Logik und Zeit – Skript zum Seminar

Christian Wurm
cwurm@phil.hhu.de

January 11, 2024

Contents

1 Literaturempfehlungen

Allgemein gibt es, meines Wissens, keinen Überblick über alle temporalen Logiken die wir hier besprechen. Die Materie bis inkl. LTL scheint aber sehr ausführlich behandelt zu werden in dem Buch "Automatentheorie und Logik" von Martin Hofmann & Martin Lange (https://epdf.tips/automatentheorie-und-logik-examenpress.html). Ansonsten orientiere ich mich gerne an den Übersichtsartikeln von Moshe Vardi – der Name steht für Qualität.

2 Propositionale Logik

2.1 Die Formelsprache $Form(Var)$

- Var – eine unendliche Menge von propsitionalen Variablen.
- Die Konnektoren $\neg$, $\wedge$
- Die Menge der Formeln $Form(Var)$ ist definiert durch:
  1. Falls $p \in Var$, dann $p \in Form(Var)$;
  2. falls $\alpha, \beta \in Form(Var)$, dann $(\alpha \wedge \beta) \in Form(Var)$;
  3. falls $\alpha \in Form(Var)$, dann $(\neg\alpha) \in Form(Var)$;

Wir definieren uns zwei Abkürzungen:

1. $\alpha \vee \beta \equiv \neg(\neg\alpha \wedge \neg\beta)$
2. $\alpha \rightarrow \beta \equiv (\neg\alpha) \vee \beta$

2.2 Die Semantik

Ein Modell für Form(Var) hat die Form $M \subseteq Var$, wobei gilt:

- $M \models p$, genau dann wenn $p \in M$.
- $M \models \alpha \wedge \beta$, gdw. $M \models \alpha$ und $M \models \beta$
- $M \models \neg\alpha$, gdw. $M \not\models \alpha$
- $M \models \alpha \vee \beta$, gdw. mindestens eines gilt: $M \models \alpha$ oder $M \models \beta$
- $M \models \alpha \rightarrow \beta$ gdw. mindestens eines gilt: $M \not\models \alpha$ oder $M \models \beta$

$M \models p \wedge q$ gdw. $p, q \in M$

$M \models \neg(p \wedge q)$
gdw. $(M \not\models p \wedge q)$
gdw. $(M \not\models p$ oder $M \not\models q)$
gdw. $p \notin M$ oder $q \notin M$

$M \models p \vee q$
gdw. $M \models \neg(\neg p \wedge \neg q)$ (definition von $\vee$)
gdw. $M \not\models \neg p \wedge \neg q$
gdw. $M \not\models \neg p$ oder $M \not\models \neg q$
gdw. $M \models p$ oder $M \models q$

$M \models p \rightarrow q$
gdw. $M \models \neg p \vee q$
gdw. $M \models \neg p$ oder $M \models q$
Angenommen, $M \models p$, also $M \not\models \neg p$. Also muss $M \models q$ gelten.

Zur Erinnerung (und in aller Kürze) die Faustregeln für Konnektoren. Bitte beachten: der Unterschied zwischen *wenn* (hinreichende Bedingung) und *gdw.* (genau dann wenn, hinreichende und notwendige Bedingung)!

- $\alpha \wedge \beta$ ist wahr, gdw. beides wahr ist
- $\alpha \wedge \beta$ ist falsch, gdw. mindestens eins der beiden falsch ist
- $\alpha \vee \beta$ ist wahr, gdw. mindestens eins der beiden wahr ist
- $\alpha \vee \beta$ ist falsch, gdw. beides falsch ist
- $\alpha \to \beta$ ist wahr, wenn α falsch ist
- $\alpha \to \beta$ ist wahr, wenn β wahr ist
- $\alpha \to \beta$ ist falsch gdw. α wahr und β falsch ist.

Hiermit kann man sich in den folgenden Aufgaben oft leichter selbst helfen.

Wir schreiben $\alpha \models \beta$, gdw. für alle M gilt: falls $M \models \alpha$, dann $M \models \beta$.

Folgendes sind die Fragen nach **Entscheidbarkeit** einer Logik. Für unsere einfache Aussagenlogik ist die Antwort natürlich immer positiv, aber für unsere Logiken werden uns diese Fragen noch oft beschäftigen!

Frage 1: Gegeben ein Modell M, eine Formel α, gilt $M \models \alpha$? – Können wir diese Frage immer entscheiden?

Frage 2: Wir schreiben $\alpha \models \beta$, gdw. für alle M gilt: falls $M \models \alpha$, dann $M \models \beta$. Gilt $\alpha \models \beta$? – Können wir diese Frage immer entscheiden?

Frage 3: Gegeben α, gibt es ein M so dass $M \models \alpha$? Gibt es M so dass $M \not\models \alpha$? – Können wir diese Frage immer entscheiden?

Übung

Aufgabe 1

Warum sind Fragen 2,3 äquivalent?

Antwort $2 \Rightarrow 3$ Nimm an, 2 ist entscheidbar. Wir möchten wissen, ob α ein Modell hat. α hat ein Modell gdw. $\neg\alpha$ in mindestens einem Modell falsch ist. Also: gdw. $\neg\alpha$ kein Theorem ist. Nimm nun eine beliebige Tautologie $p \vee \neg p$. Falls $\neg\alpha$ ein Theorem ist, gilt:

$$p \vee \neg p \models \neg\alpha$$

Falls $\neg\alpha$ kein Theorem ist, gilt

$$p \vee \neg p \not\models \neg\alpha$$

Sprich: wenn wir wissen ob $p \vee \neg p \models \neg\alpha$ wahr bzw. falsch ist, wissen wir ob $\neg\alpha$ ein Theorem ist, dann wissen wir ob α ein Modell hat.

$3 \Rightarrow 2$ $\alpha \models \beta$ gdw. $\alpha \rightarrow \beta$ ein Theorem ist. Wenn wir also entscheiden können ob es ein Modell M gibt, dass $\alpha \rightarrow \beta$ *nicht* erfüllt, dann wissen wir ob es ein Modell gibt, dass α erfüllt aber nicht β (dasselbe M nämlich)

Aufgabe 2

Wichtig an diesen einfachen Definition ist: wir haben keine Wahrheitssemantik, sondern eine Menge von Propositionen als Modell. Das ist erstmal äquivalent:

- Sei $M \subseteq \mathit{Var}$ ein Modell im obigen Sinne
- $\sigma : \mathit{Var} \rightarrow \{0, 1\}$ eine Wahrheitsfunktion
- Dann ist M äquivalent zu σ genau dann wenn $p \in M \iff \sigma(p) = 1$

Es macht also *eigentlich* keinen Unterschied welche Auffassung wir in dieser Semantik haben. Aber: unsere Semantik ist wesentlich praktischer und üblicher wenn es zu modaler Logik kommt. Das ist wichtig!

Hier nun einige Übungen zur Konstruktion von Modellen: Konstruieren Sie ein Modell M, so dass

1. $M \models p \to (q \vee r)$, $M \not\models q \to r$
2. $M \models p \to (q \to r)$, $M \not\models r$
3. $M \models \neg(p \to q)$, $M \not\models q$
4. $M \models \neg((p \wedge q) \vee r)$, $M \not\models \neg r$

   Also: 1. $r \in M$. Dann gilt auch: $M \models (p \wedge q) \vee r$, also $M \not\models \neg((p \wedge q) \vee r)$. Es gibt kein solches Modell!

Aufgabe 3

Wir schreiben $\alpha \models \beta$, falls gilt: $M \models \alpha$ impliziert $M \models \beta$.

Entscheiden/begründen Sie, ob folgende Aussagen wahr oder falsch sind. Hierbei gibt es zwei Strategien: entweder, wir sind der Meinung die Aussage ist falsch. Dann versuchen wir ein Modell zu konstruiere wie oben. Oder wir sind der Meinung, die Aussage ist wahr. Dann versuchen wir, das mit einem Argument zu belegen.

Insbesondere diese Form der halbformellen Argumentation ist schwierig und muss geübt werden; wir werden das noch viel brauchen in diesem Seminar!

1. $p \vee q \models (r \to p) \vee (r \to q)$
2. $p \vee q \models (r \to p) \vee (q \to r)$
3. $\neg((p \wedge q) \to r) \models p \to q$
4. $((p \vee r) \to q) \wedge (\neg r) \models p \to q$

3 Modale Logik

Die Temporallogiken, die wir hier betrachten, sind modale Logiken. Daher gibt es eine kleine Einführung in die modale Logik. Als Literatur empfehle ich: Goldblatt, Logics of Time and Computation. Es gibt auch Kracht, Tools and Techniques in Modal Logic, aber das ist keine leichte Kost.

Zunächst folgendes: wir werden hier nur modale Aussagenlogik betrachten; Prädikatenlogik spielt also erstmal keine Rolle (und hat auch eine ganze Reihe von Problemen). Die Syntax von unserer Modallogik ist also einfache Erweiterung der Aussagenlogik:

1. Falls $p \in Var$, dann $p \in Form(Var)$;
2. falls $\alpha, \beta \in Form(Var)$, dann $\alpha \wedge \beta \in Form(Var), \alpha \vee \beta \in Form(Var)$;
3. falls $\alpha \in Form(Var)$, dann $\neg\alpha \in Form(Var)$;
4. falls $\alpha \in Form(Var)$, dann $\Box\alpha, \Diamond\alpha \in Form(Var)$.

Wir haben also die Box $\Box$ und den Diamanten $\Diamond$ als unäre Konnektoren, oder auch sog. Modalitäten. In der philosophischen Tradition wird $\Box\alpha$ mit "es ist notwendig dass α" assoziiert, $\Diamond\alpha$ mit "es ist möglich dass α".

Es gibt mehrere Semantiken der modalen Logik; die bekannteste, einfachste und älteste die Semantik der Welten, die eine einfache Generalisierung ist der Semantik der Aussagenlogik, wie wir sie oben besprochen haben. Wir nehmen an, dass es nicht nur eine Welt gibt, sonden

- eine Menge von Welten W, und
- eine Relation $R \subseteq W \times W$, die die Zugänglichkeit zwischen Welten beschreibt.

Wir können in dieser Semantik – mittels Modalitäten – zwischen Welten wandern, während klassische Konnektoren in der aktuellen Welt evaluaiert werden. Eine Struktur

$$(W, R)$$

heißt ein **Kripke-Rahmen** (nach ihrem Erfinder Saul Kripke). Ein Kripke-Rahmen ist also im Prinzip einfach nur ein gerichter Graph. Was noch fehlt ist folgendes:

- $val : W \to \wp(Var)$ ist eine Valuation, die uns sagt welche (atomaren) Propositionen in einer Welt wahr bzw. nicht wahr sind.

Eine Struktur

$$(W, R, val)$$

ist ein **Kripke Modell**. Wir definieren nun die Wahrheit in einem Kripke Modell. Zunächst ist diese Wahrheit im Bezug auf eine Welt $w \in W$ definiert, an der wir evaluieren. Wir nutzen wiederum die induktive Formelstruktur:

- $(W, R, val), w \models p$, genau dann wenn $p \in val(w)$.
- $(W, R, val), w \models \alpha \wedge \beta$, gdw. $(W, R, val), w \models \alpha$ und $(W, R, val), w \models \beta$
- $(W, R, val), w \models \alpha \vee \beta$, gdw. mindestens eines gilt: $(W, R, val), w \models \alpha$ oder $(W, R, val), w \models \beta$
- $(W, R, val), w \models \neg\alpha$, gdw. $(W, R, val), w \not\models \alpha$
- $(W, R, val), w \models \alpha \to \beta$ gdw. mindestens eines gilt: $(W, R, val), w \not\models \alpha$ oder $(W, R, val), w \models \beta$
- $(W, R, val), w \models \Box\alpha$ gdw. für alle $w' \in W$ gilt: falls $(w, w') \in R$, dann $(W, R, val), w' \models \alpha$
- $(W, R, val), w \models \Diamond\alpha$ gdw. es ein $w' \in W$ gibt, so dass 1. $(w, w') \in R$ und 2. $(W, R, val), w' \models \alpha$

Der nächste Begriff ist eine Abstraktion dieses Begriffs der Wahrheit in einer Welt (in einem Modell). Wir sagen α ist **valide** im Modell (W, R, val), falls es in jeder Welt wahr ist; formaler:

$$(W, R, val) \models \alpha \text{ gdw. } (W, R, val), w \models \alpha \text{ f.a. } w \in W$$

Dieser Begriff der Validität im Modell ist sehr wichtig für die modale Logik.

Übung

Wir nehmen 3 Formeln:

1. $(\Diamond p) \to q$
2. $\Box(p \to \Diamond q)$
3. $\Diamond(p \to \Box q)$

Unser Kripke Modell sieht wie folgt aus:

- $W = \{w_0, w_1, w_2\}$
- $R = \{(w_0, w_1), (w_1, w_0), (w_1, w_2), (w_2, w_0)\}$
- $val(w_0) = \{p\}$, $val(w_1) = \{q\}$, $val(w_2) = \{p, q\}$

Bitte zum arbeiten aufzeichnen, das ist keine Arbeit und lohnt sich. Wir prüfen ob die Formeln valide sind im Rahmen, also wahr in allen Welten. Wir schreiben hier $w_0 \models \alpha$ als Kurzform für$(W, R, val), w_0 \models \alpha$ etc.

1. Ist valide. Wir müssen hierfür durch die Welten gehen.

$w_0 \models (\Diamond p) \to q$. Wir haben nur $(w_0, w_1) \in R$; $w_1 \not\models p$, also $w_0 \not\models \Diamond p$, also $w_0 \models (\Diamond p) \to q$.

$w_1 \models (\Diamond p) \to q$. $w_1 \models q$, also $w_1 \models (\Diamond p) \to q$ (Prämissenbelastung).

$w_2 \models (\Diamond p) \to q$. Ebenso.

2. Ist nicht valide. Hierfür brauchen wir nur eine Welt, nämlich w_1, und müssen zeigen dass $w_1 \not\models \Box(p \to \Diamond q)$. Wir machen ein Widerspruchsargument: nimm an $w_1 \models \Box(p \to \Diamond q)$. Also:

- Da $(w_1, w_2) \in R$, folgt $w_2 \models p \to \Diamond q$.
- Da $w_2 \models p$, folgt $w_2 \models \Diamond q$.
- Da es nur w_0 gibt mit $(w_2, w_0) \in R$, gilt $w_0 \models q$.
- Aber: $q \notin val(w_0)$; also $w_0 \not\models q$.

Widerspruch, also war die Annahme falsch!

3. Ist valide. Wir gehen durch die Welten.

$w_0 \models \Diamond(p \to \Box q)$ Wir haben $(w_0, w_1) \in R$, wobei gilt: $w_1 \not\models p$, also $w_1 \models p \to \Box q$ (ex falso quodlibet), also $w_0 \models \Diamond(p \to \Box q)$.

$w_1 \models \Diamond(p \to \Box q)$ Wir haben $(w_1, w_0) \in R$. Wir zeigen dass $w_0 \models p \to \Box q$: es gibt nur einen Nachfolger (d.h. erreichbare Welt) von w_0, nämlich w_1, wobei $w_1 \models q$. Also: $w_0 \models \Box q$, also $w_0 \models p \to \Box q$ (Prämissenbelastung). Mit dem obigen folgt daraus dass $w_1 \models \Diamond(p \to \Box q)$.

$w_2 \models \Diamond(p \to \Box q)$ Wir haben $(w_2, w_0) \in R$, wobei $w_0 \models p \to \Box q$ (s.o.); daraus folgt die Aussage.

Übung

Unser neues Kripke Modell sieht nur leicht anders aus:

- $W = \{w_0, w_1, w_2\}$
- $R = \{(w_0, w_1), (w_1, w_0), (w_1, w_2), (w_2, w_0)\}$
- $val(w_0) = \emptyset$, $val(w_1) = \{q\}$, $val(w_2) = \{p, q\}$

Prüfen Sie, ob die folgenden 2 Formeln valide sind:

1. $(\Box p) \to (\Diamond q)$ - ist valide, denn der Rahmen hat keinen Endpunkt (jede Welt hat einen Nachfolger). Ist also unabhängig von val!
2. $\Box((\Diamond p) \to \neg q)$

3.1 Validität im Rahmen

Es gibt auch einen noch abstrakteren Begriff, nämlich den der Validität in einem Rahmen (W, R). Dieser Begriff abstrahiert von val, also der tatsächlichen Wahrheit von Formeln, und bezieht sich nur noch auf die Struktur des Rahmens (als Graphens).

$$(W, R) \models \alpha \text{ gdw. } (W, R, val) \models \alpha \text{ f.a. } val : W \to \wp(Var)$$

Mittels dieses Begriffs kann man sehr schön strukturelle Eigenschaften des Graphen mittels Formeln beschreiben. Hier einige Beispiele:

- $(W, R) \models \Diamond p \to \Box p$ gdw. $R : W \to W$ eine partielle Funktion ist, also jede Welt höchstens einen Nachfolger hat.
- $(W, R) \models \Box p \to \Diamond p$ gdw. jede Welt mindestens einen Nachfolger hat
- $(W, R) \models \Diamond\Diamond p \to \Diamond p$ gdw. R transitiv ist.

  Beweis: $\Leftarrow$ Sei R transitiv, und nimm $w \in W$, wobei $w \models \Diamond\Diamond p$. Es folgt: es gibt $w' : (w, w') \in R$ und $w' \models \Diamond p$. Es folgt wiederum: es gibt $w'' : (w', w'') \in R$ und $w'' \models p$.

  Aber: R ist transitiv, also: $(w, w') \in R$, $(w', w'') \in R$ impliziert $(w, w'') \in R$. Also: $w \models \Diamond p$. Also: $w \models \Diamond\Diamond p \to \Diamond p$. QED

  $\Rightarrow$ Kontraposition: Sei R intransitiv, also gibt es w, w', w'' so dass $(w, w') \in R$, $(w', w'') \in R$, aber $(w, w'') \notin R$. Wir nehmen die Valuation val mit $p \in val(w'')$, $p \notin val(w) \cup val(w')$. Dann gilt $w \models \Diamond\Diamond p$, aber $w \not\models \Diamond p$, also $w \not\models \Diamond\Diamond p \to \Diamond p$.
- $(W, R) \models p \to \Diamond p$ gdw. R reflexiv ist (also: f.a. $w \in W$, $(w, w) \in R$).

  Nimm an $w \models p$, dann ist $(w, w) \in R$, also $w \models \Diamond p$.
- $(W, R) \models \Box\Box p \to \Box p$ (äquivalent zu $\Diamond p \to \Diamond\Diamond p$)
- $\Box(\Box P \to p) \to \Box p$ (Löb Axiom) – es gibt keine unendliche Kette $w_1 R w_2 R w_3 ...$ – sog. Wohlfundiertheit. Geht nur irreflexiv!

4 Die großen Fragen der temporalen Logik

Logiken sind formale Sprachen, mit denen man jeweils über eine gewisse Klasse von Modellen reden kann. Unsere Modelle werden in erster Linie Formalisierungen des Zeitbegriffes darstellen. Wir werden dabei aber sehen dass wir den intuitiven Begriff der erfahrenen Zeit auf ganz verschiedene Art und Weise auffassen und entsprechend auch formalisieren kann. Eine zentrale Frage ist z.B.: ist die Zeit diskret (d.h. es gibt kleinste minimale Zeitsprünge) oder kontinuierlich (d.h. zwischen zwei beliebigen Zeitpunkten gibt es immer einen dritten)? Davon unabhängig ist die Frage: ist die Zeit wirklich linear? Was ist in diesem Fall mit dem erfahrenen Nicht-Determinismus der Zukunft? Man kann das auf zwei Arten lösen, entweder indem man sagt: Zeit ist linear, aber unser Unwissen in Bezug auf die Zukunft besteht darin, dass wir eine *Menge* von Zeitmodellen haben, von denen wir nicht wissen welches das wahre ist. Oder man sagt: die Zeit ist ein **Baum** der sich ständig verzweigt, und jede Verzweigung stellt eben eine Auswahl zwischen verschiedenen Zukünften da. Es gibt noch exotischere Zeitmodelle (z.B. zirkuläre Zeit), das werden wir aber nicht verfolgen. Eine weitere Frage ist: sollen wir Wahrheit von Propositionen immer nur auf Zeitpunkt feststellen, oder auf Intervalle? Gibt es überhaupt so etwas wie Punkte in der Zeit? Das interessante an diesen Fragen ist: wir bleiben nicht bei philosophischer Spekulation (die auch interessant ist), sondern jeder mögliche Weg hat ganz handfeste mathematische Konsequenzen, die auch für Anwendungen der modernen Informatik sehr wichtig sind. So kann es für manche Zwecke besser sein, über Zeit als *einen* unendlichen Baum zu denken, und für manch anderen besser, sich Zeit als eine Menge von endlichen Ketten vorzustellen.

Soweit die Modelle; in der Logik gehen wir nochmal einen Schritt zurück und entwickeln Sprachen, mit denen wir über diese Modelle sprechen können. Das bringt uns auf die großen Fragen dieses Seminars:

1. Über welche Eigenschaften von Modellen können wir sprechen? Formaler gesprochen: in der Logik entspricht eine Eigenschaft einer Menge, nämlich der Menge aller Objekte die diese Eigenschaft haben; die Frage ist also: welche Mengen von Modellen können wir mit unserer Sprache charakterisieren? Man bezeichnet das als **Expressivität**.

2. Die wichtigsten Eigenschaften von logischen Formeln ϕ, χ sind die folgenden: ist ϕ eine Tautologie (d.h. ist ϕ in jedem Modell immer wahr)?

ist ϕ ein Widerspruch (d.h. ist ϕ in keinem Modell nie wahr)? und folgt aus der Wahrheit von ϕ die Wahrheit von χ (d.h. ist in jedem Modell, in dem ϕ wahr ist, auch χ wahr)? Die Frage für uns ist nun: gegeben eine Logik $\mathcal{L}$, können wir für beliebige $\mathcal{L}$-Formeln ϕ, χ diese Fragen immer definitiv beantworten? Man bezeichnet das als **Entscheidbarkeit**.

Offensichtlich sind Expressivität und Entscheidbarkeit an unterschiedlichen Polen angesiedelt: je größer die Expressivität, desto schwieriger die Entscheidbarkeit. Eine Logik, in der alle wichtigen Probleme unentscheidbar sind, ist in der Praxis kaum zu gebrauchen; ebensowenig eine Logik, in der wir wichtige Eigenschaften von Zeitmodellen nicht ausdrücken können. Das große Ziel in der angewandten Logik ist daher: Sprachen zu finden, die alle relevanten Eigenschaften ausdrücken können, aber immer noch entscheidbar sind. Es gilt also einen vorteilhaften Kompromiss zu finden von Expressivität und Entscheidbarkeit. Die wichtigen temporalen Logiken – alle die wir betrachten werden – zeichnen sich eben dadurch aus.

5 LTL

LTL steht für linear temporale Logik. Es handelt sich um eine propositionale modale Logik; sie heißt "linear", weil ihre Modelle lineare Zeitverläufe sind (und nicht etwa Bäume). Nichtdeterminismus, also die Tatsache, dass die Zukunft ungewiss ist und mehrere Möglichkeiten bestehen, wird dadurch ausgedrückt, dass ein Formel eine *Menge* von Modellen hat – dazu später mehr. Wir betrachten zunächst die Syntax von LTL, die induktiv wie folgt definiert ist:

1. $\top, \bot$ sind LTL-Formeln.
2. Falls $p \in \mathit{Var}$, dann ist p eine LTL-Formel.
3. Fall α, β LTL-Formeln sind, dann sind auch $\alpha \wedge \beta, \alpha \vee \beta, \alpha \rightarrow \beta, \neg\alpha$ LTL-Formeln.
4. Falls α eine LTL-Formel ist, dann sind auch $N\alpha$, $\Box\alpha$, $\Diamond\alpha$ LTL-Formeln.
5. Fall α, β LTL-Formeln sind, dann sind auch $\alpha U \beta$, $\alpha R \beta$ LTL-Formeln
6. Sonst ist nichts eine LTL-Formel.

$\top$ ist einfach immer wahr, $\bot$ ist immer falsch (die beiden sind interdefinierbar).

Die Intuition hinter N ist folgende: N steht für den *nächsten* Zeitpunkt, also wird $N\alpha$ erfüllt, falls der nächste Zeitpunkt α erfüllt. NB: das setzt ein diskretes Modell von Zeit voraus, wie etwa die natürlichen Zahlen – in den rationalen oder reellen Zahlen gibt es keinen eindeutigen Nachfolger einer gewissen Zahl.

$\Diamond$ und $\Box$ verhalten sich wie in normaler modaler Logik; da wir aber lineare Zeitmodelle haben, vereinfacht sich ihre Semantik wie folgt: $\Diamond\alpha$ bedeutet: es gibt einen Zeitpunkt in der Zukunft, an dem α gilt; $\Box\alpha$ bedeutet: in aller Zukunft gilt α. Das bedeutet natürlich dass wir eine transitive Zeitrelation haben!

Etwas komplizierter ist die Bedeutung von U, das von *until* kommt: $\alpha U \beta$ bedeutet dementsprechend: an einem Punkt in der Zukunft gilt β, und an allen Punkt davor gilt α. R kommt von *releases* (ab/auflösen), und $\alpha R \beta$ bedeutet: β (!) gilt solange, bis es irgendwann von α abgelöst wird. Das ist sehr ähnlich zu U, unterscheidet sich aber in zwei wichtigen Details. Wir

betrachten nun die formalen Definitionen der Semantik.

Ein **unendliches LTL-Modell** ist ein Kripke Modell $(\mathbb{N}, <, val)$, wobei

- $<$ die transitive, irreflexive kleiner-als Relation auf $\mathbb{N}$ ist; und
- $val : \mathbb{N} \to \wp(Var)$ die Valuation, die uns sagt welche Propositionen an welchem Punkt wahr sind.

Es ist also klar das LTL ein *diskretes* Modell von Zeit hat: Zeit verläuft in "Sprüngen". Das ist natürlich die Voraussetzung dafür, dass wir eine Modalität wie N haben; in einem kontinuierlichen Zeitmodell kann es keinen "nächsten Moment" geben. In einem LTL-Modell ist also der *Rahmen* $(\mathbb{N}, <)$ festgelegt als unser Zeitbegriff; was nicht festgelegt ist, ist val, also was in der Zeit passiert.

$(\mathbb{N}, <, val), n \models p$ gdw. $p \in val(n)$

$(\mathbb{N}, <, val), n \models \top$

$(\mathbb{N}, <, val), n \not\models \bot$

$(\mathbb{N}, <, val), n \models N\alpha$ gdw. $(\mathbb{N}, <, val), n+1 \models \alpha$

$(\mathbb{N}, <, val), n \models \Box\alpha$ gdw. für alle $m > n$, $(\mathbb{N}, <, val), m \models \alpha$

$(\mathbb{N}, <, val), n \models \Diamond\alpha$ gdw. es ein $m > n$ gibt so dass $(\mathbb{N}, <, val), m \models \alpha$

$(\mathbb{N}, <, val), n \models \alpha U \beta$ gdw. es ein $m > n$ gibt, so dass gilt:
1. $(\mathbb{N}, <, val), m \models \beta$, und
2. für alle $n' : n \leq n' < m$, $(\mathbb{N}, <, val), n' \models \alpha$.

$(\mathbb{N}, <, val), n \models \alpha R \beta$ gdw. für alle $m > n$ gilt: wenn es kein n' gibt so dass $n < n' < m$ und $(\mathbb{N}, <, val), n' \models \alpha$, dann gilt $(\mathbb{N}, <, val), m \models \beta$.

Der Unterschied zwischen $\alpha U \beta$ und $\beta R \alpha$ ist subtil (wohlgemerkt: wir müssen die Reihenfolge bei R umdrehen, damit die Bedeutung ähnlich wird zu U!). Es gibt 3 wichtige Unterschiede in der Bedeutung von $\alpha U \beta$ und $\beta R \alpha$. Wir beschreiben zunächst $\alpha U \beta$. Die Formel bedeutet: α Until β, also: bis β gilt muss α gelten. Hier die 3 Punkte:

1. $\alpha U \beta$ impliziert $\Diamond \beta$, sprich: irgendwann wird β eintreten (und bis dahin gilt α). Dieser Punkt, in dem β eintritt, muss in der Zukunft liegen.

