Het Alternating Bit Protocol met Time Out in Discrete Tijd

Martijn Kaart
Vakgroep Programmatuur,
Kruislaan 403, 1098 SJ Amsterdam,
Universiteit van Amsterdam
E-mail: **kaart@fwi.uva.nl**

Indra Polak
Vakgroep Programmatuur,
Kruislaan 403, 1098 SJ Amsterdam,
Universiteit van Amsterdam
E-mail: **ipolak@fwi.uva.nl**

Abstract

We specificeren en verifiëren een variant van het Alternating Bit Protocol waarbij een bovengrens aan het aantal hertransmissies wordt vastgelegd. Zowel in gewone Process Algebra ACP als in de variant met discrete tijd ACP_{dt} worden specificaties gegeven. In het discrete tijd geval gebruiken we temporele logica voor het bewijs.

1 Inleiding

Voor het vak Real Time Process Algebra dienen wij een werkstuk te schrijven over een veel bestudeerd communicatieprotocol: het *Alternating Bit Protocol*. Het is de bedoeling om dit protocol, beschreven in 'Process Algebra', J.C.M. Baeten & W.P. Weijland ([1], blz. 108), uit te breiden met een zogenaamde *time out*, om zodoende een bovengrens te stellen aan het maximaal aantal transmissies. Tevens dient dit protocol te worden gedefiniëerd in discrete tijd (Discrete Time Process Algebra).

In hoofdstuk 2 zal het basisprincipe van het Alternating Bit Protocol uiteen worden gezet, met de vier componenten: zender, ontvanger en twee (onbetrouwbare) kanalen. De specificatie van dit 'basisschema' zal in hoofdstuk 3 worden uitgebreid met een time out. De uitwerking van de verkregen specificatie, alsmede een correctheidsbewijs (verificatie) is, gezien de lengte, als appendix A aan dit werkstuk toegevoegd.

Vervolgens zal in hoofdstuk 4 de in hoofdstuk 3 verkregen specificatie worden omgezet naar discrete tijd; de zender houdt nu niet meer het exacte aantal zendingen van een datum bij, maar regelt het aantal hertransmissies met behulp van de tijd. De specificatie uit hoofdstuk 4, het protocol in discrete tijd, zal aannemelijk worden gemaakt met behulp van temporele logica.

Als extra feature zal een *disable* aan het Alternating Bit Protocol in discrete tijd worden toegevoegd. Deze zal ervoor zorgen dat het protocol 'stopt' wanneer enige tijd geen data- transmissies hebben plaatsgevonden.

Het werkstuk eindigt met een aantal bevindingen en conclusies in hoofdstuk 6 en een lijst van geraadpleegde literatuur, te vinden in hoofdstuk 7.

Graag willen we bij dezen J.A. Bergstra en F.W. Vaandrager bedanken voor hun nuttige bijdragen aan dit werkstuk.

2 Basisschema Alternating Bit Protocol

Een *communicatieprotocol* behelst het verzenden van data over een onbetrouwbaar kanaal, zodanig dat –ondanks deze onbetrouwbaarheid– er geen gegevens verloren gaan.

Een veel bestudeerd communicatieprotocol is het *Alternating Bit Protocol*.

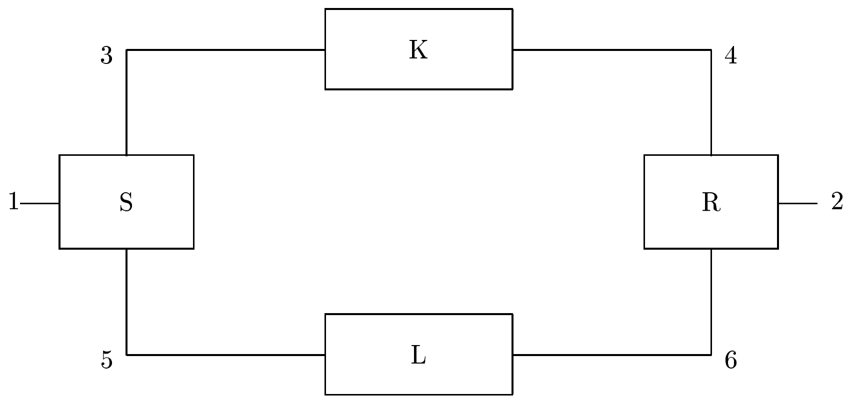

Fig. 2.1 - Alternating Bit Protocol

Het Alternating Bit Protocol bestaat uit vier componenten:

1. **component S: Sender.**
   S begint met het lezen van een data element via poort 1. Vervolgens wordt dit datum met een bijbehorend *control bit*, tezamen een *frame* genoemd, via poort 3 over kanaal K verstuurd. Elk frame wordt net zolang verstuurd, totdat een bevestiging (*acknowledgement*) via kanaal L bij S is aangekomen. (Als er een frame d0 is verstuurd, dan verwacht de zender waarde 0 en als er een frame d1 is verstuurd, dan verwacht de zender waarde 1 terug over kanaal L.)

   Is via poort 5 een juist acknowledgement binnengekomen, dan wordt een volgend data-element gelezen van poort 1. Dit zojuist gelezen datum wordt als frame over kanaal K verstuurd met een controlebit, welke de inverse waarde heeft van de laatst gebruikte boolean.

2. **component R: Receiver.**
   R leest een binnengekomen frame, bestaande uit een datum d en een boolean b. Als het control bit b de juiste is, dan wordt dit datum via poort 2 doorgestuurd als dit al niet eerder is gebeurd (gegevensoverdracht) en een acknowledgement wordt over kanaal L teruggestuurd naar de zender S.

   Receiver R blijft net zolang in een bepaalde toestand totdat een frame correct is aangekomen.

3. **component K: Unreliable data transmission channel.**
   Kanaal K zorgt voor de verbinding tussen sender S en receiver R. De frames uit S moeten via kanaal K in R terechtkomen.

   Tijdens dit transport kunnen met de frames twee dingen gebeuren. Het binnengekomen frame wordt correct naar R doorgestuurd of het frame wordt beschadigd.

   We gaan er hierbij van uit dat de receiver R in staat is onderscheid te maken tussen een correct frame f en een beschadigd frame 'ce' (= checksum error).

4. **component L: Unreliable acknowledgement transmission channel.**
   Kanaal L heeft ongeveer dezelfde werking als K, met dien verstande dat L geen frames, maar control bits (booleans) overstuurd.

De vier componenten zijn verbonden door een aantal datapoorten, genummerd 1 tot en met 6. Voor een verdere beschrijving van het Alternating Bit Protocol definiëren we een aantal verzamelingen:

- verzameling datapoorten $P = \{1, 2, 3, 4, 5, 6\}$
- verzameling waarheidswaarden $B = \{0, 1\}$
- verzameling data elementen D
- verzameling frames $(= D \times B)$ $F = \{d0, d1 : d \in D\}$

De vier processen, die de componenten S, K, L en R beschrijven, zijn gedefinieerd in Fig. 2.2. De vergelijkingen in deze specificatie van het Alternating Bit Protocol zijn min of meer een rechtstreekse 'vertaling' van de hierboven gegeven beschrijving van de componenten. De 'i' in de vergelijkingen [K1] en [L1] uit Fig. 2.2 is een atomaire (= ondeelbare) aktie. Deze aktie zorgt voor een non-deterministische keuze tussen het doorsturen van correcte gegevens (een frame in het geval van kanaal K en een boolean in het geval van kanaal L) of beschadigde gegevens (checksum errors). Non-deterministisch wil zeggen dat deze keuze niet kan worden beïnvloed door de omgeving; het is een eigenschap van het kanaal.

Unreliable data transmission channel K:

$$[K1] \quad K \quad = \quad \sum_{f \in B \times D} r_3(f) \cdot (i \cdot s_4(f) + i \cdot s_4(ce)) \cdot K$$

Unreliable acknowledgement transmission channel L:

$$[L1] \quad L \quad = \quad \sum_{b \in B} r_6(b) \cdot (i \cdot s_5(b) + i \cdot s_5(ce)) \cdot L$$

Sender S:

$$\begin{array}{lll} [S1] \quad S & = & S0 \cdot S1 \cdot S \\ [S2] \quad Sb & = & \sum_{d \in D} r_1(d) \cdot Sb_d \\ [S3] \quad Sb_d & = & s_3(db) \cdot Tb_d \\ [S4] \quad Tb_d & = & (r_5(1-b) + r_5(ce)) \cdot Sb_d + r_5(b) \end{array}$$

Receiver R:

$$\begin{array}{lll} [R1] \quad R & = & R1 \cdot R0 \cdot R \\ [R2] \quad Rb & = & (\sum_{d \in D} r_4(db) + r_4(ce)) \cdot s_6(b) \cdot Rb \\ & & + \sum_{d \in D} r_4(d(1-b)) \cdot s_2(d) \cdot s_6(1-b) \end{array}$$

Fig. 2.2 - Vergelijkingen Alternating Bit Protocol

Het Alternating Bit Protocol is de parallelle compositie van deze vier processen:

$$ABP = \delta_H(S \parallel K \parallel L \parallel R)$$

met $H = \{r_k(x), s_k(x) : x \in F \cup \{0, 1, ce\}, k = 3, ..., 6\}$

Het totale protocol, waarbij we de interne poorten 3, 4, 5 en 6 buiten beschouwing laten, voldoet aan de vergelijking:

$$\tau_I(ABP) = \sum_{d \in D} r_1(d) \cdot s_2(d) \cdot \tau_I(ABP)$$

met $I = \{c_k(x) : x \in F \cup \{0, 1, ce\}, k = 3, ..., 6\}$

Een bewijs van bovenstaande is te vinden in 'Process Algebra', J.C.M. Baeten & W.P. Weijland ([1], blz. 108-112).

3 Het Alternating bit Protocol met Time Out

In dit hoofdstuk zullen we het Alternating Bit Protocol gaan uitbreiden met een zogenaamde time out.

We gaan uit van het protocol dat in hoofdstuk twee is geïntroduceerd. Een nadeel van dat protocol is dat het versturen van een datum in principe oneindig lang kan doorlopen, wanneer één of beide kanalen de overdracht voortdurend verstoort. Dit is uiteraard een extreem geval. Het is ook mogelijk dat het protocol verhoudingsgewijs (zeer) lang bezig is met het oversturen van een relatief kleine hoeveelheid gegevens.

Om deze lange wachttijden uit het protocol te werken, breiden we de specificatie van het Alternating Bit Protocol uit met een time out. (Gezien het feit dat de factor tijd in dit ABP (nog) geen rol speelt, is de term time out een op het eerste gezicht misschien wat vreemde uitdrukking.) Hiermee hopen we te bewerkstelligen dat een datum slechts een maximaal aantal keren door sender S zal worden verzonden.

