys $L = (\Sigma, \prec)$ voidaan esittää suunnattuna syklittömänä verkkona $G = (N, E)$, missä $N = \Sigma$ ja $(a, b) \in E$ joss $a \prec b$.

- MÄÄRITELMÄ 1.1 *Tapahtuma T_i on järjestetty pari $(T_i, \prec_i)$, missä :*

  1. *$T_i \subseteq \{r_i[x], w_i[x] \mid x$ on tietoalkio $\} \cup \{a_i, c_i\}$;*
  2. *$a_i \in T_i$ joss $c_i \notin T_i$;*
  3. *jos $t = c_i$ tai $t = a_i$, niin $\forall p \in T_i (p \neq t)$, $p \prec_i t$; ja*
  4. *jos $r_i[x], w_i[x] \in T_i$, niin joko $r_i[x] \prec_i w_i[x]$ tai $w_i[x] \prec_i r_i[x]$.* $\Box$

- Toisin sanoen

  1. Tapahtuma muodostuu read, write, commit ja abort operaatioista.
  2. Tapahtumassa on joko abort tai commit operaatio.
  3. Tapahtuman viimeinen operaatio on joko abort tai commit.
  4. Jos luku ja kirjoitusoperaatio kohdistuu samaan tietoalkioon, niin niillä täytyy olla täydellinen järjestys.

- Historiat mallintavat samanaikaisten tapahtumien suoritusta. Historiat ovat osittaisjärjestyksiä, joilla on ehdot:

  1. Jos kahdella tapahtuman T_i operaatiolla on järjestys T_i:ssä, niin kaikissa historioissa, joihin T_i sisältyy täytyy kyseisillä operaatioilla olla sama järjestys.

2. Jos kaksi operaatiota ovat hisroriassa konfliktissa, niin niillä täytyy olla järjestys. Kaksi operaatiota ovat konfliktissa, jos ne kohdistuvat samaan tietoalkioon ja ainakin toinen operaatioista on kirjoitusoperaatio.

- MÄÄRITELMÄ 1.2 *Olkoon* $T = \{T_1, T_2, ..., T_n\}$ *joukko tapahtumia.* T*:n täydellinen historia* H *on osittaisjärjestys* $(H, \prec)$*, missä*

  1. $H = \bigcup_{i=1}^{n} T_i$;
  2. $\prec_H \supseteq \bigcup_{i=1}^{n} \prec_i$; *and*
  3. *Jos* $p, q \in H$*, ovat konfliktissa niin, joko* $p \prec_H q$ *tai* $q \prec_H p$. □

- Ehdolla (1) määritellään että historian H suoritus koostuu täsmälleen tapahtumien $T_1, T_2, ..., T_n$ sisältämistä operaatioista. Ehto (2) määrittelee, että historian operaatioiden järjestys noudattaa tapahtumissa määriteltyä operaatioiden järjestystä. Ehto (3) määrittelee, että konfliktoivilla operaatioilla on historiassa täydellinen järjestys.

- MÄÄRITELMÄ 1.3 *Olkoon* $L = (\Sigma, \prec)$ *ja* $L^{'} = (\Sigma^{'}, \prec^{'})$ *osittaisjärjestyksiä.* $L^{'}$ *on* L*:n rajoittuma joukossa* $\Sigma^{'}$ *jos* $\Sigma^{'} \subseteq \Sigma$ *ja* $\forall a, b \in \Sigma^{'} : a \prec^{'} b$ *joss* $a \prec b$. □

- MÄÄRITELMÄ 1.4 $L^{'}$ *on* L*:n prefiksi (merk* $L^{'} \preceq L$*), jos* $L^{'}$ *on* L*:n rajoittuma ja* $\forall a \in \Sigma^{'}$*: kaikki* a*:n edeltäjät* L*:ssä (*$\forall b \in \Sigma$ *s.e.* $b \prec a$*) ovat myös* $\Sigma^{'}$*:ssa.*□

- MÄÄRITELMÄ 1.5 *Historia on täydellisen historian prefiksi.* □

- MÄÄRITELMÄ 1.6 *Tapahtuma T_i on sitoutunut (keskeytynyt) historiassa H jos $c_i \in H$ ($a_i \in H$).*$\square$

- MÄÄRITELMÄ 1.7 *Tapahtuma T_i on aktiivinen historiassa H jos se ei ole keskeytynyt eikä sitoutunut.* $\square$

- Suoritus on sarjallistuva, jos se on ekvivalentti sellaisen sarjallisen suorituksen kansa joka muodostuu samoista tapahtumista. Lähestymistapa sarjallistuviin historioihin:

  1. Määritellään ehdot, jolloin kaksi historiaa ovat ekvivalentit,
  2. määritellään ehto, jolloin historia on sarjallinen, ja
  3. määritellään historia sarjallistuvaksi jos se on ekvivalentti jonkin sarjallisen historian kanssa.