2. Bis zu diesem Punkt, an dem β gilt, wird α gelten, und zwar inklusive Gegenwart.

3. An dem Punkt, an dem β eintritt, muss α nicht mehr gelten; es gibt also keine (zwingende) Überscheidung.

Nun im Gegenzug $\beta R \alpha$, was bedeutet: β release α, also: β löst α ab. Das bedeutet etwas ähnliches, aber:

1. $\beta R \alpha$ impliziert *nicht* dass die Ablösung irgendwann eintreten wird. Es kann auch sein, dass α für alle Zukunft gilt (also: $\Box \alpha$ impliziert $\beta R \alpha$)

2. Bis zu einem Punkt, an dem β gilt, muss α gelten (oder der Punkt wird nie kommen, siehe oben), aber jedenfalls exklusive Gegenwart. $\beta R \alpha$ macht also keine Aussage über den aktuellen Zeitpunkt!

3. An dem Punkt, an dem β eintritt (wenn es eintritt), muss α ebenfalls noch wahr sein; es gibt also eine Überscheidung bei der Ablösung.

Diese etwas seltsame Halb-Ähnlichkeit von U und R liegt daran, dass die beiden interdefinierbare Duale sind (siehe weiter unten).

Eine **wichtige Konvention** für uns ist folgende: die Validität einer LTL-Formel in einem Modell ist nicht definiert über Wahrheit an allen Zeitpunkten, sondern über Wahrheit am "Ursprung der Zeit":

$$(\mathbb{N}, <, val) \models \alpha \text{ gdw. } (\mathbb{N}, <, val), 0 \models \alpha$$

Um eine Formel im Modell wahr zu machen, machen wir sie also wahr am Punkt 0 (alle anderen sind egal). Das liegt daran, dass unsere Modalitäten alle auf die Zukunft gerichtet sind, dementsprechend ist die 0 der (einzige) Punkt, von dem aus wir alle anderen Punkte erreichen können.

5.1 Interdefinierbarkeit der Modalitäten

Offensichtlich gibt es hier einiges an Redundanz; wir brauchen tatsächlich nur 3 der atomaren Konstanten/Modalitäten, nämlich $\top, N, U$ (oder $\bot, N, R$). Wir können dann definieren:

$$\begin{aligned} \bot &\equiv \neg\top \\ \Diamond\alpha &\equiv \top U \alpha \\ \Box\alpha &\equiv \neg\Diamond\neg\alpha \\ \alpha R\beta &\equiv \neg((\neg\alpha)U(\neg\beta)) \end{aligned}$$

Diese Definitionen sind äquivalent zu den semantischen Definitionen, die oben gegeben sind. Insbesondere ist wichtig dass R dual zu U definiert wird; manchmal ändert sich die Definition von U, und dann haben wir auch eine leicht andere Definition von R. Es gibt eine weitere Modalität die sich definieren lässt, da sie häufig benutzt wird:

(1) $\quad \alpha W\beta \equiv (\alpha U\beta) \vee \Box\alpha$

d.h. $\alpha W\beta$ ist fast äquivalent zu $\alpha U\beta$, erlaubt aber die Möglichkeit dass β niemals eintritt (und ist auch fast äquivalent zu $\beta R\alpha$, aber die Überschneidungen sind anders).

5.2 Beispiele

Finden Sie jeweils eine LTL-Formel, die sicherstellt dass folgende Aussagen in allen Ihren Modellen wahr ist.

Beispiel 1 Alle p werden gefolgt von einem q, aber niemals unmittelbar.

Beispiel 2 Wir suchen eine Formel die besagt: am Anfang gilt p; und immer wenn an einem Punkt p gilt, gilt unmittelbar darauf q, und umgekehrt.

Beispiel 3 Wir suchen eine Formel die besagt: wenn an einem gewissen Punkt i p gilt, an einem Nachfolgepunkt j q, dann ist q unmittelbar gefolgt von r – vorausgesetzt, r gilt an keinem Punkt $k : i \leq k < j$.

Lösung 1: $\Box(p \rightarrow rR(\Box(q \rightarrow Nr)))$
hier Vorsicht mit dem unmittelbaren Nachfolger von p - die $\Box$ greift nicht!
Lösung 2: $(p \wedge \neg r) \rightarrow (rR((q \rightarrow Nr) \wedge (\Box(q \rightarrow Nr))))$
etwas komplizierter, aber korrekt soweit ich sehe
Lösung 3: $(p \rightarrow ((\neg r)Uq)) \rightarrow (p \rightarrow ((\neg r)U(q \wedge Nr)))$
ein alternativer Ansatz.

Beispiel 4 Wir suchen eine Formel besagt, dass ps und qs in Blöcken alternieren, und f.a. $n \in \mathbb{N}$, n Punkte an denen p gilt sollen immer gefolgt sein von n Punkten, an denen q gilt.

Vermutung: es gibt keine solche Formel!

Übung: Definierbarkeit

1. Nehmen Sie nun eine Formel α (als gesetzte Variable) und transformieren Sie sie in eine Formel α', so dass gilt: $(\mathbb{N}, <, val), n \models \alpha$ für alle $n \in \mathbb{N}$ genau dann wenn $(\mathbb{N}, <, val), 0 \models \alpha'$.

2. Nehmen Sie eine atomare Proposition p, und schreiben Sie eine Formel α, so dass $(\mathbb{N}, <, val), 0 \models \alpha$ genau dann wenn gilt: $p \in val(n) \iff$ n ist durch 3 teilbar. Anders gesagt: finden Sie eine Formel, die p als "durch drei teilbar" definiert.

5.3 Sicherheit und Lebendigkeit

Üblicherweise formuliert man eine Bedingung α in temporaler Logik so, dass $(\mathbb{N}, <, val), 0 \models \alpha$; d.h. man geht davon aus dass wir am Zeitpunkt 0 sind. Man unterscheidet in der temporalen Logik zwei Arten von Bedingungen, nämlich **Sicherheitsbedingungen** und **Lebendigkeitsbedingungen**.

- Eine Sicherheitsbedingung hat die Form: $\Box\alpha$, und das Prinzip ist einfach: wir möchten, dass in aller Zukunft immer α erfüllt wird, da es für die Sicherheit unseres Systems/Modells essentiell ist.

- Lebendigkeitsbedingungen haben die Form $\Box\Diamond\alpha$, was soviel bedeutet: in aller Zukunft gibt es eine Zukunft wo α gilt.

In unserem Rahmen $(\mathbb{N}, <)$ ist das gleichbedeutend mit: α soll *unendlich oft* stattfinden (nicht aber in einem Rahmen der Form $(\mathbb{R}, <)$!).

Wenn wir diese Bedingung auf ein System übertragen bedeutet dass soviel wie: das System darf *nie* in einen Zustand gelangen, von dem es nicht mehr in einen Zustand kommt in dem es α erfüllt. Es darf aber durchaus in einen Zustand kommen, wo es nicht mehr α erfüllt! Z.B. darf eine Fabrik nie in einen Zustand kommen, wo sie abbrennt (Sicherheitsbedingung); sie darf sehr wohl in einen Zustand kommen, wo sie nicht mehr produziert (wenn sie nämlich sonst abbrennt), aber sie darf nie in einen Zustand kommen, wo sie in keiner Zukunft mehr produziert (Lebendigkeitsbedingung). Wir sehen: Sicherheit geht vor!

Übung

Nehmen Sie zwei LTL-Formeln α, β. Wir schreiben $\alpha \models \beta$ genau dann wenn gilt: falls $(\mathbb{N}, <, val), 0 \models \alpha$, dann $(\mathbb{N}, <, val), 0 \models \beta$ (für alle Valuationen val). Entscheiden (und begründen) Sie, ob und warum die folgenden Konsequenzen (nicht) gültig sind.

1. $pU\neg p \models \Diamond\neg p$ Ja, siehe Definition von U!
2. $\Box(p \vee q) \models (\Box p) \vee (\Box q)$ Falsch, Gegenbeispiel: *pqpqp...*
3. $\Diamond(p \vee q) \models (\Diamond p) \vee (\Diamond q)$ Ja, Fallunterscheidung.
4. $\Box(p \wedge q) \models (\Box p) \wedge (\Box q)$ Ja, die Konjunkte folgen einzeln, also auch die Konjunktion!
5. $\Box\Diamond(pUq) \models \Box\Diamond(p \wedge q)$ Nein, Gegenbeispiel: *pqpqpqpqpqpq...*
6. $\Box\Diamond(pUq) \models \Box\Diamond p \wedge \Box\Diamond q$ Gilt, pUq setzt voraus dass p gilt (am selben Punkt) und q gilt (in der Zukunft)
7. $\Box\Diamond(pRq) \models \Box\Diamond(p \wedge q)$ Nein, Gegenbeispiel: *qqqqqq...* (bedenke: die Ablösung muss nicht kommen!)
8. $(\Box\Diamond p)Uq \models \Box\Diamond p$ Ja: falls n erfüllt $\alpha U\beta$, dann erfüllt n auch α. Insbesondere am Punkt 0!
9. $(\Box\Diamond p)Uq \models \Box\Diamond(pUq)$ Nein, Gegenbeispiel: p an allen geraden Punkten, q an Punkt 4, sonst nirgends.
10. $\Box\Diamond(pUq) \models (\Box\Diamond p)Uq$ ist korrekt, leicht zu zeigen mittels 6.
11. $pU(qUr) \models (pUq)Ur$

    Nein, Gegenbeispiel: nimm ein Modell M in dem gilt:

    $\{0, ..., n\} \models p$, $\{n+1, ..., n+m\} \models q$, $n+m+1 \models r$, und sonst nichts. Dann gilt: $M \models pU(qUr)$, aber: für $n+m'$, für $1 \leq m' \leq m$, gilt $n+m' \not\models pUq$, also: $M \not\models (pUq)Ur$
12. $(pUq)Ur \models pU(qUr)$

    Nein, Gegenbeispiel: nimm ein Modell M in dem gilt:

$\{0, ..., n\} \models p$, $n+1 \models q$, $n \models r$, und sonst nichts. Dann gilt: $\{0, ..., n-1\} \models pUq$, da $n \models r$ gilt $0 \models (pUq)Ur$. Aber: es gibt keinen Punkt der qUr erfüllt, also auch keinen, der $pU(qUr)$ erfüllt. Insbesondere: $0 \not\models pU(qUr)$.

13. $(pRq \wedge \Diamond p) \models \Diamond \neg q$

Hausaufgabe 3

Abgabe am 11.5.2021 vor dem Seminar! Entscheiden und bergründen Sie, ob folgende Konsequenzen gültig sind:

1. $(\Box\Diamond p) \wedge \Diamond q \models (\Box\Diamond p)Uq$
2. $\Box p \models \Box\Diamond p \models \Diamond p$
3. $(\Diamond p) \rightarrow \Box\Diamond q \models \Box(\neg p \vee \Box\Diamond q)$

6 ω-Sprachen

Eine ω-Sprache L ist eine Teilmenge von Σ^ω, wobei Σ^ω die Menge aller unendlichen Worte

$$a_1 a_2 a_3 ...$$

denotiert. Wem das zu vage ist, kann sich ein ω-Wort $\overline{w}$ auch vorstellen als eine Funktion

$$\overline{w} : \mathbb{N} \to \Sigma,$$

wobei $\overline{w}(n)$ einfach den n-ten Buchstaben angibt. Dieses Konzept ist deswegen sinnvoll, weil es eine Sache klarmacht: jeder Buchstabe in einem ω-Wort hat nur endlich viele Vorgänger. Dementsprechend ist

$$(ab)^\omega = ababab...$$

ein ω-Wort; $a^\omega b a^\omega$ aber nicht, da das b in diesem Wort bereits unendlich viele Vorgänger hat. Allerdings ist etwa die Menge

$$a^* b a^\omega = \{a^n b a^\omega,\ n \in \mathbb{N}\}$$

eine ω-Sprache, nämlich die Menge aller $a^n b a^\omega$, $n \in \mathbb{N}$.

Ein unendliches LTL-Modell $(\mathbb{N}, <, val)$ kann man tatsächlich auffassen als ein ω-Wort, indem man nämlich wiederum annimmt dass

$$\Sigma = \wp(\mathit{Var})$$

(d.h. eine Menge von atomaren Propositionen ist *ein* Buchstabe!), dass $\overline{w}$ eine Funktion ist die definiert wird durch $\overline{w}(n) = val(n)$. Wenn man sich das als Wort vorstellt, dann sieht das so aus: falls $val(0) = \{p\}, val(1) = \emptyset, val(2) = \{p, q\}$ etc., dann sieht unser Wort aus

$$\{p\}\emptyset\{p, q\}...$$

D.h. wir schreiben die Menge der Propositionen, die an einem Punkt wahr sind, ganz einfach in ihrer natürliche Folge hintereinander. Wir schreiben

$$(\mathbb{N}, <, val) \cong a_1 a_2 a_3 ... a_i ...$$

wenn gilt: für alle $i \in \mathbb{N}$ ist

(2) $\quad val(i) = a_i$

(a_i ist ja, nach Annahme, eine Teilmenge von *prop*). Wir bezeichnen die Relation $\cong$ auch als **Isomorphie**, da sie eine eindeutige Abbildung ist, die alle strukturellen Eigenschaften bewahrt. Da wir für LTL den Rahmen $(\mathbb{N}, <)$ festgelegt haben, gilt folgendes: eine LTL-Formel α denotiert eine Menge $\mathbf{X}$ von Modellen der Form $(\mathbb{N}, <, val)$; wir sagen einfach: $(\mathbb{N}, <, val) \in \mathbf{X}$ genau dann wenn $(\mathbb{N}, <, val) \models \alpha$. Da nun jedes Modell in $\mathbf{X}$ isomorph ist zu einem unendlichen Wort, können wir also sagen: eine Formel denotiert eine Menge von unendlichen Wörtern; eine sogenannte ω-Sprache. Grund genug sich das etwas näher anzuschauen.

Büchi Automaten Wir können ω-Sprachen auch unabhängig charakterisieren, nämlich mit sog. Büchi-Automaten. Ein Büchi-Automat sieht genauso aus wie ein endlicher Automat, nämlich

$$(Q, \Sigma, q_0, F, \delta)$$

wobei

- Q die Zustände,
- Σ das Alphabet,
- q_0 der Startzustand,
- F die akzeptierenden Zustände,
- $\delta \subseteq Q \times \Sigma \times Q$ die Übergangsrelation.

Ebenso induziert jedes Eingabewort eine Abfolge von Zustandsübergängen, die die Bedingungen in δ respektieren müssen. Sei

$$\overline{w} = a_0 a_1 a_2 ...$$

ein ω-Wort, $\mathfrak{A}$ ein Büchi-Automat. Dann sagen wir: $\overline{w}$ induziert $(q_0, q_1, q_2, ...)$ in $\mathfrak{A}$, falls f.a. $n \in \mathbb{N}$ gilt: $(q_n, a_n, q_{n+1}) \in \delta$. Der interessante Punkt und die eigentliche Abweichung liegt in der Akzeptanzbedingung; denn unsere Eingaben haben ja kein Ende. Die Bedingung ist nun wie folgt:

Definition 1 $\mathfrak{A}$ *akzeptiert* $\overline{w}$*, wenn es ein* $q \in F$ *gibt, das Wort* $\overline{w}$ *die Zustandsfolge* $(q_0, q_1, q_2, ...)$ *in* $\mathfrak{A}$ *induziert, und* $(q_0, q_1, q_2, ...)$ *unendlich viele Vorkommen von* q *enthält.*

Wir müssen also mindestens einen akzeptierenden Zustand *unendlich oft streifen*. Wir können einen Büchi-Automaten als eine Folge von Arbeitsanweisungen sehen. Die Verwendung von endlichen Automaten hat aber eine Reihe von weiteren Vorteilen, wie wir gleich sehen werden.

Beispiel 1 Nimm den Automaten (aus der Stunde) mit

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_2\}$
- $\delta = \{(q_0, b, q_0), (q_0, a, q_1), (q_1, a, q_2), (q_1, b, q_0), (q_2, a, q_2), (q_2, b, q_1)\}$

Wie beschreibt man diese ω-Sprache? Erstmal eine Beobachtung:

- Wir müssen immer wieder nach q_2, also:
- wenn wir einmal in q_2 sind, dann muss
- auf ein b wieder ein a folgen, und
- auf 2 oder mehr bs zwei as folgen.

Denn ω-Ausdruck macht man am besten, indem man nach Zustnden vorgeht:

$$\underbrace{b^*}_{q_0}\underbrace{a(b^*a)^*}_{q_0 \to q_1}\underbrace{a^+(ba^+)^*}_{q_1 \to q_2}$$

Dieser Ausdruck deckt alle (endlichen) Läufe ab, die keinen Übergang haben $q_2 \to q_0$. Jetzt gibt es zwei Möglichkeiten: entweder wir gehen zurück nach q_0, oder wir bleiben in $\{q_1, q_2\}$. Für mich sieht es aus wie die intuitivste Variante, wir geben dem obigen Ausdruck einen Anhang. Man muss sich hier dazu denken: wir sind bereits in q_2.

$$(\underbrace{(bb)^*}_{q_2 \to q_0}\underbrace{(b^+a)^*}_{q_0 \to q_1}\underbrace{(a^+b)}_{q_1 \to q_2})^\omega$$

Man beachte: der Übergang nach q_0 ist nicht obligatorisch! Der zugehörige Ausdruck lautet also:

$$b^*a(b^*a)^*a^+(ba^+)^*((bb)^*(b^+a)^*(a^+b))^\omega$$

Ganz schon kompliziert!

Beispiel 2 Nehmen Sie den Automaten mit

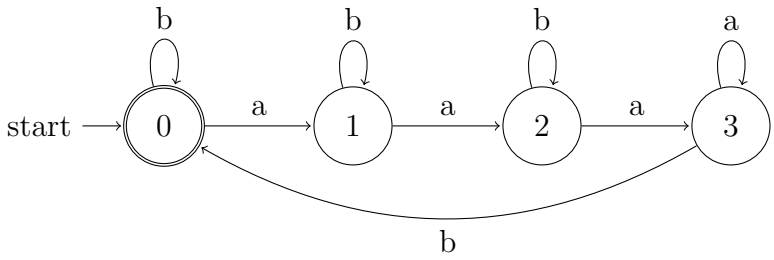
- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- $\delta = \{(q_0, a, q_0), (q_0, b, q_1), (q_1, b, q_2), (q_1, a, q_0), (q_2, a, q_2), (q_2, b, q_2)\}$

Wie beschreibt man diese ω-Sprache?

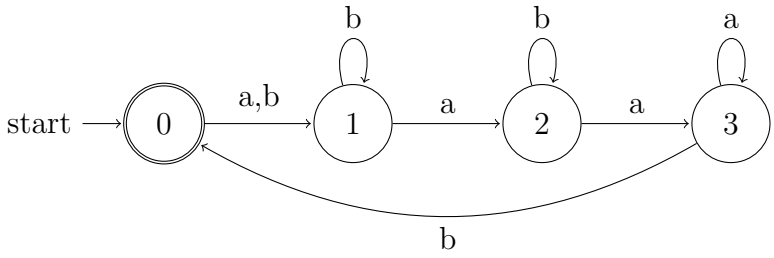
Hausaufgabe 4

Bitte bearbeiten bis zum 18.5.2021

Nehmen Sie folgenden endlichen Büchi-Automaten $\mathfrak{A}$:



1. Entscheiden Sie, ob $\mathfrak{A}$ folgendes Wort akzeptiert, und begründen Sie: $(ab)^\omega$
2. Setzen Sie $a := \{p\}$, $b := \{q\}$. Nehmen Sie an, dass ein Wort $w \in \{a, b\}^\omega$ einem Modell LTL-Modell $\mathcal{M}$ entspricht, und w wird von $\mathfrak{A}$ akzeptiert. Ist dann $\mathcal{M}$ ein Modell von $\Box\Diamond p$? Begründen Sie!


start
0
a,b
1
b
a
2
b
a
3
a
b

7 Endliche LTL-Modelle

Um den Schritt von Logik zu Sprachen zu vereinfachen werden wir zunächst kurz die endlichen LTL-Modelle betrachten, die folglich isomorph sind zu endlichen Worten (also solchen wie wir sie kennen). Ein **endliches LTL** Modell hat die Form

$$(\{0, ..., n\}, <, val),$$

wobei val als folgende eine Funktion definiert (offensichtlicherweise):

$$val : \{0, ..., n\} \to \wp(Var)$$

Eigentlich gibt es nicht besonderes zu bemerken über endliche LTL-Modelle, außer folgendes: der Begriff der Wahrheit und Konsequenz ist hier tatsächlich anders als in den unendlichen Modellen (siehe die nächste Aufgabe!) Insbesondere wenn wir an die Modalität $\Diamond$ und die Lebendigkeitsbedingungen denken:

- die Formel $\Box\Diamond p$ hat tatsächlich *kein* endliches Modell!

denn sie besagt dass an allen Punkten im Modell $\Diamond p$ gelten muss, es muss also einen $<$-größeren Punkt geben an dem p gilt. Nun haben aber alle endlichen Modellen einen $<$-maximalen Punkt – an dem kann aber nicht $\Box p$ gelten! Also kann in dem Modell nicht $\Box\Diamond p$ gelten. Ein gewichtiges Argument gegen die Verwendung von endlichen Modellen ist also: wir können Lebendigkeitsbedingungen nicht mehr sinnvoll interpretieren.

Der Grund warum wir an dieser Stelle endliche Modelle einführen ist folgender: so wie ein unendliches Modell einem ω-Wort entspricht, entspricht also ein endliches Modell einem normalen Wort. Wir werden die wichtigen Ergebnisse der folgenden Abschnitte für endliche Modelle formulieren, da auf diese Weise zumindest ein Abstraktionsschritt – von Sprachen zu ω-Sprachen – wegfällt.

8 Von LTL zu (Büchi-)Automaten

8.1 Motivation

Die Benutzung von Automaten hat eine Reihe von Vorteilen, die wir zunächst Einfachheit halber im Allgemeinen besprechen im Hinblick auf normale endliche Automaten. Nehmen wir einfach an wir haben die Korrespondenz

$$\text{Modell für eine Logik } \mathcal{L} \cong \text{(Endliches/unendliches) Wort}$$

Das führt uns natürlich auf folgende Korrespondenz:

$$\text{Modell für eine } \mathcal{L}\text{-Formel } \alpha \cong \text{Wort das erkannt wird von einem Automaten } \mathfrak{A}_\alpha$$

Wir möchten also für jede $\mathcal{L}$-Formel α einen Automaten $\mathfrak{A}_\alpha$, der modulo $\cong$ erkennt, ob ein Modell α erfüllt. Warum möchten wir das eigentlich? Zunächst folgendes: in allen Logiken ist der Automat $\mathfrak{A}_\alpha$ ein *endlicher* Automat.

Es gibt nun eine ganze Reihe von Motiven für diese Konversion. Erinnern wir uns an die 3 großen Entscheidungsprobleme der Logik:

1. ist α eine Tautologie (immer wahr),
2. ein Widerspruch (nie wahr),
3. und impliziert die Wahrheit von α die Wahrheit von β?

Diese Fragen lassen sich nun ganz einfach in automatentheoretische Fragen übersetzen. Nehmen wir die Frage: ist α ein Widerspruch? Das wird nun zur Frage: ist $L(\mathfrak{A}_\alpha) = \emptyset$? Diese Frage lässt sich wiederum sehr leicht beantworten, mit folgendes Überlegung:

- Jeder endliche Automat lässt sich determinisieren;
- jeder deterministische endliche Automat lässt sich minimisieren;
- der minimale deterministische Automat, der $\emptyset$ erkennt, ist $(Q = \{q_0\}, \Sigma, q_0, F = \emptyset, \delta = \emptyset)$.

Das lässt sich leicht prüfen, und so haben wir unser erstes Entscheidungsproblem bereits beantwortet! Diese Frage ist übrigens durchaus wichtig für die Praxis: wenn wir eine sehr komplexe Spezifikation α für ein System haben, dann lautet die entsprechende Frage: lässt sich die Spezifikation überhaupt erfüllen?

Als nächstes betrachten wir die Frage: ist α eine Tautologie? Für uns übersetzt sich diese Frage zu: ist $L(\mathfrak{A}_\alpha) = \Sigma^*$? Hier folgende Überlegung:

- Jeder endliche Automat lässt sich determinisieren;
- wenn ich in einem deterministischen endlichen Automaten $\mathfrak{A}$ die akzeptierenden und nicht-akzeptierenden Zustände vertausche (nenne das Ergebnis $\mathfrak{A}'$), dann ist $\overline{L(\mathfrak{A})} = L(\mathfrak{A}')$ (mengentheoretisches Komplement);
- wir haben $L(\mathfrak{A}) = \Sigma^*$ genau dann wenn $L(\mathfrak{A}') = \emptyset$ – was wir ja bereits entscheiden können!

Wir können also die Frage der Tautologie reduzieren auf die Frage der Universalität der Sprache, die wiederum reduzierbar ist auf die Frage der Leere der Sprache, die entscheidbar ist. Auf die Praxis bezogen hat diese Frage die folgende Bedeutung: hat eine Spezifikation überhaupt eine Bedeutung, oder ist sie einfach immer erfüllt?

Die komplizierteste Frage ist: folgt aus α β, d.h. erfüllt jedes Modell das α erfüllt auch β? Automatentheoretisch übersetzt sich das wie folgt: gilt $L(\mathfrak{A}_\alpha) \subseteq L(\mathfrak{A}_\beta)$? Dieses Problem lässt sich nun auf mehrere Arten lösen:

1. Gegeben endliche Automaten $\mathfrak{A}, \mathfrak{A}'$ können wir einen Automaten $\mathfrak{A}''$ konstruieren, so dass $L(\mathfrak{A}) \cap L(\mathfrak{A}') = L(\mathfrak{A}'')$.
2. Wir haben $L(\mathfrak{A}_\alpha) \subseteq L(\mathfrak{A}_\beta)$ gdw. $L(\mathfrak{A}_\alpha) \cap L(\mathfrak{A}_\beta) = L(\mathfrak{A}_\alpha)$. Wir müssen also nur die Äquivalenz der Automaten prüfen. Da wir (bis auf Zustandsnahmen) eindeutige minimale deterministische Automaten haben, ist das gut machbar.

Ein anderer Ansatz ist folgender (der stärker die vorigen Ergebnisse benutzt):

1. Wir haben $L(\mathfrak{A}_\alpha) \subseteq L(\mathfrak{A}_\beta)$ gdw. $L(\mathfrak{A}_\alpha) \cap \overline{L(\mathfrak{A}_\beta)} = \emptyset$.
2. Wir brauchen also nur Komplement, Schnitt, und wir sind wieder beim Problem der Leere der Sprache.

Hier eine Zusammenfassung:

Logik	**Automatentheorie**
α Deskriptiv, beschreibend	$\mathfrak{A}$ Effektiv, berechnend
Modelle $M = (\mathbb{N}, <, val)$	ω-Worte $\overline{w}$ über $\wp(Var)$
$M \models \alpha$	$\overline{w}_M \in L(\mathfrak{A}_\alpha)$
Ist α eine Tautologie?	Ist $L(\mathfrak{A}_\alpha) = \Sigma^*$?
Ist α ein Widerspruch?	Ist $L(\mathfrak{A}_\alpha) = \emptyset$?
Gilt $\alpha \models \beta$?	Gilt $L(\mathfrak{A}_\alpha) \subseteq L(\mathfrak{A}_\beta)$?

Gegeben $\mathfrak{A}$, wie bekomme ich $\mathfrak{A}'$ so dass $L(\mathfrak{A}') = \overline{L(\mathfrak{A})}$?
$F' = Q - F$, sonst alles gleich. Aber nur, wenn $\mathfrak{A}$ deterministisch ist!

Wie determiniert man? $Q \Rightarrow \wp(Q)$!
Gegeben Q, wie groß ist $\wp(Q)$? Antwort: $2^{|Q|}$
Z.B.: $|Q| = 80$

Frage Wie baue ich eigentlich $\mathfrak{A}_\alpha$ aus α?

8.2 Alternierende Automaten

Um LTL-Formeln in Automaten zu übersetzen, braucht es einen wichtigen Zwischenschritt, die sog. alternierenden Automaten.