We willen dat een datum dat correct bij de ontvanger R aankomt, via datapoort 2 wordt doorgestuurd, zoals bij het Alternating Bit Protocol uit hoofdstuk 2. Echter, wanneer zender S niet binnen het maximaal aantal pogingen een juist acknowledgement terugkrijgt, willen we dat dit via een extra datapoort kenbaar wordt gemaakt. In onderstaande figuur is daarom datapoort 7 (uitgang) toegevoegd.

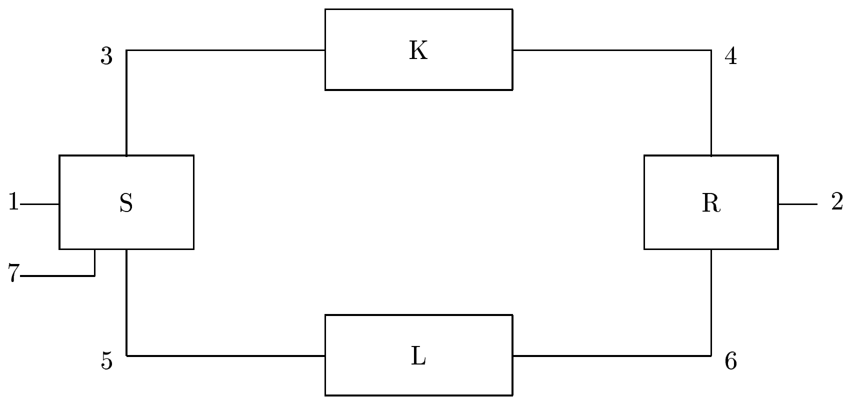

Fig. 3.1 - Alternating Bit Protocol met Time Out

Bij een eerste poging om een time out aan het protocol toe te voegen, hebben we alleen de vergelijkingen van de zender S aangepast. Een frame, bestaande uit een datum d en een boolean b, wordt nu maximaal MAX keer overgezonden. (Dit maximum is een positief geheel getal en wordt geïnitialiseerd in vergelijking [S2].)

Verloopt geen enkele van dit maximaal aantal zendingen correct, dan wordt het datum via poort 7 teruggestuurd en gaat zender S door met het volgende data-element.

Lukt het de zender wel binnen de gestelde grens een juist acknowledgement te ontvangen, dan gaat S onmiddellijk door met het volgende datum.

We komen tot de volgende specificatie voor zender S:

Sender S:

$$
\begin{array}{lllll}
[S1] & S & = & S0 \cdot S1 \cdot S \\
[S2] & Sb & = & \sum_{d \in D} r_1(d) \cdot Sb_d^{MAX} \\
[S3] & Sb_d^n & = & s_3(db) \cdot TO_b^{dn} \\
[S4] & TOb_d^1 & = & r_5(b) + (r_5(1-b) + r_5(ce)) \cdot s_7(d) \\
[S5] & TOb_d^n & = & r_5(b) + (r_5(1-b) + r_5(ce)) \cdot Sb_d^{n-1} (eis : n > 1)
\end{array}
$$

Fig. 3.2 - Vergelijkingen Zender S met Time Out

De letters TO staan voor time out. De bijbehorende n is een positief geheel getal, dat aangeeft na hoeveel foute antwoorden zender S het herhalen van een boodschap opgeeft. (Neemt men MAX=1 in de vergelijking [S2], dan verkrijgt men een protocol zonder hertransmissies.)

In de verkregen specificatie, waarin alleen zender S een 'time out mechanisme' heeft, is nu het volgende mogelijk. De zender besluit na een time out, een datum met het volgende bit, zeg frame d1, te versturen. De ontvanger R wacht nog steeds op het oude bit, want R heeft een correct datum nog niet ontvangen voor de time out van de zender. Receiver R zal het nu binnengekomen bit terugsturen. De zender denkt op zijn beurt dat het datum dan correct is aangekomen en zal doorgaan met het volgende data-element.

Op deze wijze zullen er gegevens verloren gaan tijdens het zenden, iets dat het Alternating Bit Protocol juist moet tegen gaan.

Telkens wanneer de zender en de ontvanger uit fase raken (met een verschillend bit bezig zijn) kan er data verloren gaan. We dienen er dus voor te zorgen dat ook de receiver R een time out mechanisme krijgt.

We passen de regels voor de ontvanger zo aan, dat R zich eerst een maximaal aantal keer in toestand 0 bevindt, daarna een maximaal aantal keer in toestand 1, vervolgens weer in toestand 0, enzovoort. Dit is in onderstaande figuur verwerkt.

Sender S:

$$
\begin{array}{llll}
[S1] & S & = & S0 \cdot S1 \cdot S \\
[S2] & Sb & = & \sum_{d \in D} r_1(d) \cdot Sb_d^{MAX} \\
[S3] & Sb_d^n & = & s_3(db) \cdot TOb_d^n \\
[S4] & TOb_d^1 & = & r_5(b) + (r_5(1-b) + r_5(ce)) \cdot s_7(d) \\
[S5] & TOb_d^n & = & r_5(b) \cdot STb_d^{n-1} + (r_5(1-b) + r_5(ce)) \cdot Sb_d^{n-1} (n > 1) \\
[S6] & Tb_d^1 & = & r_5(b) + r_5(1-b) + r_5(ce) \\
[S7] & Tb_d^n & = & (r_5(b) + r_5(1-b) + r_5(ce)) \cdot STb_d^{n-1} \\
[S8] & STb_d^n & = & s_3(bd) \cdot Tb_d^n
\end{array}
$$

Receiver R:

$$
\begin{array}{ll}
[R1] & R = (R1)^{MAX} \cdot (R0)^{MAX} \cdot R \\
[R2] & Rb = (\sum_{d \in D} r_4(db) + r_4(ce)) \cdot s_6(b) + \sum_{d \in D} r_4(d(1-b)) \cdot s_2(d) \cdot s_6(1-b)
\end{array}
$$

Fig. 3.3 - Vergelijkingen Zender S en Ontvanger R met Time Out

Wanneer het versturen van een datum een maximaal aantal keer fout gaat, zullen zender en ontvanger allebei naar een nieuwe toestand overgaan. Lukt het S wel binnen de gestelde grens een datum correct over te sturen, dan zal zender S hetzelfde datum blijven versturen zonder verdere consequenties, tot een volgende time out, waarop wederom S en R allebei naar een nieuwe toestand zullen gaan.

Nadeel van dit protocol is dat data-elementen verstuurd kunnen worden, terwijl deze al correct bij de ontvanger terecht zijn gekomen. Deze tijd kan uiteraard nuttiger gebruikt worden. Tevens zal een controle bit niet meer nodig zijn, omdat de zender en ontvanger zich altijd in dezelfde toestand bevinden.

In de uiteindelijke versie van het Alternating Bit Protocol laten we de time out afhangen van 'elementen' uit het protocol, die bekend zijn bij zowel de zender als de ontvanger.

Wanneer we kijken naar de kanalen K en L valt op dat er geen informatie verloren gaat. Frames of booleans komen correct of beschadigd aan, maar verdwijnen niet tijdens de transmissies. Hieruit concluderen we dat alle elementen die door zender

S en ontvanger R worden verzonden bij R resp. S terecht komen, in welke 'vorm' dan ook.

In de specificatie van Fig. 3.3 werd slechts één data-element per periode verwerkt. Wanneer de gegevensoverdracht correct verliep, werd dit datum door de zender zonder verdere consequenties verstuurd tot de eerstvolgende time out.

In de specificatie van Fig. 3.4 is het mogelijk om meer dan één data-element tijdens de periode tussen twee opeenvolgende time outs te verwerken. Op deze wijze proberen we die tijd zo efficiënt mogelijk te gebruiken.

Het Alternating Bit Protocol met time out is nu zo aangepast dat het precies werkt zoals het protocol uit hoofdstuk 2, maar dat na elke MAX zendingen het protocol wordt teruggezet naar de begintoestand (*reset*).

Unreliable data transmission channel K:

$$[K1] \quad K \quad = \quad \sum_{f \in D \times B} r_3(f) \cdot (i \cdot s_4(f) + i \cdot s_4(ce)) \cdot K$$

Unreliable acknowledgement transmission channel L:

$$[L1] \quad L \quad = \quad \sum_{b \in B} r_6(b) \cdot (i \cdot s_5(b) + i \cdot s_5(ce)) \cdot L$$

Sender S:

$$\begin{array}{ll}
[S1] & S = \sum_{d \in D} r_1(d) \cdot ST0_d^{MAX} \cdot S \\
[S2] & STb_d^1 = s_3(db) \cdot ((r_5(1-b) + r_5(ce)) \cdot s_7(d) + r_5(b)) \\
[S3] & STb_d^n = s_3(db) \cdot ((r_5(1-b) + r_5(ce)) \cdot STb_d^{n-1} \\
 & +r_5(b) \cdot \sum_{d \in D} r_1(d) \cdot ST(1-b)_d^{n-1})(eis : n > 1)
\end{array}$$

Receiver R:

$$\begin{array}{ll}
[R1] & R = R1^{MAX} \cdot R \\
[R2] & Rb^1 = (\sum_{d \in D} r_4(db) + r_4(ce)) \cdot s_6(b) + \sum_{d \in D} r_4(d(1-b)) \cdot s_2(d) \cdot s_6(1-b) \\
[R3] & Rb^n = (\sum_{d \in D} r_4(db) + r_4(ce)) \cdot s_6(b) \cdot Rb^{n-1} \\
 & + \sum_{d \in D} r_4(d(1-b)) \cdot s_2(d) \cdot s_6(1-b) \cdot R(1-b)^{n-1}(eis : n > 1)
\end{array}$$

Fig. 3.4 - Vergelijkingen Alternating Bit Protocol met Time Out

Groot voordeel bij deze versie is dat het protocol tussen twee opeenvolgende resetten precies werkt zoals het Alternating Bit Protocol beschreven in hoofdstuk 2.

In de Appendix bewijzen we dat voor deze specificatie voor alle $MAX > 0$ het volgende geldt:

$$ACP + TR \models \tau_I \circ \delta_H(S \parallel K \parallel L \parallel R)$$

$$= \sum_{d \in D} r_1(d) \cdot (s_2(d) + s_7(d) + s_2(d) \cdot s_7(d)) \cdot \tau_I \circ \delta_H(S \parallel K \parallel L \parallel R)$$

TR is het trace-axioma: $a \cdot (b + c) = a \cdot b + a \cdot c$
Met dit axioma zijn processen niet bisimulatie equivalent maar trace equivalent als de corresponderende proces termen naar elkaar herschreven kunnen worden. De oorzaak van dit resultaat kan verklaard worden als volgt:
doordat het protocol moet onthouden hoever het zich in de Time Out cyclus bevindt, ontstaan veel keuze momenten in de trace die onderscheid maken tussen een protocol met N hertransmissies en een protocol met $N - 1$ hertransmissies.

