Praktische Bewijzen in Public Announcement Logica

M. S. de Boer
s1018833
13 december 2006

begeleiders:
dr B. P. Kooi (Wijsbegeerte R*u*G)
dr L. C. Verbrugge (Kunstmatige Intelligentie R*u*G)

Kunstmatige Intelligentie
Rijks*universiteit* Groningen

Dankwoord

Hier wil ik expliciet bedanken voor hun geduld, input, onderhoud, stimulans, niet-aflatend vertrouwen en/of wijze woorden: Aafke Weller, André van Nieuwenhuijzen, Barteld Kooi, Fred D'Agostino, Gerard Renardel, Gerlof Bouma, Hans van Ditmarsch, Jan-Willem Hiddink, Leon van der Torre, Marijke Onneweer, Pablo Hoving, Rem de Boer en Rineke Verbrugge.

Samenvatting

Analyse van dynamische epistemische situaties, waarbij verandering van kennis van verschillende actoren een rol spelen, vereist goed gereedschap. De doelstelling is hier geweest dit gereedschap verder te ontwikkelen en beschikbaar te maken. In hoofdstuk 1 vindt u een uiteenzetting van een nieuwe gelabelde tableaucalculus voor public announcement logica. Dit bewijssysteem biedt de gelegenheid tot ontwikkeling van intuïtie voor geldigheid van formules in public announcement logic. Ik laat zien hoe in dit bewijssysteem eenvoudige dynamische epistemische situaties geanalyseerd kunnen worden.

In hoofdstuk 2 wordt een nieuwe implementatie voor automatische deductie voor epistemische logica $S5_n$ geïntroduceerd. Het ontwikkelde formalisme is specifiek toegspitst op $S5_n$ toegankelijkheid en biedt perspectief op efficiënte implementatie. Door public announcement logica te vertalen naar epistemische logica kan gefaciliteerd worden in de analyse van eenvoudige dynamische epistemische situaties.

Sleutelwoorden: epistemische logica, dynamische epistemische logica, public announcement logica, openbare-mededelingenlogica, semantische tableaus, gelabelde deductie, drie wijze mannen puzzel, automatische deductie, theorem-proving, leanTAP, prolog.

Inhoudsopgave

Introductie

Representatie van kennis is de hoeksteen van logica gebaseerde kunstmatige intelligentie. Statische kennis kan worden gerepresenteerd in *epistemische logica* [12]. Voor de representatie van *verandering* van kennis kunnen we gebruik maken van *dynamische epistemische logica's* [27]. Deze logica's zijn in actieve ontwikkeling en bieden krachtige representaties voor subtiele aspecten van kennis, communicatie, feitelijke veranderingen en hun interacties. Hier richten we ons op de meest eenvoudige dynamische epistemische logica: *public announcement logica* ofwel 'openbare-mededelingenlogica' [15]. In deze logica zijn bijvoorbeeld dialogen over kennis en hun gevolgen voor epistemische situaties te formaliseren.

Centraal in de logica staan *bewijzen*. Het construeren van *bewijssystemen* is een doel op zichzelf. Verschillende bewijsmethoden bieden zowel praktisch de gelegenheid om de geldigheid van formules te onderzoeken als de gelegenheid tot het ontwikkelen van inzicht in de eigenschappen van de betrokken logische taal en mogelijke gerelateerde talen. Op een snijpunt van logica, kunstmatige intelligentie en informatica ligt onderzoek naar *automatische deductie*, waarbij beslissingsprocedures voor bewijssystemen worden geïmplementeerd in software. Automatische deductie maakt verificatie of productie van bewijzen geregeld—zeker niet altijd—sneller en accurater dan een menselijke bewijzer dit kan. Automatische stellingbewijzers kunnen zo een assisterende rol spelen bij het maken van formalisaties. Implementatie kan ook tot inzicht leiden in de eigenschappen van een bewijssysteem en als inspiratie dienen voor de ontwikkeling van nieuwe bewijssystemen en logica's. Hier richten we ons op de ontwikkeling en implementatie van *semantische-tableaubewijssystemen* [17] voor epistemische logica $S5_n$ en public announcement logica. Een eenvoudige implementatie van epistemische logica is hier tenvolste uitgewerkt. Automatisering van public announcement logica is dit niet.

Hoewel de representatie van epistemische situaties in formules in public announcement logica goed is ontwikkeld is de analyse van deze formules tot nu toe problematisch. Constructie van bewijzen in het bestaande Hilbert-

stijl bewijssysteem is moeizaam en biedt geen perspectief op automatiseren. Omdat de struktuur van formules in dit bewijssysteem verloren gaat in een bewijs geeft een bewijs (of een poging daartoe) geen zicht op de semantiek van formules. De omstandigheden waaronder zich een epistemische situatie voor kan doen worden zo niet opgehelderd. Dit probleem wordt in hoofdstuk 1 opgelost door de semantiek van public announcement logica expliciet te representeren in een bewijssysteem. De doelstellingen zijn hier geweest 1) een eenvoudige, simpele, praktische methode te ontwikkelen voor het construeren van bewijzen in public announcement logica en 2) inzicht te verschaffen in de geldigheid van formules in public announcement logica. Dit 'papieren' bewijssysteem borduurt voort op de standaard tableauregels voor propositielogica, het tableausysteem KE [3] en de gelabelde tableaus voor modale logica's in [5]. De uitbreiding naar de multi-modale logica en de sprong naar openbare mededelingen zijn hier nieuw. Het ontwikkelde systeem blijkt bijzonder effectief voor de analyse van bepaalde tegen-intuitieve epistemische ontwikkelingen waarbij kennis volgt uit opeenvolgingen van mededelingen van onwetendheid.

Een belangrijk concept in de epistemische logica is *common knowledge*. Er is sprake van common knowledge in de bijzondere onstandigheid dat *iedereen* iets weet en ook iedereen weet dat iedereen het weet, iedereen weet dat iedereen het weet etc. Dit concept is goed ontwikkeld in de (dynamische) epistemische logica waarin expliciet geredeneerd kan worden met common knowledge. Omdat het ontbreekt aan ontwikkeling van common knowledge in de hier gebruikte gelabelde tableaubewijssystemen komt dit aspect van epistemische logica hier niet aan bod.

In de kunstmatige intelligentie ligt de nadruk geregeld op *implementatie* van formalismen. Omdat semantische tableaus zich goed lenen voor implementatie en hier veel bestaande resultaten over beschikbaar zijn is in hoofdstuk 2 een stellingbewijzer voor $S5_n$ geïmplementeerd in prolog geïnspireerd op het bewijssysteem uit hoofdstuk 1. De geïmplementeerde beslissingsprocedure is geïnspireerd op leanT^{A}P[2], ModLeanTAP [1] en KEM [11]. De doelstellingen is hier geweest te faciliteren in het zoeken van logische representaties voor epistemische situaties en het ontwikkelen van inzicht op dit gebied. Door zeer specifiek voor logica $S5n$ te ontwikkelen is in ieder geval een aanzet gegeven naar efficiënte automatische deductie voor epistemische logica. Er is niet alleen een kaal algoritme ontwikkeld maar er is ook een eerste aanzet gegeven tot een toegankelijk systeem. Hiertoe zijn een aantal hulpprogramma's ontwikkeld en is de implementatie online aanspreekbaar op `http:www.ai.rug.nl~mathijs/logic/piel.html`.

Hoofdstuk 1

Semantische Tableaus voor Public Announcement Logica

Public announcement logic ofwel 'Openbare-Mededelingenlogica' is een uitbreiding op de standaard epistemische logica EL ($S5_n$) met een modale 'update' operator $[\bullet]$. De formule $[\varphi]\psi$ kunnen we lezen als 'na de openbare mededeling van φ geldt ψ'. Net als dat epistemische logica een idealisering is van redeneren over kennis, met agents als perfecte logici, is het redeneren over deze zogenaamde *kennisacties* hier ook zeer geïdealiseerd. We kunnen ons deze voorstellen als ware, luide en duidelijke mededelingen (φ) waarbij alle aanwezigen erg goed opletten: nu weet niet alleen iedereen van de mededeling van φ maar ook iedereen weet dat iedereen weet van bericht φ enzovoorts. Kennisnemen van bericht φ is niet gelijk aan weten *dat* φ, dit is conform de intuïtie dat na het bericht "jij weet niet dat p en p is waar" je nu wel weet dat p en je weet dat het originele bericht *onwaar* is geworden. Zulke constructies noemen we *onsuccesvolle updates*, deze zijn zeer goed te formaliseren in PAL [6, 25]. Hoewel dynamische uitbreidingen van (epistemische) logica's snel erg complex kunnen worden is public announcement logica (zonder common knowledge[1]) te reduceren tot gewone epistemische logica [15]. Zo zijn formules in PAL te *vertalen* naar formules zonder updates met behulp van Hilbert-stijl reductie-axioma's.

Semantische tableaus zijn een intuïtieve en praktische manier om tegen bewijsbaarheid van logische formules aan te kijken. Met semantische tableaus voor epistemische logica kunnen we niet alleen een bewijs vinden voor een formule φ maar ook epistemische modellen construeren waarin deze vervulbaar is. Met semantische tableaus voor PAL kunnen niet alleen initiële

[1]Common knowledge wordt in deze totale verhandeling niet besproken. Dit is erg jammer en het behoeft eigenlijk enige verdediging om hier toch van epistemische logica te spreken.

modellen geconstrueerd worden maar ook direct de mogelijke sub-modellen die deel uitmaken van de semantiek van PAL, wat op zichzelf geen triviale taak is. Hiervoor was voorheen geen systematische procedure beschikbaar.

De Drie Wijze Mannen Puzzel is een oud raadsel dat geregeld wordt aangewend om de expressiviteit van logische formalismen te illustreren en zo gebeurt dit ook hier.

In paragraaf 1.1 wordt Public Announcement Logica formeel geïntroduceerd en in paragraaf 1.2 wordt de Hilbert-stijl axiomatisering gegeven waarmee PAL te reduceren is tot epistemische logica. In paragraaf 1.3 geef ik een iets afwijkend tableaubewijssysteem voor epistemische logica $S5_n$ gebaseerd op [5] met een aantal prettige eigenschappen. In paragraaf 1.4 wordt het tableaubewijssysteem uiteengezet voor openbare mededelingen logica en wordt correctheid en volledigheid van dit systeem aangetoond. In paragraaf 1.5 wordt ter illustratie de Drie Wijze Mannen Puzzel geanalyseerd met behulp van de hier ontwikkelde theorie.

1.1 Taal en semantiek

Modale logica met mogelijke-wereldensemantiek als epistemische logica vindt zijn oorsprong in *Knowledge and Belief*, Jaakko Hintikka (1962)[2] [12]. Uitbreiding van epistemische logica met publieke mededelingen begon zijn ontwikkeling in *Logics of Public Communications*, Jan Plaza (1989) [15]. De hier gebruikte notaties en conventies komen uit *Dynamic Epistemic Logic*, Van der Hoek, Van Ditmarsch en Kooi [27].

Definitie 1.1 (Taal voor public announcement logica PAL) Laat P een eindige verzameling propositionele variabelen zijn en A een eindige verzameling agents. Een Backus Naur Form (BNF) regel voor de taal voor openbare mededelingen $\mathcal{L}_{\mathsf{PAL}(PA)}$:

$$\varphi \ ::= \ \bot \ \mid \ p \ \mid \ \neg\varphi \ \mid \ (\varphi_1 \wedge \varphi_2) \ \mid \ \Box_i\varphi \ \mid \ [\varphi_1]\varphi_2$$

met $p \in P$ en $i \in A$. De formule $\Diamond_i\varphi$ is een afkorting voor $\neg\Box_i\neg\varphi$, $\top$ voor $\neg\bot$ (true/false), $(\varphi \vee \psi)$ voor $\neg(\neg\varphi \wedge \neg\psi)$, $(\varphi \rightarrow \psi)$ voor $\neg(\varphi \wedge \neg\psi)$ en $\langle\varphi\rangle\psi$ voor $\neg[\varphi]\neg\psi$. Haakjes () laat ik weg als dit de leesbaarheid ten goede komt. □

Definitie 1.2 (Epistemische modellen) Een model $\mathcal{M} = \langle W, R, V\rangle$ bestaat uit:

[2]Hintikka ziet Von Wright als de grondlegger van de epistemische logica.

- $W \neq \emptyset$ een verzameling mogelijke werelden,
- $R : A \to 2^{(W \times W)}$ toegankelijkheid $R(i)$ voor iedere agent i,
- $V : W \times P \to \{0, 1\}$ valuatie van propositionele variabelen per wereld.

We schrijven $w \sim_i w'$ of wR_iw' als $(w, w') \in R(i)$: 'w' is toegankelijk voor agent i vanuit w'. $(\mathcal{M}, w)$ noemen we een *gericht* (*pointed* En.) model met $w \in W$. □

Definitie 1.3 (Semantiek voor PAL) Zij gegeven epistemisch model $(\mathcal{M}, w)$ met $\mathcal{M} = \langle W, R, V\rangle$ en $w \in W$, $p \in P$, $i \in A$ en φ, $\psi \in \mathcal{L}_{\mathsf{PAL}(PA)}$.

$$
\begin{array}{lll}
\mathcal{M}, w \not\models \bot & & \\
\mathcal{M}, w \models p & :\Leftrightarrow & V_w(p) = 1 \\
\mathcal{M}, w \models \neg\varphi & :\Leftrightarrow & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & :\Leftrightarrow & \mathcal{M}, w \models \varphi \text{ en } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \Box_i\varphi & :\Leftrightarrow & \text{voor alle } v \in W \text{ met } w \sim_i v \text{ geldt } \mathcal{M}, v \models \varphi \\
\mathcal{M}, w \models \Diamond_i\varphi & :\Leftrightarrow & \text{er is een } v \in W \text{ met } w \sim_i v \text{ en } \mathcal{M}, v \models \varphi \\
\mathcal{M}, w \models [\varphi]\psi & :\Leftrightarrow & \mathcal{M}, w \models \varphi \text{ impliceert } \mathcal{M}|\varphi, w \models \psi \\
\mathcal{M}, w \models \langle\varphi\rangle\psi & :\Leftrightarrow & \mathcal{M}, w \models \varphi \text{ en } \mathcal{M}|\varphi, w \models \psi
\end{array}
$$

Het model $\mathcal{M}|\varphi := \langle W', R', V'\rangle$ is als volgt gedefinieerd:

$$
\begin{array}{lll}
W' & := & \{w' \in W \mid \mathcal{M}, w' \models \varphi\} \\
R'(i) & := & R(i) \cap (W' \times W') \\
V'_w(p) & := & V_w(p) \upharpoonright W'
\end{array}
$$

□

Het model $\mathcal{M}|\varphi$ bevat dus precies alle mogelijke werelden uit $\mathcal{M}$ waar φ geldt en precies alle toegankelijkheidsrelaties tussen die werelden uit $\mathcal{M}$. Het 'uitvoeren' van (opeenvolgingen van) updates in een *gegeven* model $\mathcal{M}$ is zo dus een eliminatiespel; in de literatuur komen diverse voorbeelden aan bod waarin dit gespeeld wordt, onder andere [27, 15, 20, 26].

1.2 Axiomatisering

Epistemische logica betreft hier uitsluitend de als $S5_n$ aangeduide klasse epistemische modellen met reflexieve, transitieve en symmetrische toegankelijkheidssrelaties; zo gaan we ervan uit dat de kennis van agents op feitelijke waarheid berust en dat agent zowel weet hebben van hun eigen kennis als weten wat zij voor mogelijk houden. Voor een uiteenzetting van modale logica's K tot $S5$ en hun framecondities zie onder andere [5, 23].

Definitie 1.4 (bewijssysteem PAL_{PA}) Zij gegeven zinnen φ, ψ, χ in taal $\mathcal{L}_{\mathsf{PAL}(PA)}$, i een agent in A en p een propositieletter in P.

Multi-$S5$ axioma's:

Taut	alle propositionele tautologieën	
Distr	$\vdash \Box_i(\varphi \to \psi) \to (\Box_i\varphi \to \Box_i\psi)$	(distributie)
T	$\vdash \Box_i\varphi \to \varphi$	(waarheid)
4	$\vdash \Box_i\varphi \to \Box_i\Box_i\varphi$	(positieve introspectie)
5	$\vdash \Diamond_i\varphi \to \Box_i\Diamond_i\varphi$	(negatieve introspectie)

PAL reductieaxioma's:

AP	$\vdash [\varphi]p \leftrightarrow (\varphi \to p)$	(atomen)
PF	$\vdash [\varphi]\neg\psi \leftrightarrow (\varphi \to \neg[\varphi]\psi)$	(partiële functie)
Dist	$\vdash [\varphi](\psi \wedge \chi) \leftrightarrow ([\varphi]\psi \wedge [\varphi]\chi)$	(distributie)
AK	$\vdash [\varphi]\Box_i\psi \leftrightarrow (\varphi \to \Box_i[\varphi]\psi)$	(kennis-update)
AC	$\vdash [\varphi][\psi]\chi \leftrightarrow [\varphi \wedge [\varphi]\psi]\chi$	(gecombineerde updates)

regels:

MP	als $\vdash \varphi$ en $\vdash \varphi \to \psi$ dan $\vdash \psi$	(modus ponens)
Nec	als $\vdash \varphi$ dan $\vdash \Box_i\varphi$	(necessitatie voor $\Box_i$)
Nec	als $\vdash \varphi$ dan $\vdash [\psi]\varphi$	(necessitatie voor $[\psi]$)

□

Definitie 1.5 (een bewijs) Een bewijs in PAL_{PA} Voor een formule φ in $\mathcal{L}_{\mathsf{PAL}(PA)}$ is een eindige reeks van formules $\varphi_1, \varphi_2, ..., \varphi_n = \varphi$ waarbij ieder φ_i met $1 \leq i \leq n$ of een instantie van een axioma in PAL_{PA} is of de conclusie van een regel in PAL_{PA} waarvan de premissen eerder in de reeks voorkomt. Als φ bewijsbaar is in PAL_{PA} dan schrijven we $\vdash_{\mathsf{PAL}_{PA}} \varphi$. □

Correctheid en volledigheid van dit bewijssysteem worden uitgebreid besproken in [27]. Daar passeren ook diverse illustratieve voorbeelden de revue.

Het staat de bewijzer natuurlijk vrij de regels en axioma's uit definitie 1.4 door elkaar heen te gebruiken maar formules in PAL zijn door toepassing van de reductieaxioma's terug te vertalen naar formules zonder updates; zoals in het volgende voorbeeld. Wat hier volgt is geen bewijs maar een vertaling van een update naar een representatie in epistemische logica zonder updates.

Voorbeeld 1.1 (update reductie)
We onderzoeken de formule

$$[p \wedge \neg\Box_a p](p \wedge \neg\Box_a p).$$

Aan de hand van axioma **Dist** kunnen we deze formule gelijk stellen aan

$$[p \wedge \neg\Box_a p]p \wedge [p \wedge \neg\Box_a p]\neg\Box_a p.$$

In verband met axioma **AP** kunnen we dit weer gelijk stellen aan

$$((p \wedge \neg\Box_a p) \rightarrow p) \wedge [p \wedge \neg\Box_a p]\neg\Box_a p.$$

Door zo stelselmatig deelformules te vervangen door hun reducties reduceren we stap voor stap met **PF**, **AK**, **AP**:

$$((p \wedge \neg\Box_a p) \rightarrow p) \wedge ((p \wedge \neg\Box_a p) \rightarrow \neg[p \wedge \neg\Box_a p]\Box_a p),$$

$$((p \wedge \neg\Box_a p) \rightarrow p) \wedge ((p \wedge \neg\Box_a p) \rightarrow \neg((p \wedge \neg\Box_a p) \rightarrow \Box_a[p \wedge \neg\Box_a p]p)),$$

en we houden een formule zonder updates over:

$$((p \wedge \neg\Box_a p) \rightarrow p) \wedge ((p \wedge \neg\Box_a p) \rightarrow \neg((p \wedge \neg\Box_a p) \rightarrow \Box_a((p \wedge \neg\Box_a p) \rightarrow p))).$$

□

Uit voorbeeld 1.1 wordt duidelijk dat zelfs achter een vrij eenvoudige PAL formule een nogal ingewikkelde formule in gewone epistemische logica schuil gaat. De formule is overigens ongeldig (contingent), na de mededeling van p weet je immers dat p. Om de geldigheid van de formule $[p \wedge \neg\Box_a p](p \wedge \neg\Box_a p)$ te onderzoeken kunnen we ons richten op de formule $((p \wedge \neg\Box_a p) \rightarrow p) \wedge ((p \wedge \neg\Box_a p) \rightarrow \neg((p \wedge \neg\Box_a p) \rightarrow \Box_a((p \wedge \neg\Box_a p) \rightarrow p)))$.