- MÄÄRITELMÄ 1.8 *Historiat H ja $H^{'}$ ovat (konflikti)ekvivantetit jos*

  1. *Ne muodostuvat samoista tapahtumista ja samoista operaatioista*
  2. *jos $P_i \in T_i \wedge q_j \in T_j$, $a_i, a_j \notin H$ ja p_i ja q_j ovat konfliktissa, niin $p_i \prec_H q_j \Rightarrow p_i \prec_{H'} q_j$. Eli sitoutuneiden ja aktiivisten tapahtumien konfliktoivat operaatiot ovat samassa järjestyksessä.*$\square$

- MÄÄRITELMÄ 1.9 *Historia H on (konflikti)sarjallistuva, jos sen sitoutunut projektio $C(H)$ on (konflikti)ekvivalentti jonkin sarjallisen historian kanssa.* $\square$

- Sarjallistuvuusverkon avulla voidaan päätellä onko historia (konflikti)sarjallistuva. Historian H sarjallistuvuusverkkoa merkitään $SG(H)$:lla.

- $SG(H)$ on suunnattu verkko. Sen solmut ovat ne tapahtumat, jotka ovat sitoutuneet H:ssa ja nuolet ovat kaikki $T_i \to T_j$ s.e. jokin T_i:n operaatio edeltää ja on konfliktissa jonkin T_j:n operaation kanssa.

- TEOREEMA 1.1 *(Sarjallistuvuusteoreema): Historia H on sarjallistuva jos ja vain jos $SG(H)$ on syklitön.*

  Todistus*: ($\Leftarrow$) Olkoon H muodostunut tapahtumajoukosta $\{T_1, T_2, ..., T_n\}$. Olkoon $\{T_1, T_2, ..., T_m\}(m \leq n)$ niiden tapahtumien joukko, jotka ovat sitoutuneet H:ssa. Siis $\{T_1, T_2, ..., T_m\}$ ovat $SG(H)$:n solmut. Koska $SG(H)$ on suunnattu syklitön verkko, sillä on topologinen lajittelu. Olkoon $i_1, ..., i_m$ lukujen $1, ..., m$ permutaatio s.e. $T_{i_1}, ..., T_{i_m}$ on $SG(H)$:n topologinen lajittelu. Väite: $C(H) \subseteqq H_s$. Olkoon $p_i \in T_i$ ja $q_j \in T_j$ ja T_i ja T_j ovat sitoutuneet H:ssa. Oletetaan, että p_i ja q_j ovat konfliktissa ja $p_i \prec_H q_j$. Tällöin $T_i \to T_j$ on $SG(H)$:n nuoli. Siis T_i edeltää T_j:tä kaikissa $SG(H)$:h topologisissa lajitteluissa. Nyt historiassa G_s kaikki T_i:n operaatiot edeltävät jokaista T_j:n operaatiota. Siis kaikille $C(H)$:n konfliktoiville operaatioille on voimasa se, että niillä on sama järjestys H_s:ssä. Siis $C(H) \subseteqq H_s$.*
  ($\Rightarrow$) Oletetaan että historia H on sarjallistuva. Olkoon H_s sarjallisen historia s.e $C(H) \subseteqq H_s$. Olkoon $T_i \to T_j$ $SG(H)$:n nuoli. Tällöin $\exists p_i$ ja q_j s.e. $p_i \prec_H q_j$. Koska $C(H) \subseteqq H_s$, niin $p_i \prec_{H_s} q_j$. Siis T_i edeltää T_j:tä H_s:ssä. Antiteesi: Oletetaan että $SG(H)$ on syklinen. Olkoon sykli $T_1 \to T_2 \to ... \to T_k \to T_1$. Tällöin $T_1 \prec_H T_2 \prec_H ... \prec_H T_k \prec_H T_1$. Eli siis $T_1 \prec_H T_1$, joka on ristiriita, joten $SG(H)$ ei voi olla syklinen. $\Box$

2 Hajautetut tapahtumat

- *Globaali historia* H hajautetussa järjestelmässä on osittaisjärjestysjoukko kaikista tietokantaoperaatioista, jotka on suoritettu hajautetussä järjestelmässä.