Voor iedere N is de normaalvorm natuurlijk verschillend, omdat hierin de verschillende keuzepunten zichtbaar moeten zijn. Indien we echter het trace axioma toevoegen, dan verdwijnen de verschillen door keuzemomenten uit de normaalvormen.

Wat als nadeel gezien kan worden is dat niet elk datum dezelfde kans krijgt om correct te worden verzonden (geen lineaire kansverdeling).

Stel het maximaal aantal zendingen MAX is gelijk aan tien. Wanneer de zender, na het correct verzenden van een datum, bijvoorbeeld bij de negende zending aan een nieuw datum begint, zal deze slechts twee maal worden geprobeerd. Na de reset wordt immers een nieuw datum gelezen, dat maximaal tien keer zal worden geprobeerd.

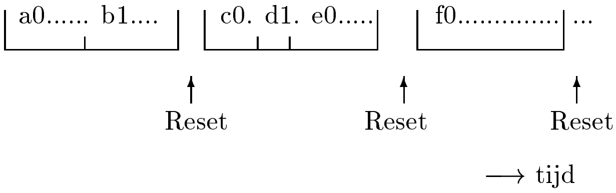

Fig. 3.5 - Tijddiagram van het Protocol $a, b, c, d, e, f \in D$

4 Het Alternating Bit Protocol met Time Out in Discrete Tijd

Tot nu toe hebben we alleen in een algebra zonder tijd gewerkt. We gaan nu proberen om hetzelfde probleem te behandelen in een algebra met tijd: de Discrete Tijd Proces Algebra. Dit is een algebra waarin we wel expliciet tijdsvoorwaarden kunnen uitdrukken. Deze algebra wordt besproken in 'Discrete Time Process Algebra', J.C.M. Baeten en J.A. Bergstra [2].

In het Alternating Bit Protocol zoals we het in eerdere hoofdstukken beschreven hebben, valt op dat berichten niet kwijt kunnen raken. Dat wil zeggen als er data wordt gelezen van poort 1, dan komen er altijd weer gegevens via poort 2, dan wel via poort 7 'naar buiten' (zie Fig. 4.1).

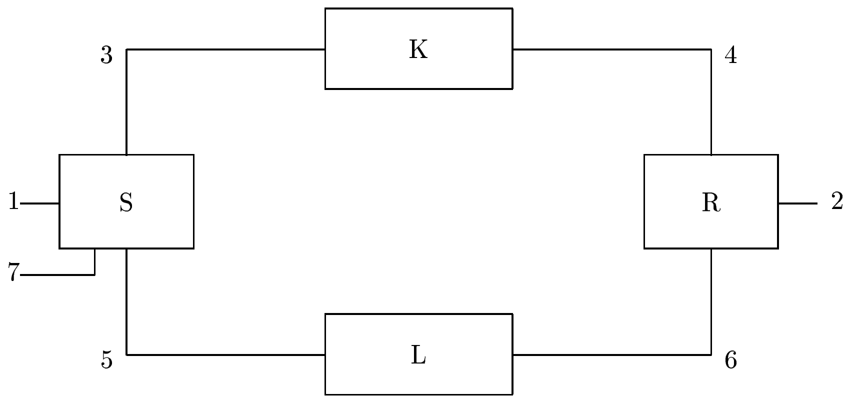

Fig. 4.1 - Alternating Bit Protocol met Time Out in Discrete Tijd

Telkens is geprobeerd om van oneindige lussen af te komen op een redelijke manier. Elk datum krijgt een aantal kansen, maar niet oneindig veel kansen om onbeschadigd de ontvanger te bereiken.

Willen we deze twee aspecten ook in een protocol met discrete tijd handhaven, dan zullen we het protocol een tweetal restricties moeten opleggen.

Restrictie 1:
Een bericht mag niet eeuwig onderweg zijn, dat wil zeggen er is een maximale tijdsduur voor een reeks interne lees- en schrijfakties over de poorten 3, 4, 5 en 6 $(s_3, r_3, s_4, r_4, s_5, r_5, s_6, r_6)$.

Restrictie 2:
Zender S en ontvanger R kunnen worden gesynchroniseerd. Dat wil zeggen, bij iedere time out en bij het begin van het systeem worden deze twee processen als het ware op nul gezet. Deze actie vindt telkens plaats in de eerste tik van iedere periode. Daarom beginnnen de zender en ontvanger pas in de tweede tik (zie het tijdsinterval bij [S1] in Fig 4.2).

We hebben deze actie niet expliciet opgenomen in de specificatie omdat we dan een soort synchronisatie-communicatie zouden hebben geïntroduceerd waarbij de communicatie enkel en alleen gebruikt wordt voor zijn synchronisatie aspecten. Dit leek ons niet de bedoeling van een communicatie-actie.

Verder is het even snel verstrijken van de tijd iets waar totaal niet over gepraat wordt in algebra's met tijd, terwijl dit naar ons idee in het geheel geen triviaal probleem is, maar buiten het bereik van dit werkstuk ligt.

We nemen aan dat de zender en de ontvanger en de beide kanalen gebruik maken van klokken die precies even snel lopen.

We volgen in principe dezelfde strategie als in het vorige hoofdstuk. We doen om een aantal $(s_3 - r_6)$-cycles een time out en controleren in de (n-1)e (voorlaatste) cycle of tijdens die cycle de overdracht correct verlopen is. Zo niet, dan melden we dit over kanaal 7.

Echter, we 'tellen' nu de $(s_3 - r_6)$-cycles met behulp van de tijd die verstreken is en niet meer met het aantal cycles. Na de time out synchroniseren zender en ontvanger.

We zorgen ervoor dat de zender alleen een bericht mag versturen, als de bevestiging van dit bericht terug is voor de eerstvolgende time out. Als de zender niet aan deze eis voldoet, dan zouden er berichten verloren kunnen gaan, doordat de ontvanger in de tussentijd door zijn time out is gegaan en in een andere toestand is gekomen.

De zender moet dus kunnen uitrekenen hoelang een bericht maximaal onderweg is. Hiervoor hebben we de maximale tijdsduur nodig. Deze tijdsduur ligt vast en wordt aangeduid met de constante DMAX.

Als de zender iets verstuurt, dan weet hij op de tik nauwkeurig wanneer hij iets terugkrijgt. Op deze wijze kan de zender bepalen of het moment waarop hij antwoord krijgt nog voor de eerstvolgende time out ligt.

We komen tot de volgende specificatie (zie Fig 4.2).

Verklaring van de in de figuur gebruikte afkortingen:

to time out.

SD Short Delay, een kleine constante, bv. 10 units.

DMAX Maximum Delay, de maximale tijdsduur van een $(s_3 - r_6)$-cykel (= 7SD).

P Period, de tijd tussen twee time outs, bv. 10*DMAX. In ieder geval moet gelden dat P = N*DMAX met $N \in 1, 2, ...$

SA, SM Send Acknowledgement, resp. Send Message.

Unreliable datatransmission channel K:

$$[K1] \quad K = \sum_{v \in N} \sum_{f \in D \times B} r_3(f)[v].K_f$$

$$[K2] \quad K_f = (i[SD] \cdot s_4(f)[SD] + i[SD] \cdot s_4(ce)[SD]) \cdot K$$

Unreliable acknowledgement transmission channel L:

$$[L1] \quad L = \sum_{v \in N} \sum_{b \in B} r_6(b)[v] \cdot L_b$$

$$[L2] \quad L_b = (i[SD] \cdot s_5(b)[SD] + i[SD] \cdot s_5(ce)[SD]) \cdot L$$

Sender S:

$$[S1] \quad S = (\sum_{v \in [2,P-1]} \sum_{d \in D} r_1(d)[v] \cdot SF^v_{d0} + to[P]) \cdot S$$

$$[S2] \quad SF^t_{d0} = SR^t_{db} < t < (P - DMAX) > s_7(d)[SD] \cdot to[P - t - SD]$$

$$[S3] \quad SR^t_{db} = s_3(db)[SD] \cdot (\sum_{v \in N}(r_5(1-b)[v] + r_5(ce)[v]) \cdot SF^{t+DMAX}_{db})+$$

$$(\sum_{v \in N} r_5(b)[v] \cdot RM^{t+DMAX+SD}_{1-b})$$

$$[S4] \quad RM^t_b = \sum_{v \in [t,P-1-t]} \sum_{d \in D} r_1(d)[v] \cdot SF^{v+t}_{db} + to[P-t]$$

Receiver R:

$$[R1] \quad R = R^2_0 \cdot R$$

$$[R2] \quad R^t_b = (\sum_{v \in [t,P-t-1]} \sum_{d \in D} r_4(d(1-b))[v] + r_4(ce)[v]) \cdot SA^{t+v}_{1-b}+$$

$$\sum_{v \in [t,P-t-1]} \sum_{d \in D} r_4(db)[v] \cdot SM^{t+v}_{db} + to[P-t]$$

$$[R3] \quad SA^t_b = s_6(b)[SD] \cdot R^{t+SD}_{1-b}$$

$$[R4] \quad SM^t_{db} = s_2(d)[SD] \cdot s_6(b)[SD] \cdot R^{t+2SD}_{1-b}$$

Fig 4.2 - Vergelijkingen Alternating Bit Protocol met Time Out in Discrete Tijd

Het Alternating Bit Protocol met time out in discrete tijd wordt nu als volgt beschreven:

$$ABPTO = \delta_H(S \parallel K \parallel L \parallel R)$$

met $H = \{r_k(x), s_k(x) : x \in F \cup \{0, 1, ce\}, k = 3, ..., 6\} \cup \{to\}$.

Nu willen we graag bewijzen dat dit protocol geabstraheerd van tijd en interne acties trace equivalent is met:

$$\tau_I(ABPTO) = \sum_{d \in D} r_1(d) \cdot (s_2(d) + s_7(d) + s_2(d) \cdot s_7(d)) \cdot \tau_I(ABPTO)$$

We volgen nu een andere strategie dan in de vorige hoofdstukken in de bewijsvoering. We hebben dit gedaan omdat we het vermoeden hebben dat het gebruik van temporele logica het bewijs een stuk inzichtelijker maakt op punten waar de toevoeging van discrete tijd invloed heeft.

We gaan eerst onderzoeken in hoeverre we van de tijdsaspecten in het protocol kunnen abstraheren. We kunnen het volgende opmerken over het protocol in discrete tijd:

- er gebeuren geen twee akties in 1 tik
- als we alle akties die plaatsvinden op een rij zetten en we projecteren vervolgens op de tik waarin die actie plaatsvindt, dan verkrijgen we een zwak stijgende rij.

Dus voor alle mogelijke executiepaden:

$$(a_1(t_1), ..., a_n(t_n), a_{n+1}(t_{n+1}), ...)$$

geldt dat de rij

$$(t_1, ..., t_n, t_{n+1}, ...)$$

een zwak stijgende rij is. Bovenstaande volgt uit de semantiek van de discrete tijd proces algebra. Combineren we beide observaties, dan volgt hieruit dat de projectie op tijd een sterk stijgende rij is.

