Skript zur Vorlesung

Ontologien im Semantic Web

(Wintersemester 2021/22)

Gehalten von Lutz Schröder
Mitgeschrieben im WS 2014/15 von Hans-Peter Deifel

Stand: 10. Februar 2022

Inhaltsverzeichnis

Überblick

Formale Wissensrepresentation (logikbasiert)

- Propositionale Logik: Algorithmik (DPLL, Tableaux)
- FOL: Unentscheidbarkeit, Ausdrucksstärke
- Beschreibungslogiken/Modallogiken
  - OWL
  - $\mathcal{ALC}/K_m$
  - ausdrucksstarke DL
    - transitive Rollen
    - Zählen
    - Nominale
    - TBoxen/ABoxen
  - leichtgewichtige DL: $\mathcal{EL}$, verwendet z.B. in SNOMED CT
  - Global Caching

Kapitel 1

Algorithmik der Aussagenlogik

Wir beginnen am weniger ausdrucksstarken Ende des Spektrums, der Aussagenlogik, und werfen einen genaueren Blick auf Algorithmen für Schlussfolgerung, äquivalenterweise Erfüllbarkeit, in Aussagenlogik; d.h. wir beschäftigen uns mit SAT-Solving-Algorithmen. Im einzelnen werden wir folgende Algorithmen behandeln:

- Resolution (Wiederholung aus GLoIn)
- DPLL (Davis/Putnam/Loge/Loveland), das den meisten „echten“ SAT-Solvern zugrundeliegende Verfahren (ein weiteres Verfahren, das wir hier aber nicht besprechen, ist der Stålmark-Algorithmus)
- Tableaux (ein im SAT-Solving eher nicht verwendetes Verfahren, das wir aber als Grundlage für Algorithmen in der Beschreibungslogik brauchen).

1.1 Aussagenlogik

Wir erinnern an die Syntax und Semantik der Aussagenlogik. Zur leichteren Definition von Normalformen nehmen wir von Anfang an Disjunktion mit in die Sprache auf:

$$\phi, \psi ::= \bot \mid \top \mid A \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \qquad (A \in \mathscr{A} \neq \emptyset)$$

Modelle sind Wahrheitsbelegungen $\kappa : \mathscr{A} \to 2 = \{\bot, \top\}$. *Erfülltheit* ist (wie schon in der Prädikatenlogik) eine Relation $\models$ zwischen Modellen und Formeln, rekursiv definiert durch

$$\begin{aligned}
\kappa &\not\models \bot \\
\kappa &\models \top \\
\kappa \models A &\iff \kappa(A) = \top \\
\kappa \models \neg\phi &\iff \kappa \not\models \phi \\
\kappa \models \phi \wedge \psi &\iff \kappa \models \phi \text{ und } \kappa \models \psi \\
\kappa \models \phi \vee \psi &\iff \kappa \models \phi \text{ oder } \kappa \models \psi.
\end{aligned}$$

Wiederum haben wir

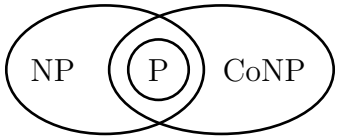
$$\begin{aligned} \phi \textit{ erfüllbar} &\iff \exists \kappa.\kappa \models \phi \\ \phi \textit{ gültig} &\iff \forall \kappa.\kappa \models \phi \\ \phi \text{ erfüllbar} &\iff \neg\phi \text{ nicht gültig} \\ \phi \text{ gültig} &\iff \neg\phi \text{ nicht erfüllbar.} \end{aligned}$$

Ferner ist eine Formel ψ eine *logische Folgerung* aus einer Menge Φ von Formeln ($\Phi \models \psi$), wenn jede Wahrheitsbelegung, die alle Formeln in Φ erfüllt, auch ψ erfüllt. Aus BFS ist bekannt:

Satz 1 (Cook)**.** *Efüllbarkeit in Aussagenlogik ist NP-vollständig.*

Wir haben bereits an Komplementärklassen erinnert; insbesondere hat man

$$\mathit{CoNP} = \{A \subseteq \{0,1\}^* \mid \{0,1\}^* - A \in \mathit{NP}\}.$$