Wir haben bislang deterministische und nicht-deterministische Automaten kennengelernt. Nehmen wir an, wir haben einen nicht-deterministischen Automaten $\mathfrak{A}$, mit

$$(q, a, q'), (q, a, q'') \in \delta$$

Wir können das auch etwas anders schreiben und auffassen: wenn wir in q sind, a lesen, dann gehen wir nach q' oder q''; erlauben wir uns also die Notation:

$$(q, a, q' \vee q'')$$

Das sagt dasselbe wie vorhin, schreibt sich aber jetzt wie ein deterministischer Automat. Nichtdeterminismus bedeutet soviel wie: existenzielle Auswahl – d.h. aus einer Menge von Möglichkeiten müssen wir eine wählen. Das entspricht dem Wesen der logischen Disjunktion. Wir werden nun dieses Konzept generalisieren, indem wir auch **Konjunktion** erlauben:

$$(q, a, q' \wedge q'')$$

bedeutet: wenn wir in q sind, a lesen, dann gehen wir in *zwei* Zustände, nämlich q' und q''. Technisch gesehen ist der Unterschied folgender:

- Bei $(q, a, q' \vee q'')$ machen wir zwei Pfade auf; wir müssen dabei aber nur mit einem einen akzeptierenden Zustand erreichen, damit wir die Eingabe akzeptieren (existenzielle Auswahl).
- Für einen Übergang $(q, a, q' \wedge q'')$ machen wir ebenfalls zwei Pfade auf, aber wir müssen nun, um die Eingabe zu akzeptieren, auf *beiden* Pfaden einen akzeptierenden Zustand erreichen (wir können das einen universellen Übergang nennen).

Alternierende Automaten haben sowohl existenzielle als auch universelle Übergänge, daher alternierend (in der ursprünglichen Formulierung haben sich die beiden immer abgewechselt).

Wir nennen $\mathbf{B}^+(Q)$ die Menge aller **positiven Booleschen Formeln** über Q, d.h.:

1. Falls $q \in Q$, dann $q \in \mathbf{B}^+(Q)$;

2. Falls $\alpha, \beta \in \mathbf{B}^+(Q)$, dann $\alpha \vee \beta, \alpha \wedge \beta \in \mathbf{B}^+(Q)$

Wir haben also alle Booleschen Konnektoren außer $\neg$; daher positiv.

Definition 2 *Ein endlicher alternierender Automat ist ein Automat* $(Q, \Sigma, q_0, F, \delta)$*, wobei alles wie gehabt ist, nur* δ *ist eine Funktion* $\delta : Q \times \Sigma \to \boldsymbol{B}^+(Q)$*.*

Wenn wir also in einem Zustand q sind, einen Buchstaben von a lesen, dann gehen wir in eine positive Boolesche Kombination von Zuständen. Was nun etwas komplizierter ist, ist die Definition von *Läufen*: in normalen nicht-deterministischen Automaten $\mathfrak{A}$ ist ein Lauf, gegeben ein Eingabewort

$$a_1 ... a_n,$$

eine Folge

$$(q_0, a_1, q_1, a_2, q_2, ..., a_n, q_n);$$

man beachte dass q_0 der Startzustand ist, aber sonst der Index q_i sich nur auf die Position des Zustandes bezieht, nicht auf die Identität. Weiterhin verlangen wir dass für alle $i : 0 \leq i < n$ gilt: $(q_i, a_i, q_{n+1}) \in \delta$.

- In einem deterministischen Automaten induziert eine Eingabe genau einen Pfad; sie wird akzeptiert, wenn der letzte Zustand akzeptierend ist.

- In einem nicht-deterministischen Automaten induziert eine Eingabe eine *Menge* von Pfaden, und sie wird akzeptiert wenn ein Pfad darunter ist, der mit einem akzeptierenden Zustand endet.

Im alternierenden Automaten gibt es die Möglichkeit, zwei Pfade **gleichzeitig** "zu betreten"; d.h.: um die Eingabe zu akzeptieren, müssen *beide* Pfade auf einen akzeptierenden Zustand führen.

Um dieses Konzept in voller Allgemeinheit ausdrücken zu können, sagt man dass der Lauf eines alternierenden Automaten, gegeben ein Wort w, ein **Baum** ist. Wir betrachten zunächst die atomaren Übergänge: wir haben einen atomaren Übergang

$$q \xrightarrow{a} (q_1, ..., q_i)$$

in einem alternierenden Automaten $\mathfrak{A}$, genau dann wenn

1. $(q, a, \theta) \in \delta$ und
2. $\{q_1, ..., q_i\} \models \theta$ (da θ eine Formel der propositionalen Logik ist, verstehen wir die Erfüllung in diesem Sinne.)

Ein Zustand q hat also für eine Eingabe a eine Liste von unmittelbaren Nachfolgern, und diese Liste muss, als Menge gesehen, die Formel θ erfüllen (im Sinne propositionaler Logik). We wir also diese lokale Bedingung induktiv auf Eingaben beliebiger Worte erweitern, dann bekommen wir für jedes Wort eine Menge von Bäumen der Form:

$$\begin{array}{c} q_0 \\ a_1 \\ q_{00}\ ,\ ,\ldots,\ q_{0i} \\ a_2 \\ q_{000}\ ,\ ,\ldots,q_{00k}\ ,\ ,\ldots,\ q_{0i0}\ ,\ ,\ldots,q_{0ij} \\ a_3 \\ \ldots \end{array}$$

Hier ist wieder q_0 der Startzustand, während die anderen Subskripte nur dazu dienen, die Position des Zustandes im Baum festzulegen. Außerdem gilt für alle lokalen Teilbäume:

1. falls $(q, a, \theta) \in \delta$, $q = q_{\vec{n}}, a = a_i$, dann dominiert $q_{\vec{n}}$ die Knoten $q_{\vec{n}0}, ..., q_{\vec{n}j}$, und
2. $\{q_{\vec{n}0}, ..., q_{\vec{n}j}\} \models \theta$.

Pfade in einem Baum sind konvexe Ketten, die nach Dominanz geordnet sind; Blätter sind die Knoten, die maximal sind nach Dominanz (also die untersten Knoten). In unseren Läufen sind alle Pfade von der Wurzel zu einem Blatt gleichlang. Ein endlicher Lauf (im Sinne eines Baumes) ist **akzeptierend**, genau dann wenn *alle* Blätter akzeptierende Zustände sind. Das ist soz. die Bedeutung der Konjunktion.

Wir sagen weiterhin: ein alternierender Automat $\mathfrak{A}$ akzeptiert ein Wort w, wenn es einen akzeptierenden Lauf gibt, der wohlgeformt ist für $\mathfrak{A}$ und w. Hiermit haben wir die Intuition formalisiert, dass für Konjunktionen von Zuständen, *jeder Pfad*, den wir gehen, akzeptierend sein muss.

Bemerkung 3 *Beachten Sie: auch wenn ein alternierender Automat eine Übergangsfunktion hat (nicht Relation), sind die Übergänge nicht deterministisch: eine Formel θ legt die Menge der Nachfolger nicht eindeutig fest!*

Hier das wichtigste Ergebnis zu alternierende Automaten:

Lemma 4 *Für jeden alternierenden Automaten $\mathfrak{A}$ gibt es einen nicht-deterministischen Automaten $\mathfrak{A}'$ so dass 1. $L(\mathfrak{A}) = L(\mathfrak{A}')$, und 2. $|\mathfrak{A}'| \leq 3^{|\mathfrak{A}|}$ ($|\mathfrak{A}|$ ist die Anzahl der Zustände des Automaten).*

Auch alternierende Automaten erkennen also genau die regulären Sprachen, allerdings geht die Umwandlung in nicht-deterministische Automaten mit einem schlimmstenfalls exponentiellen Wachstum der Zustandsmenge einher. Die Intuition hinter dieser Transformation ist relativ leicht zu verstehen: gegeben eine Zustandmenge Q für einen alternierenden Automaten bilden wir die Zustandsmenge

$$\{\bigwedge M : M \subseteq Q\} = \{q_1 \wedge ... \wedge q_i : q_1, ..., q_i \in Q\},$$

also die Menge aller Konjunktionen von Zuständen (modulo Äquivalenz).

Wenn wir nun im alternierenden Automaten einen wohlgeformten Übergang der Form

$$q \xrightarrow{a} (q_1, ..., q_i) \text{ (alternierend)}$$

haben, dann setzen wir im entsprechenden nichtdeterministischen Automaten einfach

$$(q, a, q_1 \wedge ... \wedge q_i) \in \delta \text{ (nicht-deterministisch)}$$

NB: da im alternierenenden Automaten ein Übergang (q, a, θ) eine Vielzahl von wohlgeformten Übergängen zulässt, ist der resultierende Automat nicht-deterministisch! Weiterhin setzen wir

$$(q_1 \wedge ... \wedge q_i,\ a,\ q_1^1 \wedge ... q_1^j \wedge ... \wedge q_i^1 \wedge ... \wedge q_i^j) \in \delta \text{ (nicht-deterministisch)}$$

falls alle Übergänge

$$q_1 \xrightarrow{a} (q_1^1, ..., q_1^i) \text{ etc. (alternierend)}$$

wohlgeformt sind nach dem alternierenden Automaten.

Wir können sehr leicht auch **alternierende Büchi-Automaten** definieren: so wie wir bei Büchi-Automaten verlangen, dass wir in einem unendlichen Lauf unendlich oft einen akzeptierenden Zustand streifen, so verlangen wir bei alternierenden Büchi Automaten, dass

$\Rightarrow$ in einem unendlichen Baum, der einen unendlichen Lauf darstellt, auf *jedem* unendlichen Pfad unendlich viele akzeptierende Zustände liegen.

Übung a

Nehmen Sie den alternierenden Automaten mit $Q = \{q_0, q_1\}, \Sigma = \{a, b\}, q_0, F = \{q_0\}$ und

- $\delta(q_0, a) = q_1 \wedge q_0$;
- $\delta(q_0, b) = q_0$;
- $\delta(q_1, a) = q_1 \vee q_0$
- $\delta(q_1, b) = q_1$

1. Akzeptiert der Automat das Wort aab?
2. Charakterisieren Sie die Sprache, die er beschreibt, mit einem regulären Ausruck oder einem deterministischen endlichen Automaten!

Übung b

Nehmen Sie den alternierenden Automaten mit $Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b\}, q_0, F = \{q_1, q_2\}$ und

- $\delta(q_0, a) = q_0 \vee (q_1 \wedge q_2)$
- $\delta(q_1, a) = q_0 \wedge q_2$
- $\delta(q_2, a) = q_1 \vee q_0$
- $\delta(q_0, b) = q_0$
- $\delta(q_1, b) = q_0$

- $\delta(q_2, b) = q_1 \vee q_2$

Charakterisieren Sie die Sprache, die er beschreibt, mit einem regulären Ausruck oder einem deterministischen endlichen Automaten!

Übung c

Nehmen Sie den alternierenden Automaten mit $Q = \{q_0, q_1\}, \Sigma = \{a, b\}, q_0, F = \{q_1\}$ und

- $\delta(q_0, a) = q_0 \wedge q_1$;
- $\delta(q_0, b) = q_0 \vee q_1$;
- $\delta(q_1, a) = q_0 \vee q_1$
- $\delta(q_1, b) = q_0 \wedge q_1$

Charakterisieren Sie die Sprache, die er beschreibt, mit einem regulären Ausruck oder einem deterministischen endlichen Automaten!

Übung d

(Einfach) Nehmen Sie den alternierenden Automaten mit $Q = \{q_0, q_1, q_3\}, \Sigma = \{a, b\}, q_0, F = \{q_0\}$ und

- $\delta(q_0, a) = q_0 \wedge q_1$
- $\delta(q_0, b) = q_0 \vee q_1$
- $\delta(q_1, a) = q_2 \wedge q_0$
- $\delta(q_1, b) = q_1 \wedge q_2$
- $\delta(q_2, a) = q_0$
- $\delta(q_2, b) = q_0$

Welche Sprache erkennt dieser Automat?

Übung e

(Kompliziert) Nehmen Sie den alternierenden Automaten mit $Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b\}, q_0, F = \{q_1\}$ und

- $\delta(q_0, a) = q_0 \wedge q_1$
- $\delta(q_0, b) = q_0 \vee q_1$
- $\delta(q_1, a) = q_1 \wedge q_0$
- $\delta(q_1, b) = q_1 \wedge q_2$
- $\delta(q_2, a) = q_0$
- $\delta(q_2, b) = q_0$

Welche Sprache beschreibt er? Liefern Sie einen deterministischen Automaten/regulären Ausdruck.

8.3 Von LTL-Formeln zu alternierenden Automaten

Es gibt einen prinzipiell recht einfachen Algorithmus, um von LTL-Formeln zu alternierenden Automaten zu kommen; tatsächlich ist das einer der Gründe, warum LTL für viele Anwendungen recht interessant ist.

Zunächst folgende Beobachtung: wir sagen dass eine LTL-Formel in **positiver Normalform** ist, wenn die Negation $\neg$ nur vor atomaren Formeln auftaucht. Folgendes Ergebnis lässt sich recht einfach zeigen:

Lemma 5 *Jede LTL-Formel α ist äquivalent zu einer LTL-Formel α' in positiver Normalform; d.h. $\mathcal{M} \models \alpha$ gdw. $\mathcal{M} \models \alpha'$.*

Beweis: Wir nehmen einfach die logischen Distributivgesetze für LTL; wir haben:

- $\neg(\alpha \wedge \beta) \equiv (\neg\alpha) \vee (\neg\beta)$
- $\neg\neg\alpha \equiv \alpha$
- $\neg(\alpha \vee \beta) \equiv (\neg\alpha) \wedge (\neg\beta)$
- $\neg\Box\alpha \equiv \Diamond\neg\alpha$
- $\neg\Diamond\alpha \equiv \Box\neg\alpha$
- $\neg N\alpha \equiv N\neg\alpha$
- $\neg(\alpha U\beta) \equiv (\neg\alpha)R(\neg\beta)$
- $\neg(\alpha R\beta) \equiv (\neg\alpha)U(\neg\beta)$

Das setzt natürlich voraus dass alle Konnektoren tatsächlich als atomare Teile der Sprache behandelt werden, obgleich sie eben auch teilweise definiert werden können. $\dashv$

Dieses Lemma ist extrem wichtig, denn Negation entspricht der Komplementierung; Komplementierung funktioniert aber nur mit deterministischen Automaten. Da bereits der Schritt von nicht-deterministischen Automaten zu deterministischen exponentiell sein kann, ist der Schritt von alternierenden Automaten schlimmstenfalls *hyperexponentiell*.

$\Rightarrow$ Wir müssen also für die Übersetzung die Negation nur für Formeln der Form $\neg p$ berücksichtigen, wobei $p \in Var$.

Wir betrachten nun die Automaten. Wir haben gesagt: unser Eingabealphabet sind *alle Mengen von atomaren Propositionen* Das umfasst übrigens auch die Konstanten $\top, \bot$! Nun kommen wir zu der Menge der Zustände Q.

- Gegeben eine LTL-Formel α definieren wir die Menge der Teilformeln als die Menge aller wohlgeformten Formeln, aus denen sich α zusammensetzt
- (für $(\Box p) \wedge q$ ist das z.B. $\{p, q, \Box p, (\Box p) \wedge q\}$);
- wir denotieren diese Menge mit $teil(\alpha)$.
- Die Zustandsmenge von $\mathfrak{A}_\alpha$ ist genau $teil(\alpha)$ – d.h. Teilformeln sind nun unsere Zustände!
- Zusätzlich nehmen wir noch die beiden Zustände $\top, \bot$ an.

Hier aber Vorsicht: Formeln haben natürlich eine Semantik; sobald wir aber eine Formel als einen Zustand auffassen, ist sie soz. atomar und als Name nur eine Konvention – allerdings ist diese Konvention sehr praktisch! Unsere Übersetzung ist nun ganz einfach; wir müssen nur noch δ definieren. Das machen wir induktiv. Wichtig: auf dieser Seite schreiben wir logische Formeln in einem anderen Schriftsatz, denn wir unterscheiden

1. Logische Formeln/Zustände, geschrieben in Blau wie $(\Box p) \vee q$. Man beachte dass die blauen Formeln hier bereits (atomare!) Zustände des Automaten sind.
2. das automatentheoretische $\vee$, wie gehabt in Schwarz. Wir sind ja im alternierenden Automaten!

Also: $p \vee q$ ist *ein* Zustand des Automaten, während $p \vee q$ eine Disjunktion ist zwischen *zwei* Zuständen im alternierenden Automat.

1. $\delta(\top, a) = \top$ (immer und allgemein)
2. $\delta(\bot, a) = \bot$ (immer und allgemein)
3. $\delta(FIN, a) = \bot$ (immer und allgemein)
4. $\delta(p, a) = \begin{cases} \top \text{ gdw. } p \in prop \text{ und } p \in a \\ \bot \text{ gdw. } p \in prop \text{ und } p \notin a \end{cases}$
5. $\delta(\neg p, a) = \begin{cases} \top \text{ gdw. } p \in prop \text{ und } p \notin a \\ \bot \text{ gdw. } p \in prop \text{ und } p \in a \end{cases}$
6. $\delta(\beta \wedge \gamma, a) = \delta(\beta, a) \wedge \delta(\gamma, a)$ (Achtung: induktive Definition!)
7. $\delta(\beta \vee \gamma, a) = \delta(\beta, a) \vee \delta(\gamma, a)$ (ebenso!)
8. $\delta(N\beta, a) = \beta$
9. $\delta(\Diamond\beta, a) = \beta \vee \Diamond\beta$
10. $\delta(\Box\beta, a) = (\beta \wedge \Box\beta) \vee FIN$
11. $\delta(\beta U \gamma, a) = \delta(\beta, a) \wedge (\gamma \vee \beta U \gamma)$ (ebenso!)
12. $\delta(\beta R \gamma, a) = \gamma \wedge (\beta \vee \beta R \gamma)$

Da die Übergänge teilweise induktiv definiert sind, könnte man den Eindruck haben, dass die Prozedur nicht immer endet. Allerdings kann man folgendes leicht sehen:

- wenn wir einen Übergang β, a, θ haben, θ eine positive Boolesche Formel über Zustände $\gamma_1, ..., \gamma_i$, dann sind $\gamma_1, ..., \gamma_i$ als Formeln gesehen kleiner als β –
- d.h. unsere induktive Prozedur resultiert zwangsläufig in einem endlichen Automaten!

Nachdem unsere Zustände Teilformeln von α (der Ausgangsformel) sind, die nach der induktiven Definition von δ immer kleiner werden, legen wir den **Startzustand** als α fest. Die **akzeptierenden Zustände** dagegen sind

1. $\top$, *FIN*
2. alle Zustände der Form: $\beta R \gamma$.
3. alle Zustände der Form: $\Box\beta$

Zur Erklärung:

- $\top$ ist klar;
- Zustände der Form $\Box\beta$ sind akzeptierend, weil sie 1. keine Bedingungen an die Gegenwart stellen, und 2. wenn es keine Nachfolger gibt trivial erfüllt werden (deswegen sind sie am Ende einer Eingabe immer erfüllt).
- Für $\beta R \gamma$ ist die Begründung ähnlich, nur muss man hier Klausel 12. der Übersetzung berücksichtigen um zu verstehen, warum $\beta R \gamma$ *als Zustand gesehen* für die Gegenwart

Für eine Formel α bezeichnen wir mit $|\alpha|$ die Anzahl der Konnektoren einen Formel Wir haben folgendes Ergebnis:

Lemma 6 $|teil(\alpha)| < 2(|\alpha| + 1)$.

Beweis. Induktion über Formelkomplexität: Basis ist klar. Nimm nun α, β, wobei $|teil(\alpha)| < 2(|\alpha| + 1)$, $|teil(\beta)| < 2(|\beta| + 1)$. Wir haben für $\star \in \{\wedge, \vee, \rightarrow\}$

(3) $\quad teil(\alpha \star \beta) = teil(\alpha) \cup teil\beta \cup (\alpha \star \beta)$

hence

(4) $\quad |teil(\alpha \star \beta)| = |teil(\alpha)| + |teil(\beta)| + 1$

Und da $|teil(\alpha)| < 2|\alpha|$, $|teil(\beta)| < 2|\beta|$. (es fehlen also mindestens 2, wegen <!), $|\alpha \star \beta| = |\alpha| + |\beta| + 1$, gilt

(5) $\quad |teil(\alpha \star \beta)| < 2(|\alpha \star \beta| + 1)$

Noch einfacher für unäre Konnektoren. $\dashv$

Lemma 7 *Für jede LTL-Formel α gibt es einen alternierenden Automaten $\mathfrak{A}$ so dass $|\mathfrak{A}| \leq 2|\alpha|$, und $w \in L(\mathfrak{A})$ genau dann wenn $w, 0 \models \alpha$.*

Übung

Zu bearbeiten bis zum 2.6.2020 (Abgabe *vor* dem Seminar!)

Konstruieren Sie den alternierenden Automaten $\mathfrak{A}_\alpha$ für folgende Formeln, also $\Sigma, Q, \delta, q_0, F$. Versuchen Sie die Konjunktion/Disjunktion des Automaten von den Konnektoren der Teilformeln notationell zu unterscheiden (farblich oder mit anderen Konventionen). Sie können den Automaten auch gerne zeichnen (Konvention aus der Sitzung)

1. $\alpha = (\Diamond(p_0 \vee p_1)) \wedge p_2$
2. $\alpha = (\Diamond((p \wedge q) U \neg r)) \vee \Box(p \vee r)$ [1]

[1] Siehe Kommentar

Musterlösung für 2 Wir haben Σ gegeben, $Q = teil(\alpha)$, $q_0 = \alpha$. Wir definieren als nächstes δ. Um den Unterschied zwischen automatentheoretischen und logischen Konnektoren zu verdeutlichen, nutzen wir blaue Farbe.

$$\begin{aligned}\delta(\alpha, a) &= \delta(\Diamond((p \wedge q)U\neg r))), a) \vee \delta(\Box(p \vee r), a) \\ &= ((p \wedge q)U\neg r) \vee \Diamond((p \wedge q)U\neg r)))) \vee ((p \vee r) \wedge (\Box(p \vee r))) \vee FIN\end{aligned}$$

$$\delta(p \wedge q, a) = \begin{cases} \top, \text{ falls } \{p, q\} \subseteq a \\ \bot \text{ andernfalls} \end{cases}$$

$$\delta(p \vee r, a) = \begin{cases} \top, \text{ falls } \{p, q\} \cap a \neq \emptyset \\ \bot \text{ andernfalls} \end{cases}$$

$$\delta(\neg r, a) = \begin{cases} \top, \text{ falls } r \notin a \\ \bot \text{ andernfalls} \end{cases}$$

$$\delta(\Diamond((p \wedge q)U\neg r)), a) = (p \wedge q)U\neg r \vee \Diamond((p \wedge q)U\neg r))$$

$$\delta(\Box(p \vee r), a) = (p \vee r \wedge \Box(p \vee r)) \vee FIN$$

$$\begin{aligned}\delta((p \wedge q)U\neg r, a) &= \delta(p \wedge q, a) \wedge (\neg r \vee (p \wedge q)U\neg r)) \\ &= \begin{cases} \top \wedge (\neg r \vee (p \wedge q)U\neg r)) & \text{, falls } \{p, q\} \subseteq a \\ \bot \wedge (\neg r \vee (p \wedge q)U\neg r)) & \text{, andernfalls} \end{cases} \\ &= \begin{cases} \neg r \vee ((p \wedge q)U\neg r) & \text{, falls } \{p, q\} \subseteq a \\ \bot & \text{, andernfalls} \end{cases}\end{aligned}$$

8.4 Alternierende Büchi-Automaten

Im Normalfall interessieren uns bei LTL alternierende Büchi-Automaten, wir haben nur Einfachheit halber den endlichen Fall betrachtet. Um alternierende Büchi-Automaten zu verstehen, brauchen wir wieder den Begriff des **Laufes**, der eigentlich ein **Baum** ist (wenn wir nur die Zustände betrachten):

$$
\begin{array}{c}
q_0 \\
a_1 \\
q_{00}\ ,\ ,\ldots,\ q_{0i} \\
a_2 \\
q_{000}\ ,\ ,\ldots,q_{00k}\ ,\ ,\ldots,\ q_{0i0}\ ,\ ,\ldots,q_{0ij} \\
a_3 \\
\ldots
\end{array}
$$

Im endlichen Fall konnten wir gleiche Zustände auf derselben Ebene (also nach derselben Eingabe) kollabieren lassen, so dass wir letzten Endes einen Lauf hatten der Form

$$
\begin{array}{c}
\{q_0\} \\
a_1 \\
\{q_{00}\ ,\ ,\ldots,\ q_{0i}\} \\
a_2 \\
\{q_{000}\ ,\ ,\ldots,q_{00k}\ ,\ ,\ldots,\ q_{0i0}\ ,\ ,\ldots,q_{0ij}\} \\
a_3 \\
\ldots
\end{array}
$$

Die Bedingung der Akzeptanz war: alle Zustände der Menge müssen akzeptierend sein. Im unendlichen Fall (Büchi) geht das **nicht mehr**.

Definition 8 *Ein Lauf in einem alternierenden Büchi-Automaten ist ein unendlicher Baum, wie oben definiert. Ein Lauf ist akzeptierend, falls* ***jeder Pfad*** *in diesem Baum* ***unendlich viele*** *akzeptierende Zustände enthält.*

Wir müssen also die Pfade separat betrachten: die Bedingung bedeutet nicht, das wir unendlich oft *ausschließlich* in akzeptierenden Zuständen sind, sondern ist deutlich schwächer. Als ein Beispiel nimm

(6) $\mathfrak{A}_{\Box\Diamond p}$

- wir sind jederzeit in Zustand $\Diamond p$, welcher nicht akzeptiert, aber
- wenn wir die Läufe als Baum aufmalen und die Eingabe ∞ viele p enthält, dann ist jeder Pfad ∞ oft im akzeptierenden Zustand (entweder $\top$ oder $\Box\Diamond p$)

9 Sternfreie Sprachen

Wir werden jetzt eine neue Klasse von Sprachen einführen, die echt in den regulären Sprachen enthalten ist, nämlich die sog. **sternfreien** Sprachen. Diese Klasse ist eine relativ große Teilklasse der regulären Sprachen, und meines Wissens nach die größte die allgemein geläufig und "natürlich" ist. Was "natürlich" in der Theorie der formalen Sprachen bedeutet ist: es gibt mehrere Charakterisierungen (Automaten, algebraische Ausdrücke, Grammatiken, Logiken), die nicht offensichtlich äquivalent sind, und alle unabhängig voneinander diese Klasse charakterisieren. In diesem Sinne sind die regulären Sprachen eine sehr natürliche Klasse: wir haben eine ganze Reihe von Charakterisierungen besprochen und dabei sogar noch einige wichtige ausgelassen (algebraische und logische). Auch die sternfreien Sprachen haben eine ganze Reihe von Charakterisierungen, von denen wir nur zwei relativ naheliegende besprechen werden.

9.1 Sternfreie Ausdrücke

Der Ausdruck "sternfrei" kommt daher, dass alle regulären Ausdrücke, die keinen Kleene-Stern enthalten, sternfrei sind. Das Gegenteil ist aber nicht richtig: wir brauchen noch zusätzliche Konstruktoren - sonst könnten wir ja nur die endlichen Sprachen denotieren!

Wir haben zwei neue Konstruktoren für sternfreie Ausdrücke:

- eine Konstante $\perp$, deren Syntax gleich der von Buchstaben $a \in \Sigma$ ist, und deren Semantik definiert ist durch $\| \perp \| := \emptyset$, und
- einen unären Operator$\overline{[-]}$, dessen Semantik gegeben ist durch $\|\overline{\alpha}\| := \Sigma^* - \|\alpha\|$.

Wir können also die leere Sprache explizit denotieren, und das Komplement bilden. Wir zeigen einige Beispiele.

Die Sprache Σ^* wird denotiert von $\overline{\perp}$. Die Sprache $(ab)^*$ hat eine etwas kompliziertere sternfreie Charakterisierung:

$$(7) \quad \overline{(\overline{\perp}a a\overline{\perp}) + (\overline{\perp}b b\overline{\perp}) + (b\overline{\perp}) + (\overline{\perp}a)}$$

Wenn wir diesen Ausdruck in Worte fassen, dann bekommen wir: die Sprache über $\{a, b\}$, in der niemals aa, bb vorkommt, kein Wort mit b anfängt

oder mit a aufhört. Das Beispiel zeigt bereits warum sich die Popularität der sternfreien Ausdrücke in Grenzen hält: sie sind meist wesentlich kompliziert als die regulären. Die algebraische Charakterisierung macht einige Eigenschaften sternfreier Sprachen ziemlich offensichtlich:

Satz 9 *Sternfreie Sprachen sind abgeschlossen unter Vereinigung, Komplement und Schnitt.*

Vereinigung und Komplement qua definition sternfreier Ausdrücke, Schnitt folgt aus mengentheoretischen Gründen. Ein Kommentar noch: wem nicht sofort klar ist, dass die sternfreien Sprachen in den regulären Sprachen enthalten sind, der sollte sich klar machen dass aus der Tatsache, dass reguläre Sprachen unter Komplement abgeschlossen sind, folgt, dass ein Komplementoperator keine zusätzliche Ausdrucksstärke gibt für reguläre Ausdrücke. Die leere Menge kann ebenfalls regulär denotiert werden mit dem leeren Ausdruck. Der Grund warum wir diese konstante brauchen für sternfreie Ausdrücke ist: wir haben sonst keine Möglichkeit, Σ^* zu denotieren, denn wir haben keinen Stern, und können ja nicht das Komplement des leeren Ausdrucks bilden!