We willen een aantal regels opstellen waaraan ieder Alternating Bit Protocol zou moeten voldoen. Vervolgens laten we zien dat het bovenstaande protocol hieraan voldoet. Deze regels construeren we met een soort temporele logica.

We voeren een aantal definities in:

$\forall$ = voor alle (alkwantor)
$\exists$ = er is een (existentiele kwantor)
$\neg$ = niet (ontkenning)
$\rightarrow$ = implicatie
$\wedge$ = logische en
$\vee$ = logische of
A = verzameling akties uit de discrete tijd proces algebra
T = een verzameling tijdsvariabelen (bv. $T = \{t_1, ..., t_n\}$)

Een formule is nu als volgt gedefinieerd:

1. elke aktie $a \in A$ is een formule
2. als f, g formules zijn dan zijn $\neg f, f \vee g, f \wedge g, f \rightarrow g$ ook formules
3. als $t, u \in T$, dan zijn $t < u, t > u, t = u$ etc. ook formules
4. als f een formule is, en $t \in T$, dan zijn $\forall t(f), \exists t(f)$ ook formules

De betekenis van een formule is nu als volgt:

- de betekenis van een formule $a \in A$ is dat de actie a plaatsvindt in het systeem
- de betekenis van alle andere formules zijn de bekende logische betekenissen.

We stellen een aantal formules op, waaraan een Alternating Bit Protocol met Time Out zou moeten voldoen:

Formule F1:

$$ABPTO \models \forall t(r_1(d)(t) \rightarrow \exists u((u > t) \wedge (s_2(d)(u) \vee s_7(d)(u))))$$

Formule F2:

$$ABPTO \models \forall t \forall u((r_1(d)(t) \wedge (d \neq e) \wedge (s_2(e)(u) \vee s_7(e)(u)) \wedge (u > t)) \rightarrow$$
$$\exists s((t < s < u) \wedge (r_1(e)(s) \wedge$$
$$\neg(\exists v(s < v < u) \wedge (s_2(e)(v) \vee s_7(e)(v))) \vee \exists w(s < w < u) \wedge (r_1(e)(w)))))$$

De eerste formule drukt uit dat data gelezen van poort 1, altijd op poort 2, dan wel poort 7 terecht komt. De tweede formule zegt dat er tussen een read-aktie van poort 1 en een send-aktie over poort 2 of poort 7, geen andere read-aktie van poort 1 plaats mag vinden. Een datum wordt volledig afgehandeld, voordat het protocol aan het volgende datum begint.

Vervolgens doen we de aanname dat als een ABPTO specificatie aan de bovenstaande eisen voldoet, hij dan correct is. Wat ons nu rest is om de bovenstaande formules aannemelijk te maken voor het protocol dat we eerder hebben geschreven. Dit zullen we doen door alle mogelijke executiepaden, tot op zekere diepte, van het protocol af te lopen en hiervoor de bovenstaande formules te bewijzen. Vervolgens gebruiken we het feit dat de tijd niet stil staat en dat de time out eindig is.

Allereerst bewijzen we dat de eerste aktie van het protocol, afgezien van een to-aktie, die niets anders is dan wachten tot de eerstvolgende time out, een read-aktie over poort 1 is.

Stelling 1:

$$ABPTO \models \forall t((a(t) \wedge (\forall d(a \neq r_1(d)(t))) \wedge (\neg(a = to(t))) \rightarrow \exists u \exists d((u < t) \wedge r_1(d)(u)))$$

Bewijs van stelling 1: Aangezien de processen S, K, L, R allen beginnen met een read- aktie of een time out, en r_1 de enige aktie is die niet in verzameling H voorkomt, moet een read-aktie van poort 1 wel de eerste actie van het systeem zijn, gezien het feit dat er geen twee akties zijn in 1 tik.

Vervolgens bewijzen we dat als de zender S iets van poort 1 leest, het protocol altijd via poort 3 iets communiceert of data over poort 7 schrijft.

Stelling 2:

$$ABPTO \models \forall t(r_1(d)(t) \rightarrow (s_7(d)(t + SD) \vee c_3(d0)(t + SD)))$$

Bewijs van stelling 2: Na een read-aktie op tijdstip t van poort 1 komt een SF (send frame), daarna volgt of een send-aktie over poort 7 op tijdstip t+SD of een hertransmissie (SR) van het frame, die begint met een send-aktie op tijdstip t+SD. Deze aktie leidt tot een communicatie over poort 3. Andere processen kunnen nog geen akties hebben gedaan, want die beginnen allemaal met een read-aktie uit de verzameling H.

Stelling 3:

$$ABPTO \models \forall t(c_3(db)(t + SD) \rightarrow ((c_4(db)(t + 3SD) \vee c_4(ce)(t + 3SD))$$
$$\wedge \neg \exists v((t < v < (t + 3SD)) \wedge a(v) \wedge (\neg(a = c_4(db))) \wedge (a \neq c_4(ce))))$$

Bewijs van stelling 3: Na een communicatie via poort 3 volgt een send-aktie over poort 4. We zijn nu op tijdstip t+SD+SD+SD aangeland, want: $r_1(t) \cdot c_3(t + SD) \cdot i(t + SD + SD) \cdot s_4(t + SD + SD + SD)$. Wil ontvanger R nu iets kunnen lezen, dan moet er gelden: $t + SD + SD + SD < P - 2$. Aangezien $P = 50SD$ en $t < 50SD - 10SD = 40SD$, hebben we dus $43SD < 48SD$. Dit is waar, dus er volgt een communicatie over poort 4; de to-aktie kan pas later komen, want $P - t > P - t - 1$.

Nu splitsen we het verhaal in twee delen: het ene deel behandelt de mogelijkheid $c_4(db)$ (het correct doorsturen van een frame) en de andere de mogelijkheid $c_4(ce)$ (de communicatie van een beschadigd frame).

Stelling 4:

$$ABPTO \models \forall t(r_1(d)(t) \wedge c_3(db)(t+SD) \wedge c_4(db)(t+3SD) \rightarrow s_2(d)(t+4SD).$$

Bewijs stelling 4: Triviaal. Als we op tijdstip t over poort 1 lezen en vervolgens gaan alle communicaties goed, dan kan ontvanger R nog niet van toestand zijn veranderd.

We hebben nu de helft van het verlangde bewezen: als we iets over poort 1 lezen dan kan het zijn dat we dat ook over poort 2 schrijven. Nu moeten we nog bewijzen dat als het protocol niet over poort 2 schrijft, het protocol in ieder geval over poort 7 schrijft.

Stelling 5:

$$ABPTO \models \forall t(r_1(d)(t) \wedge c_3(db)(t+SD) \wedge c_4(ce)(t+3SD) \rightarrow c_5(1-b)(t+5SD))$$

Bewijs van stelling 5: Triviaal.

Stelling 6:

$$ABPTO \models$$

$$\forall t(r_1(d)(t) \wedge c_3(db)(t+SD) \wedge c_4(ce)(t+3SD) \rightarrow (c_3(db)(t+8SD) \vee s_7(d)(t+8SD))$$

Bewijs van stelling 6: Als we over poort 5 het verkeerde bit sturen dan zal in ieder geval de zender het verkeerde bit of een error krijgen en zal zender S vervolgens het frame opnieuw sturen of door zijn time out gaan en over het datum over poort 7 sturen.

Nu zijn we in eenzelfde toestand als eerst met als enig verschil dat de tijd niet heeft stilgezeten. Dus we hebben nu gezien dat bij een executiepad $r_1 - c_3 - c_4(ce) - c_5 - c_6 - c_3$ we of over poort 7 schrijven of overnieuw beginnen later in de tijd. Aangezien de time out eindig is, kunnen we dit geval afsluiten met de constatering dat als dit executiepad zich blijft herhalen er uiteindelijk over poort 7 geschreven wordt, de zender door zijn time out gaat, en het protocol overnieuw begint.

Stelling 7:

$$ABPTO \models \forall t(r_1(d)(t) \wedge c_4(db)(t+3SD) \rightarrow (s_2(d)(t+4SD) \wedge c_5(b)(t+5SD))$$

Bewijs van stelling 7: Triviaal.

Nu hebben we nog het geval over waar de communicatie over poort 6 verkeerd gaat. In dat geval zijn er twee mogelijkheden: of de zender gaat door zijn time out en stuurt het datum over poort 7 en begint overnieuw, of de zender probeert het datum opnieuw:

Stelling 8:

$$ABPTO \models \forall t(r_1(d)(t) \wedge s_2(d)(t+4SD) \wedge c_6(ce)(t+7SD) \rightarrow s_7(d) \vee c_3(d)(t+8SD))$$

Bewijs van stelling 8: Triviaal.

Het enige wat nu nog over blijft, is bewijzen dat we nu uiteindelijk door een time out gaan. Er is namelijk al over poort 2 het goede datum geschreven, dus aan de vereiste regel F1 is al voldaan.

Welnu, we doen in iedere tik maar één aktie, de projectie op tijd is een sterk stijgende rij, dus uiteindelijk zal de time out in werking treden, aangezien het time out-tijdstip nooit binnen een $c_3 - c_4 - c_5 - c_6$ cycle kan vallen (zie verder).

Aangezien na iedere to-aktie het protocol overnieuw begint, hebben we nu ook aannemelijk gemaakt dat het ABPTO, geabstraheerd van tijd trace equivalent is met:

$$ABPTO = r_1(d) \cdot (s_2(d) + s_7(d) + s_2(d) \cdot s_7(d)) \cdot ABPTO$$

Het enige wat we in het bovenstaande nog niet behandeld hebben, is de vraag of het altijd zo is dat als we een bericht versturen, de ontvanger dat nog voor zijn time out binnen krijgt. Dit bewijs is eenvoudig.

Stel we doen een send-aktie over poort 3 op tijdstip $t + SD$. Dan geldt: $t < P - DMAX = P - 7SD$. Wanneer komt dit bericht aan bij de ontvanger? Op tijdstip $t + SD + SD + SD = t + 3SD$. Nu moet gelden dat $t + 3SD < P - 1$, dus $t < P - 3SD - 1$. We wisten al dat $t < P - 7SD$. Aangezien $P - 7SD < P - 3SD - 1$ (want $4SD > 1$), is dit waar. Conclusie: alles wat de zender S verstuurt, komt op tijd aan bij de ontvanger.

Komt het nu ook op tijd weer terug? We hebben reeds DMAX vastgesteld op 7SD. Dus stel we doen weer een send-aktie op het tijdstip $t + SD$. Dan komt dit uiterlijk terug bij poort 5 op tijdstip $t + 7SD$. Nu moet gelden, anders is de ontvanger intussen door zijn time out gegaan: $t + 7SD < P \rightarrow t < P - 7SD$, en dit was gegeven.