Hoewel zo de suggestie gewekt zou kunnen zijn dat de publieke mededelingen operator niet het bestuderen waard is toont C. Lutz aan dat zekere formules in PAL alleen exponentieel veel langer gerepresenteerd kunnen worden in EL [13].

Het valt direct op dat het laatste reductieaxioma **AC** strikt genomen niet nodig is om PAL formules te vertalen. Stel de vertaalfunctie is t, $t([\varphi][\psi]\chi) =$

$t([\varphi]t([\psi]\chi))$. Het is eenvoudig te bewijzen dat deze stap toegestaan is binnen het gegeven bewijssysteem en zo is volledigheid af te leiden zonder reductie axioma **AC**. We zullen echter zien dat in een (gelabeld) tableaubewijssysteem een dergelijke ingreep niet te plegen is en in het volledigheidsbewijs voor het te presenteren tableausysteem wordt juist enige moeite getroost de algemeengeldigheid van $[\varphi][\psi]\chi \rightarrow [\varphi \wedge [\varphi]\psi]\chi$ af te leiden in een tableau.

1.3 Semantische tableaus voor epistemische logica

De semantische tableaus hier gepresenteerd zijn een extensie van [5] en maken gebruik van *gelabelde* formules. Voor een introductie van deze specifieke tableaubewijssystemen zie [5, 8, 22]; daar worden echter alleen mono-modale logica's behandeld. Een sterk van het hier gepresenteerde systeem afwijkende uiteenzetting van gelabelde tableaus voor multi-modal logica's is Governatori's systeem KEM [10].

Enige terminologie: tableaubewijzen zijn *bomen* van gelabelde formules; deze zijn de *knopen* van de boom. De eerste knoop noemen we de *wortel* of *root* en deze staat bovenaan. "Tableaus staan botanisch gezien dus onderstboven (*vert. p.47* [5])." De boom wordt opgebouwd door *tableau-extensie-regels* toe te passen op de knopen. Knopen bovenin (dichter bij de wortel) zijn *ouders* van de lager gelegen *kinderen*. Een *blad* is een knoop zonder kinderen. Een *tak* van een tableau is een pad van de wortel naar een blad. Een knoop staat *op* een tak als deze in zijn pad ligt. Tableauregels maken een tak langer door er onderaan knopen aan toe te voegen of *vorken* de tak in tweeën en voegen onderaan beide nieuwe takken knopen toe.

Definitie 1.6 (labels) Een label is een eindige opeenvolging van natuurlijke getallen alle met een (maar mogelijk leeg) agent-subscript. Een gelabelde formule is van de vorm $\{\sigma \ \ \varphi\}$ met σ het label en φ de formule. Een label $....n_i$ noemen we een i-label. □

Labels representeren *paden* door epistemische modellen, opgebouwd uit de toegankelijkheidsrelaties voor verschillende agents. Een gelabelde formule $1.1_i.3_j \ a$ kunnen we lezen als $w_1 \sim_i w_2 \sim_j w_3$ voor werelden, agents en toegankelijkheid in een model $(\mathcal{M}, w_1)$ met $\mathcal{M}, w_3 \models a$. Deze labels zijn dus anders dan de labels die gebruikt worden in sematische tableaus voor modale logica in [22]. Daar wordt de toegankelijkheid tussen mogelijke-wereldlabels bepaald door *takkenbundels*, deelbomen van tableaus (zie *def.13.10 p.206* [22]) terwijl die hier expliciet in labels gerepresenteerd is.

Definitie 1.7 (sluiting) Een tableautak is *gesloten* als hij zowel de knopen $\sigma\ \psi$ als $\sigma\ \neg\psi$ bevat. Anders is een tak *open*. Een tableau is gesloten als al zijn takken gesloten zijn. Een gesloten tak geven we aan met $=$ en een open tak met $\uparrow$ onder het blad. □

Definitie 1.8 (tableaubewijs) Een gesloten tableau met wortel $1\ \neg\varphi$ is een *tableaubewijs* voor de formule φ. Als er een tableaubewijs is voor φ dan is φ *geldig*. Als een tak van het tableau voor $1\ \neg\varphi$ niet gesloten kan worden door het toepassen van tableauregels dan is de open tak een *tegenvoorbeeld* voor φ. □

Definitie 1.9 (propositielogische tableauregels)

$$\frac{\sigma\ \ \neg\neg\varphi}{\sigma\ \ \varphi}$$ dubbele negatie

$$\frac{\sigma\ \ \varphi\wedge\psi}{\begin{array}{l}\sigma\ \ \varphi\\ \sigma\ \ \psi\end{array}} \qquad \frac{\sigma\ \ \neg(\varphi\vee\psi)}{\begin{array}{l}\sigma\ \ \neg\varphi\\ \sigma\ \ \neg\psi\end{array}} \qquad \frac{\sigma\ \ \neg(\varphi\rightarrow\psi)}{\begin{array}{l}\sigma\ \ \varphi\\ \sigma\ \ \neg\psi\end{array}}$$ conjunctie

$$\frac{\sigma\ \ \varphi\vee\psi}{\sigma\ \ \varphi \mid \sigma\ \ \psi} \qquad \frac{\sigma\ \ \neg(\varphi\wedge\psi)}{\sigma\ \ \neg\varphi \mid \sigma\ \ \neg\psi} \qquad \frac{\sigma\ \ \varphi\rightarrow\psi}{\sigma\ \ \neg\varphi \mid \sigma\ \ \psi}$$ disjunctie

□

Voor de duidelijkheid zijn in definitie 1.9 de regels voor alle gebruikelijke propositionele operatoren expliciet gegeven (behalve $(\varphi\leftrightarrow\psi)\equiv(\neg\varphi\wedge\neg\psi)\vee(\varphi\wedge\psi)$). Er is een alternatief tableausysteem met maar één enkele vorkende regel: KE van D'Agostino[3] en Mondadori [3]. Ik geeft deze regels óók, in verkortte vorm. De meest rechter regels is het zg. bivalentieprincipe. Het maakt voor de $S5_n$ tableauregels in definitie 1.11 *niet* uit welke propositionele regels we gebruiken maar voor de komende tableaus voor PAL *wel*. In voorbeelden gebruik ik steeds de regels uit definitie 1.9 tenzij expliciet vermeldt.

Definitie 1.10 (propositielogische KE tableauregels)

$$\frac{\sigma\ \ \neg\neg\varphi}{\sigma\ \ \varphi} \qquad \frac{\sigma\ \ \varphi\wedge\psi}{\begin{array}{l}\sigma\ \ \varphi\\ \sigma\ \ \psi\end{array}} \qquad \frac{\begin{array}{l}\sigma\ \ \varphi\vee\psi\\ \sigma\ \ \neg\varphi\end{array}}{\sigma\ \ \psi} \qquad \frac{\begin{array}{l}\sigma\ \ \varphi\vee\psi\\ \sigma\ \ \neg\psi\end{array}}{\sigma\ \ \varphi} \qquad \frac{}{\sigma\ \ \neg\varphi \mid \sigma\ \ \varphi}$$

□

[3]Dit is Marcello D'Agostino, niet Fred D'Agostino.

Definitie 1.11 ($\mathsf{S5_n}$ tableauregels)

$$\mathbf{M}\ \frac{\sigma \quad \Diamond_i\varphi}{\sigma.n_i \quad \varphi}\ \sigma.n_i \text{ nieuw op de tak}\ \frac{\sigma.k_i \quad \Diamond_i\varphi}{\sigma.n_i \quad \varphi}\ \mathbf{M{=}}$$

$$\mathbf{K}\ \frac{\sigma \quad \Box_i\varphi}{\sigma.h_i \quad \varphi}\ \sigma.h_i \text{ en } \sigma \text{ niet nieuw op de tak}^4\ \frac{\sigma.k_i \quad \Box_i\varphi}{\sigma \quad \varphi}\ \mathbf{R{=}}$$

$$\mathbf{T}\ \frac{\sigma \quad \Box_i\varphi}{\sigma \quad \varphi}\ \sigma.h_i \text{ niet nieuw op de tak}\ \frac{\sigma.k_i \quad \Box_i\varphi}{\sigma.h_i \quad \varphi}\ \mathbf{K{=}}$$

Dien ten verstande dat label σ steeds *géén* i-label is en $\Diamond_i\varphi \equiv \neg\Box_i\neg\varphi$. □

De tableauregels in definitie 1.11 zijn een uitbreiding van het bewijssysteem voor $S5$ uit [5]. Ze staan hier op een wat ongebruikelijke manier bij elkaar, dit is een nieuw bewijssyteem voor $S5_n$. De tableauregels in definities 1.9 en 1.11 vormen samen het bewijssysteem ELtab (1.10 en 1.11 moeten samen natuurlijk KEEL genoemd worden). Als we in een ELtab tableau in een i-label een i-formule ($\Box_i\varphi$, $\Diamond_i\varphi$, $\neg\Box_i\varphi$ of $\neg\Diamond_i\varphi$) analyseren dan moeten de rechter '=' regels toegepast worden, in alle ander gevallen moeten de linker regels worden toegepast. Voor een formule $\Diamond_i\varphi$ is er dus steeds één mogelijke analyse waarbij een nieuw symbool n_i geïntroduceerd wordt (te noemen *introductie* van n_i). Voor een formule $\Box_i\varphi$ zijn er steeds twee regels toepasbaar: **K** en **T** of **K=** en **R=**. De **K** en **K=** regels selecteren een bestaand label $\sigma.h_i$ op dezelfde tableautak (te noemen *selectie* van h_i). Deze regels zijn dus mogelijk toepasbaar met verschillende h_i en het is aan de bewijzer de juiste h_i te selecteren om (uiteindelijk, als dit mogelijk is) het tableau te sluiten.

Voor mono-modale logica $S5$ geven Fitting en Mendelsohn [5] twee tableausystemen, de modulaire tableauregels voor $S5$ zijn de hier gegeven regels **M**, **K** en **T** + **4** en **4r**. Hier zijn de regels **4** en **4r** voor multi-modale logica:

$$\mathbf{4}\ \frac{\sigma \quad \Box_i\varphi}{\sigma.h_i \quad \Box_i\varphi} \qquad \mathbf{4r}\ \frac{\sigma.n_i \quad \Box_i\varphi}{\sigma \quad \Box_i\varphi}$$

Omdat we hier niet te maken hebben met interactie-axioma's volgt correctheid en volledigheid voor het $S5_n$ tableaubewijssysteem met regels **M**, **K**, **T**, **4** en **4r** direct uit het systeem voor $S5$ met deze regels[5]. Laten we dit systeem nu KT45tab noemen.

[4] Hier nu is σ *altijd* een bestaand label maar deze formulering komt direct nog van pas.

[5] Een mogelijk interactie-axioma is bijvoorbeeld $\vdash \Box_1\Box_2\varphi \leftrightarrow \Box_2\Box_1\varphi$. Multi-modale logica's met interactie-axioma's worden onder andere besproken in [18].

Het veel eenvoudigere en handigere alternatieve systeem van Fitting en Mendelsohn voor $S5$ bestaat uit de regels hier aangegeven als **K=** en **M=**, voor mono-modale logica als volgt:

$$\mathbf{M}{=}_1 \quad \frac{k \quad \Diamond\varphi}{n \quad \varphi} \qquad\qquad \mathbf{K}{=}_1 \quad \frac{k \quad \Box\varphi}{h \quad \varphi} \qquad n \text{ nieuw en } h \text{ een bestaand label.}$$

Dit systeem, hier te noemen S5tab, maakt gebruik van de altijd directe toegankelijkheid van alle mogelijke werelden in een $S5$ model, labels zijn voor $S5$ tableaus dan ook gedefinieerd als enkele getallen in plaats van opeenvolgingen van getallen. Voor enkele agents in $S5_n$ maken we nu ook gebruik van deze regels maar we hebben extra regels nodig om de toegankelijkheid voor verschillende agents te administreren.

De enige nieuwe regel in ELtab, **R=** kunnen we zien als toepassing van **4r**+**T**, deze regel is dus correct voor $S5_n$. Als we nu kijken naar de regels **4** en **4r** dan kunnen we zien dat deze *redundant* zijn geworden. Een formule op een tableautak is altijd het product van nul (de wortel) of meer toepassingen van tableauregels. Ik laat zien dat in iedere reeks van toegepaste regels we altijd de toepassing van **4** en **4r** kunnen vervangen door toepassing van de regels in ELtab. ieder KT45tab bewijs is zo ook te bewijzen in ELtab. We onderscheiden de volgende gevallen:

- Als we een tableautak kunnen sluiten door toepassing van **4**, d.w.z. we hebben formules $\sigma\ \Box_i\varphi$ en $\sigma.n_i\ \neg\Box_i\varphi$ op de tak dan kunnen we de tak ook sluiten door toepassing van **M=** op $\sigma.n_i\ \neg\Box_i\varphi$ en **T** op $\sigma\ \Box_i\varphi$.

- Als we een tableautak kunnen sluiten door toepassing van **4r**, d.w.z. we hebben formules $\sigma.n_i\ \Box_i\varphi$ en $\sigma\ \neg\Box_i\varphi$ op de tak dan kunnen we de tak ook sluiten door toepassing van **M** op $\sigma\ \neg\Box_i\varphi$ en **R=** op $\sigma.n_i\ \Box_i\varphi$.

- Toepassing van **4**+**4** leidt tot labels van de vorm $\sigma.n_i.m_i$, deze hebben we niet nodig in bewijzen. Een bewijs dat sluit op label $\sigma.n_i.m_i\Phi$ met σ en 1Φ labels kunnen we in ELtab altijd sluiten op label $\sigma.m_i\Phi$. Zowel bij selectie als introductie van m_i hadden we immers met regels **M=** en **K=** $\sigma.n_i$ kunnen reduceren tot $\sigma.m_i$.

- Toepassing van **4r**+**4r** is juist alleen mogelijk met labels van deze vorm $\sigma.n_i.m_i$.

- Toepassing van **4**+**4r** of **4r**+**4** leidt weer precies terug tot dezelfde gelabelde formule.

- Toepassing van **4**+**R=** is gelijk aan toepassing van **T**.

- Toepassing van **4**+**K**= is gelijk aan toepassing van **K**.
- Toepassing van **4r**+**K** is gelijk aan toepassing van **K**=,
- Toepassing van **4r**+**T** is gelijk aan toepassing van **R**=.

Verdere combinaties zijn niet mogelijk. De regels **4** en **4r** zijn dus inderdaad redundant voor ELtab en alle formules bewijsbaar in KT45tab zijn ook bewijsbaar in ELtab.

Het bewijssysteem ELtab heeft iets verloren aan algemeenheid maar is op zichzelf staand een eindige beslissingsprocedure. Het is duidelijk dat systematische toepassing van de tableauregels in definitie 1.11 op een formule alleen eindige tableaus en eindige gelabelde formules op kan leveren. Dit is een heel prettige eigenschap en precies de motivatie om dit iets minder voor de hand liggende systeem te gebruiken. Het tableausysteem met regels **4** en **4r** heeft deze eigenschap zeker niet en het is heel lastig een sluitende beslissingsprocedure voor het construeren van bewijzen te definiëren die gebruik maakt van deze regels. Verder is de keuze voor het analyseren van formules $\sigma\ \Box_i\varphi$ nu beperkt tot *twee* regels in plaats van *vier* wat de taak van de bewijzer ook vergemakkelijkt. Voor het construeren van bewijzen is systeem ELtab dus zeker te verkiezen.

Voorbeeld 1.2 (een gesloten tableau)

1	$\neg(\Diamond_1(\Box_1 p \vee \Diamond_2\Box_2\Box_1 q) \rightarrow \Box_1(\neg q \rightarrow p))$		1.
1	$\Diamond_1(\Box_1 p \vee \Diamond_2\Box_2\Box_1 q)$	($\wedge$1)	2.
1	$\neg\Box_1(\neg q \rightarrow p)$	($\wedge$1)	3.
1.1_1	$\neg(\neg q \rightarrow p)$	(**M**3)	4.
1.1_1	$\neg q$	($\wedge$4)	5.
1.1_1	$\neg p$	($\wedge$4)	6.
1.2_1	$\Box_1 p \vee \Diamond_2\Box_2\Box_1 q$	(**M**2)	7.

1.2_1	$\Box_1 p$	($\vee$7)	8.	1.2_1	$\Diamond_2\Box_2\Box_1 q$	($\vee$7)	10.
1.1_1	p	(K= 8)	9.	$1.2_1.1_2$	$\Box_2\Box_1 q$	(M10)	11.
	=	(9×6)		1.2_1	$\Box_1 q$	(**R**= 11)	12.
				1.1_1	q	(**K**= 12)	13.
					=	(13×5)	

□

In voorbeeld 1.2 zijn ter referentie de toegepaste regels beknopt aangegeven: (**M**n) achter een formule refereert aan de ouderformule $\{\sigma \quad \Diamond ... \quad n.\}$

of $\{\sigma\ \neg\Box ...\quad n.\}$, de formule is ontstaan door toepassing van regel **M** op formule n op de tak; evenzo voor $(\wedge n)$, $(\mathbf{K}{=}\ n)$, $(\vee n)$ enzovoort. Omdat het aantal toepasbare regels op een tableautak in de regel klein is en deze geregeld probleemloos voor de laag weg toegepast kunnen worden is het gebruikelijk de referenties achterwegen te laten. De cijfers en referenties indien aanwezig achter formules in een tableau maken geen deel uit van het bewijs.

Het is gebruikelijk agents in epistemische logica een nummer te geven maar dit combineert zoals tezien is in voorbeeld 1.2 niet altijd even gelukkig met de cijfers in labels. Om verwarring te voorkomen—zeker bij handgeschreven bewijzen—is het waarschijnlijk een goed idee agents aan te duiden met letters en wel zo dat dit weer geen verwarring veroorzaakt met propositieletters. Hoewel ik hier niet consequent in ben geweest lijkt het mij handig agents a, e, i, o en u en proposities p, q, r, s en t te nemen.

1.4 Semantische tableaus voor PAL

In PAL-tableaus moet er niet alleen bijgehouden worden in welke mogelijke wereld een formule geldt maar ook in welk mogelijk ge-update model. Het ligt voor de hand om dicht bij de semantiek te blijven en formules te labelen met updates.

Definitie 1.12 (update-labels) Een gelabelde PAL formule is van de vorm $\nu : \sigma\ \ \varphi$ met $\sigma\ \ \varphi$ de gelabelde formule uit definitie 1.6 en ν een opeenvolging van PAL formules tussen haakjes $()$. □

De regels uit definitie 1.9, 1.10 en 1.11 zijn nu moeiteloos te lezen met σ als $\nu : \sigma$ en het construeren van tableaus blijft hetzelfde met de toevoeging van de komende regels. Het label σ staat voor zowel 'gewone' als update-labels. De ':' in labels is handig in komende definities maar laat ik geregeld weg, een gelabelde formule ziet er dan bijvoorbeeld zo uit: $(\Box_1\Diamond_2\neg a)1.1_2\ \Box_1 a$, te lezen als 'na $(\Box_1\Diamond_2\neg a)$ in 1.1_2 geldt $\Box_1 a$'.

Wat nu volgt zijn *twee* tableaubewijssystemen voor PAL. Het eerste systeem maakt gebruik van de bivalentieregel uit de KE tableaus in definitie 1.10. De regels in definities 1.13, 1.11 en 1.10 vormen samen systeem KEPAL. We zijn ook geïnteresseerd in tableaus *zonder* bivalentieregel, het tweede tableausysteem PALtab gedefinieerd door definities 1.10, 1.11 en 1.14 geeft zo'n systeem en het is dit systeem waar ik correctheid en volledigheid voor bewijs. Het is eenvoudig te laten zien dat de tableauregels in PALtab gesimuleerd kunnen worden in KEPAL [3], hieruit volgt volledigheid voor KEPAL.