9.2 Zählerfreie Automaten

Für andere Eigenschaften sternfreier Sprachen sind sternfreie Ausdrücke eher wenig informativ. Z.B. ist $(ab)^*$ sternfrei, aber $(aa)^*$ nicht. Warum ist das so? Diese etwas merkwürdige Eigenschaft wird verständlicher, wenn wir uns eine weitere wichtige Charakterisierung sternfreier Sprachen anschauen:

Definition 10 *Sei $\mathfrak{A} = (\Sigma, Q, q_0, F, \delta)$ ein endlicher Automat, wobei $\hat{\delta}$ die Generalisierung von δ auf Ketten ist. $\mathfrak{A}$ ist **zählerfrei**, falls gilt: für alle $w \in \Sigma^*, q \in Q, n \geq 1$, falls $\hat{\delta}(q, w^n) = q$, dann ist $\delta(q, w) = q$.*

Was besagt diese Bedingung? In Worten: wenn wir in Zustand q sind, ein Wort w n-mal lesen, und damit nach q zurückkehren, dann kehren wir auch nach q zurück nachdem wir es *einmal* gelesen haben. Was wir also beschränken sind die *Zyklen* in unserem Automaten. Beispielsweise: nehmen wir an $\mathfrak{A}$ ist ein zählerfreier Automat, der $(aa)^*$ erkennt. Wir werden dann (falls er minimal ist) finden dass er nach der Eingabe aa im selben Zustand ist, wie am Anfang, nämlich q_0. Da er aber zählerfrei ist, folgt daraus: er ist auch nachdem er a gelesen hat in q_0! Das bedeutet aber wiederum: entweder

er akzeptiert *aa nicht*, oder er akzeptiert auch a - in jedem Fall erkennt er nicht die Sprache $(aa)^*$!

Dieses Argument ist leider noch unsauber: denn es kann ja einen anderen, viel größeren Automaten geben, der $(aa)^*$ erkennt und zählerfrei ist. Allerdings können wir das Argument, dass wir oben für $n = 2$ geführt haben, für jedes n führen: irgendwann (nach a^k) haben wir sicher einen Zyklus, und dann gilt: wir akzeptieren auch a^{k+1}. Ein berühmter Satz von Marcel Paul Schützenberger sagt nun:

Satz 11 *Eine Sprache L ist genau dann sternfrei, wenn es einen zählerfreien Automaten $\mathfrak{A}$ gibt, so dass $L = L(\mathfrak{A})$.*

Das, zusammen mit unseren vorigen Überlegungen, führt uns direkt zum **Pumplemma der sternfreien Sprachen**:

Lemma 12 *Eine Sprache $L \subseteq \Sigma^*$ ist genau dann sternfrei, wenn für alle $y \in \Sigma^+$ es ein $k \in \mathbb{N}$ gibt, so dass f.a. $x, z \in \Sigma^*$ gilt: falls $xy^{k'}z \in L$ für ein $k' \geq k$, dann ist $xy^n z \in L$ für alle $n \in \mathbb{N}$.*

Man beachte: auf den ersten Blick gibt es eine Ähnlichkeit zu den regulären Sprachen. Der Unterschied ist: bei den regulären Sprachen sagen wir: *es gibt* eine Zerlegung; hier sagen wir: *für alle Zerlegungen*. Der Unterschied liegt also in der Quantifikation. Man beweist das Pumplemma der sternfreien Sprachen recht einfach aus dem Satz von Schützenberger: gegeben einen zählerfreien Automaten für L, wählen wir k so, dass es den kleinsten Zyklus beschreibt (wir kommen zum zweiten Mal in denselben Zustand). Da der Automat endlich ist, gibt es so ein k.

Das können wir benutzen, um zu zeigen dass gewisse Sprachen nicht sternfrei sind: nehmen wir $L = (aa)^*$, und k eine beliebige Zahl. Dann ist entweder (i) $aa^k \in L$ oder (ii) $a^k \in L$ (je nachdem ob k gerade oder ungerade ist). Daraus folgt qua Pumplemma, dass im Fall (i) $aa^2 \in L$ - Widerspruch; oder im Fall (ii) $a^1 \in L$ - Widerspruch. Ein zweites Beispiel ist folgendes: sei $L_1 \subseteq \{a, b\}^*$, $L_1 := \{w : |w|_a \text{ gerade}\}$. Sei k beliebig. (i) k ist gerade. Dann ist $a^k \in L_1$; also nach Pumplemma auch $a \in L$ - Widerspruch. (ii) k ist ungerade. Dann ist $aa^k \in L_1$, also nach Pumplemma auch aa^2 - Widerspruch. Also ist das Pumplemma nicht erfüllt. Ein Nachtrag noch: während das Pumplemma für reguläre Sprachen notwendig ist als Kriterium, aber nicht hinreichend (es gibt nicht-reguläre Sprachen die das Lemma erfüllen,

wie etwa $\{w \in \{a,b\}^* : |w|_a = |w|_b\}$, ist das sternfreie Lemma notwendig und hinreichend: wenn eine Sprache es erfüllt, dann ist sie sternfrei. Wir können es also auch benutzen um zu zeigen dass Sprachen sternfrei sind.

Die obigen Beispiele machen klar, wie zählerfreie Automaten zu ihrem Namen gekommen sind: was endliche Automaten können ist: zählen modulo irgendeine Zahl; d.h. sie können berechnen ob die Länge eines Wortes oder die Anzahl eines gewissen Buchstaben in einem Wort durch zwei/drei/vier... teilbar ist. Zählerfreie Automaten können genau dass nicht, oder nur in sehr beschränkten Fällen.

9.3 Eine weitere Eigenschaft

Sternfreie Sprachen haben auch eine logische Charakterisierung in $FO[<]$, erststufiger Prädikatenlogik in der Signatur $[<]$ (lineare Ordnung). Wir können hier nicht auf diese Charakterisierung eingehen, allerdings lässt sich aus einem wichtigen Ergebnis der endlichen Modelltheorie (den sog. 0-1 Gesetzen) folgendes interessante Ergebnis ableiten. Wir definieren $\Sigma^{\leq n} := \{w \in \Sigma^* : |w| \leq n\}$, und für $L \subseteq \Sigma^*$, $L^{\leq n} := L \cap \Sigma^{\leq n}$. Wir sagen eine Sprache ist **dicht**, falls

$$(8) \quad lim_{n\to\infty} \frac{L^{\leq n}}{\Sigma^{\leq n}} = 1;$$

eine Sprache ist **dünn**, falls

$$(9) \quad lim_{n\to\infty} \frac{L^{\leq n}}{\Sigma^{\leq n}} = 0.$$

Satz 13 *Für jede sternfreie Sprache L gilt: L ist dünn, oder L ist dicht.*

Der Beweis würde uns viel zu weit abführen. Wir können aber zeigen dass dieser Satz für die regulären Sprachen bereits nicht mehr richtig ist: wir nehmen die reguläre Sprache $L_1 := \{w : |w|_a \text{ gerade}\}$; eine einfache Induktion zeigt uns:

$$(10) \quad lim_{n\to\infty} \frac{(L_1)^{\leq n}}{\Sigma^{\leq n}} = \frac{1}{2}.$$

Auch dieses Argument kann also benutzt werden, um zu zeigen dass eine Sprache wie $(aa)^*$ nicht sternfrei ist. *Aufgabe*: warum? Benutzen Sie Satz 50 um zu zeigen, dass $(aa)^*$ nicht sternfrei ist! Aber wohlgemerkt: das ist kein hinreichendes Kriterium um zu zeigen dass eine Sprache sternfrei ist: $\{a^n b^n : n \in \mathbb{N}\}$ ist zwar dünn, aber noch nicht einmal regulär.

9.4 Sternfreie Sprachen und Logik

Nimm an, unser Vokabulär (die Prädikate) sieht aus wie folgt:

$$(\leq, P_a : a \in \Sigma)$$

Wir haben also ein binäres (Ordnungs-) und n unaäre Prädikate, jedes entspricht einem Buchstaben. Wir nehmen weiterhin an, unsere Modelle sind **endliche lineare Ordnungen** (nach $\leq$; lineare Ordnungen sind axiomatisierbar in unserer Sprache, Endlichkeit nicht!) Dann brauchen wir noch folgende Axiome:

$\forall x. \bigvee_{a\in\Sigma} P(a)$ (überall gilt ein Buchstabe)
$\forall x. \bigwedge_{a\in\Sigma} P_a \rightarrow \neg \bigvee_{b\neq a} P_b(x)$ (und maximal einer)

Damit stellen wie sicher, dass jedes Modell als ein Wort interpretiert werden kann, und eine (zusätzliche geschlossene) Formel entspricht – Menge von Modellen – einer Sprache. Z.B.:

$$\begin{aligned} & \forall xy.(P_a(x) \wedge P_b(y)) \rightarrow x < y \\ \wedge\ & \forall x.P_b(x) \rightarrow \neg\exists y.x < y \\ \wedge\ & \exists x.P_b(x) \end{aligned}$$

Diese Formel axiomatisiert die Sprache a^*b! (Warum?) Folgendes Ergebnis ist interessant:

Theorem 14 *(Informell) Eine Sprache ist axiomatisierbar in Prädikatenlogik (mit obigem Vokabular und Modellen) gdw. sie sternfrei ist.*

Sternfrei entspricht also Prädikatenlogik! Wenn wir mehr wollen (z.B. regulär), dann brauchen wir **monadische zweitstufige Logik**, die zusätzlich Variablen für Prädikate (nicht nur Individuen) hat. Dasselbe Ergebnis gilt übrigens auch für unsere Modallogik:

Theorem 15 *(Informell) Eine Sprache ist axiomatisierbar in LTL gdw. sie sternfrei ist.*

Das Axiom für a^*b sind dann wie folgt: wir nehmen an es gibt eine propositionale Varibable pro Buchstabe in Σ, und bekommen dann:

$$
\begin{aligned}
&(a \vee b) \wedge \neg(a \wedge b) \\
\wedge\ &(a \to \Diamond b) \\
\wedge\ &(b \to \neg\Diamond a) \\
\wedge\ &(b \to \neg\Diamond\top) \\
\wedge\ &((\Diamond\top) \to \Diamond b))
\end{aligned}
$$

Wenn das unsere Formel α ist, dann wäre das Axiom wie immer $\alpha \wedge \Box\alpha$.

Übung

Zu bearbeiten bis zum 20.12. (Abgabe *vor* dem Seminar!)

Schreiben Sie folgende regulären Ausdrücke als sternfreie Ausdrücke: 1. $(a(b+c))^*$, 2. $(bab)^*$, 3. a^*b, 4. $((ab)^*c) + ((ab)^*ad)$.

9.5 Zählerfreie Automaten und LTL

Wir haben eben beobachtet dass wir aus einem Zustand β nur dann in einen Zustand γ kommen, wenn $|\gamma| \leq |\beta|$ (nicht die Konnektoren des alternierenden Automaten mit genuinen LTL-Formeln verwechseln!). Wir können das noch stärken, indem wir sagen:

$\Rightarrow$ Falls wir aus Zustand β einen Zustand γ erreichen, muss γ eine Teilformel sein von β (möglicherweise $\gamma = \beta$).

Daraus folgt, dass unser Automat $\mathfrak{A}_\alpha$ **zählerfrei** ist, d.h.

$$\hat{\delta}(q, ww) = q \Longrightarrow \delta(q, w) = q$$

– in der Tat müssen alle Zwischenschritte ebenfalls q sein, sonst kämen wir nie wieder nach q zurück!

Zählerfreie Automaten charakterisieren sternfreie Sprachen – daraus folgt also ein weiteres wichtiges Ergebnis für uns:

Lemma 16 *Sei α eine LTL-Formel. Dann erkennt $\mathfrak{A}_\alpha$ eine sternfreie Sprache.*

Wenn wir die andere Richtung zeigen wollen (für jede sternfreie Sprache L gibt es eine Formel α_L, müssen wir zeigen wie wir aus einem zählerfreien Automaten eine Formel konstruieren.

Aufgabe 3

Zu bearbeiten bis zum ... (Abgabe *vor* dem Seminar!)

Finden Sie LTL-Formeln α, γ so dass gilt: jedes unendliche Modell von α ist ein Modell von γ, **aber**: es gibt ein endliches Modell von α dass *kein* Modell von γ ist!

Aufgabe 4

Zu bearbeiten bis zum 21.5.18 (Abgabe *vor* dem Seminar!)

Schreiben Sie Büchi-Automaten für die folgenden ω-Ausdrücke. Wichtig: für einen Ausdruck gilt: die Denotation von A^ω ist die Menge

$$\{w_1 w_2 w_3 .... : w_i \text{ wird denotiert von } A, \text{ für alle } i \in \mathbb{N}\}$$

1. $(a^* bab)^\omega$
2. $b^*(ab)^\omega$
3. $a(b(c^*))^\omega$

Graphische Darstellung genügt, aber bitte eindeutig und deutlich!

10 Ein Anwendungsfall für LTL – Verifizierung

10.1 Eine konkrete Anwendung: eine Fabrik

Wir nehmen mal ein ganz konkretes Szenario. Wir haben eine Rakete mit 2 Triebwerken, und 3 Kühlungen. Welches Triebwerk wann läuft, hängt von externen Faktoren ab, Steuerkommandos etc.

Um einen sicheren Flug zu gewährleisten, ist aber folgendes zu beachten:

- Wenn 2 Triebwerke laufen, brauchen wir 3 Kühlungen
- Wenn 1 Triebwerk läuft, brauchen wir mind. 1 Kühlung
- Aber: Wenn 2 Triebwerke laufen, eine abgeschaltet wird, brauchen wir 2 Stunden lang mind. 2 Kühlungen

Diese Bedingungen können wir leicht in eine LTL-Formel α übersetzen (mit Stunden als Zeiteinheiten). Hier hilft uns die deklarative Natur der Logik.

Das sind aber nur Bedingungen, keine effektive Steuerung der Maschinen. Wir können uns nun eine solche Steuerung vorstellen als einen Automaten. Dieser Automat reagiert auf bestimmte (externe) Ein- und Ausgaben mit bestimmten Befehlen.

Die Frage ist: erfüllt unsere Steuerung $\mathfrak{A}$ die Formel α? Das lässt sich nun leicht in ein formales Problem übersetzen, nämlich: ist

(11) $L(\mathfrak{A}) \subseteq \{w : w \models \alpha\}$?

oder äquivalent:

(12) $L(\mathfrak{A}) \subseteq L(\mathfrak{A}_\alpha)$?

Diese Frage können wir, wie bereits besprochen, beantworten. Es ist also entscheidbar, inwieweit die Steuerung unsere Bedingungen erfüllt.

NB: solche Szenarien sind durchaus nicht unrealistisch. Man benutzt Verifizierung für sehr sensible Systeme, wir Atomkraftwerke oder Raketen.

Übung Wie sieht α aus? $\beta \equiv (k_1 \wedge k_2) \vee (k_1 \wedge k_3) \vee (k_2 \wedge k_3)$ (mind. 2 L.)

$$\begin{array}{lll} \square & & ((t_1 \wedge t_2) \rightarrow (k_1 \wedge k_2 \wedge k_3) \\ & \wedge & (t_1 \vee t_2) \rightarrow (k_1 \vee k_2 \vee k_3) \\ & \wedge & (t_1 \wedge t_2) \rightarrow ((N\beta) \wedge (NN\beta))) \text{ Tadaa!} \end{array}$$

Aber Wir sind davon ausgegegangen, dass unsere Steuerungen (deterministisch) auf externe Eingaben reagiert. Unsere Bedingungen dagegen sind rein intern formuliert. Die obigen Gleichungen vergleichen also Äpfel und Birnen. Wie kann man das Problem lössen?

10.2 Ein- und Ausgaben

Normalerweise interessiert uns nicht die Sprache als die Menge der akzeptierten Eingaben im engeren Sinn; ein Programm sollte im Idealfall jede Eingabe verarbeiten. Wir geben hier zwei Beispielfälle an, nämlich

1. die Eingabe ist endlich, und der Automat liefert am Ende eine Ausgabe (aus einer endlichen Menge O)
2. die Eingabe ist unendlich, und der Automat liefert für jeden Eingabebuchstaben $a \in \Sigma$ eine Ausgabe $o \in O$.

In beiden Fällen ist die akzeptierte Eingabesprache also erstmal Σ^*, denn jede Eingabe wird verarbeitet, und der Automat berechnet eigentlich eine Relation. Wir werden aber sehen, dass beide Fälle auf den Fall, den wir oben besprochen haben, sehr einfach reduziert werden kann.

Fall 1: endliche Eingabe, eine Ausgabe In diesem Fall erkennt unser Automat eine Relation $R \subseteq \Sigma^* \times O$. R ist allerdings sehr offensichtlich äquivalent zu einer Sprache

$$L_R := \{wo : (w, o) \in R\}$$

Weiterhin gilt offensichtlich: falls R von einem endlichen Automaten berechnet werden kann, dann gilt das auch für L_R. Dasselbe gilt für die Familie von Sprachen $L_o : o \in O$, definiert durch

$$L_o := \{w : (w, o) \in R\}$$

Das heißt: wir können in diesem Fall verifizieren, ob eine Eingabe, damit sie die Ausgabe o erhält (oder nicht erhält), gewisse LTL-Bedingungen erfüllt.

Fall 2: unendliche Eingabe, unendliche Ausgabe In diesem Fall erkennt unser Automat eine Relation

$$R \subseteq \Sigma^\omega \times O^\omega$$

Natürlich können wir hier nicht dieselbe Technik wie vorher anwenden; das Ergebnis wäre kein ω-Wort. Stattdessen nehmen wir ein neues Alphabet $\Sigma \times O$, und definieren

$$L_R \subseteq (\Sigma \times O)^\omega$$

Vorsicht: das setzt wirklich voraus, dass jeder Eingabe genau eine Ausgabe entspricht; aber dann gilt: wenn ein endlicher Automat die Relation R berechnet, dann gibt es auch einen endlichen Automaten, der die Sprache L_R erkennt, definiert durch

$$L_R := \{(a_1, o_1)(a_2, o_2)(a_3, o_3), ... : (a_1 a_2 a_3 ..., o_1 o_2 o_3 ...) \in R\}$$

Man kann also auch sicherstellen, dass die Eingabe-Ausgaberelation beliebige LTL-Bedingungen erfüllt.

Zurück zum Beispiel Auf unser obiges Beispiel kann man also annehmen, dass jede externe Eingabe ein Symbol hat, und es ein zusätzliches Füller-Symbol gibt $\odot$, welches besagt dass es keine neue Eingabe gibt.

10.3 Eine abstrakte Anwendung (Computerprogramme als Automaten)

Normalerweise kann man ein Computerprogramm als eine Art von Automaten auffassen: ein Zustand beschreibt den aktuellen Zustand des Programms, d.h. die Belegungen aller Variablen, der Zustand aller Zähler etc. Natürlich kann es oftmals vorkommen, das ein solches Programm theoretisch unendlich ist (nimm eine `while`-Schleife). In der Praxis wird das allerdings unmöglich sein; deswegen kann man sich im Prinzip *jedes* Programm als einen endlichen Automaten vorstellen.

Der Computer selbst ist ja endlich und kann nur in einem von endlich vielen Zuständen sein!

Nehmen wir also einmal an, wir haben einen solchen Automaten $\mathfrak{A}$. Uns interessiert die Frage:

Erfüllt $\mathfrak{A}$ die Bedingung α?

α kann alles mögliche sein: eine Sicherheitsbedingung, eine Lebendigkeitsbedingung, oder einfach ein gewisser Zustand soll erreichbar sein.

Wie können wir unsere Frage beantworten? Hier helfen uns eine Reihe von Tatsachen:

1. Gegeben zwei Automaten $\mathfrak{A}_1, \mathfrak{A}_2$ können wir einen Automaten $\mathfrak{A}_3$ konstruieren, so dass $L(\mathfrak{A}_1) \cap L(\mathfrak{A}_2) = L(\mathfrak{A}_3)$
2. Gegeben einen Automaten $\mathfrak{A}$ können wir entscheiden, ob $L(\mathfrak{A}) = \emptyset$
3. Daraus folgt: gegeben zwei Automaten $\mathfrak{A}_1, \mathfrak{A}_2$ können wir entscheiden ob $L(\mathfrak{A}_1) \subseteq L(\mathfrak{A}_2)$.

Das reicht bereits an dieser Stelle; nun müssen wir uns Gedanken machen, was es (automatentheoretisch) genau bedeutet, dass $\mathfrak{A}$ die Bedingung α erfüllt. Hier hilft uns folgende Überlegung: d

- der Automat steht für eine Menge von Worten;
- jedes Wort steht für ein LTL-Modell (endlich oder unendlich)

Was bedeutet das also? Unser Automat erfüllt die Spezifizierung α gdw. $L(\mathfrak{A}) \subseteq L(\mathfrak{A}_\alpha)$ – eine Frage, die effektiv beantwortet werden kann! NB: wir verlangen hier keinesfalls, dass $\mathfrak{A}$ ein zählerfreier Automat ist; $\mathfrak{A}_\alpha$ wird zählerfrei sein, aber dass spielt für unsere zentrale Frage keine Rolle!

11 "Die Welt weiß nicht mehr wie die Zeit verzweigt" – Computational Tree Logic

11.1 Syntax

Computational tree logic (CTL) ist eine Logik, mit der man über Zeit spricht, wobei man aber die Zeit als einen Baum auffasst. Die Verzweigungen in einem Baum entsprechen dabei verschiedenen Möglichkeiten für die Zukunft. Man redet dabei normalerweise nur über unendliche Bäume. Es gibt auch eine alternative Semantik für CTL, die wir aber zunächst hintanstellen. Wir kommen zunächst zur Syntax von CTL. Diese ist sehr ähnlich zu LTL, hat aber zusätzlich noch sog. **Pfadquantoren**, die in ihrem Auftreten aber beschränkt sind.

1. p ist eine CTL-Formel, für $p \in prop$
2. Falls α, β CTL-Formeln sind, dann sind auch $\alpha \wedge \beta, \alpha \vee \beta, \alpha \rightarrow \beta, \neg\alpha$ CTL-Formeln
3. Falls α eine CTL-Formel ist, dann sind $EN\alpha, AN\alpha$ CTL-Formeln
4. Falls α eine CTL-Formel ist, dann sind $E\Diamond\alpha, A\Diamond\alpha, E\Box\alpha, A\Box\alpha$ CTL-Formeln
5. Falls α, β CTL-Formeln sind, dann sind $E(\alpha U\beta), A(\alpha U\beta)$ CTL-Formeln
6. Falls α, β CTL-Formeln sind, dann sind $E(\alpha R\beta), A(\alpha R\beta)$ CTL-Formeln

Es ist also sehr leicht zu sehen dass jeder Modalität M in LTL jeweils *zwei* Modalitäten in CTL entsprechen, nämlich EM und AM. Die Intuition hierhinter ist folgende: A und E sind für sich genommen **Pfadquantoren**, wobei

- E soviel bedeutet wie "entlang eines Pfades",
- A soviel wie "entlang aller Pfade",

wobei sich Pfade eben auf den Baum beziehen, der das Modell darstellt – die Zeit ist ja nicht mehr linear, sondern verzweigt sich in eine Vielzahl von Pfaden! Aber diese intuitive Erklärung ist mit Vorsicht zu genießen: in CTL können A und E niemals für sich allein verwendet werden, nur zusammen mit einer Modalität, also haben sie auch streng genommen keine eigene Semantik.

11.2 Über Bäume

Ein **Baum** ist für uns eine Struktur $\mathcal{T} = (T, <)$, wobei gilt:

1. $<$ ist transitiv,
2. irreflexiv (d.h. niemals $x < x$),
3. antisymmetrisch: falls $x < y$, gilt $y \not< x$,
4. für alle $x \in T$ gilt: die Menge $\{y : y \leq x\}$ ist *linear geordnet* nach $<$ (hier ist $x \leq y$ eine Abkürzung für $x < y$ oder $x = y$), und
5. es gibt ein $r \in T$ so dass $r \leq x$ für alle $x \in T$.

Dieses r nennen wir die **Wurzel** des Baumes. Wenn es ein Objekt $x \in T$ gibt für dass es kein $x' \in T$ gibt so dass $x < x'$, dann nennen wir x ein **Blatt** in $\mathcal{T}$. Sie sind vielleicht daran gewöhnt dass Bäume Blätter haben, wir werden aber hier normalerweise Bäume *ohne* Blätter betrachten, d.h. es gibt keine maximalen Elemente, der Baum wächst einfach immer weiter.

Ein wichtiger Begriff eines Baumes ist der eines **Pfades**. Ein Pfad in $\mathcal{T}$ ist für uns an dieser Stelle (es gibt manchmal andere Verwendungen) eine Menge $M \subseteq T$, so dass gilt:

1. M ist linear geordnet nach $<$,
2. M hat ein $<$-minimales Element x und
3. falls $M \subseteq M'$, M' ebenfalls linear geordnet nach $<$ und x minimal sowohl in M' als auch in M, dann ist $M = M'$.

Pfade sind also **maximale linear geordnete Teilmengen des Baumes mit einem gegebenen minimalen Element**. Man sagt auch: M ist eine x-Pfad, falls x minimal ist in M.

In endlichen Bäumen sind das einfach alle Knoten die zwischen einem bestimmten Knoten und einem Blatt liegen (inklusive der beiden); aber für unendliche Bäume ist die Definition etwas komplizierter (es gibt ja nicht immer Blätter).

Ein weiterer wichtiger Begriff (nicht nur für Bäume) ist der der **Wohlfundiertheit**. Ein Baum $(T, <)$ (allgemeiner: eine partielle Ordnung) ist wohlfundiert, falls für alle $x \in T$ gilt: die Menge $\{y : y < x\}$ ist endlich. Jede

endliche Ordnung ist wohlfundiert, es gibt aber durchaus auch unendliche wohlfundierte Strukturen, man denke nur an die lineare Ordnung $(\mathbb{N}, <)$ (jede lineare Ordnung ist auch ein Baum). $(\mathbb{R}_0^+, <)$ (die positiven reellen Zahlen mit 0) ist ein Baum, aber nicht wohlfundiert.

Für wohlfundierte Bäume gibt es normalerweise eine recht praktische Darstellung. Wir haben Bäume einfach als eine Menge mit einer Relation $<$ aufgefasst (die man normalerweise als "Dominanz" bezeichnet). D.h. wir können den Knoten beliebige Namen geben – diese haben weiter keine Bedeutung als die Identität festzulegen (also gleicher Namen $\cong$ gleicher Knoten). Es gibt aber auch eine Art der kanonischen Namensgebung, wie man es mit sog. **Baumdomänen** macht.

- Sei Σ ein Alphabet,
- und $\prec$ eine lineare Ordnung auf Σ

(man setzt daher oft $\Sigma = \{0, 1, 2, .., 9\}$. Eine Baumdomane ist definiert als eine Menge $\mathbf{B} \subseteq \Sigma^*$, so dass gilt:

1. $\epsilon \in \mathbf{B}$,
2. falls $wbv \in \mathbf{B}$ ($w, v \in \Sigma^*$, $b \in \Sigma$), und $a \prec b$, dann ist $wa \in \mathbf{B}$,
3. falls $wv \in \mathbf{B}$, dann ist $w \in \mathbf{B}$.

Die Dominanzrelation $<$ auf einer Baumdomäne $\mathbf{B}$ ist nun einfach die Relation *pref*, wobei *pref*(w, v) soviel bedeutet wie: w ist ein Präfix von v, anders gesagt: $v = wu$ für ein u.

11.3 Bäume als CTL-Modelle

Ein CTL-Modell ist Tupel $(T, <, r, val)$, wobei $(T, <)$ ein Baum ist, r seine Wurzel, und $val : T \to \wp(prop)$ die übliche Valuation, die uns sagt welche atomaren Propositionen an welchem Punkt wahr sind. Wir gehen davon aus, dass

1. $(T, <)$ wohlfundiert ist, und
2. für alle $x \in T$ gibt es ein y so dass $x < y$ (keine Blätter)

– es gibt also keine Blätter, da jeder Knoten einen Nachfolger hat ($<$ ist irreflexiv!) Da der Baum wohlfundiert ist, ist er auch diskret, d.h. es gibt für jeden Knoten einen **unmittelbaren** Nachfolger. Wir denotieren diese Relation mit $<_u$, d.h.

$$x <_u y \Leftrightarrow x < y \wedge \neg\exists z.x < z < y$$

Wir setzen nun fest, wie wir CTL-Formeln in Modellen interpretieren; es gilt $x \in T$.