Formule $F2$ volgt uit het feit dat we nooit in de executie de mogelijkheid hadden om na een *read*-actie wederom eerst een *read*-actie te doen voordat we het datum ergens over een poort geschreven hadden.

5 Het Alternating Bit Protocol met Disable

Het is wenselijk om de vier processen (zender S, ontvanger R en twee kanalen K en L) op een nette manier te kunnen beeindigen. Tot nu toe waren deze processen recursieve processen die 'eeuwig' doorgingen.

We willen nu bekijken in hoeverre het mogelijk is om aan het eerder geschetste protocol in discrete tijd een dergelijke optie toe te voegen.

Wij veronderstellen dat zowel de zender S als de ontvanger R kunnen besluiten op een bepaald tijdstip er mee op te houden.

De meest voor de hand liggende oplossing is om de zender en ontvanger, indien ze er mee op willen houden, niets meer te laten sturen. (Het stop bericht is het stoppen zelf.)

De tegenpartij zal na een tijdje dit opmerken en concluderen dat het lopende proces beëindigd dient te worden. Dit is in het huidige protocol mogelijk omdat er nooit berichten verloren gaan in de buffers en omdat er een maximale doorlooptijd is voor ieder bericht.

Aan een dergelijke oplossing kleven wel nadelen. Wat nu bijvoorbeeld als het kanaal door een storing voor onbepaalde tijd onbruikbaar is geworden (de buffer processen vallen weg)? Beide overige processen zullen nu beëindigd worden zonder dat dit misschien de bedoeling was. Misschien is het niet eens zo gek om in zo'n geval maar te stoppen, omdat het niet zeker is wanneer het kanaal weer bruikbaar is.

Ook als de zender of de ontvanger in zo'n geval kapot gaan zullen de andere processen vanzelf ophouden omdat er niets meer gebeurt. Dit is een zeer wenselijke eigenschap zoals we die dagelijks aan bijvoorbeeld tramdeuren ondervinden. Een tramdeur zal sluiten ('houdt er mee op geopend te zijn') als er lang genoeg niemand gebruik van maakt.

We kunnen dit in het protocol in discrete tijd realiseren door van de to-aktie in bijvoorbeeld regel [S1] uit Fig. 4.2 een stop-aktie *quit* te maken.

$$[S1] \quad S = \sum_{v \in [2,P-1]} \sum_{d \in D} r_1(d)[v] \cdot SF^v_{d0} \cdot S + quit[P]$$

Bovenstaande regel zorgt ervoor dat een data-element wordt gelezen van poort 1 binnen een bepaald interval. Wanneer in dit interval geen datum wordt aangeboden, zal het proces S worden gestopt op een tijdstip P.

Voor de andere processen uit het protocol zijn de regels ook uit te breiden met een dergelijke stop-aktie. De ontvanger R bijvoorbeeld verwacht binnen een bepaald interval een frame van poort 4. Wanneer dit niet gebeurt, kan ook R besluiten het proces te stoppen middels *quit*.

Wat in ieder geval niet mogelijk is, is om een bepaald stopbericht als datum (frame) te sturen naar de verschillende processen. Het is namelijk nooit zeker of een bepaald bericht ongeschonden aankomt, dus ook van een dergelijk stopbericht niet. Indien we er voor kiezen dit stopbericht te blijven versturen totdat het uiteindelijk aankomt, dan gaat dit ten koste van het time out principe: eeuwige executiepaden zijn dan weer mogelijk.

6 Conclusie

In dit werkstuk hebben we het Alternating Bit Protocol uitgebreid met een time out. Dit is in een aantal stappen gebeurd. We zijn uitgegaan van het protocol uit 'Process Algebra' van J.C.M. Baeten en W.P. Weijland. We hebben in hoofdstuk 3 een time out-mechanisme ingebouwd, door het aantal data-elementen bij te houden. Na elke MAX zendingen wordt het protocol 'gereset', zodat het weer in de initiële toestand terechtkomt.

In hoofdstuk 4 tenslotte, is de time out niet meer gebaseerd op het aantal zendingen, maar op de verlopen tijd na de laatste time out.

Wat als nadeel gezien kan worden, bij zowel het protocol uit hoofdstuk 3 als uit hoofdstuk 4, is dat niet elk data-element dezelfde kans krijgt om de ontvanger R te bereiken. Sommige elementen kunnen de volledige tijdsduur tussen twee opeenvolgende time outs worden geprobeerd, terwijl andere elementen slechts een keer (vlak voor de time out) worden geprobeerd.

Om ervoor te zorgen dat elk datum wel dezelfde kans krijgt, kan men ervoor kiezen alleen een nieuw datum te lezen van poort 1 direct na een time out. Nadeel van deze aanpak is dat de tijd tussen een geslaagde transmissie en de eerstvolgende time out verloren gaat. (De zender S kan besluiten om niets te doen, of hetzelfde datum blijven versturen tot de eerstvolgende time out).

Tevens zal een controle bit, dat tezamen met elk datum als frame wordt verzonden, niet meer nodig zijn, omdat de zender en de ontvanger zich altijd in eenzelfde toestand zullen bevinden.

Bij een (te) ruim gekozen time out en redelijk/goed werkende kanalen K en L is het tweede alternatief geen goede keuze. Stel bijvoorbeeld dat twee verzamelingen data-elementen moeten worden verzonden. Uit het verzenden van de eerste verzameling blijkt bijvoorbeeld dat de kanalen K en L goed/redelijk werken, waardoor men besluit de time out te verkorten. Bij de tweede verzending werken de kanalen aanzienlijk minder goed, met als gevolg dat veel data-elementen niet binnen de time out onbeschadigd ontvanger R weten te bereiken.

Uiteraard is de performance van het eerste alternatief bij slecht werkende kanalen niet beter dan het tweede alternatief, maar de time out hoeft bij goed werkende kanalen K en L niet te worden aangepast.

We hebben gekozen voor de eerste mogelijkheid, omdat we niet willen dat bij elk te verzenden datum een time out hoort. We willen dat tijdens de periode tussen twee opeenvolgende time outs het protocol zoveel mogelijk data-elementen kan verwerken en zoveel mogelijk lijkt op het protocol waar we vanuit zijn gegaan.

Tevens is aan het Alternating Bit Protocol een extra uitgang toegevoegd. Uitgang 2, onder controle van de ontvanger R, heeft nog steeds dezelfde functie als het originele protocol: op deze uitgang verschijnen data-elementen die correct (onbeschadigd) bij de ontvanger terecht zijn gekomen.

Op de extra uitgang 7, onder controle van de zender S, verschijnen data-elementen, waarvan de zender, na de laatste transmissie van dat datum, niet zeker weet of ze correct bij de ontvanger R zijn aangekomen.

Het is wellicht mogelijk het protocol zo uit te breiden, dat na het verzenden van alle data van poort 1, de data-elementen die via uitgang 7 naar buiten zijn gekomen nog eenmaal te versturen (een nieuwe kans geven).

Het Alternating Bit Protocol met Time Out dient dan in ieder geval te worden uitgebreid met een buffer (opslagplaats) voor alle elementen die via uitgang 7 naar buiten komen. Helaas is hierdoor de volgorde waarin de elementen verstuurd moeten worden verdwenen. Om dit te verhelpen zijn extra gegevens (bijvoorbeeld een index) nodig, om de plaats van één enkel data-element in het geheel aan te geven.

7 Geraadpleegde literatuur

[1] 'Process Algebra'
J.C.M. Baeten, W.P. Weijland
Cambridge Tracts in Theoretical Computer Science 18
Cambridge University Press 1990

[2] 'Discrete Time Process Algebra'
J.C.M. Baeten, J.A. Bergstra
Programming Research Group Report P9208
University of Amsterdam April 1992

[3] 'Algebraic Techniques for Concurrency and their Application'
F.W. Vaandrager
Academisch Proefschrift
Centrum voor Wiskunde en Informatica 1989

[4] 'Real Time Process Algebra'
J.C.M. Baeten, J.A. Bergstra
Formal Aspects of Computing 3 (2), 1991 pp. 142 - 188

A Correctheidsbewijs van het ABP met Time Out

Wij zullen nu bewijzen dat het Alternating Bit Protocol met time out, zoals wij dat eerder in hoofdstuk 3 hebben gedefinieerd, een protocol is, dat aan onze eisen voldoet.

Wij zijn uitgegaan van een correct werkend protocol, gegeven in 'Process Algebra', J.C.M. Baeten & W.P. Weijland [1]. Dit protocol hebben we een extra eis opgelegd, namelijk dat elk data-element slechts een maximaal aantal keren door zender S zal worden geprobeerd om onbeschadigd ontvanger R te bereiken.

We zullen laten zien dat de door ons geschreven specificatie uit Fig. 3.4 eindig is, dat wil zeggen: na een aantal pogingen een data-element over te sturen zal het protocol weer in de begintoestand terechtkomen (time out).

Tevens zullen we aantonen dat elke zichtbare (externe) aktie alleen een lees-aktie van poort 1 (ingang) kan zijn, of een schrijf-aktie over poort 2, dan wel poort 7 (uitgangen).

We definiëren een tweetal verzamelingen H en I:

- verzameling H van (interne) lees- en schrijfakties:
  $H = \{r_k(x), s_k(x) : x \in F \cup \{0, 1, ce\}, k = 3, 4, 5, 6\}$
- verzameling I van alle interne akties:
  $I = \{c_k(x) : x \in F \cup \{0, 1, ce\}, k = 3, 4, 5, 6\} \cup \{i\}$

We merken op dat de zender S en de ontvanger R zich beide in twee toestanden kunnen bevinden. In totaal verkrijgen we vier toestanden die in de volgende tabel zijn weergegeven.

TOESTAND S	TOESTAND R	LETTER
0	0	A
0	1	B
1	0	C
1	1	D

We zullen nu de term $\delta_H(S \parallel K \parallel L \parallel R)$ met behulp van de regels uit Fig. 3.4 (Hoofdstuk 3) gaan herschrijven.

$\delta_H(S \parallel K \parallel L \parallel R) = \delta_H(\sum_{d \in D} r_1(d) \cdot ST0_d^{MAX} \cdot S \parallel K \parallel L \parallel R)$
$= \sum_{d \in D} r_1(d) \cdot \delta_H(ST0_d^{MAX} \cdot S \parallel K \parallel L \parallel R)$

ONDERZOEK MAX = 1.

Allereerst bekijken we mogelijkheid (B) in bovenstaande toestandstabel. (Dit stelt een protocol voor zonder hertransmissies.)