Definitie 1.13 geeft een heel herkenbaar 'modaal' en symmetrisch tableausysteem. De update-regels (1^{e} rij) volgen het patroon van label *introductie*

Definitie 1.13 (eerste tableauregels voor PAL**)**

$\dfrac{\sigma \quad \langle\varphi\rangle\psi}{(\varphi)\sigma \quad \psi}$	$\dfrac{\sigma \quad [\chi]\psi}{(\chi)\sigma \quad \psi}$	updates met $(\chi)\sigma$ een bestaand label op de tak
$\dfrac{(\varphi)\sigma \quad p}{\sigma \quad p}$	$\dfrac{(\varphi)\sigma \quad \neg p}{\sigma \quad \neg p}$	persistentie
$\dfrac{(\varphi)\sigma}{\sigma \quad \varphi}$	$\dfrac{\sigma \quad \varphi}{(\varphi)\sigma}$	correspondentie

dien ten verstande $\langle\varphi\rangle\psi \equiv \neg[\varphi]\neg\psi$ en afhankelijkheid van bivalentie. □

door $\Diamond$-formules en label *selectie* door $\Box$-formules. De correspondentieregels wisselen labels en formules uit en definiëren zo de toegankelijkheidsrelaties tussen sub-modellen. De persistentieregels zijn alleen van toepassing op atomaire formules p en $\neg p$; KEPAL (en PALtab) is net als de reductie-axiomatisering niet *substitutie-gesloten* [21], we kunnen in deze regels niet zomaar iedere formule invullen maar alleen maar losse proposities. Om KEPAL volledig te maken hebben we zeker de bivalentieregel uit definitie 1.10 nodig, waarbij tableautakken gevorkt mogen worden met arbitraire formules $\varphi \mid \neg\varphi$. Deze regel is de semantische tegenhanger van de sequente '*cut*'-regel. De bivalentieregel is analytisch te maken, om te vormen tot een *analytische-cut* waarbij alleen formules geïntroduceerd mogen worden op tableautak $\mathcal{B}$ als cut-formule φ een subformule is van een formule die al op $\mathcal{B}$ voorkomt terwijl het bewijssysteem toch volledig blijft[3]. Hier verstaan we analytisch als beantwoordend aan de *subformule eigenschap*, in een bewijs voor φ spelen alleen subformules van φ een rol[6]. De bivalentieregel kunnen we voor proposities beperken tot disjuncties [3], in KEPAL moeten we formules $\{\sigma\ [\varphi]\psi\}$ waarbij $(\varphi)\sigma$ niet op de tak staat analyseren via bivalentie en correspondentie met φ. Formules $\{(\varphi)\sigma_1\ \Box_i\psi\}$ vereisen in sommige gevallen dat we een label σ_2 introduceren in (φ) voor selectie, weer door bivalentie en correspondentie. In definitie 1.14, systeem PALtab zijn deze regels expliciet gedefinieerd en hebben we de bivalentieregel niet nodig.

In het resterende deel van deze paragraaf wordt correctheid en volledigheid van PALtab afgeleid voor zover het de regels uit definitie 1.14 betreft en

[6] Alleen omdat in de hier gebruikte notatie de waarheidswaarde van formules in tableaus steeds wordt aangegeven met '¬', in het geval een formule onwaar moet zijn komt de subformule eigenschap niet erg uit de verf.

Definitie 1.14 (tweede tableauregels voor PAL)

$\dfrac{\sigma \quad \langle\varphi\rangle\psi}{(\varphi)\sigma \quad \psi}$	$\dfrac{\sigma \quad [\varphi]\psi}{(\varphi)\sigma \quad \psi \mid \sigma \quad \neg\varphi}$	mogelijke en noodzakelijke updates
$\dfrac{(\varphi)\sigma \quad p}{\sigma \quad p}$	$\dfrac{(\varphi)\sigma \quad \neg p}{\sigma \quad \neg p}$	persistentie
$\dfrac{(\varphi)\sigma}{\sigma \quad \varphi}$	$\dfrac{(\varphi)\nu : \sigma_1}{(\varphi)\nu : \sigma_2 \mid \nu : \sigma_2 \quad \neg\varphi}$	correspondentie en update-correspondentie met $\nu : \sigma_2$ een bestaand label op de tak.

dien ten verstande $\langle\varphi\rangle\psi \equiv \neg[\varphi]\neg\psi$. □

volgen enige voorbeelden. Met PALtab kunnen in sommige gevallen overzichtelijke bewijzen weergegeven worden voor PAL formules. Voor een variatie op de onsuccesvolle update uit voorbeeld 1.1 kan nu tableau 1.3 worden gegeven waarmee de algemeengeldigheid van $[p \wedge \neg\Box_a p]\neg(p \wedge \neg\Box_a p)$ bewezen is.

Voorbeeld 1.3 (een onsuccesvolle update)

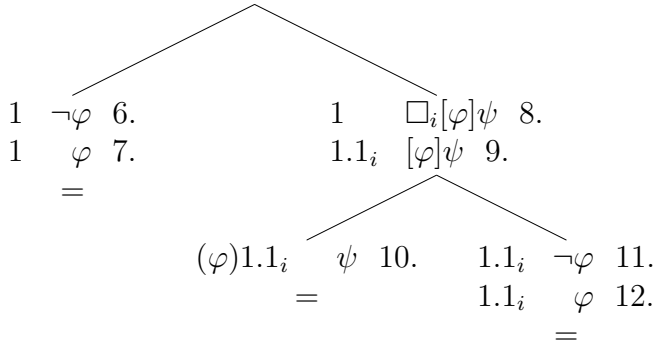
1	$\langle p \wedge \neg\Box_a p\rangle(p \wedge \neg\Box_a p)$		1.
$(p \wedge \neg\Box_a p)1$	$p \wedge \neg\Box_a p$	$(\langle\rangle 1)$	2.
$(p \wedge \neg\Box_a p)1$	p	$(\wedge 2)$	3.
$(p \wedge \neg\Box_a p)1$	$\neg\Box_a p$	$(\wedge 2)$	4.
$(p \wedge \neg\Box_a p)1.1_a$	$\neg p$	$(\Diamond 4)$	5.
1.1_a	$\neg p$	(pers.5)	6.
1.1_a	$p \wedge \neg\Box_a p$	(corr.5)	7.
1.1_a	p	$(\wedge 7)$	8.
	$=$	(8×6)	

□

Het is zaak aan te tonen dat (ge-update) epistemische modellen en semantische tableaus in overeenstemming zijn: formules hebben een gesloten tableau als ze PAL-geldig zijn (*soundness* (Eng.) ofwel correctheid) en alle PAL-geldige formules hebben een gesloten tableau (*completeness* (Eng.) ofwel volledigheid).

Voorbeeld 1.4 (een tableau voor een 'kennis-update')

$1 \quad \neg((\varphi \to \Box_i[\varphi]\psi) \to [\varphi]\Box_i\psi) \quad 1.$
$1 \quad (\varphi \to \Box_i[\varphi]\psi) \quad 2.$
$1 \quad \neg[\varphi]\Box_i\psi \quad 3.$
$(\varphi)1 \quad \neg\Box_i\psi \quad 4.$
$(\varphi)1.1_i \quad \neg\psi \quad 5.$



□

Tableaus met labels σ en epistemische modellen $\mathcal{M} = \langle W, R, V\rangle$ corresponderen als volgt: $\theta(\sigma) \in W$ is de mogelijke wereld die correspondeert met het label σ, $\eta(\sigma)$ is het ge-update model dat correspondeert met het label σ. Zo correpondeert $\eta((\varphi)(\psi) : 1.1_i)$ met $\mathcal{M}|_\psi|_\varphi$. Ik stel $\eta(\sigma)$ hier *gelijk* aan een submodel en $\eta((\varphi)(\psi)\sigma_1)$ en $\eta((\psi)\sigma_2)|\varphi$ verwijzen hier dus naar dezelfde entiteit.

Definitie 1.15 (tableau vervulbaarheid) Een set gelabelde formules S is *vervulbaar* in een model $\mathcal{M} = \langle W, R, V\rangle$ als de volgende condities gelden.

1. aan ieder label σ in S is een mogelijke wereld $\theta(\sigma) \in W$ toe te kennen,
2. voor alle σ en $\sigma.\Phi_i \in S$ met Φ_i een opeenvolging van nul of meer symbolen n_i: $\theta(\sigma) \sim_i \theta(\sigma.\Phi_i)$ en $\theta(\sigma.\Phi_i) \sim_i \theta(\sigma) \in R$ (S5 toegankelijkheid),
3. als $\{\sigma\ \varphi\} \in S$ dan $\eta(\sigma), \theta(\sigma) \models \varphi$.

Een tak van een tableau is nu vervulbaar als de set van alle gelabelde formules op die tak vervulbaar is en een heel tableau is vervulbaar als het in ieder geval één vervulbare tak heeft. □

Stelling 1.1 Een gesloten tableau is nooit vervulbaar

Bewijs Een gesloten tableau heeft gesloten takken met formules $\{\sigma\ \varphi\}$ en $\{\sigma\ \neg\varphi\}$. Een gesloten tak die tegelijkertijd vervulbaar is impliceert dat er een model $\mathcal{M}$ met $\eta(\sigma), \theta(\sigma) \models \varphi \wedge \neg\varphi$. Dit is duidelijk onmogelijk. □

Stelling 1.2 Een vervulbaar tableau is niet te sluiten met het toepassen van de PALtab tableauregels.

Bewijs Voor de propositielogische en standaard modale regels verwijs ik naar de literatuur, o.a. [5] waar deze schematische behandeling precies vandaan komt, nu dan de gepresenteerde update-tableauregels. We nemen steeds tableau $\mathcal{T}$, tak $\mathcal{B}$ en model $\mathcal{M} = (W, R, V)$ waarin $\mathcal{B}$ vervulbaar is.

MOGELIJKE UPDATE: stel $\{\sigma\ \langle\varphi\rangle\psi\}$ staat op $\mathcal{B}$ en we voegen $\{(\varphi)\sigma\ \varphi\}$ toe aan $\mathcal{B}$. Nu moet gelden $\eta(\sigma)|_\varphi, \theta(\sigma) \models \psi$. Er gold al $\eta(\sigma), \theta(\sigma) \models \langle\varphi\rangle\psi$ volgens $\{\sigma\ \langle\varphi\rangle\psi\}$ en $\eta(\sigma)|_\varphi, \theta(\sigma) \models \psi$ volgt per definitie.

NOODZAKELIJKE UPDATE: stel $\{\sigma\ [\varphi]\psi\}$ staat op $\mathcal{B}$ en we splitsen het einde van de tak. Nu moet minstens één van de twee zo ontstane takken vervulbaar zijn. links voegen we $\{(\varphi)\ \sigma\ \varphi\}$ toe en rechts $\{\sigma\ \neg\varphi\}$. Er gold dus al $\eta(\sigma), \theta(\sigma) \models [\varphi]\psi$; in de linker tak: $\eta(\sigma), \theta(\sigma) \models \varphi$ impliceert $\eta(\sigma)|_\varphi, \theta(\sigma) \models \psi$, de rechter tak blijft open in het geval $\eta(\sigma), \theta(\sigma) \not\models \varphi$ dus $\eta(\sigma), \theta(\sigma) \models \neg\varphi$ tevens per definitie.

PERSISTENTIE: stel $\{(\varphi)\ \sigma\ p\}$ staat op $\mathcal{B}$ en we voegen $\{\sigma\ p\}$ toe. Nu moet gelden als $\eta(\sigma)|_\varphi, \theta(\sigma) \models p$ dan $\eta(\sigma), \theta(\sigma) \models p$. $\theta(\sigma)$ is een mogelijke wereld in $\eta(\sigma)|_\varphi$ en $\eta(\sigma)|_\varphi = (W', R', V')$ een submodel van $\eta(\sigma) = (W, R, V)$, dus $\theta(\sigma) \in W$. Nu geldt $V'_{\theta(\sigma)}(p) \equiv V_{\theta(\sigma)}(p)$. Evenzo voor $\neg p$.

CORRESPONDENTIE: stel $(\varphi)\sigma$ is een label op $\mathcal{B}$ en we voegen $\{\sigma\ \varphi\}$ toe. $\eta(\sigma)|_\varphi = (W''R''V'')$ is een submodel van $\eta(\sigma) = (W'R', V')$; $\theta(\sigma)$ is een wereld in W' en per definitie alleen ook een wereld in W'' als $\eta(\sigma), \theta(\sigma) \models \varphi$ geldt.

UPDATE-CORRESPONDENTIE: stel $(\varphi)\nu : \sigma_1$ en $\nu : \sigma_2$ zijn labels op $\mathcal{B}$ en we vorken en voegen links label $(\varphi)\nu : \sigma_2$ toe en rechts formule $\{\nu : \sigma_2\ \neg\varphi\}$. De linker tak is vervulbaar als $\eta(\nu : \sigma_2), \theta(\nu : \sigma_2) \models \varphi$ in welk geval $\theta(\nu : \sigma_2) \in \eta(\nu : \sigma_2)|_\varphi$ per definitie en de linker tak blijft vervulbaar. Anders geldt $\eta(\nu : \sigma_2), \theta(\nu : \sigma_2) \not\models \varphi$ en dus $\eta(\nu : \sigma_2), \theta(\nu : \sigma_2) \models \neg\varphi$ en de rechter tak blijf vervulbaar.

□

Stelling 1.3 (Correctheid) als er een PALtab tableau bewijs is voor φ dan is φ PAL-geldig, d.w.z. voor ieder gepunt model $(\mathcal{M}, w)$ geldt $(\mathcal{M}, w) \models \varphi$.

Bewijs Als φ een tableau bewijs heeft dan is er een gesloten tableau $\mathcal{T}$ met wortel $\{1\ \neg\varphi\}$. Deze wortel noemen we $\mathcal{T}_0$, $\mathcal{T}$ ontstaat door het toepassen van tableau regels op $\mathcal{T}_0$. Als $\mathcal{T}$ sluit maar φ toch ongeldig zou zijn dan moet er dus een model $\mathcal{M}$ zijn met $\mathcal{M}, w \models \neg\varphi$. Als we nu nemen $w = \theta(1)$ dan volgt dat $\{1\ \neg\varphi\}$ vervulbaar is en dat $\mathcal{T}_0$ vervulbaar is. Aangezien volgens stelling 1.2 een vervulbaar tableau door toepassing van de tableauregels niet te sluiten is is $\mathcal{T}$ vervulbaar en sluitend tegelijkertijd; volgens stelling 1.1 is dit onmogelijk. □

Volledigheid van modale tableausystemen wordt over het algemeen bewezen aan de hand van *verzadigde tableaus* of *Hintikka sets*. In een verzadigd tableau zijn alle toepasbare tableauregels toegepast. Omdat we hier alleen volledigheid voor de updateregels willen aantonen en PAL formules met de reductieaxioma's uit definitie 1.4 kunnen vertalen naar formules zonder updates volstaat het om bewijsbaarheid van die reductieaxioma's af te leiden; volledigheid voor PAL is zo in tableaus ook gereduceerd tot volledigheid voor EL.

Stelling 1.4 PALtab tableaus zijn volledig voor de PAL reductie axioma's.

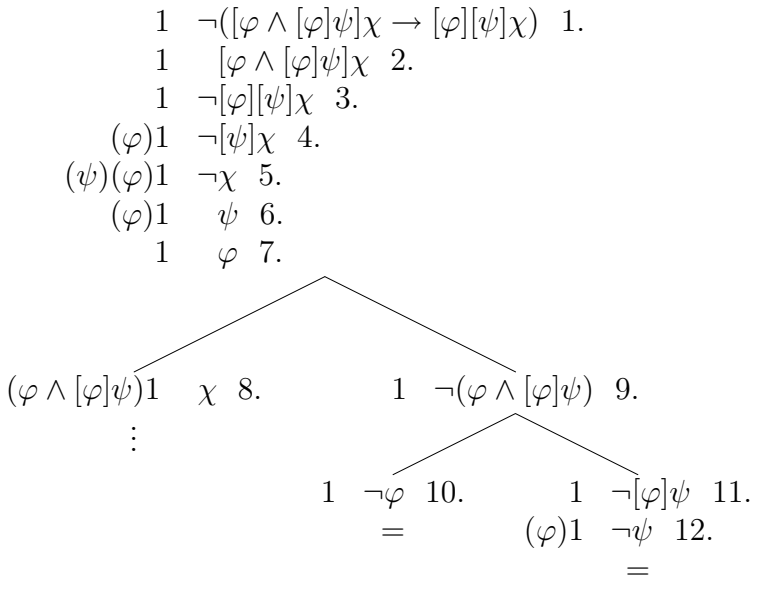
Bewijs Dit kan voor het grootste deel probleemloos, alleen het axioma 'gecombineerde updates': $\vdash [\varphi][\psi]\chi \leftrightarrow [\varphi \wedge [\varphi]\psi]\chi$ is niet direct af te leiden in een tableau en hiervoor moet iets meer werk verricht worden Het tableau voor het axioma 'kennis update' is al gegeven in voorbeeld 1.4, de andere tableaus zijn soortgelijk. De vijf tableaus voor de vijf reductieaxioma's zijn opgenomen in appendix A.2.

We bewijzen in iets algemenere termen dat de linker tak van het tableau in voorbeeld 1.5 altijd te sluiten is op tegenspraak tussen twee knopen $\{\nu_n(\varphi \wedge [\varphi]\psi)\ :\ \sigma\ \chi\}$ en $\{\nu_n(\psi)(\varphi)\ :\ \sigma\ \neg\chi\}$ met σ een mogelijke wereld label en ν_n een opeenvolging van n PAL formules $(\mu_n)(\mu_{n-1})...(\mu_1)$. ν_{n-1} Staat voor $(\mu_{n-1})(\mu_{n-2})...(\mu_1)$. Het tableau voor $[\varphi][\psi]\chi \rightarrow [\varphi \wedge [\varphi]\psi]\chi$ gaat analoog (zie tableau A.5 in de appendix). We onderscheiden vier gevallen:

1. χ is een propositionele formule p: we kunnen het tableau direct sluiten op $\{\sigma\ p, \sigma\ \neg p\}$ met de persistentie regel. Gevallen als $\neg\neg p$ sluiten zo uiteraard ook. In de komende stappen reduceren we steeds tot deze variant.

2. χ is van de vorm $\chi'_1 \wedge \chi'_2$: we moeten twee takken zien te sluiten op respectievelijk $\{\nu_n(\varphi \wedge [\varphi]\psi)\sigma\ \neg\chi'_1, \nu_n(\psi)(\varphi)\sigma\ \chi'_1\}$ en $\{\nu_n(\varphi \wedge [\varphi]\psi)\sigma\ \neg\chi'_2, \nu_n(\psi)(\varphi)\sigma\ \chi'_2\}$. De andere propositionele connectieven gaan analoog, ik reken hier onder $\chi'_1 \rightarrow \chi'_2$, $\chi'_1 \vee \chi'_2$, $\neg(\chi'_1 \wedge \chi'_2)$ etc.

Voorbeeld 1.5 (een tableau voor 'gecombineerde updates')



□

3. χ is een formule $\Box_i\chi'$: we kunnen met de knoop $\{\nu_n(\psi)(\varphi)\sigma\ \neg\Box_i\chi'\}$ expanderen tot $\{\nu_n(\psi)(\varphi)\sigma'\ \neg\chi'\}$. Met n keer de correspondentieregel en de update-correspondentieregel vorken we $\{(\varphi \wedge [\varphi]\psi)\sigma' \mid \sigma'\ \neg(\varphi \wedge [\varphi]\psi)\}$. Nu kunnen we n keer de update-correspondentieregel toepassen en krijgen we n takken die we moeten sluiten op de gewenste vorm $\{\nu_{n-x}(\psi)(\varphi)\sigma'\ \mu_x, \nu_{n-x}(\varphi \wedge [\varphi]\psi)\sigma'\ \neg\mu_x\}$. De uiterste rechter tak sluit precies als de rechter tak in voorbeeld 1.5 (maar nu in label σ') en uiterst links kunnen we nu verder met $\{\nu(\psi)(\varphi)\sigma'\ \neg\chi', \nu(\varphi\wedge[\varphi]\psi)\sigma'\ \chi'\}$. Analoog voor $\Diamond_i\chi'$, $\neg\Diamond\chi'$ en $\neg\Box_i\chi'$.

4. χ is $[\chi'_1]\chi'_2$: de mogelijke-update-regel met $\{\nu_n(\psi)(\varphi)\sigma\ \neg[\chi'_1]\chi'_2\}$ levert $\{\nu_{n+1}(\psi)(\varphi)\sigma\ \neg\chi'_2\}$ en we vorken met $\{\nu_n(\varphi \wedge [\varphi]\psi)\sigma\ [\chi'_1]\chi'_2\}$ tot $\{\nu_{n+1}(\varphi \wedge [\varphi]\psi)\sigma\ \chi'_2 \mid \nu_n(\varphi \wedge [\varphi]\psi)\sigma\ \neg\chi'_1\}$. De rechter tak moeten we nu zien te sluiten op $\{\nu_n(\varphi \wedge [\varphi]\psi)\sigma\ \neg\chi'_1, \nu_n(\psi)(\varphi)\sigma\ \chi'_1\}$ en de linker tak op $\{\nu_{n+1}(\psi)(\varphi)\sigma\ \neg\chi'_2, \nu_{n+1}(\varphi \wedge [\varphi]\psi)\sigma\ \chi'_2\}$. Evenzo voor $\langle\chi'_1\rangle\chi'_2$, $\neg\langle\chi'_1\rangle\chi'_2$ en $\neg[\chi'_1]\chi'_2$.