$(T, <, r, val), x \models p$ gdw. $p \in val(x)$

$(T, <, r, val), x \models \alpha \wedge \beta$ gdw. $(T, <, r, val), x \models \alpha$ und $(T, <, r, val), x \models \beta$

$(T, <, r, val), x \models \alpha \vee \beta$ gdw. $(T, <, r, val), x \models \alpha$ oder $(T, <, r, val), x \models \beta$

$(T, <, r, val), x \models \neg\alpha$ gdw. $(T, <, r, val), x \not\models \alpha$

$(T, <, r, val), x \models \top$

$(T, <, r, val), x \not\models \bot$

$(T, <, r, val), x \models EN\alpha$ gdw. es ein $y \in T$ gibt so dass $x <_u y$ und $(T, <, r, val), y \models \alpha$

$(T, <, r, val), x \models AN\alpha$ gdw. für alle $y \in T$ gilt: falls $x <_u y$, dann $(T, <, r, val), y \models \alpha$

$(T, <, r, val), x \models E\Box\alpha$ gdw. es einen x-Pfad $P \subseteq T$ gibt s.d. für alle $y \in P - \{x\}$ gilt: $(T, <, val), y \models \alpha$

$(T, <, r, val), x \models A\Box\alpha$ gdw. für alle x-Pfade $P \subseteq T$ und $y \in P - \{x\}$ gilt: $(T, <, val), y \models \alpha$

$(T, <, r, val), x \models E\Diamond\alpha$ gdw. es einen x-Pfad $P \subseteq T$ und ein $y \in P - \{x\}$ gibt s.d. $(T, <, val), y \models \alpha$

$(T, <, r, val), x \models A\Diamond\alpha$ gdw. für alle x-Pfade $P \subseteq T$ ein $y \in P - \{x\}$ gibt, so dass $(T, <, val), y \models \alpha$

$(T, <, r, val), x \models E(\alpha U \beta)$ gdw. es einen x-Pfad $P \subseteq T$ gibt, so dass es 1. ein $y \in P - \{x\}$ gibt, so dass $(T, <, val), y \models \beta$, und für alle $z : z \in P$, $z < y$, gilt: $(T, <, val), z \models \alpha$.

$(T, <, r, val), x \models A(\alpha U \beta)$ gdw. für alle x-Pfade $P \subseteq T$ gilt: 1. es gibt ein $y \in P - \{x\}$, so dass $(T, <, val), y \models \beta$, und 2. für alle $z : z \in P$, $z < y$, gilt: $(T, <, val), z \models \alpha$.

$(T, <, r, val), x \models E(\alpha R \beta)$ gdw. es einen x-Pfad $P \subseteq T$ gibt s.d. gilt: für alle $y \in P - \{x\}$, falls es ein kein z gibt so dass $x \leq z < y$ und $(T, <, r, val), z \models \alpha$, dann gilt $(T, <, r, val), y \models \beta$.

$(T, <, r, val), x \models A(\alpha R \beta)$ gdw. für alle x-Pfad $P \subseteq T$ gilt: für alle $y \in P - \{x\}$, falls es ein kein z gibt so dass $x \leq z < y$ und $(T, <, r, val), z \models \alpha$, dann gilt $(T, <, r, val), y \models \beta$.

Eine weitere Bemerkung: wir haben bereits gesagt dass jede lineare Ordnung ein Baum ist, wir können also alle CTL-Konnektoren ohne weiteres auf Ketten interpretieren. In diesem Fall ist es interessant, folgendes zu sehen:

Lemma 17 *Sei $(T, <, r, val)$ eine lineare Ordnung, M eine LTL-Modalität. Dann gilt: $(T, <, r, val), x \models AM\alpha$ genau dann wenn $(T, <, r, val), x \models EM\alpha$.*

Um das zu sehen, bedenke man einfach: jeder Knoten in $(T, <)$ hat mindestens einen Nachfolger nach Annahme, und da $(T, <)$ linear ist, hat er genau einen – d.h. für alle und es gibt einen fallen zusammen.

Konvention Wir sagen, $(T, <, r, val) \models \alpha$ gdw. $(T, <, r, val), r \models \alpha$

Übung Gibt es ein Modell für $E(pUq) \wedge (E\Box\neg q)$? Wie sieht es aus?

Gibt es ein Modell für $A(pUq) \wedge (E\Box\neg q)$? Wie sieht es aus?

Gibt es ein Modell für $E(pUq) \wedge (A\Box\neg q)$? Wie sieht es aus?

Gibt es ein Modell für $A(pUq) \wedge (A\Box\neg p)$? Wie sieht es aus?

Fazit Nimm an, α,β sind E-Formeln, und beide haben ein Modell. Dann gibt es auch ein Modell für $\alpha \wedge \beta$.

Z.B.

- $\alpha = E\Box A\Box p$, $\beta = E\Box A\Box \neg p$.
- Ausnahme: $E(\alpha U \beta)$, z.B. $E(pUq)$, $E(\neg pUq)$ hat kein Modell!

Wie sieht das Modell aus für $\alpha \wedge \beta$?

11.4 Äquivalenzen und Definierbarkeit

Wir brauchen strenggenommen nur die Modalitäten

1. $EN\alpha$
2. $E\Box\alpha$
3. $E(\alpha U\beta)$;

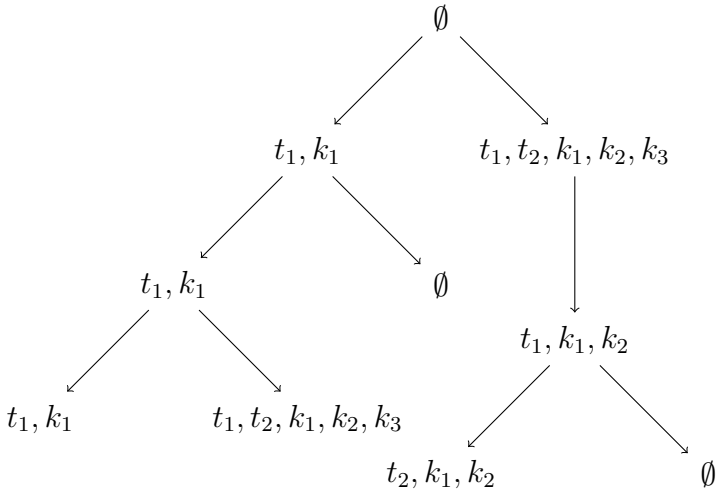
damit lassen sich die anderen definieren. Das liegt daran, dass wir weiterhin die normalen Dualitäten haben; außerdem gibt es noch eine Dualität von A, E:

- $AN\alpha \equiv \neg EN\neg\alpha$
- $E\Diamond\alpha \equiv E(\top U\alpha)$
- $A\Box\alpha \equiv \neg E\Diamond\neg\alpha$
- $A\Diamond\alpha \equiv \neg E\Box\neg\alpha$
- $A(\alpha R\beta) \equiv \neg E(\neg\alpha U\neg\beta)$
- $A(\alpha U\beta) \equiv \neg E(\neg\alpha U(\neg\alpha \wedge \neg\beta)) \wedge \neg E\Box\neg\beta$ (???)
- $E(\alpha R\beta) \equiv \neg A(\neg\alpha U\neg\beta)$

11.5 Intuition und Anwendung

Wir haben bereits eine grobe Intuition über die Bedeutung von CTL-Formeln: Dominanz im Baum entspricht dem Verlauf der Zeit, und Verzweigungen im Baum entsprechen der Ungewissheit der Zukunft, bzw. der Tatsache dass für die Zukunft viele Dinge möglich sind. Erinnern wir uns an unser Beispiel mit den beiden Maschinen und drei Kühlungen. Ob die Maschinen an oder aus sind, ist dabei von externen Faktoren abhängig, während die Kühlungen von unserem Programm in Bezug auf die Maschinen gesteuert werden. Unser Programm ist dabei deterministisch (nehmen wir an); allerdings ist der Verlauf der Ereignisse für uns nicht deterministisch: denn wir kennen ja nicht die externen Faktoren, nach denen unsere Maschinen laufen (Nachfrage, Versorgungslage etc.). D.h. wir mussten – in Bezug auf LTL-Modelle– die Ereignisse als eine *Sprache* auffassen, nämlich eine Menge von möglichen Ereignisverläufen.

Bei CTL-Modellen stehen die Dinge anders: dadurch dass wir den Nichtdeterminismus *innerhalb* der Modells erfassen, können wir nun alle möglichen Ereignissverläufe als *ein einziges Modell* beschreiben, nämlich einen Baum. In unserem Beispiel sieht der Baum wie folgt aus:



NB: dieser Baum ist so gedacht, dass an jedem Knoten (der einen gewissen Ereignisstand darstellt), wir immer jeden möglichen nachfolgenden Ereignisstand als Nachfolgeknoten erreichen.

⇒ Dieser Baum stellt also die verschiedenen Ereignisverläufe in einem Modell dar.

Das bringt uns einen großen Vorteil: vorher mussten wir prüfen, ob eine Sprache L eine Teilmenge der Sprache ist, die die die Menge aller α-Modelle kodiert. Wenn wir nun in genau demselben Sinne sagen, dass eine CTL-Formel eine Menge von Bäumen denotiert (nämlich genau die Menge aller Modelle),

$\Rightarrow$ dann müssen wir einfach nur prüfen, ob unser Baum in der Formeldenotation enthalten ist,

und schon wissen wir ob unser Programm die CTL-Formel enthält. Das ist eine deutliche Erleichterung, denn es ist unter normalen Umständen viel einfacher zu prüfen ob ein Objekt in einer Sprache enthalten ist, als ob eine (unendliche) Sprache in einer anderen enthalten ist.

Wenn wir das aber in diesem Fall machen wollen, müssen wir zunächst definieren, was eine Baumsprache ist. Grob gesagt ist des eine Menge von Bäumen; das Problem ist aber, eine solche Sprache effektiv zu charakterisieren. Man macht das üblicherweise mit **Baumautomaten** (siehe unten).

Zunächst aber können wir nun wieder die Bedingungen für Sicherheit und Lebendigkeit definieren:

- Eine **Sicherheitsbedingung**, die Sicherstellt dass in aller Zukunft immer α gilt, muss die Form haben $A\Box\alpha$ – das stellt sicher, dass egal was passiert immer α gilt.
- Eine **Lebendigkeitsbedingung** muss die Form haben: $A\Box E\Diamond\alpha$. Das beschreibt die Tatsache dass egal was passiert, gibt es immer die Möglichkeit dass in einer Zukunft α geschieht.

Übung Prüfen Sie, ob folgende Implikationen gelten:

1. $A\Diamond E\Box p \models E\Diamond A\Box p$
2. $A\Diamond E\Box p \models A\Box E\Box p$

Übung

Axiomatisieren Sie folgende Mengen von Bäumen:

- Alle Bäume, die p an genau einem Knoten erfüllen. (Hat keine Lösung. Warum?)
- Auf jeden Knoten, der ein p erfüllt, folgt ein Knoten der q erfüllt und umgekehrt.
- Jeder Knoten dominiert einen Knoten, an dem p gilt.
- Finden Sie ein unendliches CTL-Modell (blattloser Baum), der die Formel $A\Box E\Diamond p$ erfüllt, aber *nicht* die Formel $E\Diamond A\Diamond p$. (Offensichtlich können Sie das Modell nicht aufzeichnen. Charakterisieren Sie es also möglichst konzise!)

Übung

Prüfen Sie, ob folgende Implikationen gelten:

- $A\Box A\Diamond p \models E\Box p$
- $E\Box E(pUq) \models E\Box p$
- $E\Box E(pUq) \models E\Box q$
- $A\Diamond(p \wedge E\Box q) \models A(pUq)$

11.6 Baumsprachen und Bäume als Terme

Für Baumsprachen betrachten wir immer sog. *gelabelte* Bäume. Das sind Bäume, deren Knoten *Label* haben. Es ist wichtig, diese Label nicht mit ihren Namen zu verwechseln: der Name eines Knotens ist eindeutig und legt seine Identität fest – wenn zwei Knoten denselben Namen haben, dann es ein und derselbe Knoten. Label sind normalerweise eine endliche Menge, und es ist kein Problem wenn in einem Baum zwei Knoten dasselbe haben. Man kann das mit Worten wie folgt vergleichen: der Name des Knoten ist die Position im Wort, das Label der Buchstabe, der an der Stelle steht (man läßt das nur normalerweise implizit. Also bekommen wir folgende Definition:

Definition 18 *Ein **gelabelter Baum** ist eine Struktur $(T, <, \ell)$, wobei $(T, <$ $)$ ein Baum ist, und $\ell : T \to \Sigma$ eine Funktion, die jedem $t \in T$ ein Label $a \in \Sigma$ zuweist.*

(Man stelle sich z.B. vor, $\ell = val$) Im Zusammenhang mit Baumsprachen schreibt man gelabelte Bäume üblicherweise als **Terme**, das ist die kompakteste Notation. Das funktioniert zumindest für endliche Bäume sehr gut: Bäume sind z.B. Terme $a(b, a)$ oder $a(b, c(a, b))$. Wir können in dieser Form, ebenso wie wir mit Σ^* die endlichen Ketten über Σ definieren, auch die endlichen Terme über Σ definieren: $term(\Sigma)$ ist die kleinste Menge so dass

1. falls $a \in \Sigma$, dann $a \in term(\Sigma)$, und
2. falls $t_1, ..., t_i \in term(\Sigma)$, $a \in \Sigma$, dann ist $a(t_1, ..., t_i) \in term(\Sigma)$.

Allerdings gibt es hier eine Kleinigkeit zu beachten: in Termen haben die einzelnen Nachfolger eines Knoten eine *festgelegte Reihenfolge*. In den Bäumen die wir betrachten, haben sie das nicht – warum auch? Es gibt ja keinen Sinn, in dem man Verschiedene zeitgleiche Zukünfte untereinander sortieren könnte, es sei denn man legt ein völlig willkürliches Kriterium an. Wir kommen um diese kleine Widersinnigkeit aber kaum herum, denn Baumautomaten lesen Terme; wir können das aber später auflösen, indem wir verlangen, dass es dem Baumautomaten egal ist, in welcher Reihenfolge er die Nachfolger eines Knotens liest. Das ist aber ein zusätzliches Kriterium für Baumautomaten, kein "eingebautes".

11.7 Baumsprachen und Baumautomaten

Wir werden nun definieren, wie wir Automaten Terme einlesen lassen. Überlegen wir zunächst, wie es aussieht, wenn wir eine Zeichenkette $a_1...a_i$ als einen unär verzweigenden Baum auffassen, als Term $a_1(...(a_i)...)$. Der Automat ist im Startzustand, liest die Wurzel, geht in einen neuen Zustand, liest die (einzige) Tochter der Wurzel, geht in den nächsten Zustand usw. Wenn er an einem Blatt angekommen ist, muss er in einem akzeptierenden Zustand sein. Für Baumautomaten müssen wir diese Prozedur generalisieren. Nehmen wir einen Baum

$$a_1(a_2(..), a_3(..)).$$

Hier liest der Baum die Wurzel a_1 im Startzustand. Als nächstes *spaltet er sich auf*, d.h. er geht in zwei Zustände, und in dem einen Zustand prüft er, ob er $a_2(...)$ akzeptiert, und in dem anderen prüft er, ob er $a_3(...)$ akzeptiert. Am Ende werden wir, so der Baum endlich ist, eine positive oder negative Antwort erhalten für jedes Blatt des Baumes. Was dann aber noch fehlt ist folgenes: wir müssen die Antworten zu einer einzigen Antwort zusammensetzen. Damit haben wir das Prinzip des einfachen Baumautomaten beschrieben; wir kommen nun zu den Definitionen.

Definition 19 *Ein **top-down Baumautomat** ist ein Tupel $\mathfrak{A} = (Q, q_0, \delta, F, \Sigma)$, wobei alles wie bei endlichen Automaten ist, bis auf $\delta \subseteq Q \times \Sigma \times (\bigcup_{n\in\mathbb{N}} Q^n)$.*

Was mit der veränderten Übergangsfunktion ausgedrückt wird ist folgendes: eine Eingabe hat eben nicht nur *einen* Nachfolger, sondern eine beliebige endliche Anzahl davon, und jede bekommt einen Zustand zugewiesen. Die eigentlich entscheidende Definition ist die eines **Laufes** eines Automaten auf einem Baum in $term(\Sigma)$. Wir definieren die Funktion $\hat{\delta}$ wie folgt:

- falls $a(t_1, ..., t_i) \in term(\Sigma)$, $q \in Q$, $(q, a, q_1, ..., q_i) \in \delta$,
- dann ist $(\hat{\delta}(q_1, t_1), ..., \hat{\delta}(q_i, t_i)) \in \hat{\delta}(q, a(t_1, ..., t_i))$.

Eine wichtige Rolle für die Akzeptanz spielt der **leere Baum**, den wir manchmal mit () repräsentieren. Falls wir einen atomaren Term der Form a haben, und $(q, a, q_1, ..., q_i) \in \delta$, dann sagen wir

$$\hat{\delta}(q, a) = q_1, ..., q_i$$

wir lassen den leeren Term also weg. Wir sagen, dass $\mathfrak{A}$ einen Term t akzeptiert, falls es

$$(q_1, q_2, ..., q_i) \in \hat{\delta}(q_0, t) \text{ gibt, so dass } q_1, ..., q_i \in F$$

. Wir müssen also auf jedem Blatt in einem akzeptierenden Zustand angekommen sein. Zusammenfassend kann man sagen, ein Baumautomat verhält sich wie ein normaler Automat auf allen Pfaden durch den Baum, nur dass er zusätzlich noch berücksichtigt, wieviele Töchter (Nachfolger) ein Knoten hat, und auf dem wievielten Nachfolger er seinen Pfad fortsetzt.

Es gibt nicht nur *top-down* Baumautomaten, sondern auch sog. *bottom-up* Automaten, die einen Baum von unten nach oben erkennen.

Definition 20 *Ein bottom-up Baumautomat ist ein Tupel $\mathfrak{A} = (Q, \Sigma, F, \delta)$, wobei alles wie gehabt bis auf δ: δ ist eine Familie von Funktionen $\delta_a : a \in \Sigma$, $\delta_\sigma : (\bigcup_{n \in \mathbb{N}_0} Q^n) \to Q$.*

Wir definieren nun den Begriff der Akzeptanz: wir erweitern wieder δ zu $\hat{\delta}$, wobei

$$\hat{\delta}(a(t_1, ..., t_i)) = \delta_a(\hat{\delta}(t_1), ..., \hat{\delta}(t_i)) \tag{13}$$

f.a. $a \in \Sigma, t_1, ..., t_i \in term(\Sigma)$. Aus dem Baum-Term wird also ein Funktionsterm! Besondere Aufmerksamkeit verdient der Fall, wo $a = a()$ ein atomarer Baum ist; hier muss $\delta_a() \in Q$ definiert sein. Auf diese Art und Weise ist $\hat{\delta}$ eine Abbildung

$$\hat{\delta} : term(\Sigma) \to Q$$

Die Menge von Bäumen, die $\mathfrak{A}$ erkennt, ist definiert als

$$L(\mathfrak{A}) = \{t \in term(\Sigma) : \hat{\delta}(t) \in F\}$$

Ein weiterer wichtiger Begriff in Zusammenhang mit Baumautomaten ist der der **regulären Baumgrammatik**. Eine reguläre Baumgrammatik ist ein Tupel $G = (\mathcal{N}, \Sigma, S, R)$, wobei $\mathcal{N}, \Sigma$ Mengen sind von Nichtterminalen und Terminalsymbolen, $S \in \mathcal{N}$ das Startsymbol, und R eine Menge von Regeln ist der Form

$$N \to a(\alpha), \tag{14}$$

wobei $N \in \mathcal{N}, a \in \Sigma$, und α ist ein regulärer Ausdruck über Symbole in $\mathcal{N}$. Reguläre Baumgrammatiken generieren Bäume; wir definieren die Ableitung wie üblich: Sei $\alpha[N] \in \mathit{term}(\Sigma \cup \mathcal{N})$. Damit meinen wir: wir haben einen Term, in dem N als ein Blatt vorkommt. Wir schreiben

$$t[N] \vdash_G t[a(t)],$$

wobei $t[a(t)]$ das Ergebnis der Ersetzung von N durch $a(t)$ ist, gdw.

1. $N \to a(\alpha) \in R$ und
2. t eine Instanz von α ist (NB: α ist ja ein regulärer Ausdruck!).

Mit $\vdash_G^*$ bezeichnen wir die reflexive, transitive Hülle von $\vdash_G$, und bekommen so

$$L(G) = \{t \in \mathit{term}(\Sigma) : S \vdash_G^* t\}.$$

Also alles wir bei kontextfreien Grammatiken, nur das wir Bäume generieren. Im folgenden Abschnitt werde ich die wichtigsten Ergebnisse zusammenfassen.

11.8 Ein Beispiel

Sei $\mathfrak{A}$ ein Baumautomat, wobei

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_2\}$
- $\delta = \{(q_0, a, (q_1, q_1)),\ (q_0, b, q_0),\ (q_1, a, (q_1, q_1)),\ (q_1, b, q_2)\}$

Intuitiv: alle Bäume mit einer unären b-Kette am Anfang, gefolgt von binären a-Bäumen, aber die Blätter haben dann b.

Akzeptiert $\mathfrak{A}$ folgende Bäume?

1. $b(b(a(b), a(b)))$
2. $b(a(a(b), a(b)))$
3. $b(a(a(b, b), a(b, b)))$
4. $b(b(b(a, b)))$
5. $b(b(a(a, a)))$
6. $b(b(a(a(b, b)), a(a(b, b), a(b, b)))))$

3.

$$\begin{aligned}\hat{\delta}(q_0, b(a(a(b, b), a(b, b)))) &= \hat{\delta}(q_0, a(a(b, b), a(b, b))))\\ &= (\hat{\delta}(q_1, a(b, b)), \hat{\delta}(q_1, a(b, b)))\\ &= (\hat{\delta}(q_1, b), \hat{\delta}(q_1, b), \hat{\delta}(q_1, b), \hat{\delta}(q_1, b))\\ &= (q_2, q_2, q_2, q_2)\end{aligned}$$

2. Nein:

$$\begin{aligned}\hat{\delta}(q_0, b(a(a(b), a(b)))) &= \hat{\delta}(q_0, a(a(b), a(b))))\\ &= (\hat{\delta}(q_1, a(b)), \hat{\delta}(q_1, a(b)))\\ &= (\bot, \bot)\end{aligned}$$

1. Nein:

$$\begin{aligned}\hat{\delta}(q_0, b(b(a(b), a(b)))) &= \hat{\delta}(q_0, b(a(b), a(b))))) \\ &= \perp \text{ (kein Übergang mit } q_0, b \text{ und zwei Nachfolgern)}\end{aligned}$$

11.9 Übung

Sei $\Sigma = \{a, b\}$, $L \subseteq term(\Sigma)$ die Menge aller binären Bäume, die genau einmal den Buchstaben a enthalten.

1. Schreiben Sie einen top-down Baumautomaten, der die Sprache erkennt.
2. Schreiben Sie einen bottom-up Baumautomaten, der die Sprache erkennt.
3.
4. Schreiben Sie eine reguläre Baumgrammatik, die die Sprache generiert.

11.10 Die wichtigsten Ergebnisse in Kürze

Wir machen es kurz:

Lemma 21 *Sei $L \subseteq term(\Sigma)$ eine Menge von Bäumen. Die folgenden drei Aussagen sind äquivalent:*

1. *Es gibt eine reguläre Baumgrammatik G so dass $L = L(G)$.*
2. *Es gibt einen bottom-up Baumautomaten $\mathfrak{A}$ so dass $L = L(\mathfrak{A})$.*
3. *Es gibt einen top-down Baumautomaten $\mathfrak{A}$ so dass $L = L(\mathfrak{A})$.*

Eine Menge von Bäumen nennt man auch einfach eine Baumsprache. Die drei Charakterisierungen sind also äquivalent, und charakterisieren jeweils dieselbe Klasse von Baumsprachen. Wir nennen diese Klasse auch die Klasse der **regulären Baumsprachen**. Einen wichtigen Unterschied gibt es zwischen top-down und bottom-up Automaten. Wir sagen, ein Automat ist **deterministisch**, falls seine Übergangsrelation δ eine Funktion ist. Man beachte, dass wir bottom-up Automaten deterministisch definiert haben, top-down Automaten nicht; der Grund dafür ist folgender:

Lemma 22 *Für jeden nichtdeterministischen bottom-up Baumautomaten $\mathfrak{A}$ gibt es einen deterministischen bottom-up Baumautomaten $\mathfrak{A}'$, so dass $L(\mathfrak{A}) = L(\mathfrak{A}')$.*

Der Beweis ist wie üblich die Potenzmengenkonstruktion über Zustände. Für die top-down Methode funktioniert diese Konstruktion nicht:

Lemma 23 *Es gibt eine Baumsprache L, so dass es einen nichtdetermistischen top-down Automaten $\mathfrak{A}$ gibt mit $L = L(\mathfrak{A})$, aber keinen determistischen top-down Automaten $\mathfrak{A}'$ mit $L = L(\mathfrak{A}')$.*

Der Grund hierfür ist folgender: Falls $\mathfrak{A}'$ deterministisch ist, die Bäume $a(b,c)$, $a(c,b)$ akzeptiert, dann geht ist also $\hat{\delta}(q_0, a) = (q, q')$; $q(b), q(c), q'(b), q'(c) \in F$. Folglich akzeptiert der Automat auch den Baum $a(b,b)$. Also gibt es keinen deterministischen top-down Automaten, der $\{a(b,c), a(c,b)\}$ erkennt. Diese Baummenge ist aber regulär:

Lemma 24 *Jede endliche Menge von Bäumen ist eine reguläre Baummenge.*

Das zu beweisen ist recht einfach: wir nehmen einfach für jeden auftretenden Fall einen Zustand im Baumautomaten; so bleiben wir immer mit endlich vielen Zuständen. Als nächstes sollten wir kurz besprechen, wir reguläre Baummengen mit kontextfreien Sprachen zusammenhängen. Sei t ein Baum. Mit

$$b(t) \in \Sigma^*$$

meinen wir die eine Zeichenkette w, die entsteht, wenn man die Label der Blätter von t hintereinanderschreibt. Man kann b auffassen als Funktion

$$b : term(\Sigma) \to \Sigma^*.$$

Wir heben diese Funktion auf die kanonische Art und Weise auf Mengen an.

Lemma 25 *Falls L eine reguläre Baumsprache, dann ist $b(L)$ eine kontextfreie Sprache; und falls L eine kontextfreie Sprache ist, dann ist gibt es eine reguläre Baumsprache L_B, so dass $L = b(L_B)$.*

Der Beweis ist in beide Richtungen einfach; wir können tatsächlich kontextfreie Grammatiken umformen in reguläre Baumgrammatiken, die deren Ableitungsbäume beschreiben: wir transformieren einfach alle Regeln der Form $N \to \alpha$ zu Regeln $\overline{N} \to N(\alpha)$; die Terminale der neuen Grammatik sind $\mathcal{N} \cup \Sigma$, die Nichtterminale sind die Menge $\{\overline{N} : N \in \mathcal{N}\}$. Wenn wir nun allerdings diese Bäume betrachten, dann sehen wir, dass Klasse der kontextfreien Ableitungsbäume, die wir auf diese Art und Weise erhalten, echt kleiner ist als die Klasse der regulären Baumsprachen. Ein einfaches Beispiel ist folgende Baumsprache: $\{a, a(a)\}$. Die ist endlich, also regulär, aber nicht kontextfrei, da wir auf kontextfreie Art und Weise mindestens noch $a(a(a)), a(a(a(a)))...$ ableiten können. Diese kontextfreien Ableitungssprachen heißen auch **lokale Baumsprachen**, und verhalten sich zu den regulären Baumsprachen genau so, wie sich die streng lokalen Sprachen zu den regulären Sprachen verhalten (dazu sage ich einiges im oben angeführten Automaten-Skript). Dazu gehört u.a. folgender Satz. Ein **Baumhomomorphismus** ist eine Abbildung $h : \Sigma \to T$, die wie folgt auf Terme erweitert wird: $h(a(t_1, ..., t_i) = h(a)(h(t_1), ..., h(t_i))$.

Lemma 26 *Für jede reguläre Baummenge L gibt es eine lokale Baummenge L' und einen Homomorphismus h, so dass gilt: $L = h[L']$. Umgekehrt, falls L' lokal ist, h ein Homomorphismus, dann ist $h[L']$ regulär.*

Der Beweis des ersten Teiles geht so: wir konstruieren ein Alphabet das aus Paaren in $\Sigma \times Q$ besteht; dadurch können wir den Automaten mit einfachen Buchstaben simulieren. Der zweite Teil ist offensichtlich; wir können z.B. den Homomorphismus einfach in der Baumgrammatik einsetzen und bekommen das gewünschte Ergebnis.