- H_i on operaatioiden järjestys solmussa i. Tälöin jos $a, b \in H_i$ ja $a \prec_{H_i} b$ niin a edeltää b:tä ohjelman suoritusjärjestyksessä.

- *Efektiivinen operaation suoritusaika* on operaation alun ja lopun välinen aika (oletetaan se positiiviseksi nollasta eroavaksi kokonaisluvuksi).

- Jos a:n efektiivinen aika on t, niin tällöin a suoritettiin ajanhetkellä t.

- Jos D on joukko operaatioita, niin D:n *serialisointi* (serialization) on lineaarinen sarja S sisältäen D:n kaikki operaatiot siten, että jokainen lukuoperaatio palauttaa kyseisen olion viimeisimmän sitoutuneen arvon.

- Olkoon $\sim$ osittaisjärjestysrelaatio joukossa D. Tällöin S:n serialisointi noudattaa $\sim$:ä jos $\forall a, b \in D$ s.e. $a \sim b$ ja pätee että a edeltää b:tä S:ssä.

- Historia H on *lineaariserialisointi* (linearizability) (LIN) jos on olemassa serialisointi, joka noudattaa operaatioiden efektiivisiä suoritusaikoja.

- Strikti sarjallistuvuus on määritelty historian avulla, joka muodostuu tapahtumista, ja edellyttää serialisoinnin olemassaoloa, joka noudattaa operaatioiden tosiaikajärjestystä (real-time order).

- LIN voidaan nähdä striktin sarjallistuvuuden erikoitapauksena, missä jokaisella tapahtumalla on ennaltamäärätty operaatio yhteen olioon.

- MÄÄRITELMÄ 2.1 *Syysuhderelaatiossa (causality relation) $\rightarrow$ viestinvälitysjärjestelmälle operaatioiden järjestystä H:ssa voidaan muuttaa. Olkoon $a, b, c \in H$. Tällöin $a \rightarrow b$, eli a syysuhdeellisesti edeltää b:tä, jos yksi seuraavista ehdoista on voimassa:*

  1. *a ja b on suoritettu samassa solmussa ja a on suoritettu ennen B:tä.*
  2. *b lukee a:n kirjoittaman olion*
  3. *$a \rightarrow c$ ja $c \rightarrow b$.*$\square$

- Kaksi erillistä operaatiota a ja b ovat *samanaikaisia* (concurrent) jos yksikään edellisistä ehdoista ei ole voimassa.

- Olkoon H_{i+w} joukko operaatioita H_i:stä lisättynä kaikki kirjoitusoperaatiot H:sta. Historia H täyttää *syysuhde-eheyden* (causal consistency, CC) jos jokaisessa solmussa i on olemassa serialisointi H_{i+w}:lle, joka noudattaa syysuhdejärjestystä $\rightarrow$. Eli, jos $a, b, c \in H$ ovat sellaisia että a kirjoittaa arvon v olioon x, c lukee saman arvon v oliosta x ja b kirjoitaa arvon v' olioon x, niin ei päde että $a \rightarrow b \rightarrow c$.