In de gevallen 2–4 hebben we steeds open tableautakken geconstrueerd met twee knopen $\{\nu'(\psi)(\varphi){:}\sigma'\ \omega, \quad \nu'(\varphi\wedge[\varphi]\psi){:}\sigma'\ \neg\omega\}$ met ω een deelformule van χ of een formule uit label ν. Door inductie op de opbouw van χ en ν

zijn alle takken te sluiten op het eerste propositionele geval en sluit het hele tableau voor alle formules $[\varphi \wedge [\varphi]\psi]\chi \rightarrow [\varphi][\psi]\chi$. De tableaus voor de overige axioma's sluiten direct en er hoeven geen verdere stappen genomen te worden. □

Hoewel we in paragraaf 1.2 in verband met definitie 1.4 hebben gezien dat het axioma **AC** niet strikt noodzakelijk was voor volledigheid van het systeem PAL_{PA} door uit $[\varphi][\psi]\chi$ eerst $[\psi]\chi$ (deels) te vertalen tot $t([\psi]\chi)$ en dan voort te bewijzen met $[\varphi]t([\psi]\chi)$ is het duidelijk dat we deze route niet kunnen nemen in PALtab, de analyse van $[\psi]\chi$ zou immers zelf een tableau opleveren.

Stelling 1.5 tableaus zijn volledig voor update reductie $\varphi \leftrightarrow \varphi^t$ voor ieder PAL formule φ en formule φ^t *zonder updates*.

Bewijs (schets) Dit volgt direct uit volledigheid voor de reductieaxioma's van tableaus en volledigheid van de reductie axioma's zoals gegeven in [27]. Uit volledigheid van tableaus voor epistemische logica en de reductieaxioma's kunnen we voor ieder formule $\varphi = \varphi^{t_0}$ met $\varphi^{t_n} = \varphi^t$ en iedere i $(0 \leq i \leq n)$ in een tableau bewijzen $\varphi^{t_i} \leftrightarrow \varphi^{t_{i+1}}$ met $\varphi^{t_{i+1}}$ de formule φ^{t_i} met één instantie van de linkerhelft van een reductie axioma vervangen door zijn rechterhelft zoals in voorbeeld 1.1. Correctheid en volledigheid van de PAL reductieaxioma's garandeert geldigheid van de formules $\varphi^{t_0} \leftrightarrow \varphi^{t_1}, ..., \varphi^{t_{n-1}} \leftrightarrow \varphi^{t_n}$ en propositioneel volgt $\varphi \leftrightarrow \varphi^t$. □

Stelling 1.6 (volledigheid) Voor iedere geldige PAL formule φ is er een tableaubewijs.

Bewijs Voor ieder PAL formule φ met vertaling φ^t naar gewone epistemische logica zonder updates kunnen we een tableaubewijs leveren voor $\varphi \leftrightarrow \varphi^t$. Tableaus zijn correct en volledig voor epistemische logica zonder updates dus als φ PAL geldig is kunnen we een tableaubewijs vinden voor φ^t. Als er een tableaubewijs is voor $\varphi \leftrightarrow \varphi^t$ en voor φ^t maar *niet* voor φ dan is er geen tableaubewijs voor $\neg(\varphi \leftrightarrow \varphi^t) \vee \neg\varphi^t \vee \varphi$, alledrie de formules zijn immers onbewijsbaar. Echter, deze formule is een propositielogische tautologie en tegenspraak volgt. □

Het zo verkregen volledigheidsbewijs is om eerlijk te zijn niet erg bevredigend. Hoewel er wellicht enig inzicht uit te ontlenen is met betrekking tot de relatie tussen tableaus en reductie axioma's zou een meer gebruikelijk volledigheidsbewijs mogelijk korter en overzichtelijker zijn.

1.5 De drie wijze mannen puzzel

Om te laten zien dat semantische tableaus voor public announcement logica nuttig zijn voor de analyse van lastige epistemische situaties analyseren we de welbekende Drie Wijze Mannen Puzzel in tableaus. De Drie Wijze Mannen Puzzel gaat als volgt:

> Drie wijze mannen verschijnen voor de koning van Perzië, deze wenst te achterhalen wie de wijste is van de drie. Hij schildert een witte stip op ieders voorhoofd en zegt dan: 'jullie hebben allemaal een witte of een zwarte stip op jullie voorhoofd maar in ieder geval één stip is wit; de eerste die zijn kleur weet is de wijste'. De wijzen kijken elkaar aan en de eerste zegt 'ik weet het niet'; de tweede zegt 'ik weet het ook niet'; de derde zegt nu 'ik weet het!'.

Deze variant van de Drie Wijze Mannen Puzzel is toe te schrijven aan McCarthy [14]. In de originele variant van deze puzzel zeggen de wijzen niets tot de wijste zegt "ik weet het"; het redeneren van de wijzen gaat dan, informeel en bondig, zo. De wijste van de drie ziet twee witte stippen, concludeert: als ik een zwarte stip had dan weten de anderen dat. Als de anderen een witte en een zware stip zien dan weten ze allebei: als mijn stip zwart is dan weet de ander het direct; er wordt niets gezegd dus dat zal niet het geval zijn, de wijste van de andere twee kan nu concluderen dat zijn stip wit moet zijn maar er wordt nog niets gezegd; De wijste weet nu: ik heb ook een witte stip. Zo wordt er niet alleen geredeneerd over de feiten maar ook over de snelheid waarmee de wijzen redeneren, iets wat we hier niet kunnen modelleren.

Het 'antwoord' op de puzzel is al gegeven, de wijzen hebben allemaal een witte stip. Er hoeft niets opgelost te worden maar de puzzelaar moet een achterliggende waarheid ontdekken. In dat perspectief geef ik hier ook niet helemaal een sluitende verhandeling, als u niet gelooft dat het Drie Wijze Mannen Puzzel polyloog te voeren is zonder dat een van de wijzen of de koning liegt—iets wat niet geformuleerd kan worden in PAL—kan er aangevoerd worden dat niet alle feiten in ogenschouw genomen zijn en dat de tegenspraak zo verdoezeld is.

De Drie Wijze Mannen Puzzel in de hier gekozen vorm is al eens geformaliseerd door McCarthy voor de stellingbewijzer FOL [14], wordt als voorbeeld gebruikt in de Logics Workbench Manuals[7] en Van Ditmarsch geeft

[7]De LWB is een populaire propositionele stellingbewijzer ontwikkeld aan de Universiteit van Bern, Zwitserland; http://www.lwb.unibe.ch

een alternatieve formulering voor de puzzel voor de LWB[8]. Deze laatste en meest begrijpelijke variant zal ik hier ook gebruiken. Genoemde formalisaties worden typisch aangeboden aan een stellingbewijzer (computerprogramma) en die levert dan als uitvoer over het algemeen `true`—als het goed is—of—jammer—`false` of zelfs `ERROR`. De formalisaties zijn zelf eigenlijk ook computerprogramma's en niet altijd even begrijpelijk. Dit is allerminst informatief, gezien de aard van de Drie Wijze Mannen Puzzel.

De Drie Wijze Mannen Puzzel is te bespreken aan de hand van epistemische modellen, Van Benthem behandelt in [20] op deze wijze de vergelijkbare Modderige Kinderen Puzzel. Dit levert een heel begrijpelijk verhaal maar geen direct aanwijsbaar *bewijs* voor de geldigheid van een of andere conclusie uit een conversatie. Omdat deze polyloog puzzels steeds over feitelijke waarheden, propositioneel of anderzijds gaan en niet direct over 'modellen' is het ook niet altijd direct duidelijk hoe een (initieel) epistemisch model er precies uit moet zien. In [7] wordt een axiomatische uiteenzetting gegeven voor de Modderige Kinderen Puzzel in dynamische epistemsiche logica.

Hoe representeren we de Drie Wijze Mannen Puzzel in PAL? Laat $1, 2, 3$ staan voor de drie wijze mannen en a, b, c voor de feiten respectievelijk '1,2,3 heeft een witte stip'. De koning vermeldt dat er in ieder geval één witte stip is, mogelijk meer, dit vertaalt naar $(a \vee b \vee c)$. De Wijzen kijken elkaar aan en leren van elkaar of ze een witte stip hebben (niet van zichzelf). Het blijkt (aldus Van Ditmarsch) dat het toereikend is om dit zo te representeren:

$$(\neg a \to (\Box_2 \neg a \wedge \Box_3 \neg a)) \wedge (\neg b \to (\Box_1 \neg b \wedge \Box_3 \neg b)) \wedge (\neg c \to (\Box_1 \neg c \wedge \Box_2 \neg c))$$

'Als 1 geen witte stip heeft dan weten 2 en 3 dat'—enzovoort.

Na de bekendmaking van deze begincondities zegt Wijze 1 "ik weet het niet"; dit wordt $\Diamond_1 \neg a$: Wijze 1 houdt voor mogelijk dat hij geen witte stip heeft. Na dezelfde mededeling voor Wijze 2: $\Diamond_2 \neg b$ moet nu gelden $\Box_3 c$: Wijze 3 weet dat hij een witte stip heeft.

Voor de overzichtelijkheid sta ik mezelf nu nog een vrijheid toe en voeg de volgende regel toe aan het arsenaal:

$$\textbf{MP} \qquad \begin{array}{ll} \sigma & \varphi \\ \sigma & \varphi \to \psi \\ \hline \sigma & \psi \end{array} \qquad \text{(modus ponens)}$$

Dit is natuurijk een verschijningsvorm van de disjunctieregel uit systeem KE.

Nu zijn we klaar om een tableaubewijs te maken voor de Drie Wijze Mannen Puzzel, dit levert tableau 1.6 waarmee de algemeengeldigheid van het puzzel polyloog bewezen is. Om het tableau niet te lang te maken zijn

[8] http://tcw2.ppsw.rug.nl/mas/LOK/lwb/

enige afleidingen verkort weergegeven, deze zijn aangegeven met een a-tje en de voornaamste regel zoals (corr.a.6) in formule 1.6-10.

Voorbeeld 1.6 (bewijs voor de Drie Wijze Mannen Puzzel)

$$\mathsf{WM} \equiv (a \vee b \vee c) \wedge (\neg a \rightarrow (\Box_2 \neg a \wedge \Box_3 \neg a)) \wedge (\neg b \rightarrow (\Box_1 \neg b \wedge \Box_3 \neg b)) \wedge (\neg c \rightarrow (\Box_1 \neg c \wedge \Box_2 \neg c))$$

1	$\neg[\mathsf{WM}][\Diamond_1\neg a][\Diamond_2\neg b]\Box_3 c$		1.
$(\mathsf{WM})1$	$\neg[\Diamond_1\neg a][\Diamond_2\neg b]\Box_3 c$	$(\langle\rangle 1)$	2.
$(\Diamond_1\neg a)(\mathsf{WM})1$	$\neg[\Diamond_2\neg b]\Box_3 c$	$(\langle\rangle 2)$	3.
$(\Diamond_2\neg b)(\Diamond_1\neg a)(\mathsf{WM})1$	$\neg\Box_3 c$	$(\langle\rangle 3)$	4.
$(\Diamond_2\neg b)(\Diamond_1\neg a)(\mathsf{WM})1.1_3$	$\neg c$	$(\mathbf{M}4)$	5.
$(\Diamond_1\neg a)(\mathsf{WM})1.1_3$	$\Diamond_2\neg b$	(corr.5)	6.
$(\Diamond_1\neg a)(\mathsf{WM})1.1_3.1_2$	$\neg b$	$(M6)$	7.
$(\mathsf{WM})1.1_3.1_2$	$\Diamond_1\neg a$	(corr.7)	8.
$(\mathsf{WM})1.1_3.1_2.1_1$	$\neg a$	$(\mathbf{M}8)$	9.
1.1_3	WM	(corr.a.6)	10.
1.1_3	$\neg c$	(pers.a.5)	11.
1.1_3	$\Box_2\neg c$	(MP a. 10,11)	12.
$1.1_3.1_2$	$\neg c$	$(\mathbf{K}12)$	13.
$1.1_3.1_2$	WM	(corr.8)	14.
$1.1_3.1_2$	$\neg b$	(pers.a.7)	15.
$1.1_3.1_2$	$\Box_1\neg c$	(MP a. 13,14)	16.
$1.1_3.1_2$	$\Box_1\neg b$	(MP a. 15,14)	17.
$1.1_3.1_2.1_1$	$\neg c$	$(\mathbf{K}16)$	18.
$1.1_3.1_2.1_1$	$\neg b$	$(\mathbf{K}17)$	19.
$1.1_3.1_2.1_1$	$\neg a$	(pers.9)	20.
$1.1_3.1_2.1_1$	WM	(corr.9)	21.
$1.1_3.1_2.1_1$	$(a \vee b \vee c)$	$(\wedge 21)$	22.
	$=$	$(\text{a.}[18, 19, 20] \times 22)$	

□

Een aantal zaken vallen op in het tableaubewijs voor de Drie Wijze Mannen Puzzel. We zien dat alleen de modale tableauregels **M** en **K** zijn gebruikt in dit bewijs. Dit betekent dat we bewezen hebben dat dit polyloog in *alle* modellen is te voeren en dat ware kennis, positieve en negatieve introspectie niet vereist zijn. Dit was al bekend en volgt bijvoorbeeld uit de succesvolle implementatie in de LWB voor K_n. Wat niet direct opvalt is dat $(\neg a \rightarrow (\Box_2 \neg a \wedge \Box_3 \neg a))$, de premisse dat als Wijze 1 geen witte stip heeft de

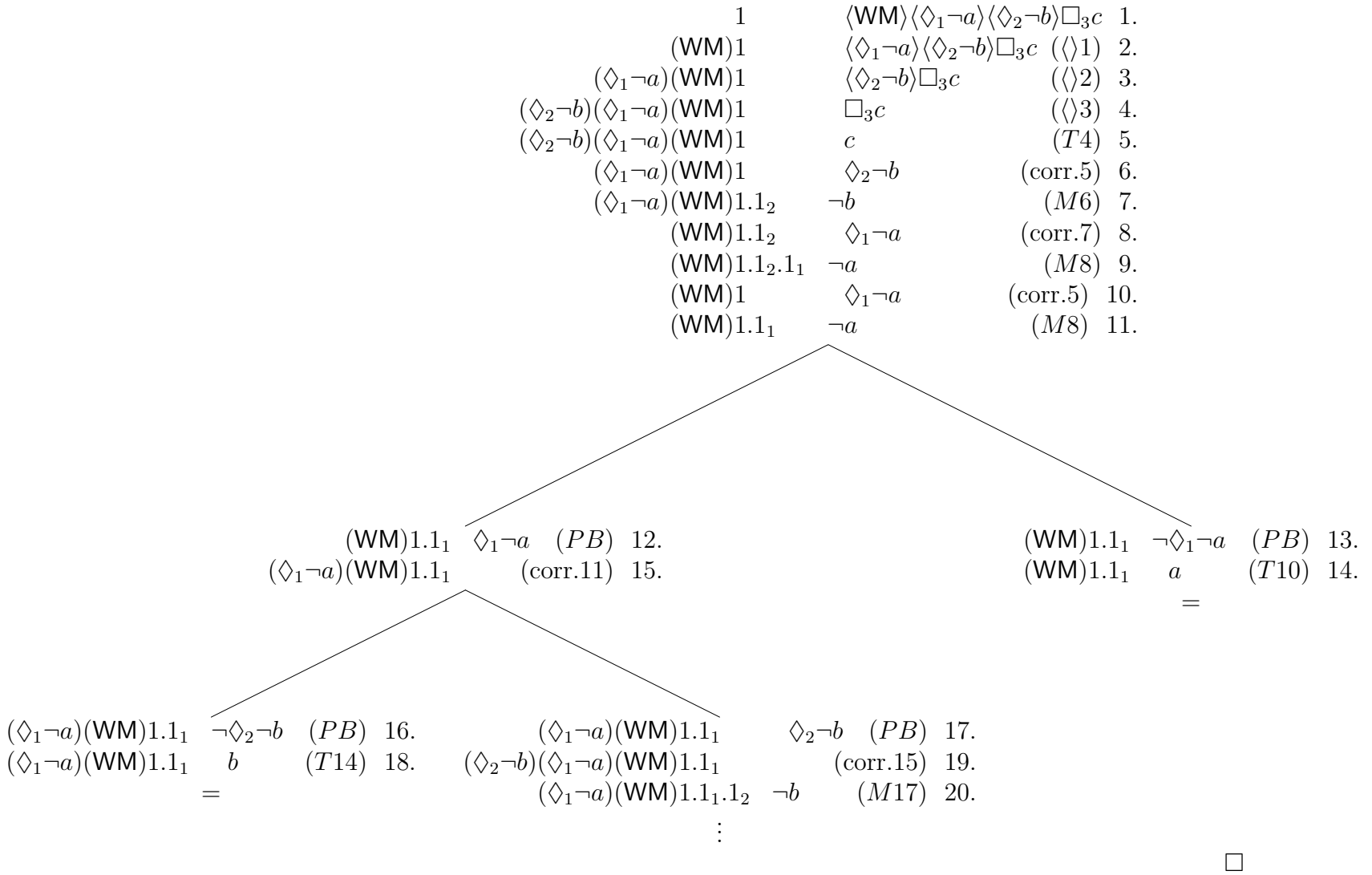
anderen dit weten, geen rol speelt in het bewijs[9]. Het bewijs sluit al voordat we zijn toegekomen aan een toepassing van (MP a. 20,21). Dit komt er op neer dat als Wijze 1 bijvoorbeeld zijn hand voor zijn stip houdt het polyloog nog steeds te voeren is en Wijze 3 nog steeds te weten komt dat hij een witte stip heeft. De redenatie gaat dan als volgt. Als Wijze 1 een hand voor zijn voorhoofd houdt zodat de anderen zijn stip niet kunnen zien en zegt "ik weet niet of ik een witte stip heb" dan ziet Wijze 1 dus één of twee witte stippen. Als Wijze 3 géén witte stip heeft dan wéét Wijze 2 dus dat hij zelf een witte stip moet hebben. Wijze 2 zegt nu "ik weet ook niet of ik een witte stip heb". Wijze 3 weet nu dus wél dat hij zelf een witte stip heeft.

Misschien zijn we ook geïnteresseerd in het epistemische model en zijn sub-modellen waarin de Drie Wijze Mannen Puzzel uitgevoerd kan worden. Om een model te vinden voor een formule φ maken we een tableau met root $\{1\ \varphi\}$, dit in tegenstelling tot het vinden van een bewijs. Tableau 1.6 bewijst al dat formule $[\mathsf{WM}][\Diamond_1\neg a][\Diamond_2\neg b]\Box_3 c$ geldig is in *ieder* epistemisch model, een tableau voor deze formule is dan ook niet zo interessant. We nemen nu dan ook $\{1\ \langle\mathsf{WM}\rangle\langle\Diamond_1\neg a\rangle\langle\Diamond_2\neg b\rangle\Box_3 c\ \}$. Aan de hand van definitie 1.15 kunnen we open tableautakken omschrijven naar epistemische (sub-)modellen. In het geval van $\{1\ \langle\mathsf{WM}\rangle\langle\Diamond_1\neg a\rangle\langle\Diamond_2\neg b\rangle\Box_3 c\ \}$ krijgen we zo dus een model $\mathcal{M}$ met $(\mathcal{M}, w_1) \models c$. Wereld w_1 verstaan we als de 'echte' wereld, waar *wij* al weten $a \wedge b \wedge c$. Een model gegenereerd door een tableau is altijd een *boom* en niet een cyclische graaf zoals epistemische modellen geregeld worden weergegeven. Zo'n boom vertegenwoordigd een 'plaatje' van de paden door een mogelijk cyclische model vanuit mogelijke wereld $\theta(1)$. In voorbeeld 1.7 laat ik nu een gedeeltelijke expansie van het tableau voor $\{1\ \langle\mathsf{WM}\rangle\langle\Diamond_1\neg a\rangle\langle\Diamond_2\neg b\rangle\Box_3 c\ \}$ zien. We kunnen nu voor iedere mogelijke-wereldlabel σ steeds onderzoeken of het in een submodel $\mathcal{M}|\varphi$ thuis hoort door middel van een vork $\{\sigma\ \varphi \mid \sigma\ \neg\varphi\}$. In tableau 1.7, knoop 17 is bijvoorbeeld afgeleid dat na de mededeling van Wijze 2 "ik weet het niet" Wijze 1 nog steeds toegang heeft tot mogelijke wereld $\theta(1.1_1)$ en dus nog niet weet of hij zelf een witte of een zwarte stip heeft. Mogelijke werelden $\theta(1.1_1.1_2)$ en $\theta(1.1_2.1_1)$ krijgen bij nader onderzoek dezelfde valuatie $\neg a, \neg b, c$.

[9]Barteld Kooi wees mij hier indirect op en Rem de Boer heeft mij er vervolgens van kunnen overtuigen dat het toch klop. Volgens Rineke Verbrugge is dit weldegelijk onderdeel van de "folklore" van de epistemische logica.