Reguläre Baumsprachen haben sehr viele Eigenschaften mit regulären Sprachen gemeinsam: neben der Charakterisierung durch Automaten, Grammatiken auch die Charakterisierung durch eine Logik; und außerdem die äußerst wichtigen Abschlusseigenschaften:

Lemma 27 *Seien $L_1, L_2 \subseteq term(\Sigma)$ reguläre Baummengen. Dann sind $L_1 \cup L_2, L_1 \cap L_2, term(\Sigma) - L_1$ ebenfalls reguläre Baummengen.*

Wir haben also Abschluss unter Vereinigung unter Vereinigung, Schnitt (man zeigt dass mit den Standardkonstruktionen für Automaten) und Komplement (das folgt nur, weil wir die bottom-up Automaten deterministisch halten können!). Dieses Ergebnis ist eigentlich überraschend, denn die kontextfreien Sprachen sind nicht abgeschlossen unter Schnitt und Komplement, und die regulären Baumsprachen sind in gewissem Sinne äquivalent. Der Unterschied ist dass wir bei Baumsprachen die mengentheoretischen Operationen über die Bäume definieren, nicht über die Ketten. Wenn wir $b[L_1] \cap b[L_2]$ schneiden, dann gibt es im allgemeinen Fall keine reguläre Baumsprache L_3, so dass $b[_3] = b[L_1] \cap b[L_2]$. Etwas genauer gesagt:

(15) $b[L_1 \cap L_2] \subseteq b[L_1] \cap b[L_2]$

denn in der rechten Seite sind Worte, welche durch unterschiedliche Bäume "generiert wurden.

- $b[L_1 \cap L_2]$ ist kontextfrei, falls L_1, L_2 reguläre Baumsprachen sind
- $b[L_1] \cap b[L_2]$ ist im Allgemeinen *nicht* kontextfrei!

11.11 Von CTL zu Baumautomaten

Ähnlich wie wir aus einer LTL-Formel einen alternierenden Automaten kompilieren können, so können wir aus einer CTL-Formel einen alternierenden Baumautomaten kompilieren. Die Konstruktion ist sehr ähnlich, und im

Prinzip ist der Schritt von alternierenden Automaten zu alternierenden Baumautomaten kein Problem. Da alllerdings auch das Konzept der Alternation Bäume benötigt, wird die Beschreibung etwas unübersichtlich. Aus diesem Grund, und weil sich alles aus den behandelten Konzepten leicht zusammensetzen läßt, lassen wir dieses Thema aus, und beschränken uns auf die Feststellung: für jede CTL-Formel α gibt es einen Baumautomaten $\mathfrak{A}_\alpha$, so dass $\mathfrak{A}_\alpha$ eine Baum $\mathcal{T}$ erkennt genau dann wenn $\mathcal{T}$ ein Modell von α ist.

Damit hat – was Entscheidbarkeit betrifft – CTL die gleichen vorteilhaften Eigenschaften wir LTL.

Hausaufgabe 8

Bitte bearbeiten zum 22.6.2021

Schreiben Sie einen top-down Baumautomaten, der folgende Baumsprache $L \subseteq term(\{a, b\})$ erkennt: $t \in L$ gdw.

1. t ist binär;
2. t eine *gerade Anzahl* von bs enthält, sowie eine beliebige Anzahl von as.

Tipp: das ist eine komplexe Aufgabe, die eine Reihe von Zwischenschritten verlangt. Insbesondere gilt es an jeder Verzweigung zu Unterscheiden: für einen Teilbaum mit einer geraden Anzahl von as, verzweigen wir uns a) nach zwei Teilbäumen mit jeweils einer ungeraden Anzahl von as, oder b) nach zwei Teilbäumen mit einer geraden Anzahl von as. Wenn wir diese Entscheidung getroffen haben, müssen wir sie uns merken (um Sicherzustellen dass jeder Teilbaum die richtige gerade/ungerade Anzahl hat). Ebenso für jeden Teilbaum mit einer ungeraden Anzahl von as. Hier heißt "merken" soviel wie: wir nehmen einen Zustand hierfür.

12 Programme als Modelle

12.1 Programmsemantik für CTL

Es gibt noch eine alternative Semantik zu CTL, die etwas allgemeiner ist als die Baumsemantik. In gewissen Sinne ist diese Semantik weniger intuitiv wenn wir sie uns als *Zeit* vorstellen wollen; für viele Anwendungen hingegen ist sie unmittelbarer nutzbar. Das ist die sog. **Programmsemantik**, in der ein Modell eben nicht mehr ein Baum ist, sondern ein Programm, was in unserem Fall bedeutet: ein gerichteter Graph mit einem Wurzelknoten und einer Valuation. Im Prinzip kann man sich das auch wie einen Automaten vorstellen, nur dass wir keine akzeptierenden Zustände haben und Kanten eben nicht mit Buchstaben besetzt sind.

Definition 28 *Ein* ***Programm*** *ist eine Struktur* (W, R, w^0, val)*, wobei*

- $w^0 \in W$ *der Ausgangspunkt ist,*
- $R \subseteq W \times W$ *eine totale Übergangsrelation (d.h. für alle* $w \in W$ *gibt es* $w' \in W$ *so dass* $(w, w') \in R$*),*
- *und* $val : W \to \wp(prop)$ *ein Funktion ist.*

Ein Programm ist ***endlich****, falls* W *endlich ist.*

Die Intuition ist hier folgende: ein Zustand in W ist ein Zustand eines Programmes; das umfasst alle möglichen Parameter: Werte von Variablen, Zählern, Speicherbelegung und Zeiger. R sagt uns, von welchem Zustand in wir in welchen anderen Zustand kommen; wohlgemerkt ist R eine Relation, Programme können also nicht-deterministisch sein. *val* ist die übliche Valuation, die uns sagt welche Propositionen in welchem Zustand wahr sind (z.B. $i \leq 20$). Die Annahme, dass R total ist, erlaubt es die Definitionen zu vereinfachen. Eine weitere Vereinfachung besteht in der Annahme, dass Programmausführungen "enden", indem sie ihren letzten Zustand unendlich oft wiederholen.

Eine weitere wichtige Beobachtung ist folgende: während wir bislang immer nur mit transitiven Relationen $<$ zu tun hatten, macht das in der Programmsemantik keinen Sinn: denn R ist nicht notwendig antisymmetrisch, d.h. wir können z.B. einen Zirkel haben

$$w_0 R w_1 R ... R w_0$$

– und in diesem Fall bekommen wir jede Menge Übergänge, die wir eigentlich nicht möchten.

Definition 29 *Sei $\mathcal{P}$ ein Programm. Ein **Pfad** P für $\mathcal{P}$ ist eine unendliche Abfolge von Zuständen $(w_0, w_1, w_2, ...)$, so dass für alle $i \geq 0$ gilt: $(w_i, w_{i+1}) \in R$; wir nennen einen solchen Pfad auch einen w_0-Pfad.*

NB: wir verlangen *nicht* dass $w_0 = w^0$, der Pfad muss also nicht mit w_0 beginnen! Wir kommen nun zu den Definitionen; wir nehmen an dass $\mathcal{P} = (W, w_0, R, val)$ und $w \in W$.

$\mathcal{P}, w \models p$ gdw. $p \in val(w)$

$\mathcal{P}, w \models \alpha \wedge \beta$ gdw. $\mathcal{P}, w \models \alpha$ und $\mathcal{P}, w \models \beta$

$\mathcal{P}, w \models \alpha \vee \beta$ gdw. $\mathcal{P}, w \models \alpha$ oder $\mathcal{P}, w \models \beta$

$\mathcal{P}, w \models \neg\alpha$ gdw. $\mathcal{P}, w \not\models \alpha$

$\mathcal{P}, w \models \top$

$\mathcal{P}, w \not\models \bot$

$\mathcal{P}, w \models EN\alpha$ gdw. es ein $v \in W$ gibt so dass $(w, v) \in R$ und $\mathcal{P}, v \models \alpha$

$\mathcal{P}, w \models AN\alpha$ gdw. für alle $v \in W$ gilt: falls $(w, v) \in R$, dann $\mathcal{P}, v \models \alpha$

$\mathcal{P}, w \models E\Box\alpha$ gdw. es einen w-Pfad $(w_0, w_1, ...)$ in $\mathcal{P}$ gibt s.d. für alle $i > 0$ gilt: $\mathcal{P}, w_i \models \alpha$

$\mathcal{P}, w \models A\Box\alpha$ gdw. für alle w-Pfade $(w_0, w_1, ...)$ in $\mathcal{P}$ und alle $i > 0$ gilt: $\mathcal{P}, w_i \models \alpha$

$\mathcal{P}, w \models E\Diamond\alpha$ gdw. es einen w-Pfad $(w_0, w_1, ...)$ in $\mathcal{P}$ und ein $i > 0$ gibt s.d. gilt: $\mathcal{P}, w_i \models \alpha$

$\mathcal{P}, w \models A\Diamond\alpha$ gdw. für alle w-Pfade $(w_0, w_1, ...)$ in $\mathcal{P}$ es ein $i > 0$ gibt

s.d. gilt: $\mathcal{P}, w_i \models \alpha$

$\mathcal{P}, w \models E(\alpha U \beta)$ gdw. gdw. es einen w-Pfad $(w_0, w_1, ...)$ in $\mathcal{P}$ und ein $i > 0$ gibt s.d. $\mathcal{P}, w_i \models \Diamond\beta$ und für alle $j < i$ gilt: $\mathcal{P}, w_j \models \alpha$

$\mathcal{P}, w \models A(\alpha U \beta)$ gdw. für alle w-Pfade $(w_0, w_1, ...)$ in $\mathcal{P}$ es ein $i > 0$ gibt s.d. $\mathcal{P}, w_i \models \Diamond\beta$ und für alle $j < i$ gilt: $\mathcal{P}, w_j \models \alpha$

$\mathcal{P}, w \models E(\alpha R \beta)$ gdw. es einen w-Pfad $(w_0, w_1, ...)$ in $\mathcal{P}$ gibt s.d. für alle $i > 0$ gilt: falls für alle j:$0 \leq j \leq i$ gilt $\mathcal{P}, w_j \not\models \Diamond\alpha$, dann gilt $\mathcal{P}, w_j \models \Diamond\beta$. dann

$\mathcal{P}, w \models A(\alpha R \beta)$ gdw. für alle w-Pfade $(w_0, w_1, ...)$ in $\mathcal{P}$ und alle $i > 0$ gilt: falls für alle j:$0 \leq j \leq i$ gilt $\mathcal{P}, w_j \not\models \Diamond\alpha$, dann gilt $\mathcal{P}, w_j \models \Diamond\beta$.

Wir setzen

$$\mathcal{P} \models \alpha \text{ genau dann wenn } \mathcal{P}, w^0 \models \alpha$$

Diese Semantik ist eine echte Generalisierung, denn wir können uns das Programm auch als einen unendlichen Baum vorstellen oder als unendliche Kette – aber eben auch als beliebigen endlichen oder unendlichen Graphen.

12.2 Programmsemantik für LTL

Das interessante ist dass wir auch sehr einfach die Programmsemantik für LTL definieren können. Das basiert auf der Tatsache, dass R total ist, und wir dementsprechend in einem Programm eine Menge unendlicher Pfade haben.

Definition 30 *Sei α eine LTL-Formel, $\mathcal{P} = (W, w^0, R, val)$. Wir sagen $\mathcal{P} \models \alpha$, gdw. alle w^0-Pfade P in $\mathcal{P}$ gilt: $(P, <, val), w^0 \models \alpha$.*

Wir identifizieren also ein Programm mit der Menge aller unendlichen Pfade; die Totalität von R stellt sicher, dass jedes Programm durch die Menge seiner unendlichen Pfade eindeutig charakterisiert ist. Das bedeutet aber nicht, dass wir beliebige Programme durch LTL-Formeln unterscheiden können – denn wir bestimmen ja nur Propositionen, und deren Gültigkeit wiederum wird von *val* bestimmt.

Wie wir sehen ist es nicht klar, wie ein Programm mit unserem intuitiven Zeitbegriff zusammen hängt. Wenn wir allerdings, wie üblich, die Eigenschaften eines Systems verifizieren wollen, dann ist die Programmsemantik wesentlich naheliegender!

12.3 LTL und CTL - Ein Vergleich

Übung Entscheiden Sie für ein Modell, ob es folgende Formeln erfüllt: ...

Nachdem wir also Programmsemantik sowohl von LTL als auch von CTL betrachtet haben, können wir einen direkten Vergleich der Expressivität anstellen. Damit meinen wir:

Gibt es für jede LTL-Formel β eine CTL-Formel α so dass $\mathcal{P} \models \beta$ genau dann wenn $\mathcal{P} \models \alpha$, und umgekehrt?

Überraschenderweise ist **beides falsch**, d.h. es gibt sowohl LTL-Formeln, die keine CTL-Entsprechung haben, wie auch umgekehrt. Das mag unintuitiv scheinen, v.a. da CTL auf den ersten Blick mächtiger erscheint. Die Restriktionen, denen Pfadquantoren unterliegen, und v.a. deren obligatorische Nutzung beschränken aber die Mächtigkeit von CTL (was wiederum seiner Entscheidbarkeit zugute kommt).

Eine LTL-Formel ohne Entsprechung in CTL ist z.B.

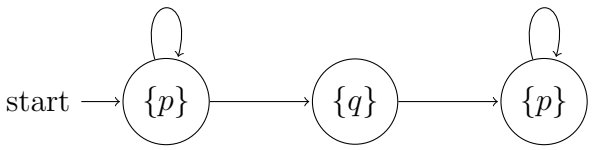
$$\Diamond\Box p$$

Betrachten wir zunächst was das bedeutet in der Programmsemantik: das bedeutet auf allen Pfaden gibt es einen Punkt, in dem wir nur noch in Zustände kommen in denen p gilt.

Nehmen wir einmal eine mögliche CTL-Entsprechung, nämlich die Formel

$$A\Diamond A\Box p$$

Was bedeutet diese Formel? Etwas ganz ähnliches, nämlich dass es auf jedem Pfad einen Punkt gibt, in dem alle Pfade nur noch Zustande umfassen, in denen p gilt. Das ist aber nicht ganz dasselbe, denn diese Bedingung schließt unter anderem *Übergangszustände* aus, die wir nur einmal berühren können, in denen aber p nicht gilt. Man betrachte folgendes Beispiel:



Man sieht: der Pfad, der immer im ersten Zustand bleibt, erfüllt $\Diamond\Box p$, allerdings erfüllt er nicht $A\Diamond A\Box p$, denn es besteht immer die Möglichkeit in den zweiten Zustand zu wechseln. Die CTL-Formel ist also **stärker**. Wenn wir allerdings ein A durch ein E ersetzen, dann wird die Formel schwächer – unsere LTL-Formel $\Diamond\Box p$ hat keine CTL-Entsprechung!

Eine CTL-Formel ohne Entsprechung in LTL ist z.B.

$$A\Box E\Diamond p$$

Diese Formel besagt: von jedem Zustand, den ich erreichen kann, kann ich wiederum einen Zustand erreichen, in dem p gilt. Anders gesagt: es wird niemals unmöglich, einen Zustand zu erreichen in dem p gilt. Die natürliche LTL-Entsprechung wäre:

$$\Box\Diamond p$$

Allerdings bedeutet das eine ganz andere Sache: es besagt dass ich auf jedem Pfad immer wieder einen Zustand treffe, in dem p gilt, anders gesagt: auf jedem Pfad im Programm gilt: ich treffe jeder Pfad enthält unendlich viele Zustände, in denen p gilt (das entspricht also grob dem Kriterium der Akzeptanz in Büchi-Automaten). $A\Box E\Diamond p$ kann aber in einem Programm gelten, in dem es Pfade gibt, auf denen *niemals* p gilt:

Tatsächlich gibt keine LTL-Formel, die $A\Box E\Diamond p$ entspricht.

Ein konkretes Szenario Stellen Sie sich vor Sie sitzen in einem Raum ohne Chance, ihn zu verlassen; aber Sie haben einen Kuchen. p bedeutet: Sie haben Hunger. Wenn Sie den Kuchen essen, haben Sie erstmal keinen Hunger mehr, aber einen Kuchen kann man nur einmal essen.

Dann gilt: in ihrem Szenario ist $A\Box E\Diamond\neg p$ sicher korrekt, denn Sie können den Kuchen immer weiter hinaus schieben. Aber in LTL lässt sich nichts vergleichbares sagen.

12.4 Noch zwei Beispiele

In LTL ist es Beispielsweise möglich, folgendes zu sagen:

(16) $(\Box\Diamond p) \rightarrow (\Box\Diamond q)$

Nehmen wir an, p bedeutet soviel wie: ein Prozess x wird gestartet, und q bedeutet soviel wie: derselbe Prozess x wird erfolgreich abgeschlossen. Dann bedeutet die Formel: wenn der Prozess x immer wieder gestartet wird, dann wird er auch immer wieder zu einem erfolgreichen Ende geführt. Das bedeutet aber nicht, dass jeder Start zur einem erfolgreichen Ende führt, sondern nur, dass er nur endlich oft hintereinander scheitert, also irgendwann ein Start zum Erfolg führen muss.

Umgekehrt kann man die CTL-Formel

(17) $A\Box E\Diamond p$

wie folgt lesen: bedeute p soviel wie: "Neustart". Dann heißt das ganze: es ist immer möglich, das System neu zu starten – ohne dass wir es je tun müssten. Das lässt sich also in LTL nicht ausdrücken.

Aufgabe

Zu bearbeiten bis zum 17.1. (Abgabe vor dem Seminar).

Gibt es eine CTL-Formel, die $\Box\Diamond p$ entspricht (unter Annahme der Programmsemantik)? Falls ja, geben Sie die Formel und eine Begründung der Äquivalenz; falls nein, liefern Sie eine Begründung.

13 CTL*

Die Tatsache, dass sich manche Dinge nicht mit LTL ausdrücken lassen und manche nicht mit CTL lässt uns die Frage stellen, ob es eine Sprache gibt die die Ausdrucksstärke beider Sprachen vereint. Das gibt uns die Sprache CTL*, die auf natürliche Weise entsteht, wenn wir die Pfadquantoren von den Temporaloperatoren ablösen. Sie lässt sich also ganz einfach wie folgt charakterisieren:

1. p ist eine CTL*-Formel, für $p \in prop$
2. Falls α, β CTL*-Formeln sind, dann sind auch $\alpha \wedge \beta, \alpha \vee \beta, \alpha \rightarrow \beta, \neg\alpha$ CTL*-Formeln
3. Falls α eine CTL*-Formel ist, dann sind $N\alpha, N\alpha$ CTL*-Formeln
4. Falls α eine CTL*-Formel ist, dann sind $\Diamond\alpha, \Diamond\alpha, \Box\alpha, \Box\alpha$ CTL*-Formeln
5. Falls α, β CTL*-Formeln sind, dann sind $\alpha U \beta, \alpha R \beta)$ CTL*-Formeln
6. Falls α eine CTL*-Formel ist, dann sind $A(\alpha), E(\alpha)$ CTL*-Formeln

Aus dieser Definition lässt sich leicht ablesen das insbesondere jede LTL-Formel *und* jede CTL-Formel eine CTL*-Formel ist. Die Semantik ist intuitiv recht offensichtlich, aber um den Pfadquantoren eine unabhängige Semantik zu geben, ist doch einige Arbeit erforderlich, daher überspringen wir sie.

Tatsächlich ist es so, dass eine LTL- und CTL-Formeln *nicht* äquivalent sind zu ihren CTL*-homonymen. Allerdings lassen sich beide Logiken in CTL* einbetten. Das Problem ist dass sich nicht mehr ohne weiteres Baumautomaten aus CTL*-Formeln kompilieren lassen!

14 Logiken für Intervalle I: Allens Relationen

14.1 Einleitung

Bislang haben wir Logiken betrachtet, in denen Propositionen zu Zeipunkten evaluiert wurden. Das ist für viele Anwendungen nicht ausreichend, v.a. aus folgendem Gründen:

1. Oft haben verschiedene Eigenschaften eine *zeitliche Ausdehnung*, und so kann es sein dass sich Dinge überschneiden, überlappen etc. Wir haben bislang keine Handhabe, um über derartige Dinge zu sprechen.

2. Oft möchten wir, dass eine Eigenschaft zeitlich direkt an eine andere anschließt. Wir können das ausdrücken mit N in LTL und CTL; allerdings setzt das ein diskretes Zeitmodell voraus. Gerade das ist in den meisten Fällen unrealistisch. Auf der anderen Seite kann man verlangen, dass zwei Intervalle direkt aneinander anschließen.

Es gibt also einige schlagende Gründe, Zeitintervalle statt Zeitpunkte zu betrachten. Genau das werden wir jetzt tun; bevor wir allerdings logische Sprachen einführen, betrachten wir etwas genauer über was wir eigentlich reden wollen.

14.2 Allens Algebra der Intervalle

Allens Intervallalgebra umfasst die wichtigsten Relationen von Intervallen, die oft auch Allens Relationen genannt werden. Wir werden zunächst diese Relationen umfassen. Es sind insgesamt 13, von denen man aber normalerweise nur 6/7 als grundelegend annimmt.

Definition 31 *Ein **Intervall** ist für uns ein Paar von Zahlen $\langle x, y\rangle$, wobei $x, y \in \mathbb{R}$ und $x \leq y$. Ein Intervall ist **echt**, falls $x < y$.*

Die beiden Zahlen repräsentieren jeweils Anfangs- und Endpunkt des Intervalles. Wir werden annehmen, dass Ereignisse eine zeitliche Ausdehnung haben, also können wir sagen dass ein Ereigniss stattfindet in einem bestimmten echten Intervall (verwechseln Sie Intervalle in unserem Sinne nicht mit den offenenen/geschlossenen Intervallen, die man aus der Analysis kennt: diese sind *Mengen* von Zahlen, nicht Paare).

Nachdem Intervalle Paare sind und Relationen Mengen von Paaren, sind also Allens Relationen Mengen von Paaren von Paaren. Das sind die folgenden:

1.	"Präzediert":	$\mathsf{p} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : x < x' < y < y'\}$
2.	"Während":	$\mathsf{w} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : y < x < x' < y'\}$
3.	"Überschneidet":	$\mathsf{u} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : x < y < x' < y'\}$
4.	"Trifft":	$\mathsf{t} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : x < x' = y < y'\}$
5.	"Beginnt":	$\mathsf{b} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : x = y < x' < y'\}$
6.	"Endet":	$\mathsf{e} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : y < x < x' = y'\}$
7.	"Identität":	$\mathsf{1} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : x = y < x' = y'\}$

Aus diesen 7 Relationen bekommen wir 13, indem wir die Relationen invertieren: für R eine Relationen schreiben wir $R^{-1} = \{\langle x, y\rangle : \langle y, x\rangle \in R\}$.

8.	$\mathsf{p}^{-1} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : y < y' < x < x'\}$
9.	$\mathsf{w}^{-1} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : x < y < y' < x'\}$
10.	$\mathsf{u}^{-1} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : y < x < y' < x'\}$
11.	$\mathsf{t}^{-1} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : y < y' = x < x'\}$
12.	$\mathsf{b}^{-1} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : x = y < y' < x'\}$
13.	$\mathsf{e}^{-1} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : x < y < y' = x'\}$
7.	$\mathsf{1}^{-1} = \{\langle\langle x, x'\rangle, \langle y, y'\rangle\rangle : x = y < x' = y'\}$

Wie man leicht sehen kann ist $1^{-1} = 1$; wir kommen also mit 7 Relationen und ihren Inversionen auf 13. Man verzichtet aber normalerweise darauf, den Relationen 8.-13. eigene Namen zu geben. Diese Relationen sind die konzeptuelle Grundlage für jede Intervall-basierte Temporallogik. Wichtig ist, dass diese Relationen

1. disjunkt sind (d.h. ihre Schnittmenge ist immer leer),
2. universell sind im Sinne dass zwischen zwei Intervallen immer mindestens (und genau) eine Allen Relation gilt,
3. und dass diese Relationen alle konzeptuellen Konstellationen von zwei Intervallen unterscheiden.

Wir fassen sie hier als Relationen im mengentheoretischen Sinne auf, was bedeutet: wir können alle möglichen mengentheoretischen Operationen mit ihnen durchführen, wie z.B. Schnittmenge und Vereinigung bilden. So ist es leich zu sehen dass Ein wichtiges Konzept für Relationen ist das der **Komposition**: wir setzen

$$R \circ R' = \{\langle x, y\rangle : \exists z.\langle x, z\rangle \in R \& \langle z, y\rangle \in R'\}$$

Damit ist es z.B. leicht zu sehen dass

(18) $\mathsf{p} = \mathsf{t} \circ \mathsf{e}^{-1}$

Es ist also keinesfalls so, dass die Relationen nicht noch weiterhin reduziert werden können. Ein weiteres Beipiel für eine Reduktion ist folgendes:

(19) $\mathsf{w} = \mathsf{b} \circ \mathsf{e}$

und

(20) $\mathsf{w} = \mathsf{e} \circ \mathsf{b}$

Ebenso:

(21) $\mathsf{e} = (\mathsf{p}^{-1} \circ \mathsf{b}) \cap (\mathsf{t} \circ \mathsf{t}^{-1})$

(22) $\mathsf{t} = (\mathsf{b} \circ \mathsf{e}^{-1}) \cap \overline{\mathsf{p}} \cap \overline{\mathsf{u}}$

Tatsächlich ist es so, dass mit den Operationen

$$\cap, \overline{[-]}, [-]^{-1}, \circ$$

sämtliche Allen Relationen aus einer einzigen erzeugt werden können, nämlich wahlweise p, t, oder u. Das zu zeigen ist aber nicht wirklich einfach und gehört ins Feld der **Relationen-Algebren**.

Definition 32 *Eine Relationen Algebra ist eine Struktur* $(\boldsymbol{R}, \cup, \cap, \overline{[-]}, [-]^{-1}, \circ, 1, \bot, \top)$*, wobei* $\boldsymbol{R}$ *eine Menge von Relationen ist, abgeschlossen unter den vorigen Operationen, und* $\top = \bigcup \boldsymbol{R}$*,* $\bot = \bigcap \boldsymbol{R}$*, und* $1 = 1^{-1}$ *das neutrale Element der Komposition ist.*

Wenn wir eine Menge M haben, Operationen $f_1, ..., f_n$ auf diesen Mengen, dann sagen wir M *generiert* eine Algebra $(h[M], f_1, ..., f_n)$, wenn $h[M]$ die kleinste Menge ist so dass

1. $M \subseteq h[M]$,
2. falls $1 \leq i \leq n$, f_i eine $k$äre Relation ist, $m_1, ..., m_k \in h[M]$, dann ist $f_i(m_1, ..., m_k) \in h[M]$

$h[M]$ nennt man auch die Hülle oder den Abschluss von M unter den Operationen. Wie wir gesehen haben wird Allens Relationenalgebra der Intervalle generiert von einer kleinen Anzahl von Relationen, z.B. u.

Was für die Logik wichtig sein wird ist die Menge

$$\mathsf{A} = \{\mathsf{p}, \mathsf{w}, \mathsf{t}, \mathsf{u}, \mathsf{b}, \mathsf{e}, 1, \mathsf{p}^{-1}, \mathsf{w}^{-1}, \mathsf{t}^{-1}, \mathsf{u}^{-1}, \mathsf{b}^{-1}, \mathsf{e}^{-1}\},$$

also die Menge der Allen Relationen. Diese Relationen haben auch eine wichtige Eigenschaft in der Relationenalgebra, die sie generieren.

- Wir sagen, eine Algebra ist **geordnet** (nach $\leq$), falls $\leq$ eine partielle Ordnung ist.
- Wir sagen eine geordnete Algebra ist **begrenzt**, falls $\leq$ ein eindeutiges minimales Element $\bot$ und maximales Element $\top$ hat.
- Ein **Atom** in einer geordneten, begrenzten ist ein Element x, so dass gilt: falls $y \leq x$, dann ist $y = \bot$ oder $y = x$.
- (Also die Atome der Booleschen Algebra sind genau die Objekte, die unmittelbar über der 0 sitzen).