Korollar 2. *Gültigkeit in Aussagenlogik ist CoNP-vollständig.*

1.2 Normalformen

Die meisten SAT-Solver arbeiten nicht auf beliebigen Formeln, sondern verlangen geeignete Normalformen, meist die konjunktive Normalform. Wir erinnern an die relevanten Begriffe aus GLoIn:

$$\begin{aligned} \phi \text{ NNF (Negationsnormalform)} &\iff \phi \text{ erzeugt durch} \\ &\quad \phi, \psi ::= A \mid \neg A \mid \phi \wedge \psi \mid \phi \vee \psi \\ \phi \text{ CNF (Konjunktive Normalform)} &\iff \phi \text{ erzeugt durch} \\ &\quad L ::= A \mid \neg A \\ &\quad C ::= \bot \mid L \vee C \text{ (Klauseln)} \\ &\quad \phi ::= \top \mid C \wedge \phi \end{aligned}$$

Man schreibt CNFs für viele Zwecke, insbesondere als Eingaben für Algorithmen, oft als Mengen von Klauseln, die wiederum als Mengen von Literalen geschrieben werden. Zur Unterscheidung notiert man dabei die leere CNF als $\top$ und die leere Klausel als $\square$. Dual zur CNF hat man *disjunktive Normalformen (DNF)*, also Disjunktionen von *konjunktiven Klauseln*, d.h. von Konjunktionen von Literalen.

Normalisierung zu NNF

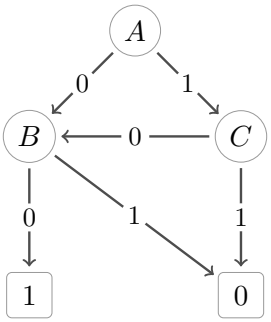
$$\begin{aligned} \neg(\phi \wedge \psi) &\equiv \neg\phi \vee \neg\psi \\ \neg(\phi \vee \psi) &\equiv \neg\phi \wedge \neg\psi \\ \neg\neg\phi &\equiv \phi \end{aligned}$$

Normalisierung von NNF zu CNF

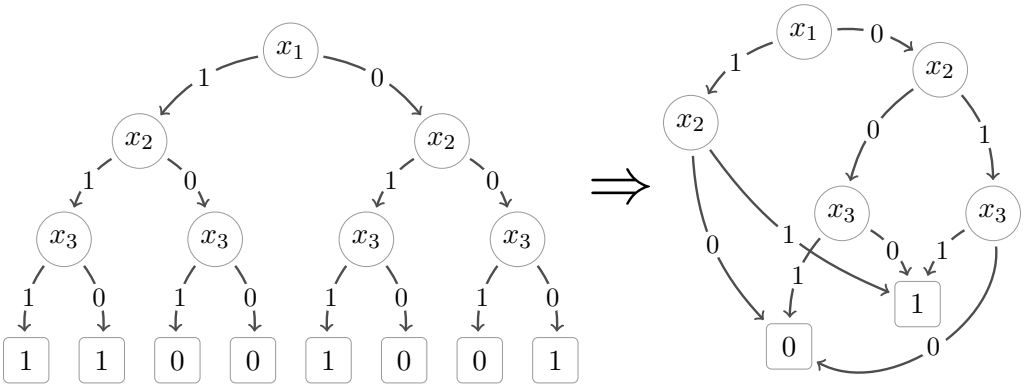
$$\phi \vee (\psi \wedge \xi) \longrightarrow (\phi \vee \psi) \wedge (\phi \vee \xi)$$
$$(\psi \wedge \xi) \vee \phi \longrightarrow (\psi \vee \phi) \wedge (\xi \vee \phi)$$

In schlechten Fällen ist die Größe einer CNF von ϕ exponentiell in der von ϕ. Dies lässt sich mittels zusätzlicher Literale zur Abkürzung von Teilformeln vermeiden; dann ist allerdings die CNF nur noch *erfüllbarkeitsäquivalent*, nicht mehr logisch äquivalent, zur ursprünglichen Formel.