Voorbeeld 1.7 (modelonderzoek voor de Drie Wijze Mannen Puzzel)

$\mathsf{WM} \equiv (a \vee b \vee c) \wedge (\neg a \to (\Box_2 \neg a \wedge \Box_3 \neg a)) \wedge (\neg b \to (\Box_1 \neg b \wedge \Box_3 \neg b)) \wedge (\neg c \to (\Box_1 \neg c \wedge \Box_2 \neg c))$



□

Hoofdstuk 2

Automatische Deductie voor Epistemische Logica

In dit hoofdstuk wordt een stellingbewijzer gepresenteerd voor epistemische logica EL ($S5_n$) gebaseerd op leanT^AP[2], een *lean* (snelle, flexibele) tableaustellingbewijzer voor predikaatlogica in Prolog. Verkregen programma ELtap implementeert een vrije-variabelen gelabelde tableaucalculus met labelunificatie op basis van het tableausysteem uit hoofdstuk 1. Door formules in public announcement logica door middel van de reductieaxioma's te vertalen naar EL kunnen we met dit programma ook formules in PAL bewijzen.

Modale variaties op leanT^AP zijn niet nieuw: Bernhard Beckert en Rajeev Goré implementeren *lean labelled deduction* (Eng.) in ModLeanTAP voor logica's K, KD, KT en S4 [1]. Melvin Fitting's dirsecK implementeert een sequente-calculus voor K gebaseerd op leanT^AP [4]. De hier gekozen implementatie is een combinatie van deze programma's en Governatori's modulaire tableausysteem KEM [11, 10]. Inzichten uit deze publicaties worden hier zoveel mogelijk opgevolgd.

De erfenis van leanT^AP houdt onder andere in dat de hier gepresenteerde programma's niet zozeer een efficiënt alswel een begrijpelijk, compact en flexibel geheel vormen. Er is hier gekozen om zo dicht mogelijk bij de calculus uit hoofdstuk 1 te blijven en steeds te proberen de meest eenvoudige oplossingen te kiezen. Elementaire kennis van Prolog is voldoende om het merendeel van de programma's te vatten; het een en ander valt bijvoorbeeld na te lezen in Sterling en Shapiro's *The Art of Prolog* [19]. De meer geavanceerde datastructuren en algoritmes voor leanT^AP uit [16] zijn onverkort van toepassing op ELtap.

2.1 Voorbereidingen

Omdat tekens en notaties $\rightarrow, \wedge, \vee, \Box_i$ etc. over het algemeen niet beschikbaar zijn in een programmeeromgeving representeren we formules intern in Prologtermen; `F = update(box(1,and(a,b)),not(dia(2,imp(a, or(b,c)))))` staat zo voor $[\Box_1(a \wedge b)]\neg\Diamond_2(a \rightarrow (b \vee c))$. In verband met de leesbaarheid wordt in appendix A.1.1 module PALparse, `[parse/2]` geboden die zo veel mogelijk gebruikelijke ASCII weergaven van logische formules kan lezen en omzetten; bovenstaande is resultaat van `parse("[[1](a & b)]~<2>(a => (b|c))",F)`.

Definitie 2.1 (plaintext-PAL syntax) Een ABNF voor PAL-formules in platte tekst:

```
fml  =  ALPHA *(ALPHA / DIGIT / "_") / "(" fml ")" /
        fml "<->" fml / fml "<=>" fml / fml "->" fml /
        fml "=>" fml / fml SP "v" SP fml / fml "|" fml /
        fml "&" fml / "~" fml / "-" fml /
        "box" 1*DIGIT fml / "[" 1*DIGIT "]" fml /
        "dia" 1*DIGIT fml / "<" 1*DIGIT ">" fml /
        "[" fml "]" fml / "<" fml ">" fml /
        LWSP fml / fml LWSP /
```

Symbolen hebben hun gebruikelijke betekenis en de meeste variaties zijn toegestaan. Zo kunnen we dus zowel `boxbobdiamary(a|b)` voor $\Box_{bob}\Diamond_{mary}(a \vee b)$ als `[1]<2>(p v q)` voor $\Box_1\Diamond_2(p \vee q)$ schrijven. String `[bob]a` wordt, als `bob` niet eerder expliciet als agent geïntroduceerd is door een `...boxbob...` of `...diabob...`, geïnterpreteerd als $[bob]a$. Strings `box`, `dia`, `v` betreffen steeds onderkast letters (ABNF is officieel case-insensitive). Extra spaties en newlines worden genegeerd. Voorbeeld: `([1](g <-> box1 p) & <1>~g) => ~g` (wees liever consistent). □

Operatorprioriteit is standaard: (), [], $\langle\rangle$, $\Box$, $\Diamond$ en $\neg$ binden het sterkst, $\leftrightarrow$ sterker dan $\rightarrow$ dan $\vee$ dan $\wedge$. Het is goed mogelijk Prolog-operatoren te gebruiken om formules op te bouwen en deze direct in bewijzen te gebruiken. De verlangst om '`[fml1]fml2`' voor updates te gebruiken en '`boxbob X`' (toegestaan) voor '$\Box_{bob}X$' weerhoudt mij hiervan. De volledige specificatie is gegeven in Prolog DCG-notatie in de appendix; zie ook *def. 2.5*.

2.2 Automatisch epistemische logica

2.2.1 regels en definities

In de tableaubewijzen uit hoofdstuk 1 moet bij toepassing van de $\Box_i$ regels: $\sigma\ \Box_i\varphi$ analyseren we tot $\sigma'.n_i\ \varphi$ danwel $\sigma'\ \varphi$, het correcte label n_i 'geraden' worden. In *vrije-variabelen gelabelde tableaus* stellen we deze keuze uit tot het moment waarop we een tableautak of het hele tableau sluiten.

Definitie 2.2 (vrije variabelen labels) Een vrije-variabelen label is een eindige opeenvolging van constanten en *variabelen*, alle met een (maar mogelijk leeg) agent-subscript. Variabelen schrijven we als X_i, Y_i, Z_i en een label is bijvoorbeeld $1.2_1.X_2.3_1.Y_3$. De termen X_i en n_j zijn mogelijke-wereld-symbolen en $1....n_i$ is een *i-label*, i is een agent-suffix (*vgl. def. 1.6*). In Prolog representeren we labels in standaard Prolog-lijsten, deze worden in dit licht *achterstevoren* afgedrukt. Hoewel de punt-notatie van labels al bijna Prolog is nemen we de gebruikelijke weergave: `L = [Y:3,3:1,X:2,2:1,1]`. □

De bekende $\Box_i$ regels vervangen we nu door regels die variabelen in labels introduceren in plaats van constanten selecteren: $\sigma\ \Box_i\varphi$ analyseren we tot $\sigma'.X_i\ \varphi$ met X een nieuwe variabele in het tableau. Hoe liberaal er met de toekenning van variabelen omgegaan kan worden is afhankelijke van de opgelegde framecondities [1]. Het zal blijken dat we in tableaus voor EL heel vrij om kunnen gaan met variabelen instantiatie en dat vrije-variabelen-tableaus eenvoudig zijn te relateren aan tableaus zonder variabelen.

Voorbeeld 2.1 (variabelen vs. herhaalde $\Box_i$-regel toepassing)

1	$\Box_i(\Diamond_i a \wedge \neg a)$	1.	1	$\Box_i(\Diamond_i a \wedge \neg a)$	1.
1	$\Diamond_i a \wedge \neg a$	2.	$1.X_i$	$\Diamond_i a \wedge \neg a$	2.
1.2_i	a	3.	1.2_i	a	3.
1	$\neg a$	4.	$1.X_i$	$\neg a$	4.
1.2_i	$\Diamond_i a \wedge \neg a$	5.		$[X] = [2]$	
1.2_i	$\neg a$	6.			
	$=$				

□

In voorbeeld 2.1 ziet het variabelen-tableau rechts er nogal anders uit dan het tableau zonder variabelen. X instantieert tot 2, terwijl 1.2_i ná $1.X_i$ op het tableau verscheen, iets wat in tableaus zonder variabelen niet te realiseren is. Tableaubewijzen met variabelen zijn echter te herschrijven als

(veel langere) tableaus zonder variabelen, waarmee de correctheid van vrije-variabelen-tableaus aangetoond kan worden.

In de hier gepresenteerde vrije-variabelen-tableaus is het voor bewijzen voor bijvoorbeeld systeem K niet correct variabelen met elkaar te unificeren, d.w.z. $\nvdash_K \Box\varphi \rightarrow \Diamond\varphi$. Voor $S5_n$ tableaus is dit wel toegestaan omdat iedere mogelijke wereld w^n in een epistemische model *altijd* voor iedere agent i ten minste één toegankelijkheidsrelatie telt, nl.: $w^n \sim_i w^n$. In termen van gelabelde tableaus: een formule $1....n_j.X_i.m_k...\ \varphi$ op een tableautak wijst op de toelaatbaarheid van een verder identiek label $1....n_j.m_k...\ \varphi$ op diezelfde tak. In dit perspectief is het verleidelijk om $S5_n$ tableaus te sluiten op formules $1.X_1.2_2\ \varphi$ en $1.Y_2\ \neg\varphi : [X_1, 2_2] = [\emptyset, Y_2]$ met de interpretatie van $n_i.\emptyset_j.m_k$ als $n_i.m_k$. Hiermee is de keuze voor analyse van $\sigma\ \Box_i\varphi$ tot $\sigma'.X_i\ \varphi$ danwel $\sigma'\ \varphi$ *ook* uitgesteld tot het moment van sluiting. De lezing van $1.2_1.\emptyset_3.3_1$ moet nu 1.3_1 zijn, $1.2_1\ \Diamond_1\varphi$ analyseren we immers tot $1.3_1\ \varphi$.

De volgend prologpredikaten `red/3`, `lprove/2` en `extend/1` bevat zg. *red cuts*. In Prolog betekent een red cut toepassing van '`!`' (cut) zodat mogelijke nieuwe oplossingen die een programma zou kunnen vinden van ons worden afgesneden. In verband met de efficiëntie is het hier zeker wenselijk ons te verlagen tot deze praktijken.

Definitie 2.3 (label reductie) Een label σ met n variabelen $V = \{X_a^1, ..., X_z^n\}$ heeft maximaal 2^n interpretaties. Voor iedere interpretatie van σ instantiëren we variabelen $X \in S \subseteq V$ tot $\emptyset$ (`0`). De corresponderende '*oppervlaktevorm*' van σ is het label dat ontstaat als we nu van voor naar achter door σ lopen en steeds achtereenvolgens een $\emptyset_i$ verwijderen en de zo mogelijk ontstane opeenvolgingen van mogelijke-wereld-symbolen met gelijke agent-suffixen reduceren tot het meest rechter (laatst geïntroduceerde) symbool. De oppervlaktevorm van $1.\emptyset_1.2_3.\emptyset_2.Z_3$ is zo $1.Z_3$. Deze definitie is geformaliseerd in predikaat `red/3` waarmee alle reducties van een label gegenereerd en/of gecontroleerd kunnen worden:

```
red(S,[_:I|L],[X:I|R]) :- !,
        red(S,L,[X:I|R]).
red(S,L,[0:_|R]) :-
        red(S,L,R).
red(S,[X|L],R) :- !,
        red(S,L,[X|R]).
red(S,[],S).
```

Dit predikaat is aan te roepen als `red(Oppervlaktevorm,Label,[])`. Dit predikaat maakt gebruik van een variatie op een zgn. *difference list* om deels gereduceerde labels in te representeren. □

Definitie 2.4 (label unificatie) Twee labels σ_1 en σ_2 *unificeren* $U(\sigma_1,\sigma_2)$ als deze te interpreteren zijn met oppervlaktevormen σ_1' en σ_2' zodat toekenning $\sigma_1' = \sigma_2'$ slaagt, d.w.z. variabelen unificeren met variabelen of constanten en constanten unificeren alleen met gelijke constanten. Unificatie van twee labels is te bewerkstelligen door aanroep `red(O,L1,[]), red(O,L2,[])`. Dit is correct en geeft tegelijkertijd veel ruimte tot verbetering. Twee labels $1....n_j$ en $1....m_k$ kunnen duidelijk nooit unificeren en zo is er gaandeweg ook steeds gelegenheid onmogelijke unificaties kort te sluiten. De lezer wordt uitgenodigd een efficiënt direct unificatie-predikaat te definiëren. De hier gepresenteerde vorm van label-unificatie is een specifiek voor epistemische logica vereenvoudigde variatie op het raamwerk uit [11]. □

Met betrekking tot de introductie van *constante* mogelijke-wereld symbolen n_i op tableautakken $\mathcal{B}$ door toepassing van $\Diamond_i$-regels is tot nu toe alleen opgemerkt dat n_i 'nieuw' moet zijn, d.w.z. uniek voor $\mathcal{B}$. Het is echter afdoende een gelabelde formule $\sigma\ \Diamond_i\varphi$ op $\mathcal{B}$ te analyseren tot $\sigma.n_i\ \varphi$ met n_i uniek voor φ, d.w.z. iedere formule $\Diamond_i\varphi$ in een tableau analyseren we met identieke n_i. In navolging van [1] wordt deze *finite diamond-rule* op een directe manier toegepast: $n = \varphi$. Voor de (grafische) weergave van tableaus is dit niet wenselijk, de komende voorbeelden blijven de bekende notatie gebruiken. Een implementatie zou baat kunnen hebben bij globale toekenning van constanten aan iedere modale subformule van een te sluiten tableau. De specifieke ins en outs van de toelaatbaarheid van deze constructie komen hier niet aan bod. Introductie van variabelen X_i wordt hier eenvoudig overgelaten aan Prolog door middel van anonieme variabelen '_'.

Definitie 2.5 (tableauregels in Prolog) Tableauregels gedefinieerd in `rule/3` en `rule/4` met premisse, type en conclusie(s). Dit zijn de Prolog equivalenten van de tableauregels zoals we die in hoofdstuk 1 hebben gezien in definities 1.9 en 1.11. We hebben weer conjunctie '`conj`', disjunctie '`disj`', $\Box$ '`know`' en $\Diamond$ '`poss`'. Argumenten zijn steeds links de premisse en rechts de conclusie(s).

```
rule(L:and(A,B),            conj, L:A,L:B).           % A & B
rule(L:not(imp(A,B)),       conj, L:A,L:not(B)).      % ~(A -> B)
rule(L:not(or(A,B)),        conj, L:not(A),L:not(B)). % ~(A v B)
rule(L:or(A,B),             disj, L:A,L:B).           % A v B
rule(L:imp(A,B),            disj, L:not(A),L:B).      % A -> B
rule(L:not(and(A,B)),       disj, L:not(A),L:not(B)). % ~(A & B)
rule(L:eq(A,B),             disj, L:and(A,B),L:and(not(A),not(B))).
rule(L:not(eq(A,B)),        disj, L:and(A,not(B)),L:and(not(A),B)).
rule(L:not(bra(F)),         doub, L:not(F)).          % ~(F)
rule(L:bra(F),              doub, L:F).               % (F)
rule(L:not(not(F)),         doub, L:F).               % ~~F
rule([_:I|L]:box(I,F),      know, [_:I|L]:F).         % <I>[I]F
rule([_:I|L]:not(dia(I,F)), know, [_:I|L]:not(F)).    % [I]~<I>F
rule(L:box(I,F),            know, [_:I|L]:F).         % [I]F
rule(L:not(dia(I,F)),       know, [_:I|L]:not(F)).    % ~<I>F
rule([_:I|L]:dia(I,F),      poss, [F:I|L]:F).         % [I]<I>F
rule([_:I|L]:not(box(I,F)), poss, [not(F):I|L]:not(F)).% <I>~[I]F
rule(L:dia(I,F),            poss, [F:I|L]:F).         % <I>F
rule(L:not(box(I,F)),       poss, [not(F):I|L]:not(F)).% ~[I]F
```

□

De expliciete representatie van '(φ)' als '`bra(Fml)`' zoals te zien in *def. 2.5* kan gezien worden als een schoonheidsfoutje onstaan onder invloed van de wat ongelukkige implementatie van operatorprioriteit in PALparse. Let ook op de '`not(F):I`' labels in de laatste en voor-voorlaatste regels.

2.2.2 beslissingsprocedure

In deze meest eenvoudige implementatie/specificatie worden tableautakken gerepresenteerd in standaard Prologlijsten en bewijzen worden geconstrueerd door standaard Prolog *depth-first search* over tableautakken. Het is niet gezegd dat als we een tableautak sluiten op een bepaalde toekenning van labels dat dit ook de 'juiste' sluiting is *en* de *volgorde* van analyse van formules is mede beslissend voor tableaubewijzen. Voorbeeld: $\{1\ \Box_1(a \vee b) \wedge (\Diamond_1(\neg a \wedge \neg b) \vee \Diamond_1(\neg b \wedge \neg a))\}$ waarbij $\Box_1(a \vee b)$ in beide takken $\Diamond_1(\neg a \wedge \neg b)$

en $\Diamond_1(\neg b \wedge \neg a)$ geanalyseerd moet worden. Voor het overgrote deel volg ik steeds [1].

Definitie 2.6 (tableauconstructie en sluiting)

```
lprove([L:Fml|Fs],Branch) :-
        (rule(L:Fml,doub,NlF), !, lprove([NlF|Fs],Branch));
        (rule(L:Fml,poss,NlF), !, lprove([NlF|Fs],Branch));
        (rule(L:Fml,conj,A,B), !, lprove([A,B|Fs],Branch));
        (rule(L:Fml,disj,A,B), !,
           lprove([A|Fs],Branch), !,
           lprove([B|Fs],Branch)).

lprove([L:Fml|_],Branch) :-
        (Fml = not(Neg); Neg = not(Fml)),
        member(K:Neg,Branch),
        red(S,L,[]), red(S,K,[]), !.

lprove([L:Fml|Fs],Branch) :- !,
        lprove(Fs,[L:Fml|Branch]).

lprove([],Branch) :- !, extend(Branch).
```

□

Het predikaat `lprove/2` is aan te roepen met `lprove([[]:Fml],[])`, `[]:Fml` de wortel van het tableau ('`[[1]:Fml]`' mag uiteraard ook). Formule `[]:Fml` is de *actieve* formule, te analyseren door `lprove/2`, formules in `Branch` zijn *passief*. Alle '`doub`', '`poss`', '`conj`' en '`disj`' regels worden toegepast op de actieve formules. Een vork, disjunctie, wordt eerst links en bij succes dan rechts geanalyseerd. *Sluiting* van tableaus gebeurt zo alleen op '`atom`' en '`know`' formules, tegenspraak volgt als `L:not(F)` ∈ `Branch`, `K:F` ∈ `Branch` en `K`, `L` unificeren (*def. 2.4*). Als de tak niet sluit dan wordt `L:Fml` passief. Als alle actieve formules geanalyseerd zijn en de tableautak sluit niet dan kan de tak mogelijk uitgebreid worden d.m.v. predikaat `extend/1` waarin de '`know`' regels worden toegepast op elementen van `Branch`. De stijl van deze declaratie is mede geïnspireerd op [9].