$\delta_H(ST0_d^{MAX} \cdot S \parallel K \parallel L \parallel R) = \delta_H(ST0_d^1 \cdot S \parallel K \parallel L \parallel R)$

$= \delta_H(s_3(d0) \cdot ((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel \sum_{d \in D} r_3(d0)(i \cdot s_4(d0) + i \cdot s_4(ce)) \cdot K \parallel L \parallel R)$

$= c_3(d0)\{i \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel s_4(d0) \cdot K \parallel L \parallel R)+$
$i \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel R)\}$

$= c_3(d0)\{i \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel s_4(d0) \cdot K \parallel L \parallel R_1^1 \cdot R)+$
$i \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel R_1^1 \cdot R)\}$

$= c_3(d0)\{i \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel s_4(d0) \cdot K \parallel L \parallel$
$((\sum_{d \in D} r_4(d1) + r_4(ce)) \cdot s_6(1) + \sum_{d \in D} r_4(d0) \cdot s_2(d) \cdot s_6(0)) \cdot R)+$
$i \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel$
$((\sum_{d \in D} r_4(d1) + r_4(ce)) \cdot s_6(1) + \sum_{d \in D} r_4(d0) \cdot s_2(d) \cdot s_6(0)) \cdot R)\}$

$= c_3(d0)\{i \cdot c_4(d0) \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel L \parallel s_2(d) \cdot s_6(0) \cdot R)+$
$i \cdot c_4(ce) \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel L \parallel s_6(1) \cdot R)\}$

$= c_3(d0)\{i \cdot c_4(d0) \cdot s_2(d) \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel L \parallel s_6(0) \cdot R)+$
$i \cdot c_4(ce) \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel L \parallel s_6(1) \cdot R)\}$

$= c_3(d0)\{i \cdot c_4(d0) \cdot s_2(d) \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel$
$r_6(0)(i \cdot s_5(0) + i \cdot s_5(ce)) \cdot L \parallel s_6(0) \cdot R)+$
$i \cdot c_4(ce) \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel r_6(1)(i \cdot s_5(1) + i \cdot s_5(ce)) \cdot L \parallel s_6(1) \cdot R)\}$

$= c_3(d0)\{i \cdot c_4(d0) \cdot s_2(d) \cdot c_6(0)\{i \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel s_5(0) \cdot L \parallel R)+$
$i \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel s_5(ce) \cdot L \parallel R)\}+$
$i \cdot c_4(ce) \cdot c_6(1)\{i \cdot \delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel s_5(1) \cdot L \parallel R)+$
$\delta_H(((r_5(1) + r_5(ce)) \cdot s_7(d) + r_5(0)) \cdot S \parallel K \parallel s_5(ce) \cdot L \parallel R)\}$

$= c_3(d0)\{i \cdot c_4(d0) \cdot s_2(d) \cdot c_6(0)\{i \cdot c_5(0) \cdot \delta_H(S \parallel K \parallel L \parallel R)+$
$i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R)\}+$
$i \cdot c_4(ce) \cdot c_6(1)\{i \cdot c_5(1) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R)+$
$i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R)\}\}$

Bovenstaande term is grafisch weergegeven in Figuur 1.

Wanneer we met de τ-operator alle interne stappen verwijderen, krijgen we de volgende vergelijking voor MAX = 1:

$ABPTO = \tau_I \cdot \delta_H(S \parallel K \parallel L \parallel R)$
$= \sum_{d \in D} r_1(d) \cdot (s_2(d) \cdot (ABPTO + s_7(d) \cdot ABPTO) + s_7(d) \cdot ABPTO$
$= \sum_{d \in D} r_1(d) \cdot (s_2(d) \cdot (\epsilon + s_7(d)) + s_7(d)) \cdot ABPTO$

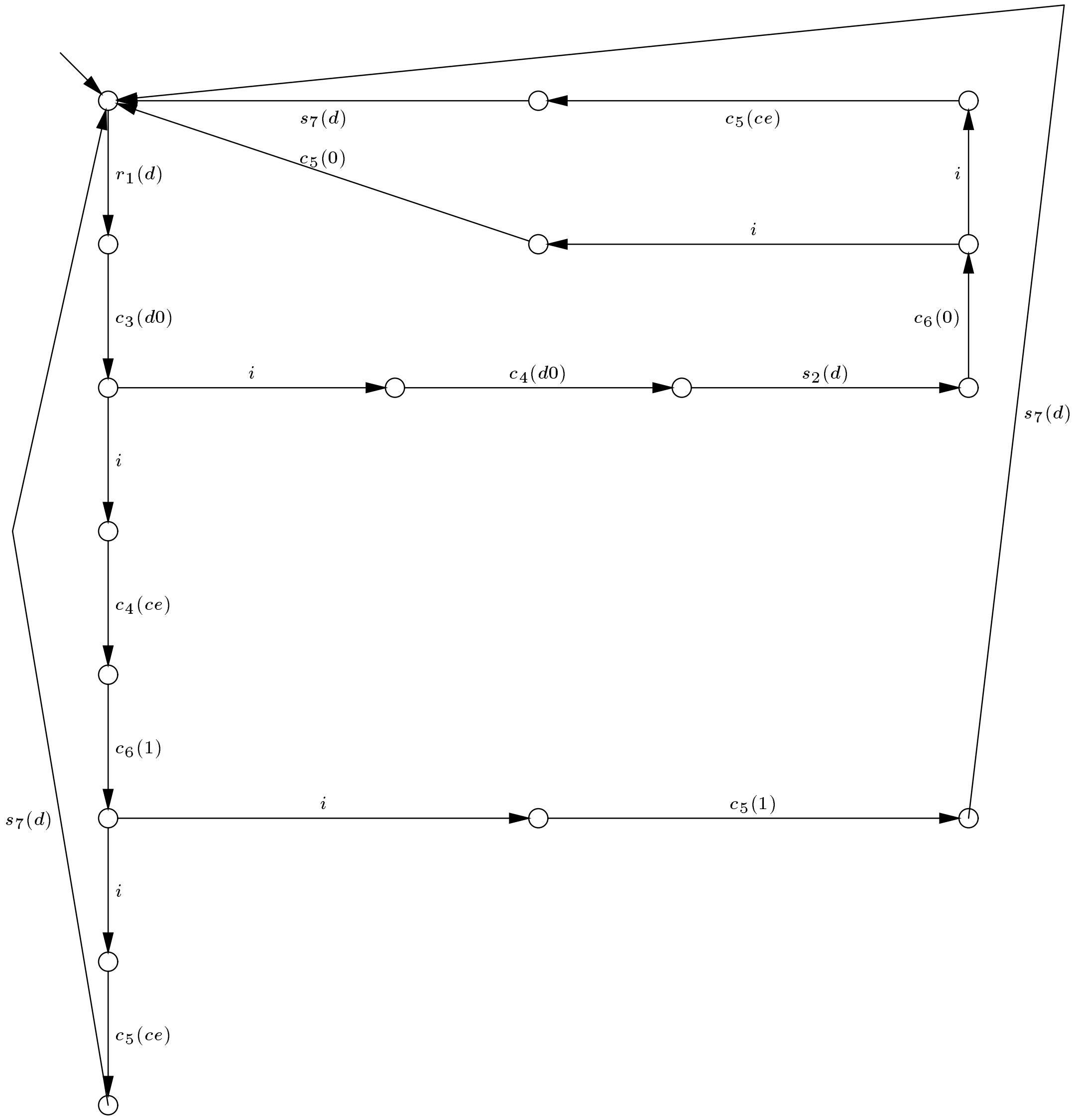

Figure 1: Grafische Weergave van het Alternating Bit Protocol met Time Out ($MAX = 1; B$)

Vervolgens bekijken we mogelijkheid (C) uit de toestandstabel (met MAX = 1). We krijgen de volgende uitwerking:

$\delta_H(ST1_d^{MAX} \cdot S \parallel K \parallel L \parallel R) = \delta_H(ST1_d^1 \cdot S \parallel K \parallel L \parallel R)$

$= \delta_H(s_3(d1) \cdot ((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel$
$\sum_{d \in D} r_3(d1)(i \cdot s_4(d1) + i \cdot s_4(ce)) \cdot K \parallel L \parallel R)$

$= c_3(d1)\{i \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel s_4(d1) \cdot K \parallel L \parallel R) +$
$i \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel R)\}$

$= c_3(d1)\{i \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel s_4(d1) \cdot K \parallel L \parallel R_0^1 \cdot R) +$
$i \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel R_0^1 \cdot R)\}$

$= c_3(d1)\{i \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel s_4(d1) \cdot K \parallel L \parallel$
$((\sum_{d \in D} r_4(d0) + r_4(ce)) \cdot s_6(0) + \sum_{d \in D} r_4(d1) \cdot s_2(d) \cdot s_6(1)) \cdot R) +$
$i \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel$
$((\sum_{d \in D} r_4(d0) + r_4(ce)) \cdot s_6(0) + \sum_{d \in D} r_4(d1) \cdot s_2(d) \cdot s_6(1)) \cdot R)\}$

$= c_3(d1)\{i \cdot c_4(d1) \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel L \parallel s_2(d) \cdot s_6(1) \cdot R) +$
$i \cdot c_4(ce) \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel L \parallel s_6(0) \cdot R)\}$

$= c_3(d1)\{i \cdot c_4(d1) \cdot s_2(d) \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel L \parallel s_6(1) \cdot R) +$
$i \cdot c_4(ce) \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel L \parallel s_6(0) \cdot R)\}$

$= c_3(d1)\{i \cdot c_4(d1) \cdot s_2(d) \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel$
$r_6(1)(i \cdot s_5(1) + i \cdot s_5(ce)) \cdot L \parallel s_6(1) \cdot R) +$
$i \cdot c_4(ce) \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel r_6(0)(i \cdot s_5(0) + i \cdot s_5(ce)) \cdot L \parallel s_6(0) \cdot R)\}$
$= c_3(d1)\{i \cdot c_4(d1) \cdot s_2(d) \cdot c_6(1)\{i \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel s_5(1) \cdot L \parallel R) +$
$i \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel s_5(ce) \cdot L \parallel R)\} +$
$i \cdot c_4(ce) \cdot c_6(0)\{i \cdot \delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel s_5(0) \cdot L \parallel R) +$
$\delta_H(((r_5(0) + r_5(ce)) \cdot s_7(d) + r_5(1)) \cdot S \parallel K \parallel s_5(ce) \cdot L \parallel R)\}$

$= c_3(d1)\{i \cdot c_4(d1) \cdot s_2(d) \cdot c_6(1)\{i \cdot c_5(1) \cdot \delta_H(S \parallel K \parallel L \parallel R) +$
$i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R)\} +$
$i \cdot c_4(ce) \cdot c_6(0)\{i \cdot c_5(0) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R) +$
$i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R)\}\}$

Voor de mogelijkheden (A) en (D), met MAX = 1, vermelden we direct (zonder herschrijvingen) de eindtermen.