Ordered binary decision diagrams (OBDD) Ein BDD ist ein gerichteter Graph mit einer Wurzel und genau zwei Senken, die mit 0 und 1 gekennzeichnet sind; alle anderen Knoten sind mit Atomen aus $\mathscr{A}$ gekennzeichnet. Aus jedem Knoten (außer den Senken) starten zwei Kanten, die mit 0 und 1 markiert sind. Ein BDD berechnet in offensichtlicher Weise einen Wahrheitswert aus einer Wahrheitsbelegung als Eingabe.



Ein OBDD ist ein BDD mit der Eigenschaft, dass auf jedem Pfad von der Wurzel zu den Senken die Atome (so weit sie vorkommen) eine gegebene Reihenfolge (von kleiner zu größer) einhalten. (Ist obiger BDD ein OBDD?) Eine zentrale Rolle spielt dabei die *Reduktion* von OBDDs (Entfernung von Knoten mit irrelevantem Wert; Identifikation von Knoten, unter denen isomorphe Teilgraphen hängen), wie im folgenden Beispiel:



$$\phi = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2) \vee (x_2 \wedge x_3), \quad x_1 < x_2 < x_3$$

(Was passiert, wenn man ϕ auf $\phi' = (\neg x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_2) \vee (x_2 \wedge x_3)$ abändert?)

Die Komplexität der veschiedenen Schlussfolgerungsprobleme und Berechnungsprobleme in der Aussagenlogik hängt stark von der verwendeten Darstellung ab:

	Gültigkeit	Erfüllbarkeit	$\neg$	$\wedge$	$\vee$
Formeln	CoNP	NP	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
CNFs	Logspace	NP (SAT)	schwer	$\mathcal{O}(1)$	schwer
DNFs	CoNP	Logspace	schwer	schwer	$\mathcal{O}(1)$
OBDD	P	P	$\mathcal{O}(1)$	mittel	mittel

(In etlichen Fällen hängt hierbei die konstante Laufzeit von Operationen an einer passenden Repräsentation von Datenstrukturen auf dem Heap.)

1.3 Resolution

(Siehe GLoIn) Der Resolutionsalgorithmus besteht in der erschöpfenden Anwendung der *Resolutionsregel*

$$\frac{\{A\} \cup C \qquad \{\neg A\} \cup D}{C \cup D}\ \textsc{Res}$$

auf eine CNF, in dem Sinne, dass man zu einer CNF, die die beiden Klauseln in der Prämisse enthält, die Klausel in der Konklusion hinzufügt. Wir fassen hierbei CNFs als Mengen von Klauseln und Klauseln als Mengen von Literalen auf.

Der Algorithmus kann auf eine von zwei Weise terminieren:

- entweder (Res) ist irgendwann nicht mehr anwendbar und die dann erreichte CNF enthält die leere Klausel $\square$ nicht. Dann ist die CNF erfüllbar.
- Die leer Klausel $\square$ wird irgendwann erzeugt; dann ist die CNF nicht erfüllbar.

Beispiel 3. Die Herleitung

$$\dfrac{\dfrac{\dfrac{\{\neg C, B, \neg A\}\ \{C, B, \neg A\}}{\{B, \neg A\}}\quad \{\neg B, \neg A\}}{\{\neg A\}}\quad \dfrac{\{\neg B, A\}}{\{\neg B\}}\qquad \dfrac{\dfrac{\{D, B, \neg C\}\ \{D, C\}}{\{D, B\}}\quad \{\neg D, B\}}{\{B\}}}{\square}$$

zeigt, dass die aus den Klauseln

$$\{\neg C, B, \neg A\}, \{C, B, \neg A\}, \{\neg B, \neg A\}, \{\neg B, A\}, \{D, B.\neg C\}, \{D, C\}, \{\neg D, B\}$$

bestehende CNF unerfüllbar ist.