Definitie 2.7 (tableau extensie en label expansie)

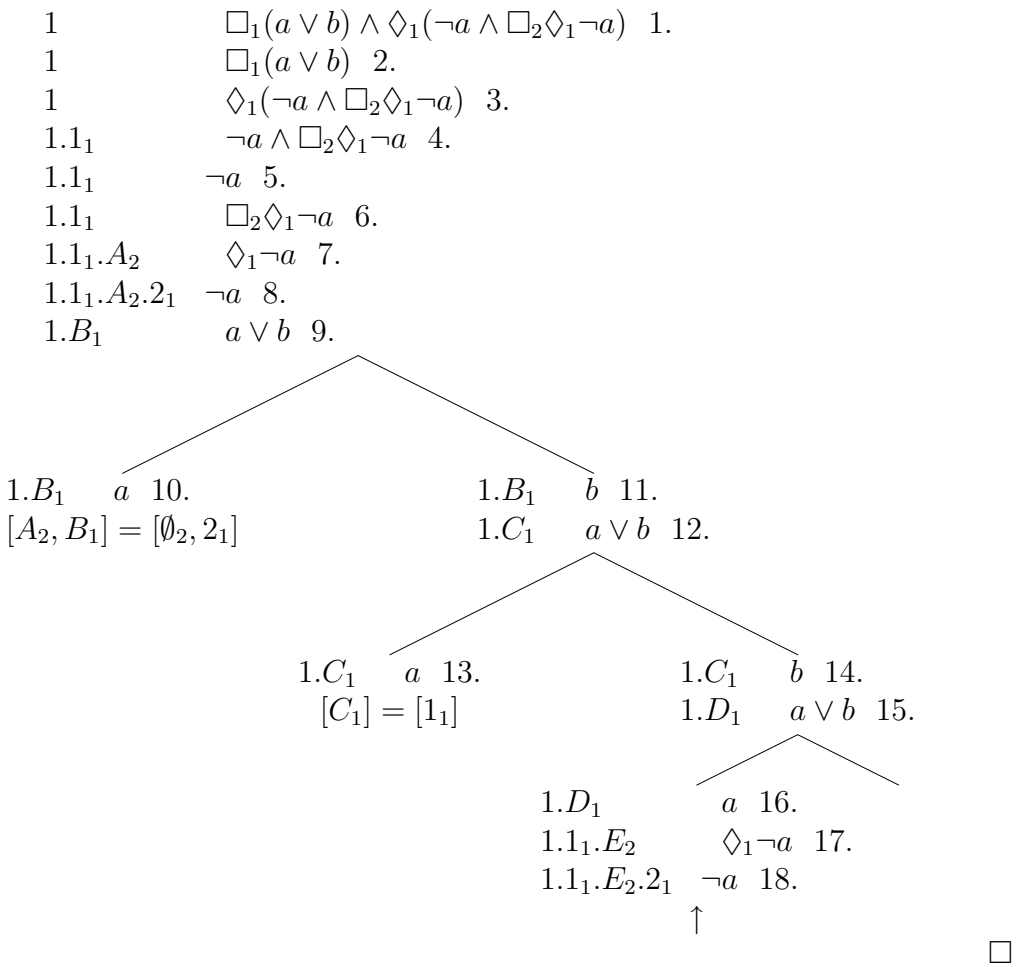
```
extend(Branch) :-
        select(InLbl:Fml,Branch,RestB),
        expand(M,InLbl,Inst,L),
        rule(L:Fml,know,M:F), !,
        lprove([M:F],[(Inst:L):Fml|RestB]).

expand([X:_|_],(Inst:L),[X|Inst],L) :- !,
        ground(Inst), freeze(X,\+member(X,Inst)).

expand([X:_|_],L,[X],L).
```

□

Gelabelde formules $\sigma\ \Box_i\varphi$ moeten mogelijk meerdere malen geanalyseerd worden, wat enige administratie vereist. Als alle actieve formules op een tableautak geanalyseerd zijn selecteren we in `extend/1` een passieve formule `InLbl:Fml` van type '`know`'. Als premisse `InLbl:Fml` niet eerder geanalyseerde is wordt `X`-conclusie `M:F` de actieve formule, `([X]:L):Fml` plaatsen we aan het begin van de passieve staart van de tableautak ter indicatie van de bestaande analyse van `L:Fml` met `X`. Als premisse `InLbl:Fml` van de vorm `([Y|Inst]:Lbl):Fml` is dan is deze formule al eerder geanalyseerd op deze tableautak. Als `Y` een variabele is dan selecteren we een andere formule. De her-analyse van `InLbl:Fml` levert ons dan immers niets op, alle zo te introduceren formules waarmee de tak mogelijk te sluiten is zouden al op de tak moeten staan. Als `Y` geïnstantieerd is wordt `X`-conclusie `M:F` de actieve formule met `([X,Y|Inst]:Lbl):Fml` de nieuwe passieve tegenhanger in de staart. Restrictie op `X` is nu `X` $\notin$ `[Y|Inst]`, precies deze analyses van `L:Fml` staan immers al op de tak. Deze restrictie moet nageleefd worden in de toekenning van `X`. Deze toekenningen moeten voor `X` dus *geblokkeerd* worden. Ik noem dit ook wel *verboden* toekenningen. Hiertoe is gebruik gemaakt van standaard Prolog *coroutining*: zodra `X` geïnstantieerd is wordt `\+member(X,[Y|Inst])` aangeroepen, met backtracking over de verboden toekenningen tot gevolg[1].

[1] In [1] worden restricties op de toekenning van variabelen bijgehouden in een lijst, een andere mogelijkheid is gebruik te maken van het *constraint* systeem waarbij het noodzakelijk is formules op getallen te mappen in de '`poss`' regels. Tests wijzen uit dat prestaties hierbij vooralsnog gelijk blijven.

Voorbeeld 2.2 (grafische weergave van een ELtap tableau)



□

Definities 2.3, 2.5, 2.6 en 2.7 tezamen vormen tableaubewijzer ELtap. In voorbeeld 2.2 wordt een ELtap-beslissingsprocedure weergegeven, verdere executie wordt bij 18 kortgesloten door de onder invloed van A_2 en B_1 geblokkeerde toekenning van $[E_2, D_1] = [\emptyset_2, 2_1]$. Gevolg: $\Box_2\Diamond_1\neg a$ en $\Box_1(a \vee b)$ worden niet opnieuw actief omdat D_1 en E_2 ongeïnstantieerd zijn en het tableau blijft open.

2.2.3 correctheid en volledigheid

Correctheid bespreek ik hier door de tableaubewijzen in ELtap te relateren aan de tableaus uit hoofdstuk 1. In [16] relateren Posegga en Schmitt de executie van prolog programma's aan 'computationele bomen'. Het is gemakkelijk ELtap-tableaus als voorbeeld 2.2 te laten genereren en 'ELtap ta-

bleaus' mogen zo verstaan worden. Volledigheid van ELtap komt hier alleen schetsmatig aan bod.

Stelling 2.1 (correctheid) sluitende ELtap tableaus zijn te herschrijven als sluitende Eltab tableaus zoals gedefinieerd in hoofdstuk 1

Bewijs (schets) Voor bewijs $\mathcal{T}$ kunnen we de volgende procedure toepassen.

1. Alle (schijnbare) toepassingen van $\Box_i$ regels $\sigma\ \Box_i\varphi/\sigma'.n_i\ \varphi$ die symbolen n_i selecteren die op dat moment nog niet geïntroduceerd zijn op de tak vervangen we door $\sigma\ \Box_i\varphi/\sigma'\ \varphi$ analyses. Kopieer de volledige $\sigma\ \Box_i\varphi/\sigma'.n_i\ \varphi$ analyse naar de bladeren onder $\sigma\ \Box_i\varphi/\sigma'\ \varphi$.

2. Alle ongeïnstantieerde variabelen in $\mathcal{T}$ krijgen toekenning $\emptyset$, ook als deze toekenning geblokkeerd is en alle labels in $\mathcal{T}$ vervangen we door hun nu enige interpretaties.

1 en 2 preserveren sluiting en leveren precies de gewenste vorm tableaus. □

De hier voorgestelde tableauprocedure is 'gevaarlijk' en wel om de volgende reden. Het gaat om unificatie van labels zoals te produceren met formules als $\Box_i\Diamond_j\Diamond_i(\Box_j a \wedge \Diamond_j\Box_i\neg a)$. Labels $A_i.1_j.2_i.B_j$ en $A_i.1_j.2_i.3_j.C_i$ unificeren: $[A_i, B_j, C_i] = [A_i, 3_j, \emptyset_i]$. Wat als we, door een ingewikkeld gefabriceerde formule φ een unificatie kunnen maken waarin variabele(n) A_i in het *ene* label een andere toekenning zou moeten krijgen dan in het *andere* label? Dit is, logisch gezien, niet ontoelaatbaar of onvoorstelbaar en zou een her-analyse van boosdoener φ moeten bewerkstelligen. ELtap biedt hier geen gelegenheid voor: de eerste $U(\sigma_1,\sigma_2)$ faalt, variabelen worden niet geïnstantieerd en verdere analyses geblokkeerd. Een mogelijk correct bewijs faalt in ELtap met mogelijk onvolledigheid tot gevolg. Ik onderzoek dit geval systematisch en laat zien dat voor $S5_n$ tableaus dit probleem toch niet bestaat.

Definitie 2.8 (*Zilant*-unificatie)
$U(\sigma_1,\sigma_2)$ is een Zilant-unificatie als:

1. Zilant-labels σ_1, σ_2 hebben een gemeenschappelijke staart '1 . . .' en twee verschillende koppen.

2. kopieen σ_1^c en σ_2^c unificeren waarbij een of meer gemeenschappelijke/gekopieerde variabelen $X_i^{c_1}$ en $X_i^{c_2}$ in de staart *verschillende* toekenningen krijgen. Noem $X_i^{c_1}$ en $X_i^{c_2}$ *vleugels*.

3. In $U(\sigma_1,\sigma_2)$ geldt voor alle vleugels $X_i = X_i$.

□

Stelling 2.2 (*Zilant*-unificatie slaagt)

Bewijs Ik behandel drie gevallen:

1. Het eerste (achterste) paar vleugels $X_i^{c_1} = \emptyset_i$ en $X_i^{c_2} = n_i$. Op de positie van $X_i^{c_1}$ is in de interpretaties van σ_1^c en σ_2^c dus n_i terecht gekomen. Tussentijds resultaat in de interpretatie van σ_1^c moet dan geweest zijn $\ldots X_i^{c_1}.n_i \ldots$ wat reduceert tot $\ldots n_i \ldots$ *zonder* toekenning van $X_i^{c_1}$. Conflicterende toekenningen $X_i^{c_1} = \emptyset_i$ en $X_i^{c_2} = n_i$ zijn dus onmogelijk vereist voor unificatie.

2. Vleugels $X_i^{c_1} = m_i$ en $X_i^{c_2} = n_i$. Uit 1 volgt dat de deel-interpretaties van de staarten van σ_1^c en σ_2^c even lang zijn door gelijke toekenning van $\emptyset$. Posities van toegekende $X_i^{c_1}$ en $X_i^{c_2}$ in σ_1^c en σ_2^c moeten dus gelijk zijn. Interpretaties van σ_1^c met n_i op de positie van $X_i^{c_1}$ en σ_2^c met m_i op de positie van $X_i^{c_2}$ vereisen reducties $\ldots X_i^{c_1}.n_i \ldots$ tot $\ldots n_i \ldots$ en $\ldots X_i^{c_2}.n_i \ldots$ tot $\ldots m_i \ldots$. $X_i^{c_1}$ en $X_i^{c_2}$ moeten nu in unificatie tegelijkertijd gereduceerd *en* toegekend worden, wat onmogelijk is.

3. Vleugels $X_i^{c_1} = Y_i$ en $X_i^{c_2} = x_i$ met $Y_i = x_i$ geblokkeerd. In het geval dat dit zou kunnen gebeuren, ook in het geval $x \equiv \emptyset$, is σ_2^c onderdeel van een her-analyse onder invloed van `extend/1` met voor het overige gelijke formules met $Y_i^c \equiv x_i$ al op de tableautak ter Zilant-unificatie aanwezig. Afhankelijk van de representatie/volgorde in `Branch` moet dit een unificatie uitgevoerd op een reeds gesloten tak zijn, dit gebeurt duidelijk niet.

Zilant-unificatie slaagt voor $S5_n$. Voor logica's als $S4_n$ zou een dergelijke benadering mogelijk niet van de grond komen. De indruk is zo gewekt dat dit bewijssysteem erg specifiek is voor $S5$ toegankelijkheidsrelaties. □

Volledigheid van tableaus bewijzen we aan de hand van *verzadigde tableaus*. Een tableautak is verzadig als alle mogelijke toepassingen van tableauregels daadwerkelijk zijn toegepast op de tak. Als een verzadigde tableautak niet sluit dan is deze tak een tegenvoorbeeld voor de te bewijzen formule. Het is dus zaak aan te tonen dat als in de ELtap beslissingsprocedure een tak niet sluit deze tak is te verzadigen zonder hem te sluiten.

Stelling 2.3 (volledigheid) Als ELtap faalt in tableautak $\mathcal{B}$ van een bewijspoging voor φ dan is $\mathcal{B}$ een tegenvoorbeeld voor φ

Bewijs (schets) `Branch`, corresponderend met $\mathcal{B}$, bevat alleen elementen

van de vorm `L:not(P)`, `L:P` en `([V|_]:L):K` met `var(V)`, `atom(P)` en `rule(_:K,know,_)`. De lijst actieve formules is `[]`, leeg. Sluiting op `Branch` is sluiting van $\mathcal{B}$ op atomaire formules $\sigma\ p$ en $\sigma\ \neg p$ en volledig. Op $\mathcal{B}$ zijn alle formules nu in ieder geval ééns geanalyseerd tot op het atomaire niveau. De set S bevat alle formules op $\mathcal{B}$, S^+ is een multiset van S. $\mathcal{B}$ sluit op elementen uit S of is open. $\mathcal{B}^+$ is een tableautak met dezelfde elementen als $\mathcal{B}$ ... Her-analyse van formules niet van type '`know`' op $\mathcal{B}$ leidt tot $\mathcal{B}^+$, triviaal. Een vork leidt hierbij ook altijd tot in ieder geval één te sluiten tak van de vorm $\mathcal{B}$. Her-analyse van formules `([V|_]:L):K` met `var(V)` leidt ook alleen tot méér identiek formules op $\mathcal{B}$. □

2.2.4 geselecteerde optimalisaties en prestaties

ELtap is eerder een *specificatie* dan een volwaardige implementatie. Het is al opgemerkt dat de unificatie van labels zoals hier voorgesteld bijna de meest inefficiënte voorstelbare is. Dat ELtap soms toch vlot bewijzen levert is te danken aan de kwaliteit van prolog implementaties. Om tot een competitieve stellingbewijzer te komen kunnen bijvoorbeeld de optimalisaties voor leanT^AP in [16] worden aangepast. Alleen *universele variabelen tableaus* zijn hier geïmplementeerd; de efficiënte representatie van gelabelde tableaus blijft steken in aanbevelingen. Een andere weg die hier niet genomen wordt is her-implementatie in een imperatieve of functionele programmeertaal. Het programmeergemak van prolog heeft een zekere prijs in prestatie.

ELtap is niet onderworpen aan uitbreidde prestatie-tests. ELtap is *niet* gemeten tegen de *TANCS Non Classical System Comparison* S5n probleemset. In appendix A.1.4 zijn eenvoudige testformules opgenomen.

universele variabelen

Een implementatie van universele variabelen tableaus die sterk afwijkt van [1] is opgenomen in appendix A.1.3, ELtap^u. Een variabele X geïntroduceerd door $\sigma\ \Box_i\varphi$ in $\sigma.X_i\ \varphi$ kan onder bepaalde omstandigheden als 'universeel' aangemerkt worden. Deze universele variabelen, 'wildcards' of 'jokers', aangegeven met '$\star$' worden alleen lokaal, op een enkele tableautak toegekend en blijven zo beschikbaar in de overige takken van $\mathcal{T}$ voor sluiten op andere toekenningen van X. Hoewel de precieze omstandigheden waaronder variabelen universeel kunnen blijven niet helemaal geformaliseerd zijn kunnen we een aantal eenvoudige regels toepassen die een implementatie ruimschoots ten goed komen. Een variabele is op moment van introductie altijd universeel. Variabelen in een gelabelde formule $\sigma\ \varphi$ verliezen hun universaliteit door een vork in het tableau over φ, wat overeen komt

met het feit dat kennis/noodzakelijkheid niet distribueerd over disjunctie: $\Box_i(\varphi \vee \psi) \not\equiv (\Box_i\varphi \vee \Box_i\psi)$. In de overige tableautakken kunnen we $\star$ steeds zien als een copie van een variabele X in plaats van X zelf. We kunnen formules $\sigma\ \Box_i\varphi$ dus analyseren tot $\sigma.\star_i\ \varphi$ en iedere toekenning aan $\star$ onmiddellijk weer vergeten/terugdraaien. Bij de analyse van gelabelde formules $\sigma\ \varphi \vee \psi$ "bevrijden" we alle $\star$'s in σ en vervangen deze door verse vrije variabelen. ELtap^u is veel efficiënter dan ELtap zonder universele variabelen omdat 1) bewijzen zijn mogelijk exponentieel veel korter, 2) universele variabelen zijn weergegeven als constanten `'*'` in plaats van Prolog variabelen, 3) vork en extensie van takken is één operatie geworden in `extend/1`.

representatie van tableaus

De weergave van tableautakken in standaard prolog lijsten is overzichtelijk maar bijzonder inefficiënt. Het doorzoeken van `Branch` kan efficiënter door de tak in een *hash* of *dictionary* te representeren of tableautakken te compileren tot prolog programma's en gebruik te maken van prolog matching. Tableaus kunnen expliciet in bomen gerepresenteerd worden in plaats van in losse lijsten. We kunnen labels bij elkaar nemen en representeren in bomen of strukturen anderzijds die zowel compact zijn als het zoeken van complementaire formules én unificatie vergemakkelijken. Of en hoe dergelijke representaties te realiseren zijn is hier vooralsnog niet onderzocht. Het staat vast dat hier grote prestaties te leveren zijn.

2.3 Automatisch public announcement logica

In appendix A.1.2 vindt u module PAL2EL, `[pal2el/2]` waarin updatereductie zoals gedefinieerd in *def. 1.4* is geïmplementeerd. Automatisering van PAL is hiermee een feit. Tegelijkertijd hebben we hiermee een procedure in handen om gemakkelijk betekenisvolle formules in EL te genereren die *niet* in afzienbare tijd zijn te bewijzen door ELtap.

Een trouwe implementatie van PALtab als extensie van ELtap is problematisch. Wat gemakkelijk en inzichtelijk is voor een menselijke bewijzer: de juiste labels selecteren voor introductie in submodellen is algoritmisch niet uitgewerkt. Een equivalent van vrije-variabelentableaus voor updates om hierin te faciliteren is niet gevonden. ELtap biedt in de huidige vorm waarschijnlijk geen perspectief op een directe implementatie van PAL-tableaus die merkbaar efficiënter is dan een benadering gebaseerd op updatereductie.

De formules die ontstaan door updatereductie bevatten veel propositionelen redundantie, identieke formules geldig in dezelfde of verschillende la-

bels. Deze redundanties zijn niet uniek voor updatereductie en vragen om ontwikkelen van bewijsstrategieën onafhankelijk van updates. Ontwikkeling van bewijsstrategieën om redundanties te herkennen in formules is dus meer urgent dan expliciete directe analyse van updates en heeft een veel breder toepassingsgebied.

Om inzicht te krijgen in de specifieke mogelijkheden van directe automatisering van PAL kan ook het beste eerst naar bestaand onderzoek in deze gerelateerde problematiek gekeken worden. Dit zijn zaken die hier niet aan bod komen.

2.4 Voorbeeldsessie

Ter indicatie van het gebruik van de gepresenteerde programma's volgen enige voorbeelden. Hier is onder andere weer gekeken naar de formules omtrend de Drie Wijze Mannen Puzzel. Hier is `UELtap.pl` met SWI-Prolog versie 5.6.16 gebruikt. Programma `UELtap.pl` zoals gegeven in appendix A.1.3 laad modules `pal2el`, `palparse` en `test`. Deze programma's zijn ook ongewijzigd te gebruiken in SICStus Prolog. Een interface naar `UELtap.pl` is ook online beschikbaar op

`http://www.ai.rug.nl~mathijs/logic/piel.html.`

De functionaliteit daar is beperkt tot het parsen en bewijzen van formules. De respons van ELtap^u is hier beperkt tot '`yes`', als er een bewijs is voor de gegeven formule, '`no`', als er geen bewijs gevonden is. Om vergissingen te voorkomen geeft de online-versie atomaire formules automatisch letterlijk terug. Resultaat van `a`' is dus ook '`a`'. Een verzoek tot bewijs van een lege formule '' levert een vriendelijke begroeting op ter indicatie dat het programma in ieder geval wel *iets* doet. Online bewijzen worden na enige tijd afgebroken, dit resulteert in een 'Internal Server Error'.

2.4.1 start een sessie

```
$ pl -f UELtap.pl
%  palparse compiled into palparse 0.01 sec, 18,144 bytes
%  test compiled into test 0.01 sec, 22,172 bytes
%  pal2el compiled into pal2el 0.01 sec, 3,520 bytes
%  library(lists) compiled into lists 0.01 sec, 11,628 bytes
%  UELtap.pl compiled 0.05 sec, 65,988 bytes
Welcome to SWI-Prolog (Version 5.6.16)
Copyright (c) 1990-2006 University of Amsterdam.
```

```
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?-
```

Programma `UELtap` is succesvol geladen en de vertrouwde Prolog prompt verschijnt. We beginnen onmiddelijk met een eenvoudig bewijs.

2.4.2 een eenvoudig bewijs

```
?- prove("a -> a").
% 18 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips)

Yes
```

Formule $a \rightarrow a$ is duidelijk bewijsbaar en prolog geeft enige statistieken en '`Yes`'.

2.4.3 een bewijs voor de drie wijze mannen puzzel

```
?- prove("[(a|b|c) &
           (~a -> [2][3]~a) &
           (~b -> [1][3]~b) &
           (~c -> [1][2]~c)][~[1]a][~[2]b][3]c").
% 34,318 inferences, 0.06 CPU in 0.07 seconds (84% CPU, 549088 Lips)

Yes
```

Dit is meteen al vrij indrukwekkend. Hier is de notatie `[1]` voor $\Box_1$ en `|` voor $\vee$ gebruikt. het is ook toegestaan `box1a`, `dia1 a` en `a v b` te schrijven. Hier is ook gebruik gemaakt van het feit dat in $S5_n$ uit $\Box_1\Box_2\neg a$ volgt dat $\Box_1\neg a$ en $\Box_2\neg a$. Vergelijk de uiteenzetting in hoofdstuk 1. In tableau 1.6 zagen we al dat `(~a -> [2][3]~a)` niet nodig was voor een bewijs en dat kunnen we hier ook proberen. Wat als we ook `(~b -> [1][3]~b)` weglaten?