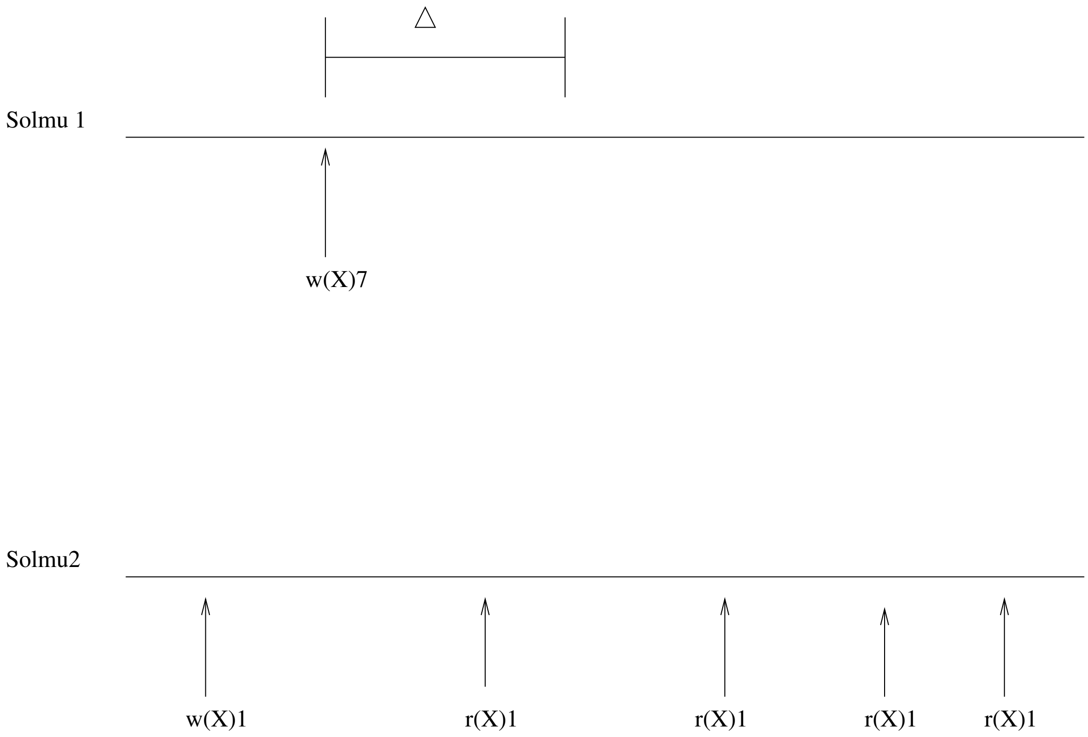
- CC olettaa että syysuhteessa olevat operaatiot nähdään jokaisessa solmussa samassa järjestyksessä vaikka paikalliset solmut havaitsevat samanaikaiset operaatiot eri järjestyksessä.

3 Ajastettu eheys

- Ajastetuissa malleissa joukko arvoja, jotka lukuoperaatio voi palauttaa on rajoitettu edellisen kirjoitusoperaatiosta kuluneeseen aikaan. Oletetaan aluksi että kellot ovat tarkasti synkronoitu.

- Lukuoperaatio on ajastettu jos operaatio ei palauta tietokannan tilaa jos on olemassa tietokannan tila, joka on ollut voimassa enemmän kuin Δ aikayksikköä.

- Kuvan 1 suoritus on SC ja CC, mutta ei LIN. Toiseen operaatioon asti solmussa 1 suoritus nudattaa ajastettua suoritusta. Samassa pisteessa LIN ehtoja rikotaan. Tämän pisteen jälkeen suoritus ei ole edes ajastettu, koska lukuoperaatio solmussa 1, joka alkaa Δ aikayksikköä kirjoituksen jälkeen ei palauta uusinta arvoa.



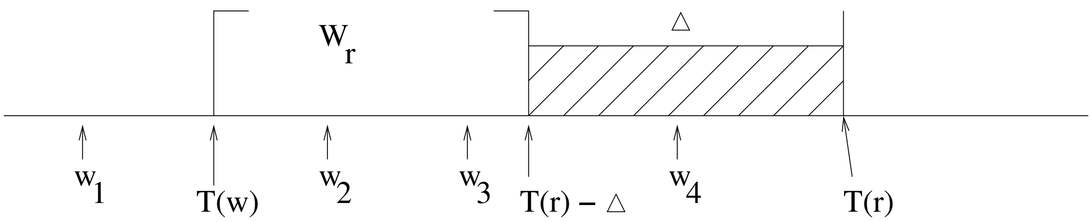
Kuva 1: Ei ajastettu sarjallisesti eheä (SC) suoritus

- MÄÄRITELMÄ 3.1 *Olkoon* $D \subseteq H$ *joukko operaatioita ja* S *D:n serialisointi. Olkoon* $w, r \in D$ *siten, että* w *kirjoitaa arvon olioon* X*, jonka* r *lukee myöhemmin. Tällöin :*

$$W_r = \{w^{'} \in D : (w^{'} \textit{ kirjoittaa arvon olioon } X) \\ \wedge (T(w) < T(w^{'}) < (T(r) - \delta))\}$$

Operaatio r *tapahtuu tai on ajastettu serialisoinnissa* S*, jos* $W_r = \emptyset$. S *on ajastettu jos jokainen lukuoperaatio* S*:ssä tapahtuu ajoissa.* $\square$