Atome sitzen also direkt über dem Minimum, sind also die kleinsten nichttrivialen Teilchen. Relationenalgebren, wie wir sie benutzen, haben eine natürliche Ordnung

$$\subseteq \text{ (mengentheoretisch)},$$

ein minimales Element $\emptyset$ und ein maximales Element $\overline{\emptyset}$. Folgendes Lemma beschreibt die Funktion von Allens Relationen in der generierten Relationenalgebra:

Lemma 33 *Sei* $\boldsymbol{AA}$ *die Relationenalgebra über Allens Intervalle.* x *ist ein Atom in* $\boldsymbol{AA}$ *gdw.* $x \in \mathsf{A}$.

Wir sprechen also hier algebraisch über Atome, und das ist, was diese Relationen konzeptuell so besonders macht. Daraus folgt folgendes

Lemma 34 *Jede Relation in* $\boldsymbol{AA}$ *kann erzeugt werden als eine Vereinigung* $\bigcup X$*, wobei* $X \subseteq \mathsf{A}$.

Die anderen Operationen sind hilfreich, aber nicht nötig! Weiterhin, erinnern wir uns dass es für jede endliche BA $\mathbf{B}$ eine (endliche) Menge M gibt, so dass

(23) $\wp(M) \cong \mathbf{B}$

– jede endliche BA ist isomorph (strukturgleich) mit einer Potenzmengenalgebra. Die Atome von $\wp(M)$ sind dabei die Mengen mit einem Element:

(24) $A(\wp(M)) = \{\{x\} : x \in M\}$

Auf unserem Fall übertragen bedeutet das:

(25) $\mathbf{AA} = \wp(\mathsf{A})$

woraus unmittelbar folgt:

Lemma 35 $\boldsymbol{AA}$ *enthält* $2^{13} = 8192$ *Relationen.*

Also doch schon einige.

14.3 Halpern und Shohams Logik HS

Die Syntax von HS ist sehr einfach:

1. p ist eine HS-Formel, für $p \in prop$
2. Falls α, β HS-Formeln sind, dann sind auch $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), (\neg\alpha)$ HS-Formeln
3. falls α eine HS-Formel ist und $\mathsf{x} \in \mathsf{A}$, dann ist auch $(\langle \mathsf{x} \rangle \alpha)$ eine HS Formel;
4. sonst ist nichts eine HS-Formel.

Wir haben also für jede Allen-Relation eine eigene Modalität. Als nächstes betrachten wir die Modelle. Für $\mathbb{R}$ die Menge der reellen Zahlen denotieren wir mit $\mathbb{I}(\mathbb{R})$ die Menge aller Intervalle über reelle Zahlen (inklusive (x, x), dem trivialen Intervall). Ein HS-Modell ist ein Tupel

$$(\mathbb{R}, val),$$

wobei

$$val : \mathbb{I}(\mathbb{R}) \rightarrow \wp(prop)$$

die unvermeidliche Valuation ist, die uns sagt welche Propositionen in welchem Intervall wahr sind. Die Erfüllung von Formeln ist definiert im Hinblick auf gewisse Referenzintervalle, ansonsten wie üblich induktiv über den Formelaufbau:

$(\mathbb{R}, val), \langle x, y \rangle \models p$ gdw. $p \in val(\langle x, y \rangle)$, für $p \in prop$.

$(\mathbb{R}, val), \langle x, y \rangle \models \neg\alpha$ gdw. $(\mathbb{R}, val), \langle x, y \rangle \not\models \alpha$

$(\mathbb{R}, val), \langle x, y \rangle \models \alpha \wedge \beta$ gdw. $(\mathbb{R}, val), \langle x, y \rangle \models \alpha$ und $(\mathbb{R}, val), \langle x, y \rangle \models \beta$

$(\mathbb{R}, val), \langle x, y \rangle \models \langle \mathsf{x} \rangle \alpha$ gdw. es ein Intervall $\langle z, z' \rangle$ gibt, so dass 1. $\langle \langle x, y \rangle, \langle z, z' \rangle \rangle \in \mathsf{x}$ und $(\mathbb{R}, val), \langle z, z' \rangle \models \alpha$.

Man kann die letzte Klausel auch ausschreiben für alle $\mathsf{x} \in \mathsf{A}$, aber damit ist wohl niemandem wirklich geholfen. Was sehr wichtig ist, ist die Modalität $[-]$, die dual ist zu $\langle - \rangle$ und wie üblich definiert wird durch

$$[\mathsf{x}]\alpha \equiv \neg\langle\mathsf{x}\rangle\neg\alpha$$

Man kann ihr aber auch – wie üblich – eine atomare Semantik zuweisen durch

$(\mathbb{R}, val), \langle x, y\rangle \models [\mathsf{x}]\alpha$ gdw. für alle Intervalle $\langle z, z'\rangle$ gilt: falls $\langle\langle x, y\rangle, \langle z, z'\rangle\rangle \in \mathsf{x}$, dann $(\mathbb{R}, val), \langle z, z'\rangle \models \alpha$.

Validität wird wie üblich definiert durch

$$M \models \alpha \text{ gdw. für alle } x, y \in \mathbb{R} \text{ gilt: } M, \langle x, y\rangle \models \alpha$$

Eine Formel ist **erfüllbar** falls es $x, y \in \mathbb{R}$ gibt so dass $x < y$ gilt und $M, \langle x, y\rangle \models \alpha$. Natürlich ist α valide genau dann wenn $\neg\alpha$ nicht erfüllbar ist.

Übung A Was bedeuten folgende Formeln (d.h. wie sehen die Modelle aus)? ($M = (\mathbb{R}, val)$): (Lösung auskommentiert)

1. $M, \langle x, y\rangle \models \langle \mathsf{p}\rangle q$
2. $M, \langle x, y\rangle \models \langle \mathsf{w}\rangle q$
3. $M \models \langle \mathsf{p}\rangle q$
4. $M \models \langle \mathsf{w}\rangle q$
5. $M \models (\langle \mathsf{p}^{-1}\rangle q) \to \langle \mathsf{p}\rangle q$
6. $M \models [\mathsf{e}](q \to \langle \mathsf{e}\rangle r)$

Übung B Schreiben Sie folgende Relationen als Mengen der Form: $\{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : \xi\}$, wobei ξ eine Bedingung ist der z.B. Form $y < x < y' < x'$, oder $y = y'$, oder: falls $x < x'$, dann $y < y'$. Das sind natürlich nur Beispiele; finden Sie die richtige Bedingung! (Zeichnung hilft natürlich)

1. $t \circ t^{-1}$
2. $(e \cup e^{-1}) \circ t$
3. $\overline{w \cup w^{-1}}$
4. $w \circ w^{-1}$

Hausaufgabe 9

Bearbeiten bis zum 28.6.2021

Bearbeiten Sie Teil 6 der Übung A, sowie Teil 2 der Übung B.

Übung

Es gibt die sog. Vendler Klassen von Verben, mit 4 Kategorien:

1. State: wollen haben
2. Activity: schwimmen, laufen (überträgt sich auf Teilintervalle)
3. Accomplishment: ein Haus bauen, 2 Pizzas essen (Ausdehnung, aber keine Übertragung auf Teilintervalle)
4. Achievement: finden, ankommen (punktuell)

Finden Sie HS-Formeln, die soviel besagen wie

- q ist eine activity
- q ist ein accomplishment
- q ist ein achievement

Lösung Activity:

(26) $q \to ([w^{-1}]q \wedge [b^{-1}]q \wedge [e^{-1}]q)$

Accomplishment: Vererbung nach oben. Das wird dann:

(27) $q \to ([w]q \wedge [b]q \wedge [e]q)$

Achievement: wenn ein Intervall $\langle x, x' \rangle$ die Proposition q erfüllt, dann ist $x = x'$.

Erinnern wir uns an die Formel $\Box q \to \Diamond q$ (falsch wenn es keinen Nachfolger gibt). NB: das triviale Intervall hat keinen w^{-1}-Nachfolger!

Also sind Propositionen der Form $[w^{-1}]q$ in diesem Intervall *immer* wahr.

Sei $\langle x, x' \rangle$ *nicht*-trivial, und nimm an, $M, \langle x, x' \rangle \models [w^{-1}]q$. Dann gilt: $M, \langle x, x' \rangle \models \langle w^{-1} \rangle q$. Also:

(28) $q \to (([w^{-1}]q) \wedge \neg(\langle w^{-1} \rangle q))$

Sprich: q muss in einem trivialen Intervall wahr sein, sonst ist die rechte Seite von $\to$ inkonsistent.

14.4 Expressivität von HS

Was können wir mit HS ausdrücken? Die erste Antwort ist: wir beschreiben Modelle der Form $(\mathbb{R}, val)$ mit der axiomatischen Methode, wobei für eine HS-Formel α wir definieren:

(29) $\mathsf{Mod}(\alpha) = \{(\mathbb{R}, val) : (\mathbb{R}, val) \models \alpha\}$

Wir nutzen also den Begriff der Validität um mittels Formeln Klassen von Modellen zu beschreiben. Was für Klassen können wir also beschreiben?

Beispiel 1 Sei $P \subseteq prop$; nenne **Dis(P)** die Menge aller Modelle $(\mathbb{R}, val)$ so dass für alle Intervalle $\langle x, x'\rangle, \langle y, y'\rangle \in \mathbb{I}(\mathbb{R})$, $p, q \in P$ mit $p \neq q$ gilt:

$$\text{falls } p \in val(\langle x, x'\rangle),\ q \in val(\langle y, y'\rangle), \text{ dann ist } [x, x'] \cap [y, y'] = \emptyset.$$

D.h. wir stellen sicher dass die Intervalle, in denen verschiedene Propositionen in P gelten, disjunkt sind. Anders gesagt: wenn wir P als Menge von Prozessen auffassen, verlangen wir dass niemals zwei Prozesse in P gleichzeitig laufen. Wie axiomatisieren wir diese Menge?

Zunächst gehen wir zurück zu den Allen Relationen. Wir könnten versuchen, die Eigenschaft der Disjunktheit aus ihnen zu definieren; aus logischen Gründen ist es aber sinniger, ihr *Komplement* zu definieren. Wir nennen diese Relation s (für "schneidet"); sie lässt sich definieren als:

(30) $\mathsf{s} := \mathsf{w} \cup \mathsf{u} \cup \mathsf{b} \cup \mathsf{e} \cup \mathsf{t} \cup \mathsf{1} \cup \mathsf{w}^{-1} \cup \mathsf{u}^{-1} \cup \mathsf{b}^{-1} \cup \mathsf{e}^{-1} \cup \mathsf{t}^{-1}$

Man beachte:

(31) $\mathsf{s} = \mathsf{s}^{-1}$

Das sehr hilfreich; wir können damit eine Modalität $\langle \mathsf{s} \rangle$ definieren:

(32) $\langle \mathsf{s} \rangle \alpha := \langle \mathsf{w} \rangle \alpha \vee \langle \mathsf{u} \rangle \alpha \vee ... \vee \langle \mathsf{e}^{-1} \rangle \alpha \vee \langle \mathsf{t}^{-1} \rangle \alpha$

Man beachte dass man s auch anders und äquivalent definieren kann in Allens Algebra:

(33) $\mathsf{s} = \overline{\mathsf{p} \cup \mathsf{p}^{-1}}$

Allerdings lässt sich dieser Term nicht so einfach in HS übersetzen, da

(34) $\langle \mathsf{s} \rangle \alpha \not\equiv \neg(\langle \mathsf{p} \rangle \alpha \vee \langle \mathsf{p}^{-1} \rangle \alpha)$

wie sich leicht verifizieren lässt.

Es handelt sich also um eine rein syntaktische Abkürzung, die aber durchaus die intendierte semantische Bedeutung hat. Nun nehmen wir eine etwas vereinfachte Form unseres Beispiels, nämlich: wir haben zwei Propositionen p, q, und möchten dass diese nun in disjunkten Intervallen gelten. Die entsprechende Formel ist dann:

(35) $p \rightarrow [\mathsf{s}] \neg q$

Nun ist es einfach, aus mittels dieser Formel die eigentlich gewünschte zu konstruieren:

(36) $$\alpha_{Dis} := \bigwedge_{p \in P} \left(p \rightarrow \bigwedge_{q \in P - \{p\}} [\mathsf{s}] \neg q \right)$$

Damit lässt sich leicht das folgende zeigen:

Lemma 36 $\mathsf{Mod}(\alpha_{Dis}) = \mathbf{Dis}(\mathbf{P})$.

Beachten Sie aber dass (36) nur eine Abkürzung ist; die vollständige Formel wäre viel komplizierter. Die Schritte, die wir hier durchlaufen haben um zu (36) zu kommen sind aber sehr lehrreich, da man in Logik und Modelltheorie sehr oft ähnlich vorgeht um komplexe Formeln zu konstruieren.

Beispiel 2 Nehmen wir wieder unsere Menge $\mathbf{P}$, und nehmen wir an, s, s' sind zwei besondere Prozesse, für die wir folgendes verlangen: wann immer ein Prozess in $\mathbf{P}$ von s beendet wird, soll jeder unmittelbar anschließende Prozess (in $\mathbf{P}$) mit s' beginnen. Wir nennen die Klasse der Modelle, die das erfüllt

$$\mathbf{T}(s, s')$$

Wir können diese Eigenschaft natürlich versuchen sicherstellen, indem wir verlangen:

(37) $\beta := s \rightarrow \langle \mathsf{t} \rangle s'$

wobei es leicht zu sehen ist dass $\mathsf{Mod}(\beta) \supseteq \mathbf{T}(s, s')$ (vorausgesetzt $s, s' \notin \mathbf{P}$). In $\mathbf{T}(s, s')$ gibt es aber durchaus Modelle mit Intervallen, die s erfüllen, an die sich kein Intervall anschliesst das s' erfüllt! Also müssen wir eine etwas komplexere Formel finden. Die Formeln, die besagen dass s p beendet (und s' p anfängt) sehen wie folgt aus:

(38) $$s \wedge \langle \mathsf{e} \rangle p$$

(39) $$s' \wedge \langle \mathsf{b} \rangle p$$

Das ist vielleicht unintuitiv, läßt sich aber leicht verifizieren. Übrigens ist das äquivalent zu

(40) $$p \wedge \langle \mathsf{e}^{-1} \rangle s$$

(41) $$p \wedge \langle \mathsf{b}^{-1} \rangle s'$$

(nicht im Hinblick auf ein konkretes Intervall, aber im Hinblick auf Gültigkeit der Formel in *allen* Intervallen. Zusätzlich brauchen wir noch eine Formel, die besagt dass zwei Intervalle, in denen jeweils p bzw. q gilt, aneinander anschließen. Das ist

(42) $p \wedge \langle \mathsf{t} \rangle q$

Das ist aber noch nicht was wir brauchen; schließlich müssen wir über *jedes* anschließende Intervall sprechen, in dem p gilt.

(43) $(s \wedge \langle \mathsf{e} \rangle p) \rightarrow [\mathsf{t}](q \rightarrow \langle \mathsf{b}^{-1} \rangle s')$

Hier bettet $[\mathsf{t}]$ die nachfolgende Formel komplett ein; sonst würde sich $\langle \mathsf{b}^{-1} \rangle$ noch auf das Referenzintervall beziehen, nicht auf das anschließende Interval.

Die Formel ist aber noch nicht vollständig, weil wir hier nur über ein p und q reden. Die vollständige Formel lautet also:

$$(44) \quad \alpha_{s,s'} := \bigwedge_{p,q \in P} (s \wedge \langle \mathsf{e} \rangle p) \to [\mathsf{t}](q \to \langle \mathsf{b}^{-1} \rangle s')$$

Das ist ausreichend:

Lemma 37 $\mathsf{Mod}(\alpha_{s,s'}) = \boldsymbol{T}(s, s')$.

Beispiel 3 Wir haben gesehen dass A, die Allen Relationen, die Atome sind von Allens Intervallalgebra. Daraus folgt: jede Relation R in der Algebra hat die Form

(45) $R = \bigcup X$ wobei $X \subseteq \mathsf{A}$

(beachte den Spezialfall $X = \emptyset$). Das bedeutet: sei $R \in \mathbf{AA}$ eine beliebige Relation in Allens Algebra. Dann können wir

(46) $\langle R \rangle \alpha$

(mit der offensichtlichen Bedeutung) ausdrücken als

(47) $\bigvee_{\mathsf{a} \in X} \langle \mathsf{a} \rangle \alpha$

Ebenso können wir :

(48) $[R]\alpha$

(mit der offensichtlichen Bedeutung) ausdrücken als

(49) $\bigwedge_{\mathsf{a} \in X} [\mathsf{a}]\alpha$

Aber wie gesagt Vorsicht: Operationen wie $\overline{\ }$ lassen sich nicht einfach übersetzen als Negation!

Beispiel 4 Wir betrachten nun linguistische Beispiele:

(1) a. Er war am lesen als ich ankam.
b. Während ich saß, fing er an zu rennen.

Hier können wir die Ereignisse "lesen(er)", "ankommen(ich)" etc. als atomare Propositionen auffassen, uns interessiert nur wie sie in Relation stehen. Wie sehen die entsprechenden Formeln aus?

Auch interessant: das sog. "imperfective paradox", das so im Englischen besser funktioniert:

(2) a. Max built a house.
b. Max was building a house.

a) bedeutet: es gibt ein Intervall in der Vergangenheit wo Ereignis p stattgefunden hat, b) bedeutet: es gibt ein Intervall in der Vergangenheit, welches in einem Intervall liegt, in welchem p stattfindet. Interessant ist:

(3) Max was building a house $\not\models$ Max built a house

Wir knnen diese Konzepte teilweise in HS fassen, kommen da aber an unsere Grenzen.

14.5 Lokale und globale Konsequenz

Für modale Logiken gibt es tatsächlich zwei Arten, Konsequenz zu definieren. HS eignet sich hervorragend, um das zu illustrieren.

Definition 38 *Wir schreiben $\alpha \models \beta$ falls $M \models \alpha$ implizert $M \models \beta$. Man nennt das die globale Konsequenz: aus der globalen Wahrheit von α in M (implizite universelle Quantifikation über Referenzintervalle!) folgt die globale Wahrheit von β in M.*

Definition 39 *Wir schreiben $\alpha \models_l \beta$ falls f.a. (x, y) gilt. $M, (x, y) \models \alpha$ implizert $M, (x, y) \models \beta$. Man nennt das die lokale Konsequenz: aus der lokalen Wahrheit von α in $M, (x, y)$ folgt die lokale Wahrheit von β in $M, (x, y)$ (also an derselben Stelle). Erst danach wird universell quantifiziert!*

Wichtig ist: die beiden Fallen nicht zusammen, lokale Konsequenz ist echt stärker als globale:

Lemma 40 *Falls $\alpha \models_l \beta$ dann gilt $\alpha \models \beta$.*

Beweis Nimm an, $\alpha \models_l \beta$, also falls $M, (x, y) \models \alpha$, dann $M, (x, y) \models \beta$. Nimm nun an, für alle (x, y) gilt $M, (x, y) \models \alpha$. Dann gilt natürlich auch $M, (x, y) \models \beta$. Also $\alpha \models \beta$. ⊣

Lemma 41 *Es gibt Formeln α, β so dass $\alpha \models \beta$, aber nicht $\alpha \models_l \beta$.*

Beweis Nimm $\alpha = \langle e \rangle p$, $\beta = \langle w \rangle p$. ⊣

Wir haben also zwei Konsequenzrelationen!

Beweis Nimm $\alpha = \langle e \rangle p$, $\beta = \langle w \rangle p$. ⊣

Übung Entscheiden Sie, ob folgende Implikationen **global** gültig sind. Es muss kein streng formaler Beweis sein, sondern ein überzeugendes Argument. Eine Skizze ist normalerweise wesentlich einfacher und besser zu verstehen.

Also schreiben Sie erst, wenn Sie sich eine genaue Idee gemacht haben!

(50) $(\langle w\rangle q) \to [t]r \models (\langle e\rangle q) \to \langle b\rangle r$

(51) $\langle w\rangle(r \to [w^{-1}]q) \models r \to q$

(52) $\langle p^{-1}\rangle q \models [p]\langle p^{-1}\rangle q$

(53) $[p]\langle p^{-1}\rangle q \models \langle p^{-1}\rangle q$

(54) $p \to [w]\neg p \models (\langle w\rangle p) \to \neg p$

(55) $(\langle t\rangle q \wedge \langle t^{-1}\rangle q) \to q \models (\langle e\rangle q) \to (\langle b\rangle q)$

(Musterlösung für 1. und 2.)

Ich gebe hier die Lösung für 1. und 2. Wichtig ist zunächst folgendes: wenn wir sagen $M \models \alpha$, dann heißt dass, für alle $\langle x, y\rangle$ gilt $M, \langle x, y\rangle \models \alpha$. Das bedeutet aber insbesondere dass wir hier nicht das Prinzip der Bivalenz haben: für beliebige α gilt entweder $M, \langle x, y\rangle \models \alpha$ oder $M, \langle x, y\rangle \models \neg\alpha$ (eines der beiden muss wahr sein); aber es kann durchaus sein dass $M \not\models \alpha$ und $M \not\models \neg\alpha$ – beide Formeln können in verschiedenen Referenzintervallen falsch sein. Das sollten wir also im Kopf behalten.

Lösung zu 1 (Wichtig: Skizze machen, was entscheidend ist, ist die Relation der zwei Intervalle in denen jeweils q und r gilt!). Korrekt. Nimm an, wir haben $M \models (\langle w\rangle q) \to [t]r$. Nimm weiterhin an,(für beliebiges $\langle x, y\rangle$) dass $M, \langle x, y\rangle \models \langle e\rangle q$ (sonst ist die Formel trivial erfüllt an diesem Referenzintervall). Das bedeutet, es gibt $x' < x$ so dass $M, \langle x', y\rangle \models q$.

Wir zeigen nun, dass gilt: $M, \langle x, y\rangle \models \langle b\rangle r$, was bedeutet: es gibt y' so dass $M, \langle x, y'\rangle \models r$

Nimm nun das Intervall $\langle z_1, x\rangle$, so dass $x' < z_1 < x (< y)$. Natürlich gilt, qua Annahme, $M, \langle z_1, x\rangle \models \langle w\rangle q$, und deswegen, da $M \models (\langle w\rangle q) \to [t]r$, für beliebigies $z_2 > x$, $M, \langle x, z_2\rangle \models r$. Also insbesondere: $M, \langle x, y + 1\rangle \models r$.

Daraus folgt nun: $M, \langle x, y\rangle \models \langle b\rangle r$, also, da $\langle x, y\rangle$ beliebig gewählt war, $M \models (\langle e\rangle q) \to \langle b\rangle r$.

Lösung zu 2 Falsch; wir konstruieren also ein Gegenbeispiel. Entscheidend ist hier: wenn in einem Intervall r gilt, gilt in jedem echten Teilintervall q, aber nicht in dem Intervall selbst. Wir setzen also z.B.:

$val(\langle 0, 1\rangle) = r$, $val(\langle x, y\rangle) = q$ für all $0 < x < y < 1$;

ansonsten $v(\langle z, z'\rangle) = \emptyset$. Es ist leicht zu verifizieren dass $M \models \langle w\rangle(r \to [w^{-1}]q)$, aber $M, \langle 0, 1\rangle \not\models r \to q$.

Lösung zu 3 Korrekt. Beweis mit Kontraposition. Nimm an, $M \not\models [p]\langle p^{-1}\rangle q$, also: es gibt (x, x') so dass $M, (x, x') \not\models [p]\langle p^{-1}\rangle q$. Also: es gibt (z, z') so dass $x' < z$ und $M, (z, z') \not\models \langle p^{-1}\rangle q$. Daraus folgt bereits: $M \not\models \langle p^{-1}\rangle q$, QED.

Übung Gelten die obigen Konsequenzen lokal?

14.6 Entscheidbarkeit von HS

Wir sagen eine HS-Formel α ist ein **Theorem**, falls für alle Modelle $(\mathbb{R}, val)$ gilt:

$$(\mathbb{R}, val) \models \alpha$$

Weiterhin schreiben wir, wie üblich:

$$\alpha \models \beta \text{ gdw. } (\mathbb{R}, val) \models \alpha \text{ impliziert } (\mathbb{R}, val) \models \beta$$

Die interessantesten Ergebnisse sind:

Theorem 42 *(Unentscheidbarkeit von HS)*

1. *Es ist i.A. unentscheidbar, ob eine HS-Formel α ein Theorem ist.*
2. *Es ist i.A. unentscheidbar, ob für zwei HS-Formeln α, β $\alpha \models \beta$ gilt.*

Der Beweis ist einigermaßen kompliziert und läuft über die Kodierung von Turingmaschinen.

Übung

Zu bearbeiten bis zum 2.7.2018, Abgabe wie immer vor dem Seminar!

1. Konstruieren Sie die Relation u aus den anderen Allen Relationen (ausgenommen natürlich u^{-1}) mittels $\cup, \cap, \circ, [-]^{-1}, \overline{[-]}$.
2. Schreiben Sie eine HS-Formel α, so dass gilt: $\mathcal{M} \models \alpha$, genau wenn in $\mathcal{M}$ gilt: an jedes Intervall, in dem p gilt, schließt immer unmittelbar ein Intervall an, in dem q gilt, und umgekehrt, ausgenommen es läuft bereits in Intervall mit q (bzw. p) zum Endzeitpunkt des Intervalls von p (bzw. q); in diesem Fall schließt nicht der andere Prozess an.

15 Logiken für Intervalle II: Chop seine Pseudo-Residuen

15.1 Syntax und Semantik der Logik CDT

Wir betrachten nun eine weitere Intervall-basierte Logik, die in gewissem Sinne "logischer" ist. Wir nennen diese Logik CDT, nach ihren drei Operatoren. Die Syntax ist schnell definiert:

1. Falls $p \in Var$, dann ist p eine CDT-Formel;
2. falls α, β CDT-Formeln sind, dann sind es auch $(\alpha \wedge \beta), (\alpha \vee \beta), (\neg\alpha), (\alpha \rightarrow \beta)$;
3. falls α, β CDT-Formeln sind, dann sind es auch $(\alpha C \beta)$, $(\alpha D \beta)$, $(\alpha T \beta)$;
4. π ist eine CDT-Formel
5. und sonst nichts.

Wir haben also eine Aussagenlogik mit 3 zusätzlichen binären Operatoren, die man nur bedingt als Modaloperatoren auffassen kann. Modelle sind ebenso definiert wie HS-Modelle, Wahrheit ist definiert im Hinblick auf Intervalle; die klassischen Konnektoren (und Atome) werden klassisch interpretiert, und wir müssen also nur die Semantik der neuen Konnektoren betrachten. Hier steht

- C für *chop*, was soviel heißt wie "abschneiden"; wir sehen gleich dass C ein Intervall in zwei Teile schneidet
- D steht für *done*,
- T für *to come*,

denn D bezieht sich immer auf einen Punkt der *vor* unserem Referenzintervall liegt, T auf einen Punkt der *nach* diesem Intervall liegt.

$(\mathbb{R}, val), \langle x, y\rangle \models p$ gdw. $p \in val(\langle x, y\rangle)$, für $p \in prop$.

$(\mathbb{R}, val), \langle x, y\rangle \models \neg\alpha$ gdw. $(\mathbb{R}, val), \langle x, y\rangle \not\models \alpha$

$(\mathbb{R}, val), \langle x, y\rangle \models \alpha \wedge \beta$ gdw. $(\mathbb{R}, val), \langle x, y\rangle \models \alpha$ und $(\mathbb{R}, val), \langle x, y\rangle \models \beta$

$(\mathbb{R}, val), \langle x, y\rangle \models \alpha C \beta$ gdw. es ein $z : x \leq z \leq y$ gibt, so dass $(\mathbb{R}, val), \langle x, z\rangle \models \alpha$ und $(\mathbb{R}, val), \langle z, y\rangle \models \beta$

$(\mathbb{R}, val), \langle x, y\rangle \models \alpha D \beta$ gdw. es ein $z : z \leq x$ gibt, so dass $(\mathbb{R}, val), \langle z, x\rangle \models \alpha$ und $(\mathbb{R}, val), \langle z, y\rangle \models \beta$

$(\mathbb{R}, val), \langle x, y\rangle \models \alpha T \beta$ gdw. es ein $z : y \leq z$ gibt so dass $(\mathbb{R}, val), \langle y, z\rangle \models \alpha$ und $(\mathbb{R}, val), \langle x, z\rangle \models \beta$

$(\mathbb{R}, val), \langle x, y\rangle \models \pi$ gdw. $x = y$.

π hat eine sehr einfache Semantik: es denotiert einfach nur das Punktintervall. Um den Zusammenhang von C, D, T besser zu verstehen, malt man sich das ganze am besten auf. Die Bedingungen, die die drei verbinden haben zu tun mit dem Gesetz des Residuums (dazu später mehr); anders gesagt: D und T sind so etwas wie die Residuen von C (technisch stimmt das nicht ganz, aber die Vorstellung hilft).