2.4.4 géén bewijs voor drie wijze mannen en twee handen

```
?- prove("[(a|b|c) & (~c -> [1][2]~c)][~[1]a][~[2]b][3]c").
```

Dit ziet er niet hoopvol uit, ELtap^u vindt geen bewijs maar we krijgen onze prompt ook niet terug. Hoewel ELtap^u uiteindelijk wel zal ophouden met zoeken naar een bewijs, dit bewijs is er duidelijk niet, houd ik het voor gezien met `Ctrl-c`:

```
^C
Action (h for help) ? abort
% Execution Aborted
?-
```

Bij nadere inspectie wordt het iets duidelijker waarom dit zo lang kan duren. Zelfs een korte formule met een update is in vertaling naar EL al behoorlijk lang en updatereductie leidt hier tot kennis over meerdere disjuncties, juist het type formule wat moeilijk te analyseren is.

2.4.5 PAL naar EL vertalen

```
?- parse("[a][a|b][1](b->a)",X), pal2el(X,E).

X = update(a, update(or(a, b), box('1', bra(imp(b, a)))))
E = imp(or(imp(a, a), imp(a, b)), imp(a, box('1', imp(or(
imp(a, a), imp(a, b)), imp(imp(or(imp(a, a), imp(a, b)),
imp(a, b)), imp(or(imp(a, a), imp(a, b)), imp(a, a)))))))

Yes
```

Met `trace/2` kunnen we in SWI-Prolog kijken naar waar precies een bewijs voor een formule faalt of slaagt, dit is een eenvoudige manier om bijvoorbeeld tegenvoorbeelden of tableaus af te drukken.

Conclusie

In deze scriptie is het volgende bereikt. Het tableaubewijssystemen PALtab uit hoofdstuk 1 is een nieuw eenvoudig en overzichtelijk bewijssysteem binnen een bekend formalisme waarmee formules in publieke mededelingen logica op een systematische manier zijn te onderzoeken en presenteren. Het systeem geeft mogelijk nieuw zicht op het redeneren over kennis en kennisverandering en handvatten voor nader onderzoek naar de eigenschappen van public announcement logica. Eenvoudige dynamische epistemische situaties zoals de Drie Wijze Mannen Puzzel zijn gemakkelijk te analyseren met behulp van PALtab.

De stellingbewijzer ELtap uit hoofdstuk 2 geeft een nieuwe beslissingsprocedure voor het bewijzen van formules in epistemische logica $S5_n$. Door de implementatie zeer specifiek op logica $S5_n$ te richten is ELtap efficiënt in het verifiëren van complexe toegankelijkheidsrelaties voor verschillende agents. De gegeven implementatie laat echter te wensen over op het propositionele-vlak. De eenvoudige representatie van tableaus en labels maakt een serieuze uitbreiding naar directe tableaus voor public announcement logica niet haalbaar. De huidige implementatie maakt gebruik van automatische vertaling van PAL naar EL om formules in PAL te kunnen bewijzen. Logica PAL is zowel in vertaling als in directe bewijzen sterk disjunctief is over kennis *en* is sterk redundant over noodzakelijke evaluatie van identiek formules in verschillende mogelijke werelden (of labels). Voor de bewijsstrategie ontwikkeld in ELtap is dit zeer problematisch. In hoofdstuk 1 is de indruk gewekt dat er zinvolle representaties en strategieën zijn te ontwikkelen voor automatische bewijzen in PAL. Het is moeilijk gebleken de inzichten uit hoofdstuk 1 te verbinden aan bestaande formalismen voor de implementatie van tableaus. Om aan de gewekte verwachtingen te voldoen is nog veel werk te verzetten.

Op basis van het gepresenteerde werk is enig nader onderzoek te verrichten. Omdat semantische tableaus geregeld uit te breiden zijn met bijvoorbeeld kwantificatie is de lijn van onderzoek in hoofdstuk 1 mogelijk voort te zetten naar complexere dynamisch epistemische logica's [28]. Een natuurlijke extensie van PAL is *private* update logica waarbij de epistemische toestand

van alleen één specifieke agent ge-update wordt. Het is interessant om te zien of private update logica in een natuurlijke extensie van PALtab te formaliseren is.

Zoals vermeld in de introductie is *common knowledge* een belangrijk concept in het redeneren over epistemische situaties. Hier is dit concept niet aan bod gekomen omdat redeneren met common knowledge in gelabelde tableaus niet ontwikkeld is. Een uitwerking van gelabelde tableaus voor common knowledge is op zijn plaats. Een hele andere benadering van common knowledge in een sequente calculus wordt onderzocht in [24].

In de implementatie van ELtap is vooral geprobeerd om zicht te bieden op de mogelijkheden tot verbetering zonder zelf al te complex te worden. Er is zo de gelegenheid gecreëerd om in vervolgonderzoek direct te kunnen werken aan efficiëntie. De extensie van ELtap naar een directe stellingbewijzer voor PAL of nog andere logica's kan dan ook zeker doorgang vinden. Ook het genereren van meer betekenisvolle uitvoer dan '`yes`' of '`no`'—bijvoorbeeld tableaus of tegenmodellen—kan dan op de agenda geplaats worden.

Appendices

A.1 Prologprogramma's

A.1.1 Een parser voor plaintext-PAL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% palparse.pl - basic PAL formula DCG parser
%
% Mathijs de Boer - KI RuG - 2004
%
% parse(+"Source",-Formula).
%

:- module(palparse,[parse/2]).
:- dynamic agent/1.

parse(S,T) :- fml(T,S,[]), !.
parse(S,C) :- fml(_,S,X), !,
        append("Syntax error here: ",X,E),
        atom_codes(C,E).
parse([],'true\n') :- !.
parse(_,'syntax error!? (no helpfullness available)\n').


fml(Y) --> atom(X), gnm(X,Y).

atom(X)                         --> space, !, atom(X).
atom(not(X))                    --> "~", !, atom(X).
atom(not(X))                    --> "-", !, atom(X).
atom(bra(X))                    --> "(", !, fml(X), ")".
atom(box(I,X))                  --> "box",  agent(I), !, atom(X).
atom(dia(I,X))                  --> "dia",  agent(I), !, atom(X).
```

```
atom(box(I,X))             --> "box", !, var(I), {assert(agent(I))}, atom(X).
atom(dia(I,X))             --> "dia", !, var(I), {assert(agent(I))}, atom(X).
atom(box(I,X))               --> "[", agent(I), !, "]", atom(X).
atom(dia(I,X))               --> "<", agent(I), !, ">", atom(X).
atom(update(X1,X2))          --> "[", !, fml(X1), "]", atom(X2).
atom(not(update(X1,not(X2)))) --> "<", !, fml(X1), ">", atom(X2).
atom(X)                       --> var(X).


% A variable (or agent) might be a string [a-zA-Z].[0-9a-zA-Z_]*
var(A) --> space, !, var(A).
var(A) --> alpha(C), vartail(Cs), { atom_codes(A,[C|Cs]) }, !.
vartail([C|Cs]) --> alnum(C), vartail(Cs).
vartail([]) --> [].


agent(I) --> num(I).
agent(I) --> var(I), {agent(I)}.


num(N) --> digit(A), numtail(As), !, { atom_codes(N,[A|As]) }.
numtail([A|As]) --> digit(A), numtail(As).
numtail([]) --> [].


gnm(X1,Y) --> space, !, gnm(X1,Y).
gnm(X1,Y) --> "<->", !, fml(X2), gnm(eq(X1,X2),Y).
gnm(X1,Y) --> "<=>", !, fml(X2), gnm(eq(X1,X2),Y).
gnm(X1,Y) -->  "->", !, fml(X2), gnm(imp(X1,X2),Y).
gnm(X1,Y) -->  "=>", !, fml(X2), gnm(imp(X1,X2),Y).
gnm(X1,Y) -->  "v",  !, fml(X2), gnm(or(X1,X2),Y).
gnm(X1,Y) -->  "|",  !, fml(X2), gnm(or(X1,X2),Y).
gnm(X1,Y) -->  "&",  !, fml(X2), gnm(and(X1,X2),Y).


% It ain't much but it's operator precedence from the bottom up,
% defaulting to () >> <-> >> -> >> v >> &. Brackets () are denoted
% by bra(), to be stripped after parsing. There must be a better way.
gnm(imp(A,eq(B,C)),eq(imp(A,B),C)) --> [], !.
gnm(or(A,eq(B,C)),eq(or(A,B),C)) --> [], !.
gnm(or(A,imp(B,C)),imp(or(A,B),C)) --> [], !.
```

```
gnm(and(A,eq(B,C)),eq(and(A,B),C)) --> [], !.
gnm(and(A,imp(B,C)),imp(and(A,B),C)) --> [], !.
gnm(and(A,or(B,C)),or(and(A,B),C)) --> [], !.
gnm(A,A) --> [].


% like SWI-prolog builtins code_type/2
space --> [C],
        { (C==32);          % ' '
          (C>8, 14>C) }.    % \b\t\n\v\f\r

digit(C) --> [C],
{ C>47, 57>C }.     % 0..9

alpha(C) --> [C],
{ (C>64, 91>C);     % A..Z
  (C>96, 123>C) }.  % a..z

alnum(C) --> [C],
        { (C>47, 57>C);    % 0..9
          (C>64, 91>C);    % A..Z
          (C==95);         % _
          (C>96, 123>C) }. % a..z
```

A.1.2 PAL omzetten naar EL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pal2el.pl - translates updates in formula's to pure
%              epistemic logic.
%
% Mathijs de Boer - KI RuG - 2004

:- module(pal2el,[pal2el/2]).

% Dummy variables are done.
pal2el(update(_,X),_) :- var(X), !.
pal2el(X,X) :- var(X), !.

% On encounter of a new update [u]fml, postpone u's translation to uel
% until after the analysis of [uel]fml and fill in the (many) uel's all
% at once in the end. What is the real space complexity of PAL reduction?
```

```
pal2el(update(U,Fml),ElFml) :-
        \+var(U), !,
        pal2el(update(Uel,Fml),ElFml),
        pal2el(U,Uel).

pal2el(update(U,Fml),ElFml) :-
        ( Fml = not(A)   -> ElFml = imp(U,not(Ael));
          Fml = bra(A)   -> ElFml = imp(U,Ael);
          Fml = box(I,A) -> ElFml = imp(U,box(I,Ael));
          % q -> ~(q -> [1](q -> ~[q]x)) <-> q -> <1>(q & [q]x)
          Fml = dia(I,A) -> ElFml = imp(U,dia(I,and(U,Ael)));
          Fml = and(A,B) -> ElFml = and(Ael,Bel);
          % q -> (<q>a v <q>b) <-> [q]a v [q]b (inefficient!)
          Fml = or(A,B)  -> ElFml = or(Ael,Bel);
          Fml = imp(A,B) -> ElFml = imp(Ael,Bel);
          Fml = eq(A,B)  -> ElFml = eq(Ael,Bel);
          atomic(Fml)    -> ElFml = imp(U,Fml);
          pal2el(Fml,A)  -> ElFml = Ael % Fml = update(_,_)
        ), !,
        pal2el(update(U,A),Ael),
        pal2el(update(U,B),Bel). % A, B might be dummies.

pal2el(Fml,ElFml) :-
        ( Fml =.. [O,A]   -> ElFml =.. [O,Ael];
          Fml =.. [O,A,B] -> ElFml =.. [O,Ael,Bel];
          ElFml = Fml % must be atomic in wffs
        ), !,
        pal2el(A,Ael),
        pal2el(B,Bel). % A, B might be dummies.
```

A.1.3 ELtap met universele variabelen

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% UELtap.pl
%
%               Prove Epistemic Logic Formulas with Free and
%                   Universal Variable Labeled Tableaus
%
% prove(+"Fml").
%
```

```
% Mathijs de Boer - KI RuG - 2004

:- use_module(palparse,[parse/2]).
:- use_module(test,[test/1]).
:- use_module(pal2el,[pal2el/2]).
:- use_module(library('lists')).


prove(FmlString) :-
        parse(FmlString,FmlTerm),
        pal2el(not(FmlTerm),Fml),
        time(prove([[]:Fml],[])), !. % SWI prolog 'time/1'


% first all non-forking rules on a branch are analysed
prove([LFml|Fs],Branch) :-
        (rule(LFml,_tpe,NLF), !, prove([NLF|Fs],Branch));
        (rule(LFml,conj,A,B), !, prove([A,B|Fs],Branch)).

prove([L:Fml|Fs],Branch) :-
        (Fml = not(Neg); Neg = not(Fml)),
       (memberunify(L:Neg,Branch); prove(Fs,[L:Fml|Branch])), !.


% disjunction and variable introduction in one operation
prove([],Branch) :-
        select(Label:Fml,Branch,RestB),
        rule(_:Fml,disj,_:A,_:B),
        expand(Label,Inst,M,_),
        (Label = _:L; Label = L), !,
        prove([M:A],[(Inst:L):Fml|RestB]),
        prove([M:B],[(Inst:L):Fml|RestB]).


expand(([[X|Xs]|Xss]:['*'+I|Ls]),[[X|Xs]|Yss],[X:I|Ms],G) :-
        var(X), !,
        expand((Xss:Ls),Yss,Ms,G).

expand(([Xs|Xss]:['*'+I|Ls]),[[X|Xs]|Yss],[X:I|Ms],_) :- !,
        freeze(X,\+member(X,Xs)),
        expand((Xss:Ls),Yss,Ms,true).
```

```
expand((Xss:[C|Ls]),Yss,[C|Ms],G) :- !,
        expand((Xss:Ls),Yss,Ms,G).

expand(([]:[]),[],[],G) :- nonvar(G).


expand(['*'+I|Ls],[[X]|Xss],[X:I|Ms],_) :- !,
        expand(Ls,Xss,Ms,_).

expand([L|Ls],Xss,[L|Ms],_) :- !,
        expand(Ls,Xss,Ms,_).

expand([],[],[],_).


memberunify(L:A,Ls) :-
        member(K:A,Ls),
        red(O,L,[]),
        red(O,K,[]), !.


red(O,[_:I|L],[X:I|R]) :- !,
        red(O,L,[X:I|R]).

red(O,L,[O:_|R]) :-
        red(O,L,R).

red(O,L,['*'+I|R]) :- !,
        red(O,L,[_:I|R]).

red(O,[X|L],R) :- !,
        red(O,L,[X|R]).

red(O,[],O).


rule(L:and(A,B),            conj, L:A,L:B).
rule(L:not(imp(A,B)),       conj, L:A,L:not(B)).
rule(L:not(or(A,B)),        conj, L:not(A),L:not(B)).
rule(L:or(A,B),             disj, L:A,L:B).
```

```
rule(L:imp(A,B),            disj, L:not(A),L:B).
rule(L:not(and(A,B)),       disj, L:not(A),L:not(B)).
rule(L:eq(A,B),          disj, L:and(A,B),L:and(not(A),not(B))).
rule(L:not(eq(A,B)),      disj, L:and(A,not(B)),L:and(not(A),B)).
rule(L:not(bra(F)),         doub, L:not(F)).
rule(L:bra(F),              doub, L:F).
rule(L:not(not(F)),         doub, L:F).

% Symbol '*' a universal variables. Notation '*'+L to
% make certain no unification of variables with '*' can
% take place.
rule([_:I|L]:box(I,F),      know, ['*'+I|L]:F).
rule([_:I|L]:not(dia(I,F)), know, ['*'+I|L]:not(F)).
rule([_+I|L]:box(I,F),      know, ['*'+I|L]:F).
rule([_+I|L]:not(dia(I,F)), know, ['*'+I|L]:not(F)).
rule(L:box(I,F),            know, ['*'+I|L]:F).
rule(L:not(dia(I,F)),       know, ['*'+I|L]:not(F)).
rule([_:I|L]:dia(I,F),      poss, [F:I|L]:F).
rule([_:I|L]:not(box(I,F)), poss, [F:I|L]:not(F)).
rule([_+I|L]:dia(I,F),      poss, [F:I|L]:F).
rule([_+I|L]:not(box(I,F)), poss, [not(F):I|L]:not(F)).
rule(L:dia(I,F),            poss, [F:I|L]:F).
rule(L:not(box(I,F)),       poss, [not(F):I|L]:not(F)).
```

A.1.4 Testformules

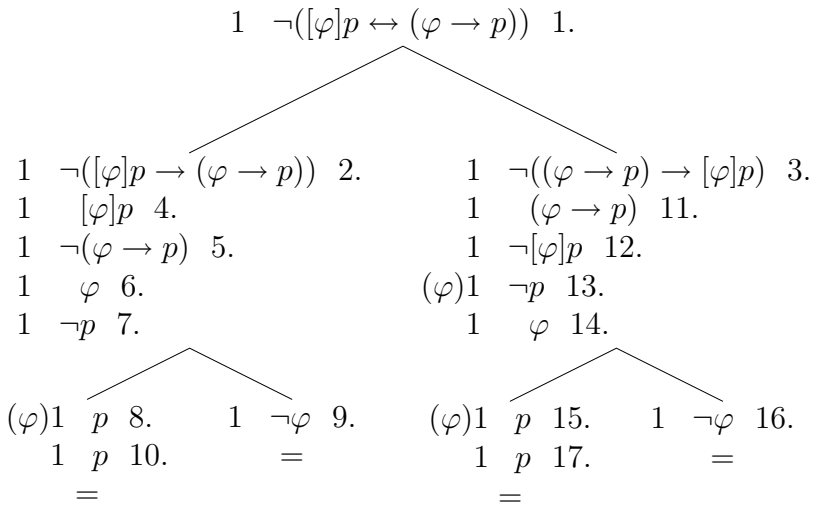
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% test.pl - some testformules for ELtap
%
% Mathijs de Boer - KI RuG - 2004

:- module(test,[test/1]).

% some s5n-test-theorems
test(1) :- prove("[1]<1><3>[3][2][1]a -> a").
test(2) :- prove("~([1](a | ~b) & <1>~a & <1>(~a & b))").
test(3) :- prove("~<1>(~a & a)").
test(4) :- prove("[1]<1><3>[3][2][3]a -> <1><3>a").
test(5) :- prove("~([1](a|b) & <1>~a & <1>[1]<3>~b &
                    (<1>[2]<1>(~a&~b) | <1>(~a&~b) | [1]a))").
```

```
test(6) :- prove("~([1]<1><3>[3][2][3]a & [1]<3><2>[3][2]~a)").
test(7) :- prove("~([1]<1><3>[3][2][3]a &
                    [1]<3>[3][2](~a|x) & [1]~x)").
test(8) :- prove("[1](a -> b) -> ([1]a -> [1]b)"). % distribution
test(9) :- prove("[1]a -> a"). % truth
test(10) :- prove("[1]a -> [1][1]a"). % positive introspection
test(11) :- prove("<1>a -> [1]<1>a"). % negative introspection
test(12) :- prove("<1>([1]p | <2>[2][1]q) -> [1](~q -> p)").
test(13) :- prove("((p & ~[1]p) -> ~p) |
                   ((p & ~[1]p) -> ~((p & ~[1]p) ->
                  ~[1]((p & ~[1]p) -> p)))"). % [p & ~[1]p]~(p & ~[1]p)
test(14) :- prove("([1](a|b) & (c -> [1]~b) & ([1]a -> d)) -> (c -> d)").
test(15) :- prove("([1](g <-> [1]p) & <1>g) -> (g & p)"). % Anselm
test(16) :- prove("([1](g <-> [1]p) & <1>~g) -> ~g"). % Findlay
test(17) :- prove("([1]p & [1]((a & p) -> b)) -> [1](a -> b)").

% and some non-s5n-test-theorems
test(18) :- \+prove("([1]p & [1]((a & p) -> b)) -> (<1>~b & [1](a -> b))").
test(19) :- \+prove("[1]<1><3>[3][2][3]a -> a").
test(20) :- \+prove("~([1](a | ~b) & <1>~a & <1>b)").
test(21) :- \+prove("[1]<1><3>[3][2]<3>a -> [1][3]a").
test(22) :- \+prove("~([1](a | b) & <1>~a &
                       (<1>(~a&~b) | <1>a))").
test(23) :- \+prove("~([1]<1><3>[3][2][3]a &
                       [1]<3>[3][2](~a|x) & ~x)").
test(24) :- \+prove("((p & ~[1]p) -> p) &
                     ((p & ~[1]p) -> ~((p & ~[1]p) ->
                    [1]((p & ~[1]p) -> p)))"). % [p & ~[1]p](p & ~[1]p)
test(25) :- \+prove("~([1]<3>(<1>a | [4]b) &
       [3]([1][3]~a & <3>[4][3]<6>~b))").

% some PAL-test-theorems
test(26) :- prove("[a & <1>~a]~(a & <1>~a)").
test(27) :- prove("[a v b][1]x <-> ((a v b) -> [1][a v b]x)").
test(28) :- prove("[a][b]c <-> [a & [a]b]c").
test(29) :- prove("~<a & <1>~a>(a & <1>~a)").
test(30) :- prove("[(a|b) & (~a -> [2]~a) & (~b -> [1]~b)][<2>~b][<1>~a](a & ~a)").
test(31) :- prove("[a -> [1]x][b|<2>~q][3]<1>~(c -> d) <->
   [(a -> [1]x) & [a -> [1]x](b|<2>~q)][3]<1>~(c -> d)").
test(32) :- prove("[(a v b v c) &
    ( ~a -> (box2 ~a & box3 ~a)) &
```