(A) $\delta_H(ST0_d^1 \cdot S \parallel K \parallel L \parallel R0^1 \cdot R)$
$\rightarrow c_3(d0)\{i \cdot c_4(d0) \cdot c_6(0)\{i \cdot c_5(0) \cdot \delta_H(S \parallel K \parallel L \parallel R)$
$+ i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R)\}$
$+ i \cdot c_4(ce) \cdot c_6(0)\{i \cdot c_5(0) \cdot \delta_H(S \parallel K \parallel L \parallel R)$
$+ i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R)\}\}$

(D) $\delta_H(ST1_d^1 \cdot S \parallel K \parallel L \parallel R1^1 \cdot R)$
$\rightarrow c_3(d1)\{i \cdot c_4(d1) \cdot c_6(1)\{i \cdot c_5(1) \cdot \delta_H(S \parallel K \parallel L \parallel R)$
$+ i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R)\}$
$+ i \cdot c_4(ce) \cdot c_6(1)\{i \cdot c_5(1) \cdot \delta_H(S \parallel K \parallel L \parallel R)$
$+ i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(S \parallel K \parallel L \parallel R)\}\}$

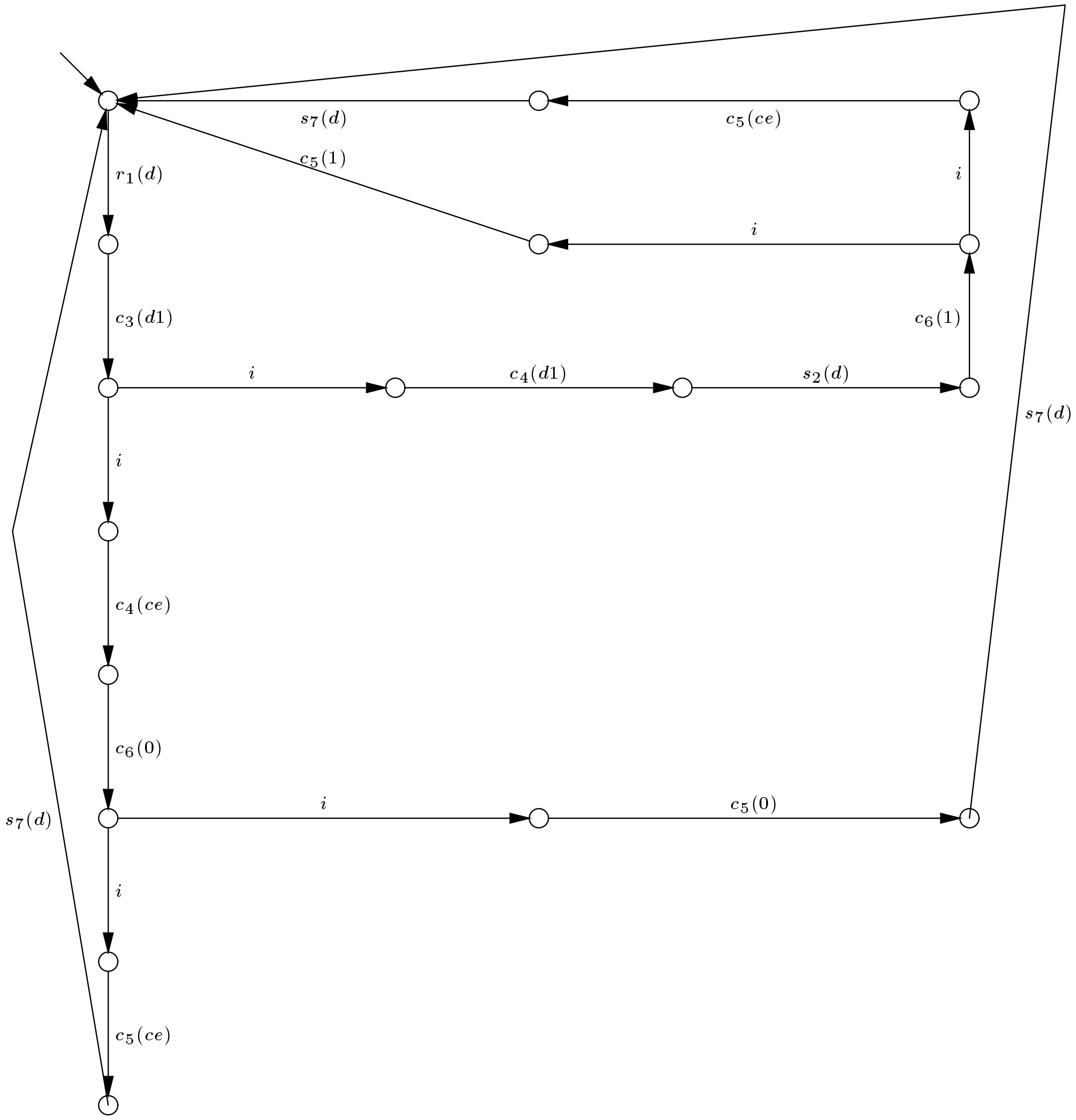

Figure 2: Grafische Weergave van het Alternating Bit Protocol met Time Out $(MAX = 1; C)$

ONDERZOEK MAX > 1.

Wederom bekijken we eerst mogelijkheid (B). Dit is de starttoestand van ons Alternating Bit Protocol met hertransmissies.

$$\delta_H(ST0_d^{MAX} \cdot S \parallel K \parallel L \parallel R) = \delta_H(ST0_d^n \cdot S \parallel K \parallel L \parallel R)$$

$$= \delta_H(s_3(d0) \cdot ((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum_{d \in D} r_1(d) ST1_d^{n-1}) \cdot S \parallel$$
$$\textstyle\sum_{d \in D} r_3(d0)(i \cdot s_4(d0) + i \cdot s_4(ce)) \cdot K \parallel L \parallel R)$$

$$= c_3(d0)\{i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel$$
$$s_4(d0) \cdot K \parallel L \parallel R) + i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel$$
$$s_4(ce) \cdot K \parallel L \parallel R)\}$$

$$= c_3(d0)\{i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel$$
$$s_4(d0) \cdot K \parallel L \parallel R_1^n \cdot R) +$$
$$i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel$$
$$s_4(ce) \cdot K \parallel L \parallel R_1^n \cdot R)\}$$

$$= c_3(d0)\{i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel$$
$$s_4(d0) \cdot K \parallel L \parallel \textstyle\sum r_4(d1) + r_4(ce)) \cdot s_6(1) \cdot R_1^{n-1} +$$
$$\textstyle\sum r_4(d0) \cdot s_2(d) \cdot s_6(0) \cdot R_0^{n-1})R) + i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} +$$
$$r_5(0) \cdot \textstyle\sum r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel \textstyle\sum r_4(d1) + r_4(ce)) \cdot s_6(1) \cdot R_1^{n-1} +$$
$$\textstyle\sum r_4(d0) \cdot s_2(d) \cdot s_6(0) \cdot R_0^{n-1})R)\}$$

$$= c_3(d0)\{i \cdot c_4(d0) \cdot s_2(d) \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel$$
$$K \parallel L \parallel s_6(0) \cdot R_0^{n-1} \cdot R)$$
$$+ i \cdot c_4(ce) \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum_{d \in D} r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel K \parallel L \parallel s_6(1) \cdot R_1^{n-1} \cdot R)\}$$
$$= c_3(d0)\{i \cdot c_4(d0) \cdot s_2(d) \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum_{d \in D} r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel$$
$$K \parallel r_6(0) \cdot (i \cdot s_5(0) + i \cdot s_5(ce)) \cdot L \parallel s_6(0) \cdot R_0^{n-1} \cdot R) +$$
$$i \cdot c_4(ce) \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum r_1(d) \cdot ST1_d^{n-1}) \cdot S \parallel$$
$$K \parallel r_6(1) \cdot (i \cdot s_5(1) + i \cdot s_5(ce)) \cdot L \parallel s_6(1) \cdot R_1^{n-1} \cdot R)\}$$

$$= c_3(d0)\{i \cdot c_4(d0) \cdot s_2(d) \cdot c_6(0)\{i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum_{d \in D} r_1(d) \cdot ST1_d^{n-1}) \cdot S$$
$$\parallel K \parallel s_5(0) \cdot L \parallel R_0^{n-1} \cdot R) +$$
$$i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum_{d \in D} r_1(d) \cdot ST1_d^{n-1}) \cdot S$$
$$\parallel K \parallel s_5(ce) \cdot L \parallel R_0^{n-1} \cdot R)\} +$$
$$i \cdot c_4(ce) \cdot c_6(1)\{i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum_{d \in D} r_1(d) \cdot ST1_d^{n-1}) \cdot S$$
$$\parallel K \parallel s_5(1) \cdot L \parallel R_1^{n-1} \cdot R) +$$
$$i \cdot \delta_H((r_5(1) + r_5(ce)) \cdot ST0_d^{n-1} + r_5(0) \cdot \textstyle\sum_{d \in D} r_1(d) \cdot ST1_d^{n-1}) \cdot S$$
$$\parallel K \parallel s_5(ce) \cdot L \parallel R_1^{n-1} \cdot R)\}\}$$

$$= c_3(d0)\{i \cdot c_4(d0) \cdot s_2(d) \cdot c_6(0)\{i \cdot c_5(0) \cdot \delta_H(\textstyle\sum r_1(d) \cdot ST1_d^{n-1} \cdot S \parallel K \parallel L \parallel R_0^{n-1} \cdot R) +$$
$$i \cdot c_5(ce) \cdot \delta_H(ST0_d^{n-1} \cdot S \parallel K \parallel L \parallel R_0^{n-1} \cdot R)\} +$$
$$i \cdot c_4(ce) \cdot c_6(1)\{i \cdot c_5(1) \cdot \delta_H(ST0_d^{n-1} \cdot S \parallel K \parallel L \parallel R_1^{n-1} \cdot R) +$$
$$i \cdot c_5(ce) \cdot \delta_H(ST0_d^{n-1} \cdot S \parallel K \parallel L \parallel R_1^{n-1} \cdot R)\}\}$$

Vervolgens bekijken we mogelijkheid (C). We krijgen de volgen- de uitwerking:

$\delta_H(ST1_d^{MAX} \cdot S \parallel K \parallel L \parallel R) = \delta_H(ST1_d^n \cdot S \parallel K \parallel L \parallel R)$

$= \delta_H(s_3(d1) \cdot ((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum_{d\in D} r_1(d) ST0_d^{n-1}) \cdot S \parallel$
$\sum_{d\in D} r_3(d1)(i \cdot s_4(d1) + i \cdot s_4(ce)) \cdot K \parallel L \parallel R)$

$= c_3(d1)\{i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel s_4(d1) \cdot K \parallel L \parallel R)+$
$i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel R)\}$

$= c_3(d1)\{i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel s_4(d1) \cdot K \parallel L \parallel R_0^n \cdot R)+$
$i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel R_0^n \cdot R)\}$

$= c_3(d1)\{i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel s_4(d1) \cdot K \parallel L \parallel$
$\sum_{d\in D} r_4(d0) + r_4(ce)) \cdot s_6(0) \cdot R_0^{n-1} + \sum_{d\in D} r_4(d1) \cdot s_2(d) \cdot s_6(1) \cdot R_1^{n-1})R)+$
$i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel s_4(ce) \cdot K \parallel L \parallel$
$\sum_{d\in D} r_4(d1) + r_4(ce)) \cdot s_6(0) \cdot R_0^{n-1} + \sum r_4(d1) \cdot s_2(d) \cdot s_6(1) \cdot R_1^{n-1})R)\}$