Wir schreiben M für ein Modell $(\mathbb{R}, val)$; wir schreiben $M \models \alpha$ falls für alle $x, y \in \mathbb{R}$ $(x \leq y)$ gilt: $M, \langle x, y\rangle \models \alpha$.

15.2 Übung

Wir axiomatisieren (Lösung auskommentiert):

1. Jeder Punkt in einem Modell erfüllt
2. Falls ein Intervall p erfüllt, erfüllt ein Punkt darin p
3. Falls ein Intervall p erfüllt, erfüllt jeder Punkt darin p.
   Äquivalent das Kontrapositum: falls ein Punkt nicht p erfüllt, liegt er nicht in einem Intervall, das p erfüllt.
4. Was bedeutet $M, \langle x, y\rangle \models qT(qDp)$?

15.3 Übung

Wir simulieren HS, und zwar folgende Modalitten:

1. $\langle w\rangle\alpha$
2. $\langle t\rangle\alpha$
3. $\langle p\rangle\alpha$
4. $\langle u\rangle\alpha$
5. $\langle b\rangle\alpha$
6. $\langle e\rangle\alpha$

15.4 Expressivität – HS simulieren

Betrachten wir einmal die Formel $\neg\pi D(\neg\pi Tp)$. Das wird erfüllt in $M, \langle x, y\rangle$ genau dann wenn p erfüllt wird in $M, \langle z_1, z_2\rangle$, wobei $z_1 < x \leq y < z_2$. Hier sehen wir also folgenes: da HS Modelle genauso aussehen wir CDT Modelle, haben wir

(56) $M, \langle x, y\rangle \models \langle w\rangle p \quad \Leftrightarrow M, \langle x, y\rangle \models \neg\pi D(\neg\pi Tp)$

Wir können also die Modalität $\langle w\rangle$ *simulieren* in CDT. Wir machen das jetzt mit anderen Modalitäten:

(57) $M, \langle x, y\rangle \models \langle t\rangle p \quad \Leftrightarrow \quad M, \langle x, y\rangle \models pT\neg\pi$

(58) $M, \langle x, y\rangle \models \langle p\rangle p \quad \Leftrightarrow \quad M, \langle x, y\rangle \models (\neg\pi Cp)T\neg\pi$

(59) $M, \langle x, y\rangle \models \langle u\rangle p \quad \Leftrightarrow \quad M, \langle x, y\rangle \models \neg\pi C(\neg\pi Tp)$

Die entsprechenden Box-Modalitäten lassen sich leicht definieren; erinnern wir uns, dass $[\mathsf{x}]\alpha \equiv \neg\langle\mathsf{x}\rangle\neg\alpha$; dementsprechend:

(60) $M, \langle x, y\rangle \models [t]p \quad \Leftrightarrow \quad M, \langle x, y\rangle \models \neg((\neg p)T\neg\pi)$

(61) $M, \langle x, y\rangle \models [p]p \quad \Leftrightarrow \quad M, \langle x, y\rangle \models \neg((\neg\pi C\neg p)T\neg\pi)$

(62) $M, \langle x, y\rangle \models [u]p \quad \Leftrightarrow \quad M, \langle x, y\rangle \models \neg(\neg\pi C(\neg\pi T\neg p))$

Die Inversion der Relation ist aber nicht ganz einfach: so gilt z.B.

(63) $M, \langle x, y\rangle \models \langle w^{-1}\rangle p \quad \Leftrightarrow M, \langle x, y\rangle \models \neg\pi C(pC\neg\pi)$

Wir haben gezeigt, dass wir in gewissem Sinne HS in CDT simulieren können: anstelle von p in den obigen Beispielen können wir ja eine beliebige Formel α setzen. Damit haben wir gezeigt (noch nicht ganz):

Lemma 43 *Für jede Formel α in HS gibt es eine stark äquivalente Formel in CDT α^*, so dass $M, \langle x, y\rangle \models \alpha$ gdw. $M, \langle x, y\rangle \models \alpha^*$*

Beweis : Wir können α^* induktiv aus α definieren:

$$\begin{array}{rcl} p^* & = & p, \text{ für } p \in Var \\ (\alpha \wedge \beta)^* & = & \alpha^* \wedge \beta^* \\ (\alpha \vee \beta)^* & = & \alpha^* \vee \beta^* \\ (\alpha \rightarrow \beta)^* & = & \alpha^* \rightarrow \beta^* \\ (\neg\alpha)^* & = & \neg\alpha^* \\ (\langle w\rangle\alpha)^* & = & \neg\pi D((\alpha^* \wedge \neg\pi)T\neg\pi) \\ ([w]\alpha)^* & = & \neg(\langle w\rangle\neg\alpha)^* \\ & etc. & \end{array}$$

⊣

Das bedeutet soviel wie: wir können HS **einbetten** in CDT. Daraus folgt unmittelbar:

Lemma 44 *Die Logik CDT ist unentscheidbar.*

Denn wäre sie entscheidbar, dann wäre ja auch HS entscheidbar (qua Übersetzung).

Die Frage ist nun: können wir eine CDT Formel finden, die sich nicht in HS ausdrücken lässt? Die Antwort ist erstmal eindeutig ja: wir können ja in HS nicht über Punkte sprechen.

Lemma 45 *Folgende Formeln haben keine stark äquivalenten Formeln in HS:*

1. *pCq*
2. *pTq*
3. *pDq*

Wir müssen also gar nicht lange suchen um solche Formeln zu finden, es sind die einfachsten modalen Formeln. Um zu verstehen warum das so ist, betrachten wir einmal folgende Beispiel (1. ist klar, da HS keine trivialen Intervalle zulässt):

$$p \wedge \langle t \rangle q \tag{64}$$

$$\langle b^{-1} \rangle p \wedge \langle e^{-1} \rangle q \tag{65}$$

$$\langle t \rangle p \wedge \langle b \rangle q \tag{66}$$

$$\langle t^{-1} \rangle p \wedge \langle e \rangle q \tag{67}$$

Was ist der Unterschied zwischen den HS-Formeln und den obigen CDT Formeln?

- Der Unterschied zwischen 1 und 64 liegt im Referenzintervall, dass jeweils anders ist. Für 65 hingegen ist das Referenzintervall gleich, aber der Endpunkt von p und Anfangspunkt von q sind verschieden.
- Der Unterschied zwischen 2 und 66 liegt darin, dass in 66 die Endpunkte der beiden Intervalle unabhängig sind, in 2 aber identisch;
- parallel für 4 und 67.

Wir haben also in jeder versuchten HS-Übersetzung einen Punkt (bzw. eine Identität von Punkten), die man nicht ausdrückt. Das ist natürlich kein Beweis, sondern nur eine Illustration. Was das aber vor allem zeigt ist die Mächtigkeit binärer Modalitäten: binäre Modalitäten können unäre Modalitäten normalerweise sehr einfach simulieren; das Gegenteil ist normalerweise sehr schwierig oder unmöglich.

Etwas formaler können wir folgendes sagen: jede CDT Modalität führt, in einem Referenzinterval $\langle x, y \rangle$ zwei Intervalle und 3 Gleichungen ein; wir haben die beiden Intervalle $I_p = \langle a_1, a_2 \rangle$, $I_q = \langle b_1, b_2 \rangle$, und die Gleichungen sind:

- Für pCq,
  1. $a_1 = x$
  2. $b_2 = y$
  3. $a_2 = b_1$

- Für pDq,
    1. $a_2 = x$
    2. $b_2 = y$
    3. $a_1 = b_1$
- Für pTq,
    1. $a_1 = y$
    2. $b_1 = x$
    3. $a_2 = b_2$

Dem gegenüber führt HS mit jeder Modalität höchstens eine Gleichungen ein, z.B. für $\langle t \rangle p$ haben wir $a_1 = y$. Ausserdem können wir mit neuen Formeln nicht mehr auf die alten Variablen von alten Intervallen zugreifen; so kann man zeigen, dass sich die Bedingungen aus CDT nicht ausdrcken lassen.

15.5 Expressivität global

Wir haben gesehen, dass es für jede HS-Formal eine lokal äquivalente CDT-Formel gibt. Wir definieren jetzt das KOnzept der globalen Äquivalenz:

Definition 46 *Zwei Formeln α, β sind **global äquivalent**, falls gilt: $M \models \alpha$ gdw. $M \models \beta$.*

Offensichtlich gilt: falls α, β stark äquivalent sind, dann sind sie auch global äquivalent. Das Gegenstück gilt allerdings nicht (beachte die vielen Beispiele). Daraus folgt natürlich auch:

- Für jede HS-Formel α gibt es eine global äquivalente CDT-Formel (nämlich α^* wie oben)
- Aber gilt diesmal auch das Gegenstück? Ich glaube nicht, aber die Sache ist deutlich komplizierter. Ich lasse das hier mal offen.

Nehmen wir die Formel pCq. Es gilt $M \models pCq$ gdw. jedes Intervall sich choppen lässt in zwei Teilintervalle, eines erfüllt p, eines q. Nimm also ein beliebiges Intervall $\langle x, y\rangle$. Nimm es, $M, \langle x, y\rangle \not\models p$.

Übung Was bedeuten die folgenden Formeln für Modelle?

1. $(pDq)T(q \wedge r)$
2. $p \rightarrow (pTq)$

Aufgabe 10

1. Definieren Sie die übrigen 3 Allen Relationen (ohne die Inversionen).
2. Erklären Sie, warum es die Inversionen aber dann praktisch frei Haus gibt.

16 CDT und die Algebra der Relationen

16.1 Interpretation als Relatationen

Wir haben bislang immer eine Valuation $val : \mathbb{I}(\mathbb{R}) \to \wp(Var)$ gehabt. Gegeben ein Modell $M = (\mathbb{R}, val)$ können wir auch umgekehrt Formeln als Relationen interpretieren. Wir definieren dazu

$$int_M(p) = \{\langle x, y\rangle : p \in val(\langle x, y\rangle)\}$$

Wir haben also eine **Interpretation** in einem Model. Das ist soweit geradeaus. Das interessante ist: wir können int_M auf beliebige Formeln erweitern.

$$\begin{aligned}
int_M(\alpha \wedge \beta) &= int_M(\alpha) \cap int_M(\beta) \\
int_M(\alpha \vee \beta) &= int_M(\alpha) \cup int_M(\beta) \\
int_M(\neg(\alpha) &= \overline{int_M(\alpha)} \\
int_M(\alpha \to \beta) &= \overline{int_M(\alpha)} \cup int_{M\,M}(\beta) \\
int_M(\alpha C \beta) &= int_M(\alpha) \circ int_M(\beta) \qquad (!)
\end{aligned}$$

Soweit, sogut: wir haben Boolesche Operationen, und wir haben Komposition von Relationen, bekannt aus Allens Intervallalgebra. Was uns fehlt ist die **relationale Interpretation von T,D**. Wir können das natürlich aufschreiben, aber um das etwas besser zu verstehen sollten wir uns erst noch einmal die Algebra der Relationen anschauen.

16.2 Residuen von Relationen

Wir nehmen an, wir haben eine Relationenalgebra

$$(\mathbf{R}, \cap, \cup, \circ, \overline{[-]}, [-]^{-1}, \emptyset, \top, 1).$$

Hier ist

- 1 die Identitätsrelation $\{(x,x) : x \in M\}$, M die Domäne von $\mathbf{R}$.
- $\top = M \times M$.

Wir können in diesen Algebren auch das sog. **Residuum** definieren, und zwar für die Komposition. Komposition ist nicht kommutativ, daher gibt es das linke und das rechte Residuum; denotiert mit $R\backslash R'$, R'/R. Sie werden definiert durch:

$$R_1 \subseteq R_3/R_2 \text{ gdw. } R_1 \circ R_2 \subseteq R_3 \text{ gdw. } R_2 \subseteq R_1\backslash R_3$$

Die Residuen kann man auch explizit definieren:

$$R_1/R_2 = \{(x,y) : \text{ falls } (y,z) \in R_2, \text{ dann } (x,z) \in R_1\} \tag{68}$$

$$= \{(x,y) : \{(x,y)\} \circ R_2 \subseteq R_1\} \tag{69}$$

Parallel das linke Residuum:

$$R_2\backslash R_1 = \{(x,y) : \text{ falls } (z,x) \in R_2, \text{ dann } (z,y) \in R_1\} \tag{70}$$

$$= \{(x,y) : R_2 \circ \{(x,y)\} \subseteq R_1\} \tag{71}$$

Beispiel 1 Nimm die Algebra der Intervalle in $\mathbb{R}$, also

$$(\mathbb{I}(\mathbf{R}), \cap, \cup, \circ, \overline{[-]}, \emptyset, \top, 1).$$

WIr haben keine Inversion $[-]^{-1}$ in diesem Fall. Hier ist

- 1 die Identität $\{(x,x) : x \in \mathbb{R}\}$
- $\top = \{\langle x,y\rangle : x \leq y\}$

Sei $R_1 = \{\langle x,y\rangle : 0 \leq x \leq y \leq 1\}$, $R_2 = \{\langle x,y\rangle : x + 0.5 \leq y\}$.

- Was ist $R_1 \circ R_2$? $\{(x,y) : x \in [0,1],\ x \leq y - 0.5\}$
- Was ist R_1/R_2? $\emptyset$
- Was ist $R_2\backslash R_1$? $\emptyset$
- Was ist $R_1\backslash R_2$? $\{(x,y) : \text{falls } x \in [0,1], \text{ dann } x \leq y - 0.5\}$
- Was ist R_1/R_1?

Beispiel 2 (mit Allens Intervallrelationen) Erstmal ein Wort der Vorsicht zu den Konventionen: eine Allensche Intervallrelation ist eine Menge von Paaren von Paaren, da jedes Intervall wiederum ein Paar ist. Um zu vermeiden, dass wir und totschreiben, nehmen wir nun folgende Konvention:

- x, y, z etc. stehen für reelle Zahlen;
- I_1, I_2, I_3 stehen für Intervalle, also (bestimmte) Paare von reellen Zahlen.
- Wenn wir z.B. von I_1 sprechen, dann ist $I_1 = \langle x_1, y_1 \rangle$. Wir greifen also mit x_n auf den linken Rand von I_n zu, mit y_n auf den rechten Rand.

Aber Vorsicht: die Konvention beschränken sich auf den Fall dass wir von Allens Intervallrelationen sprechen; sonst stehen x, y etc. für beliebige Objekte!

Übung 1 Was sind folgende Relationen?

1. $e/1$
2. $1/e$
3. $\overline{1}/e$
4. $e\backslash\overline{1}$

Wir beantworten die Fragen der Reihe nach.

$$\begin{aligned} e/1 &= \{(I_1, I_2) : \text{ falls } (I_2, I_3) \in 1, \text{ dann } (I_1, I_3) \in e\} \\ &= \{(I_1, I_2) : \text{ falls } I_2 = I_3, \text{ dann } (I_1, I_3) \in e\} \\ &= \{(I_1, I_2) : (I_1, I_2) \in e\} \\ &= e \end{aligned}$$

Das ist einfach und eigentlich klar: wir haben $R \subseteq e/1$ genau dann wenn $R \circ 1 \subseteq e$; aber $R \circ 1 = R$, also ist e die größte Relation R, die das erfüllt.

Das nächste ist etwas schwieriger. Wir haben

$$\begin{aligned} 1/e &= \{(I_1, I_2) : \text{ falls } (I_2, I_3) \in e, \text{ dann } (I_1, I_3) \in 1\} \\ &= \{(I_1, I_2) : \text{ falls } (I_2, I_3) \in e, \text{ dann } I_1 = I_3\} \end{aligned}$$

Wir zeigen dass $1/e = \emptyset$. Denn nimm an, wir haben $(I_1, I_2) \in 1/e$. Wir haben dann unendlich viele Intervalle I_3 so dass $(I_2, I_3) \in e$ (der Startpunkt x_3 von I_3 ist ja beliebig, wir brauchen nur $x_3 > x_2$). Es gibt also insbesondere zwei konkrete Intervalle $I_3 \neq I_3'$. Nach Annahme folgt dass $I_1 = I_3$, aber ebenso $I_1 = I_3' \neq I_3$ - Widerspruch!

Wir kommen zu 3:

$$\overline{1}/e = \{(I_1, I_2) : \text{ falls } (I_2, I_3) \in e, \text{ dann } (I_1, I_3) \in \overline{1}\} \tag{72}$$

$$= \{(I_1, I_2) : \text{ falls } (I_2, I_3) \in e, \text{ dann } (I_1, I_3) \notin 1\} \tag{73}$$

$$= \{(I_1, I_2) : \text{ falls } (I_2, I_3) \in e, \text{ dann } I_1 \neq I_3\} \tag{74}$$

(75)

$(I_2, I_3) \in e$ bedeutet: $y_2 = y_3, x_2 > x_3$. Wir schliessen jetzt einmal umgekehrt was nicht in der Relation ist: wir haben $(I_1, I_2) \notin \overline{1}/e$, falls gilt: $y_1 = y_2$, und $x_1 \leq x_2$, denn sonst haben wir I_3 mit $y_3 = y_2 = y_1$, und $x_3 = x_1$, und $(I_2, I_3) \in e$. Daraus folgt also:

$$\overline{1}/e = \overline{e^{-1}} \tag{76}$$

Wir kommen zu 4:

$$e\backslash\overline{1} = \{(I_1, I_2) : \text{ falls } (I_3, I_2) \in e, \text{ dann } (I_3, I_1) \in \overline{1}\}$$

$$= \{(I_1, I_2) : \text{ falls } (I_3, I_1) \in e, \text{ dann } I_3 \neq I_2\}$$

Wir nehmen denselben Ansatz: wir beschreiben erstmal, was nicht darin ist. Wir haben $(I_1, I_2) \notin e\backslash\overline{1}$, falls es ein I_3 gibt so dass $(I_3, I_1) \in e$ und $I_3 = I_2$, also falls $(I_2, I_1) \in e$. Das bedeutet, wir haben auch hier ein paralleles Ergebnis.

$$e\backslash\overline{1} = \overline{1}/e = \overline{e^{-1}} \tag{77}$$

Übung 2 Woher kommt der Parallelismus, sprich die Kommutativität der obigen Terme?

Das liegt natürlich an der 1, die wir verwendet haben; wir haben ja $R \circ 1 = 1 \circ R = R$. Allerdings gilt es hier vorsichtig zu sein: $R \circ R' \subseteq 1$ impliziert nicht $R' \circ R \subseteq 1$.

Z.B. kann man auch leicht zeigen:

$$e \circ e^{-1} = \{(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle : y_1 = y_2\} = e^{-1} \circ e$$

Allerdings haben wir

$$\begin{aligned} p \circ e &= \{(I_1, I_2) : y_1 < y_2\} \\ &\neq p \\ &= e \circ p \end{aligned}$$

Übung 3 Was sind folgende Relationen?

1. e/w
2. $\overline{e}/w =$
3. $t \backslash t$
4. p/u
5. $\overline{p}/u$

16.3 Residuen in Relationenalgebren

Residuen lassen sich definieren in Relationen Algebren: Wir haben

$$R_1/R_2 = \{(x,z) : \text{falls } (z,y) \in R_2, \text{ dann } (x,y) \in R_1\}$$

Wir machen einen kleinen Umweg über das Komplement $\overline{R_1/R_2}$; da das Residuum über eine Implikation definiert ist, nehmen wir also einfach den Fall, wo die Implikation falsifiziert wird:

$$\begin{aligned}
\overline{R_1/R_2} &= \{(x,y) : \text{ es gibt } z : (y,z) \in R_2 \& (x,z) \notin R_1\} \\
&= \{(x,y) : \text{ es gibt } z : (y,z) \in R_2 \& (x,z) \in \overline{R_1}\} \\
&= \{(x,y) : \text{ es gibt } z : (z,y) \in R_2^{-1} \& (x,z) \in \overline{R_1}\} \\
&= \{(x,y) : \text{ es gibt } z : (x,z) \in \overline{R_1} \& (z,y) \in R_2^{-1}\} \\
&= \overline{R_1} \circ R_2^{-1}
\end{aligned}$$

Damit können wir nun das Residuum definieren:

$$\begin{aligned}
R_1/R_2 &= \overline{\overline{R_1} \circ R_2^{-1}} \\
&= R_1 \dagger \overline{R_2^{-1}}
\end{aligned}$$

Die letzte Umformung ist nicht so wichtig.

Wir definieren eine Residuierte Boolesche Algebra wie folgt:

$$(\mathbf{R}, \cap, \cup, \circ, \overline{[-]}, /, \backslash, \emptyset, \top, 1).$$

Eine RBA ist, im einfachen Fall, tatsächlich äquivalent zur einer Relationenalgebra:

Lemma 47 $R^{-1} = \overline{(R \backslash \overline{1})}$

$\overline{1}$, das Komplement der Identität, wird auch die Diversität genannt; sie setzt alles in Relation, was verschieden ist.

Beweis:

$\subseteq$ Nimm an, $(x,y) \in R^{-1}$, $x \neq y$. Zu zeigen: $(x,y) \notin R \backslash \overline{1}$. Nimm an $(x,y) \in R \backslash \overline{1}$. Dann ist für alle $(z,x) \in R$, $(z,y) \in \overline{1}$. Wir haben aber $(y,x) \in R$, und $(y,y) \notin \overline{1}$.

$\supseteq$ Nimm an, $(x,y) \in \overline{(R \backslash \overline{1})}$. Also ist $(x,y) \notin R \backslash \overline{1}$. Also gibt es $(y,z) \in R$ so dass $(x,z) \notin 1$. Also muss $z = x$ sein, also $(y,x) \in R$, also $(x,y) \in R^{-1}$. $\dashv$

Was wir also gezeigt haben (wir lassen ein Paar Details ausser Acht):

Lemma 48 *Die residuierte Boolesche Algebra der über Allens Intervallrelationen ist äquivalent zur Allens Relationenalgebra.*

16.4 Die residuierte Boolesche Algebra der reellen Intervalle

Nimm die Algebra der Intervalle in $\mathbb{R}$, also

$$\mathbf{A}\mathbb{R} = (\mathbb{I}(\mathbf{R}), \cap, \cup, \circ, \overline{[-]}, \emptyset, \top, 1).$$

Wir haben keine Inversion $[-]^{-1}$ in diesem Fall. Hier ist

- 1 die Identität $\{(x, x) : x \in \mathbb{R}\}$
- $\top = \{\langle x, y\rangle : x \leq y\}$

Dazu fehlt uns aber noch das folgende Ergebnis. Wie wir sehen werden, ist das das algebraische Gegenstück zu unserer logischen Übersetzung von Formeln.

Sei $val : I(\mathbb{R}) \to \wp(Var)$ die bekannte Intervall-Interpretation: jede Proposition wird interpretiert als eine Relation, die Intervalle in denen sie war ist, und

$$p \in val(\langle x, y\rangle \text{ gdw. } \langle x, y\rangle \in int_M(p)$$

Wir haben bereits gezeigt, wie wir die meisten Konnektoren von CDT hier interpretieren können in $\mathbf{A}\mathbb{R}$. Was noch fehlt ist T und D (und C und π). Wir machen das wir folgt:

$int_M(\pi) = id$

$$\begin{aligned} int_M(\alpha C\beta) &= \{\langle x, y\rangle : \text{ es gibt } z.\langle x, z\rangle \in int_M(\alpha), \langle z, y\rangle \in int_M(\beta)\} \\ &= int_M(\alpha) \circ int_M(\beta) \end{aligned}$$

$$\begin{aligned} int_M(\alpha D\beta) &= \{\langle x, y\rangle : \exists z.(\langle z, x\rangle \in int_M(\alpha) \text{ und } \langle z, y\rangle \in int_M(\beta)\} \\ &= \overline{\{\langle x, y\rangle : \forall z.(\langle z, x\rangle \in \overline{int_M(\alpha)} \text{ oder } \langle z, y\rangle \in \overline{int_M(\beta)}\}} \\ &= \overline{\{\langle x, y\rangle : \forall z.(\neg(\langle z, x\rangle \in int_M(\alpha) \text{ oder } \langle z, y\rangle \in \overline{int_M(\beta)}\}} \\ &= \overline{\{\langle x, y\rangle : \forall z.(\text{ falls } \langle z, x\rangle \in int_M(\alpha), \text{ dann } \langle z, y\rangle \in \overline{int_M(\beta)}\}} \\ &= \overline{int_M(\alpha) \backslash \overline{int_M(\beta)}} \end{aligned}$$

Parallel bekommen wir dazu

$$int_M(\alpha T\beta) = \overline{\overline{int_M(\beta)}/int_M(\alpha)}$$

Lemma 49 *Jede CDT-Formel über kann interpretiert werden als ein RBA-Term über* $\{int_M(p) : p \text{ kommt in } \alpha \text{ vor}\}$

Beweis. Wir interpretieren wie folgt:

$$
\begin{aligned}
&int_M(p) = val^{-1}(p)\\
&int_M(\alpha \wedge \beta) = int_M(\alpha) \cap int_M(\beta)\\
&int_M(\alpha \vee \beta) = int_M(\alpha) \cup int_M(\beta)\\
&int_M(\alpha \to \beta) = \overline{int_M(\alpha)} \cup int_M(\beta)\\
&int_M(\neg\alpha) = \overline{int_M(\alpha)}
\end{aligned}
$$

$$
\begin{aligned}
int_M(\alpha C \beta) &= \{\langle x, y\rangle : \text{ es gibt } z.\langle x, z\rangle \in int_M(\alpha), \langle z, y\rangle \in int_M(\beta)\}\\
&= int_M(\alpha) \circ int_M(\beta)
\end{aligned}
$$

$$int_M(\pi) = id$$

Soweit, sogut. Mit D, T wird es etwas komplizierter. Wir haben:

$$
\begin{aligned}
int_M(\alpha D \beta) &= \overline{\{\langle x, y\rangle : \exists z.(\langle z, x\rangle \in int_M(\alpha) \text{ und } \langle z, y\rangle \in int_M(\beta)\}}\\
&= \overline{\{\langle x, y\rangle : \forall z.(\langle z, x\rangle \in \overline{int_M(\alpha)} \text{ oder } \langle z, y\rangle \in \overline{int_M(\beta)}\}}\\
&= \overline{\{\langle x, y\rangle : \forall z.(\neg(\langle z, x\rangle \in int_M(\alpha) \text{ oder } \langle z, y\rangle \in \overline{int_M(\beta)}\}}\\
&= \overline{\{\langle x, y\rangle : \forall z.(\text{ falls } \langle z, x\rangle \in int_M(\alpha), \text{ dann } \langle z, y\rangle \in \overline{int_M(\beta)}\}}\\
&= \overline{int_M(\alpha) \backslash \overline{int_M(\beta)}}
\end{aligned}
$$

Parallel bekommen wir dazu

$$int_M(\alpha T \beta) = \overline{\overline{int_M(\beta)}/int_M(\alpha)}$$

⊣

Außerdem gilt natürlich folgendes:

Lemma 50 $M, \langle x, y\rangle \models \alpha$ *impliziert* $M, \langle x, y\rangle \models \beta$ *f.a.* $M, \langle x, y\rangle$ *(lokale Konsequenz) gdw. f.a.* int_M, $int_M(\alpha) \subseteq int_M(\beta)$.

Wir haben also ein sog. algebraische Semantik für unsere Logik CDT.

16.5 Relationale Addition

Jetzt wird es noch interessanter. Wir müssen zuerst definieren einen Konnektor †, dual zur Komposition ∘:

$$R \dagger R' = \overline{\overline{R} \circ \overline{R'}} = \{(x, y)\text{: für alle } z\text{, entweder } (x, z) \in R\text{, oder } (z, y) \in R'\}$$

Man nennt † auch die **relationale Addition**; sie ist DeMorgan dual zur Komposition. Man beachte dass wir haben:

$$int_M(\alpha) \dagger int_M(\beta) = int_M(\neg((\neg\alpha)C(\neg\beta)))$$

⊣

16.6 Zwei Algebren

Wir haben nun zwei Algebren:

1. Die (residuierte) Algebra der Intervalle.
2. Die Relationen Algebra der Intervallrelationen.

Wie verhalten sich diese beiden Algebren zueinander? Erstmal sind sie unvergleichbar, denn ihre Objekte sind jeweils anders.

Von 1. zu 2.: jeder Term mit zwei Variablen generiert eine Relation – die maximale Menge der Belegungen. So kann ich Relationen definieren. ? Aber wie genau, algebraisch? Logisch ist das klar...

Von 2. zu 1. Ebenfalls logisch??

Hier gibt es aber folgendes zu beachten: die Diversität existiert nicht, auch ist das Komplement nicht mengentheoretisch. Deswegen gibt es auch keine Inversion (das wren ja keine Intervalle). Irgendwie bleibt das unklar.