- Kuva 2 esittää Määritelmän 3.1 eri tapauksia. Operaatio r lukee w:n aikaisemmin kirjoittaman arvon. Koska operaatio w_1 suoritetaan ennen w:tä ei sillä ole vaikutusta operaatioon r. Vastaavasti, vaikkakin w_4 suoritetaan myöhemmin kuin w, operaatio r lukee vanhan arvon, koska ei ole kulunut Δ aikayksikköä w operaatiosta. Toisaalta operaatiot w_2 ja w_3 on suoritettu operaation w jälkeen ja kirjoitetut arvot ovat olleet nähtävillä yli Δ ajan kun operaatiota r suoritetaan. Joten w_2 ja w_3 ovat W_r:ssä ja siksi operaatio r ei ole ajastettu. Harmaa alue kuvassa kuvaa aikaväliä, jonka aikana W_r Määritelmän 3.1 mukaan pitää olla tyhjä jos r on ajastettu. Eli yhtään kirjoitusoperaatiota ei saa tapahtua samaan tietoalkioon tänä aikavälinä.



Kuva 2: Operaatio r ei ole ajastettu.

- Hajautetussa järjestelmässä kellot eivät pysy täsmälleen synkronissa vaan niiden välillä on pientä heittoa. Edellä esitetty malli voidaan laajentaa myös lähes synkronissa olevien kellojen tapaukseen. Oletetaan että tietyn solmun kello ei heitä todellisesta ajasta enempää kuin $\epsilon/2$ aikayksikköä.

- Jos tietyn operaation a suoritusajaksi jokin solmu ilmoittaa $T(a)$ tarkoittaa tämä aikapalvelimen kannalta aikaväliä $[T(a)-\epsilon/2, T(a)+\epsilon/2]$.

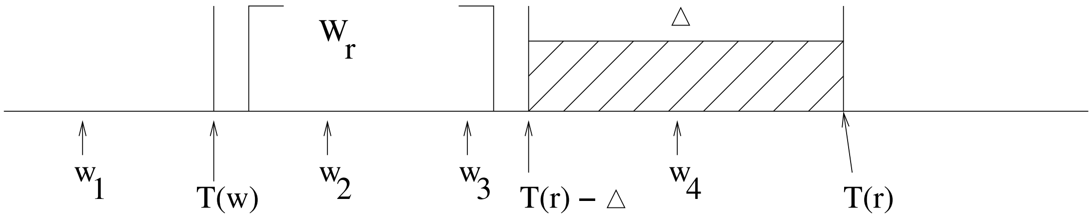
- MÄÄRITELMÄ 3.2 *$\forall a, b \in H$, a tapahtuu ennen kuin b, jos $T(a)+\epsilon/2 < T(b)-\epsilon/2$ tai yhtäpitävästi jos $T(a)+\epsilon < T(b)$.* □

- MÄÄRITELMÄ 3.3 *Olkoon $D \subseteq H$ joukko operaatioita ja S D:n serialisointi. Olkoon $w, r \in D$ siten, että w kirjoittaa arvon olioon X, jonka r lukee myöhemmin. Tällöin :*

$$W_r = \{w^{'} \in D : (w^{'} \textit{ kirjoittaa arvon olioon } X)$$
$$\wedge (T(w) + \epsilon < T(w^{'}) \wedge (T(w^{'}) + \epsilon < T(r) - \delta))\}$$

Operaatio r tapahtuu tai on ajastettu serialisoinnissa S, jos $W_r = \emptyset$. S on ajastettu jos jokainen lukuoperaatio S:ssä tapahtuu ajoissa. □

- Kuvassa 3 on esimerkki tästä määritelmästä.



Kuva 3: Operaatio r ei ole ajastettu.

4 TSC ja TCC

- Yhdistään yheysmallien SC ja CC vaatimukset ajastetun lukemisen vaatimuksella.
- MÄÄRITELMÄ 4.1 *Historia H on ajastetusti sarjalliseheä (TSC) jos on olemassa ajastettu H:n serialisointi S, joka noudattaa ohjelman järjestystä jokaisessa solmussa.* $\square$
- MÄÄRITELMÄ 4.2 *Historia H on ajastetusti syysuhde-eheä (TCC) jos jokaiselle solmulle i on olemassa ajastettu H_{r+w}:n serialisointi S, joka noudattaa syysuhdejärjestystä $\rightarrow$.* $\square$