$= c_3(d1)\{i \cdot c_4(d1) \cdot s_2(d) \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1}+$
$r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel K \parallel L \parallel s_6(1) \cdot R_1^{n-1} \cdot R)+$
$i \cdot c_4(ce) \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel K \parallel L \parallel s_6(0) \cdot R_0^{n-1} \cdot R)\}$

$= c_3(d1)\{i \cdot c_4(d1) \cdot s_2(d) \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1}+$
$r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel K \parallel r_6(1) \cdot (i \cdot s_5(1) + i \cdot s_5(ce)) \cdot L \parallel s_6(1) \cdot R_1^{n-1} \cdot R)+$
$i \cdot c_4(ce) \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1}+$
$r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel K \parallel r_6(0) \cdot (i \cdot s_5(0) + i \cdot s_5(ce)) \cdot L \parallel s_6(0) \cdot R_0^{n-1} \cdot R)\}$
$= c_3(d1)\{i \cdot c_4(d1) \cdot s_2(d) \cdot c_6(1)\{i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1}+$
$r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel K \parallel s_5(1) \cdot L \parallel R_1^{n-1} \cdot R)+$
$i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel K \parallel s_5(ce) \cdot L \parallel R_1^{n-1} \cdot R)\}+$
$i \cdot c_4(ce) \cdot c_6(0)\{i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1}+$
$r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel K \parallel s_5(0) \cdot L \parallel R_0^{n-1} \cdot R)+$
$i \cdot \delta_H((r_5(0) + r_5(ce)) \cdot ST1_d^{n-1} + r_5(1) \cdot \sum_{d\in D} r_1(d) \cdot ST0_d^{n-1}) \cdot S \parallel K \parallel s_5(ce) \cdot L \parallel R_0^{n-1} \cdot R)\}\}$

$= c_3(d1)\{i \cdot c_4(d1) \cdot s_2(d) \cdot c_6(1)\{i \cdot c_5(1) \cdot \delta_H(\sum_{d\in D} r_1(d) \cdot ST0_d^{n-1} \cdot S \parallel K \parallel L \parallel R_1^{n-1} \cdot R)+$
$i \cdot c_5(ce) \cdot \delta_H(ST1_d^{n-1} \cdot S \parallel K \parallel L \parallel R_1^{n-1} \cdot R)\}+$
$i \cdot c_4(ce) \cdot c_6(0)\{i \cdot c_5(0) \cdot \delta_H(ST1_d^{n-1} \cdot S \parallel K \parallel L \parallel R_0^{n-1} \cdot R)+$
$i \cdot c_5(ce) \cdot \delta_H(ST1_d^{n-1} \cdot S \parallel K \parallel L \parallel R_0^{n-1} \cdot R)\}\}$

Voor de mogelijkheden (A) en (D), met MAX > 1, vermelden we direct (zonder herschrijvingen) de eindtermen.

(A) $\delta_H(ST0_d^n \cdot S \parallel K \parallel L \parallel R0^1 \cdot R)$
$\rightarrow c_3(d0)\{i \cdot c_4(d0) \cdot c_6(0)\{i \cdot c_5(0) \cdot \delta_H(\sum_{d\in D} r_1(d) \cdot ST1_d^{n-1} \cdot S \parallel K \parallel L \parallel R0^{n-1} \cdot R)$
$+i \cdot c_5(ce) \cdot \delta_H(ST0_d^{n-1} \cdot S \parallel K \parallel L \parallel R0^{n-1} \cdot R)\}$
$+i \cdot c_4(ce) \cdot c_6(0)\{i \cdot c_5(0) \cdot \delta_H(\sum_{d\in D} r_1(d) \cdot ST1_d^{n-1} \cdot S \parallel K \parallel L \parallel R0^{n-1} \cdot R)$
$+i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(ST0_d^{n-1} \parallel K \parallel L \parallel R0^{n-1} \cdot R)\}\}$

(D) $\delta_H(ST1_d^n \cdot S \parallel K \parallel L \parallel R1^1 \cdot R)$
$\rightarrow c_3(d1)\{i \cdot c_4(d1) \cdot c_6(1)\{i \cdot c_5(1) \cdot \delta_H(\sum_{d\in D} r_1(d) \cdot ST0_d^{n-1} \cdot S \parallel K \parallel L \parallel R1^{n-1} \cdot R)$
$+i \cdot c_5(ce) \cdot \delta_H(ST1_d^{n-1} \cdot S \parallel K \parallel L \parallel R1^{n-1} \cdot R)\}$
$+i \cdot c_4(ce) \cdot c_6(1)\{i \cdot c_5(1) \cdot \delta_H(\sum_{d\in D} r_1(d) \cdot ST0_d^{n-1} \cdot S \parallel K \parallel L \parallel R1^{n-1} \cdot R)$
$+i \cdot c_5(ce) \cdot s_7(d) \cdot \delta_H(ST1_d^{n-1} \parallel K \parallel L \parallel R1^{n-1} \cdot R)\}\}$

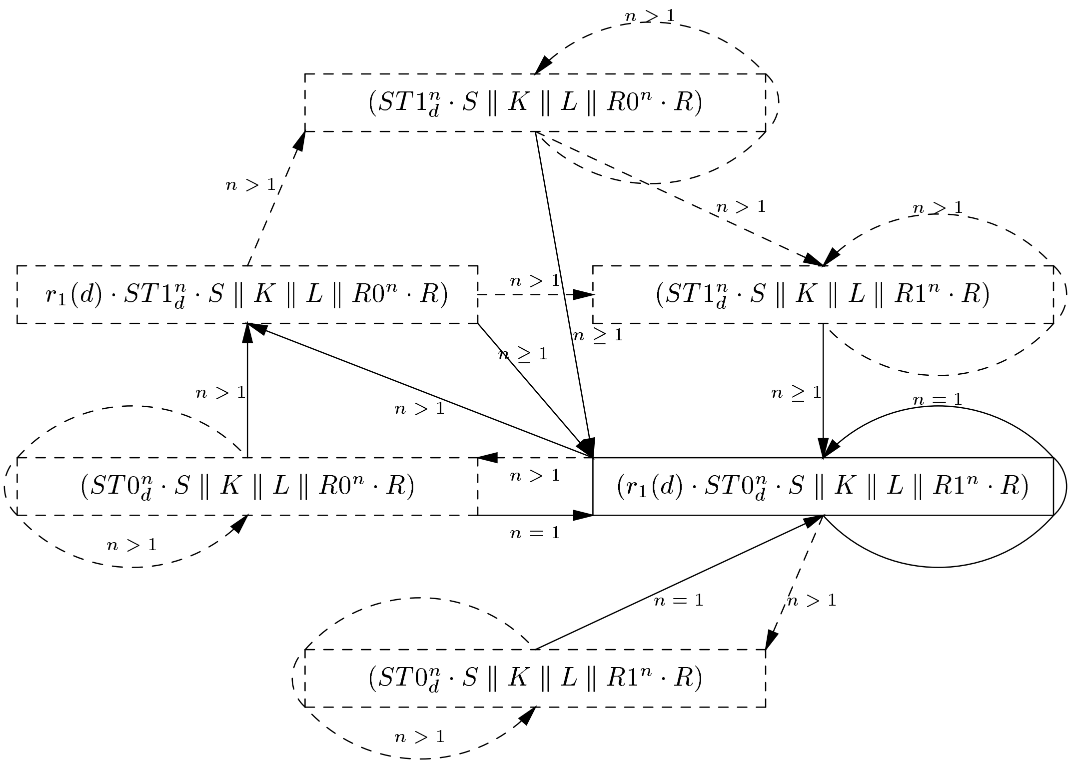

Figure 3: Abstracte Weergave van het Alternating Bit Protocol met Time Out

Verklaring van de toestandsovergangen bij bovenstaande figuur:
$\cdot\cdot \rightarrow$ er wordt geen nieuw datum gelezen van poort 1
$\longrightarrow$ er wordt wel een nieuw datum gelezen van poort 1
Bij elke transitie wordt de waarde van n met 1 verlaagd.

Elke term uit bovenstaande figuur stelt een toestand voor van het Alternating Bit Protocol met Time Out op een bepaald moment (na een aantal zendingen).

We kunnen met behulp van Figuur 3 goed zien dat de processen K en L (data respectievelijk acknowledgement transmission channel) geen directe invloed hebben op de overgangen tussen de hierboven beschreven toestanden. (De kanalen K en L fungeren als doorgeefluik en bepalen alleen of een datum correct of beschadigd wordt doorgegeven.)

De overgangen worden volledig bepaald door de toestand waarin de processen S (zender) en R (ontvanger) zich bevinden en het aantal transmissies n.

We merken op dat de vergelijkingen van de specificatie van het Alternating Bit Protocol (Fig. 3.4) eindig zijn. Dat wil zeggen: na elke MAX transmissies zal het protocol weer in de initiële toestand terugkeren. In de figuur is dit te zien, doordat vanuit elke toestand er een eindig pad bestaat, die eindigt in een toestand, die gelijk is aan de begintoestand.

Door het nalopen van elk pad vanuit de initiële toestand bij een willekeurige waarde van MAX, is na te gaan aan welke vergelijking het Alternating Bit Protocol met Time Out bij een waarde MAX voldoet.

Bij elke toestand uit bovenstaande figuur is 'berekend' welke akties het Alternating Bit Protocol zal nemen en naar welke (nieuwe) toestand het protocol zal gaan. Deze nieuwe toestand is weer een toestand die uit het beschreven protocol. (Er ontstaan geen ongewenste toestanden, waarin het protocol bijvoorbeeld oneindig lang kan doorlopen.)

De enige akties, als we de interne akties buiten beschouwing laten, die het protocol zou kunnen nemen zijn:

- Een read-aktie van poort 1, om een nieuw datum te lezen. Dit gebeurt in ieder geval na elke time out, na elke geslaagde gegevensoverdracht (correcte transmissie van gegevens over zowel kanaal K, als kanaal L) en voor de allereerste data transmissie.
- Een send-aktie over poort 2. Dit gebeurt voor elke datum, dat onbeschadigd ontvanger R weet te bereiken.
- Een send-aktie over poort 7. Een datum wordt alleen geretourneerd over deze uitgang, als de laatste transmissie voor de time out foutief is verlopen.

Hiermee hebben wij aangetoond dat de specificatie, gegeven in hoofdstuk 3 (Fig. 3.4) aan onze eisen eindigheid (het 'resetten' van het protocol na een eindig aantal transmissies: de time out) en signalering (het voorkomen van data-elementen op de uitgangen 2 en 7) voldoet.