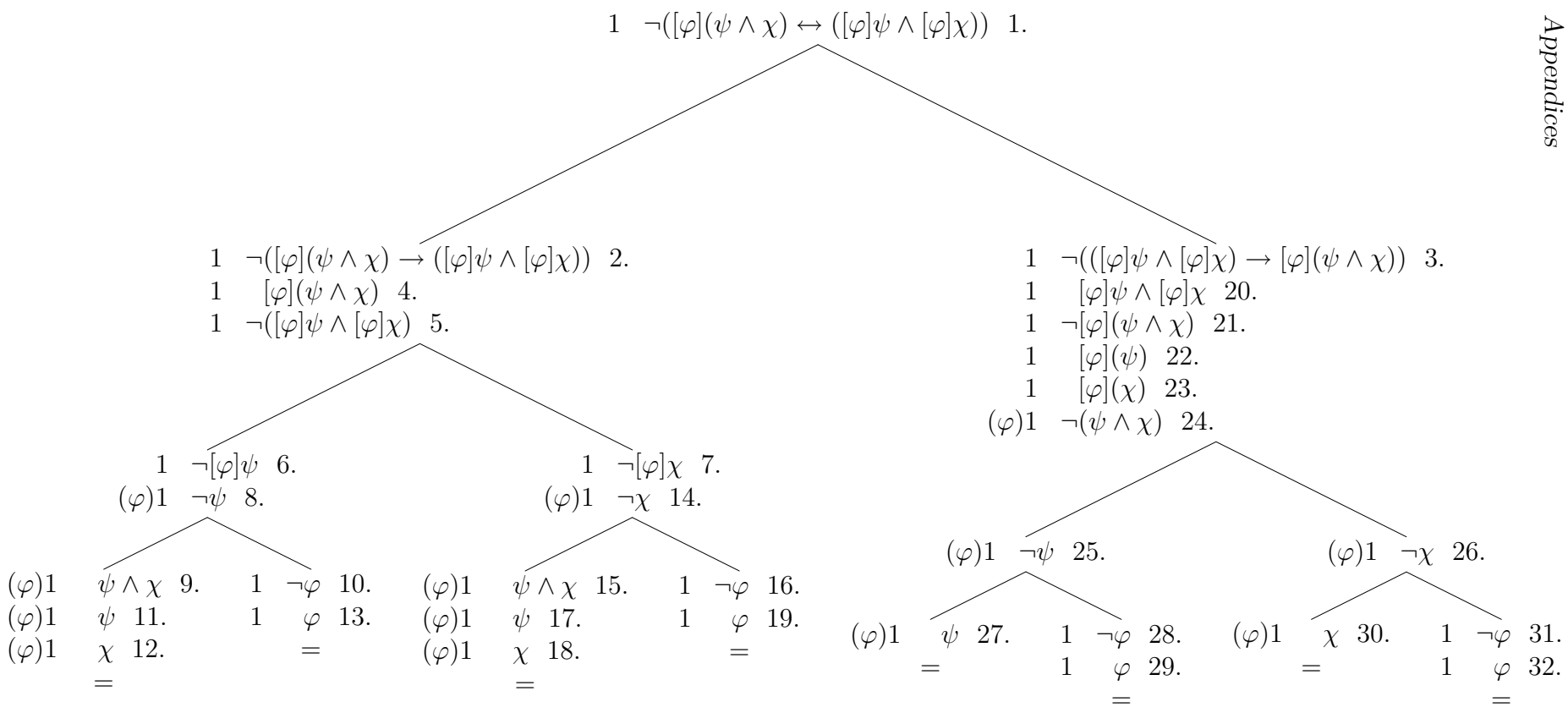
```
    ( ~b -> (box1 ~b & box3 ~b)) &
    ( ~c -> (box1 ~c & box2 ~c))
   ][~box1 a][~box2 b]box3 c"). % the three wise men puzzle
test(33) :- prove("[(a|b|c) & (~b -> [1][3]~b) & (~c -> [1][2]~c)
   ][<1>~a][<2>~b][3]c"). % S5 three wise men puzzle

%
% Some statistics of UELtap.pl with these formula's and SWI prolog
% Used hardware and OS is not very relevant but:
% FreeBSD 6.1-SECURITY #0: Mon Aug 28 05:21:08 UTC 2006
% CPU: Intel Celeron (633.81-MHz 686-class CPU)
% real memory  = 334430208 (318 MB)
%
%
% test(N)
%
% 62 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 1;
% 90 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 2 ;
% 24 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 3 ;
% 94 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 4 ;
% 272 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 5 ;
% 115 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 6 ;
% 153 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 7 ;
% 148 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 8 ;
% 23 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 9 ;
% 28 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 10 ;
% 30 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 11 ;
% 99 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 12 ;
% 222 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 13 ;
% 193 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 14 ;
% 312 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 15 ;
% 225 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 16 ;
% 122 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 17 ;
% 468 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 18 ;
% 92 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 19 ;
% 422 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 20 ;
% 71 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 21 ;
% 476 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 22 ;
% 718 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 23 ;
% 1,999 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 24 ;
% 950 inferences, 0.01 CPU in 0.00 seconds (354% CPU, 121600 Lips) N = 25 ;
```

```
% 292 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 26 ;
% 1,238 inferences, 0.01 CPU in 0.01 seconds (147% CPU, 158464 Lips) N = 27 ;
% 346 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 28 ;
% 294 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips) N = 29 ;
% test(30) took to long to mesure
% test(31) took to long to mesure
% 41,776 inferences, 0.08 CPU in 0.08 seconds (97% CPU, 534733 Lips) N = 32
```

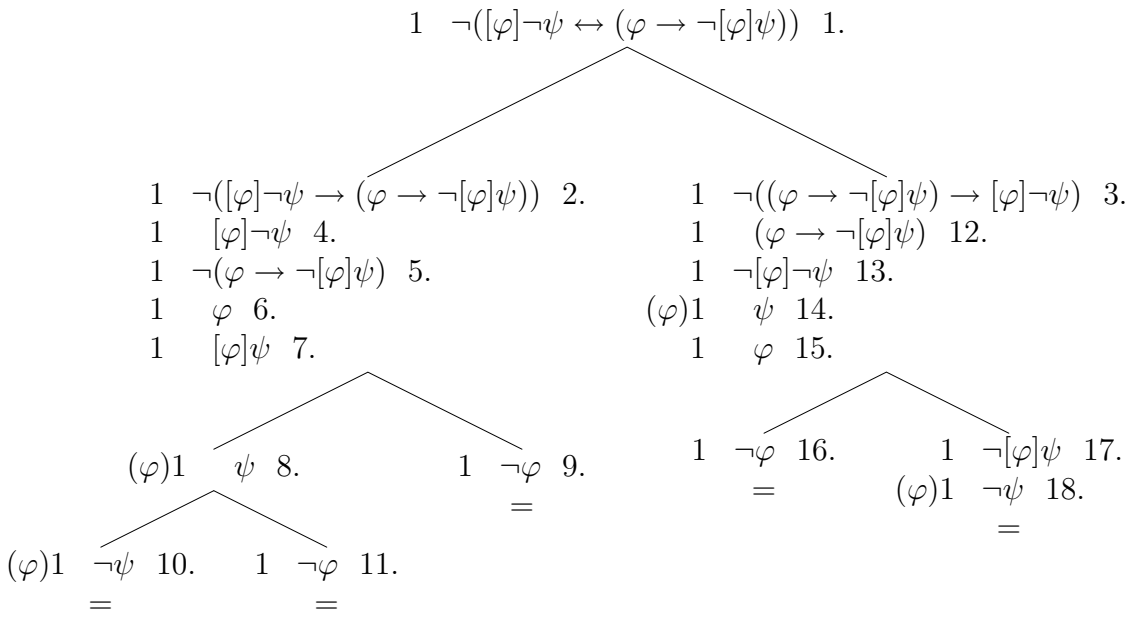
A.2 Tableaus voor reductieaxioma's



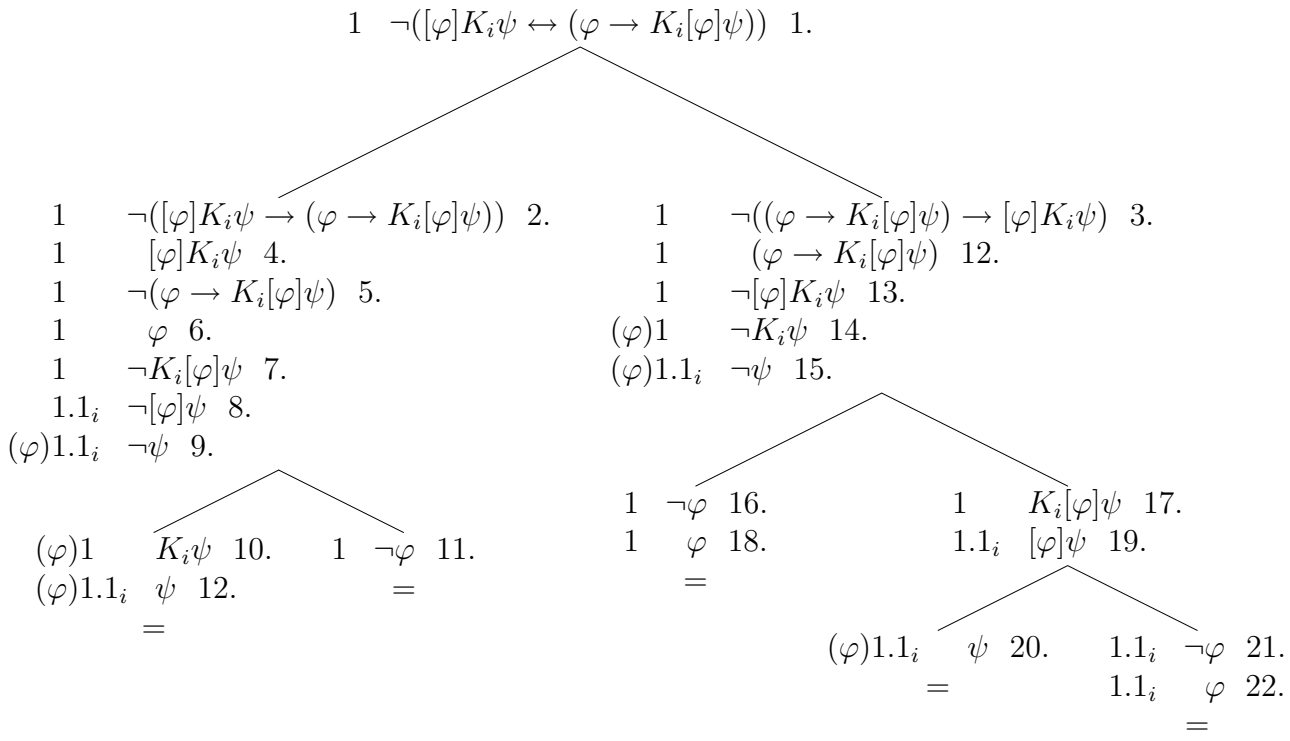
Figuur A.1: tableau voor het axioma 'atomen'. Let op dat in het algemeen niet geldt $\vdash [\varphi]\psi \leftrightarrow (\varphi \rightarrow \psi)$.



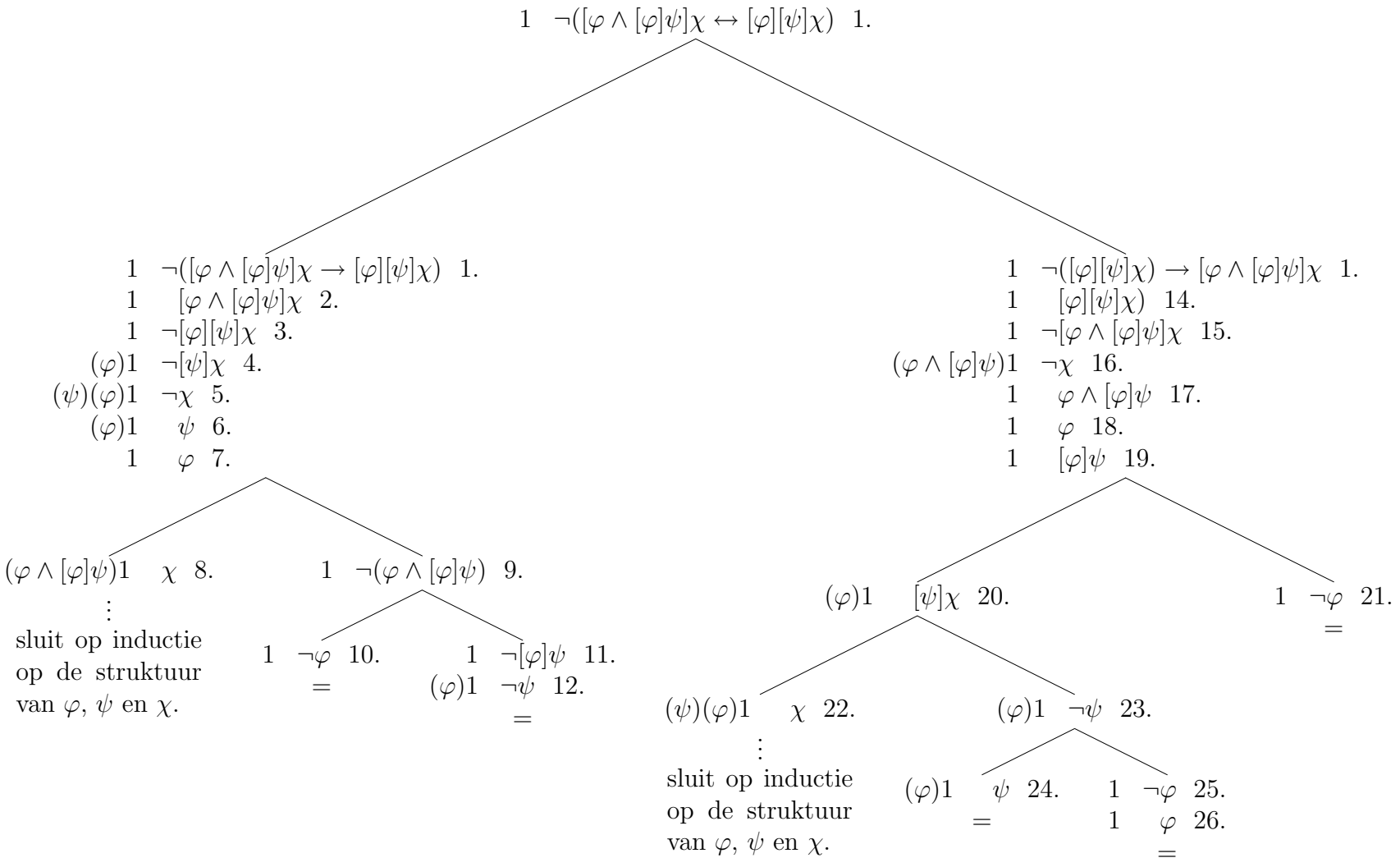
Figuur A.2: tableau voor het axioma ‘distributie’.



Figuur A.3: tableau voor het axioma 'partiële functie'.



Figuur A.4: tableau voor het axioma 'kennis-update'.



Figuur A.5: tableau voor het axioma ‘gecombineerde updates’.

Bibliografie

[1] B. Beckert and R. Goré. Free variable tableaux for propositional modal logics. In *Proc. International Conference on Theorem Proving with Analytic Tableaux and Related Methods, Pont-a-Mousson, France*, volume LNCS 1227, pages 91–106. Springer-Verlag, 1997.

[2] B. Beckert and J. Posegga. leanTAP: Lean, tableau-based deduction. *Journal of Automated reasoning*, 15(3):339–358, 1995.

[3] M. D'Agostino and M. Mondadori. The taming of the cut. *Journal of Logic and Computation*, 4(3):285–319, 1994.

[4] M. Fitting. leanTAP revisited. *Journal of Logic and Computation*, 8(1):33–47, 1998.

[5] M. Fitting and R.L. Mendelsohn. *First-Order Modal Logic*, volume 277 of *Synthese Library*. Kluwer Academic Publishers, 1999.

[6] J. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, ILLC Dissertation Series, Amsterdam, 1999.

[7] J. Gerbrandy and W. Groeneveld. Reasoning about information change. *Journal of Logic, Language and Information*, 6(2):147–169, 1997.

[8] R. Goré. Tableau methods for modal and temporal logics. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Kluwer Academic Publishers, 1999.

[9] Jean Goubault and Joachim Posegga. BDDs and automated deduction. In *8th International Symposium on Methodologies for Intelligent Systems (ISMIS)*, Lecture Notes in Artificial Intelligence, pages 541–550. Springer-Verlag, 1994.

[10] G. Governatori. Labelled tableaux for multi-modal logics. In P. Baumgartner, R. Hähnle, and J. Possega, editors, *Theorem Proving with Analytic tableaux and Related Methodes*. Springer-Verlag, Berlin, 1995.

[11] G. Governatori. A duplication and loop checking free proof system for S4. In *Short Papers: TABLEAUX'96*, 1996.

[12] J. Hintikka. *Knowledge and belief, An Introduction of the Logic of the Two nations*. Cornell University Press, 1962.

[13] C. Lutz. Complexity and succinctness of public announcement logic. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 137–143. ACM Press, 2005.

[14] J. McCarthy. Formalisation of two puzzles involving knowledge. In V. Lifschitz, editor, *Formalizing Common Sense: Papers by John McCarthy*, Ablex series in artificial intelligence. Northwood, New Jersey: Ablex Publishing Corporation, 1990.

[15] J.A. Plaza. Logics of public communication. In M.S. Pfeifer, M. Hadzikadic, and Z.W. Ras, editors, *Proceedings of the* 4^{th} *International Symposium on Methodologies for Intelligent Systems*, pages 201–216. Oak Ridge National Laboratory, 1989.

[16] J. Possega and P.H. Schmitt. Implementing semantic tableaux. Technical report, Universität Karlsruhe, Fakultät für Informatik, 1996.

[17] R.M. Smullyan. *First-Order Logic*. Courier Dover Publications, 1995 (1968).

[18] E. Spaan. *Complexity of Modal Logics*. PhD thesis, Universiteit van Amsterdam, 1993.

[19] L. Sterling and E. Shapiro. *The Art of Prolog*. MIT press, second edition, 1994.

[20] J.F.A.K. van Benthem. One is a lonely number: on the logic of communication. Technical Report PP–2002–27, ILLC, Amsterdam, 2002. to appear in P. Köpke, editor, *Colloquium Logicum*, Münster, AMS Publications, Providence.

[21] J.F.A.K. van Benthem. Open problems in logical dynamics. In M. Zakharysashev D. Gabbay, S. Goncharov, editor, *Mathematical Problems from Applied Logic I*, International Mathematical Series , Vol. 4, pages 137–192. Springer, New York & Novosibirsk, 2006.

[22] J.F.A.K. van Benthem, H.P. van Ditmarsch, J. Ketting, and W.P.M. Meyer-Viol. *Logica voor Informatici*. Addison-Wesley, 1991.

[23] W. van der Hoek and J.-J.Ch. Meyer. *Epistemic Logic for AI and Computer Science*. Number 41 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1995.

[24] H.P. van Ditmarsch and R. Dyckhoff. Sequent calculi for logics with common knowledge. In A. Voronkov, editor, *Eighth Workshop on Automated Reasoning*, pages 36–37, 2001.

[25] H.P. van Ditmarsch and B.P. Kooi. The secret of my success. *Synthese*, 151(2):201–232, 2006.

[26] H.P. van Ditmarsch, J. Ruan, and R. Verbrugge. Model checking sum and product. In *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence (AI 2005)*, pages 790–795. Springer Verlag, 2005. LNAI 3809.

[27] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. *Dynamic epistemic logic*. Manuscript, 2006.

[28] J. van Eijck and G. Cepparello. Dynamic modal predicate logic. In M. Kanazawa and C.J. Piñon, editors, *Dynamics, Polarity and Quantification*, pages 251–276. CSLI, Stanford, 1994.