Wir zeigen kurz die Korrektheit der Resolutionsregel:

Lemma 4. *Die Klausel $C \cup D$ ist eine logische Folgerung aus $C \cup \{A\}$ und $D \cup \{\neg A\}$ (d.h. $C \cup \{A\}, D \cup \{\neg A\} \models C \cup D$).*

Beweis. Sei $\kappa \models C \cup \{A\}$ und $\kappa \models D \cup \{\neg A\}$; zu zeigen ist dann $\kappa \models C \cup D$. Wir unterscheiden zwei Fälle:

1. $\kappa(A) = \top \Rightarrow \kappa \models D \Rightarrow \kappa \models C \cup D$

2. $\kappa(A) = \bot \Rightarrow \kappa \models C \Rightarrow \kappa \models C \cup D$ □

Satz 5 (Korrektheit des Resolutionsverfahrens)**.** *Das Resolutionsverfahren terminiert. Wenn es die Antwort „Ja“ liefert, dann ist die Eingabeformel erfüllbar. Wenn es die Antwort „Nein“ liefert, dann ist die Eingabeformel nicht erfüllbar.*

Beweis. Terminierung ist klar (mit welcher Zeitschranke?), ebenso mit Lemma 4 die Korrektheit der Antwort „Nein“. Wir zeigen, dass die Antwort „Ja“ korrekt ist.

Eine CNF ϕ (in Mengendarstellung) heißt *resolutionsabgeschlossen* (ra), wenn ϕ mit Klauseln der Form $C \cup \{A\}, D \cup \{\neg A\}$ stets auch $C \cup D$ enthält, wenn sich also mittels der Resolutionsregel keine Klauseln mehr zu ϕ hinzufügen lassen. Zu zeigen ist also: Wenn ϕ ra ist und $\square \notin \phi$, dann ist ϕ erfüllbar. Wir verwenden Induktion über $n = |\mathsf{At}(\phi)|$:

Induktionsanfang ($n = 0$): $\mathsf{At}(\phi) = \emptyset, \square \notin \phi \Rightarrow \phi = \emptyset = \top$, also ϕ erfüllbar.

Induktionsschritt ($< n \to n$): Sei $A \in \mathsf{At}(\phi)$. Setze

$$\phi/A = \{C \setminus \{\neg A\} \mid C \in \phi, A \notin C\} \qquad (\text{„}\phi \text{ angenommen } A\text{“})$$
$$\phi/\neg A = \{C \setminus \{A\} \mid C \in \phi, \neg A \notin C\} \quad (\text{„}\phi \text{ angenommen } \neg A\text{“})$$

Dann ist $\square \notin \phi/A$ oder $\square \notin \phi/\neg A$, denn sonst: $\{\neg A\} \in \phi \ni \{A\} \overset{\phi \text{ ra}}{\Rightarrow} \square \in \phi$, Widerspruch.

Ohne Einschränkung sei $\square \notin \phi/A$. Noch zu zeigen ist: ϕ/A ist ra.

Sei nun $C \cup \{B\}, D \cup \{\neg B\} \in \phi/A$. Dann existieren $\tilde{C}, \tilde{D}$ mit $\tilde{C} \setminus \{\neg A\} = C$, $\tilde{D} \setminus \{\neg A\} = D$, $\tilde{C} \cup \{B\}, \tilde{D} \cup \{\neg B\} \in \phi$, und $A \notin \tilde{C} \cup \tilde{D}$. Da ϕ ra ist, folgt $\tilde{C} \cup \tilde{D} \in \phi$ und damit $C \cup D = (\tilde{C} \cup \tilde{D}) \setminus \{\neg A\} \in \phi/A$.

Damit können wir die Induktionsvoraussetzung anwenden, haben also κ mit $\kappa \models \phi/A$. Dann gilt $\kappa[A \mapsto \top] \models \phi$ (insbesondere ist ϕ wie verlangt erfüllbar):

Sei $D \in \phi$; zu zeigen ist $\kappa[A \mapsto \top] \models D$.

Fall 1: $A \in D \Rightarrow \kappa[A \mapsto \top] \models D$.

Fall 2: $A \notin D \Rightarrow D \setminus \{\neg A\} \in \phi/A \Rightarrow \kappa \models D \setminus \{\neg A\} \Rightarrow \kappa[A \mapsto \top] \models D \setminus \{\neg A\} \Rightarrow \kappa[A \mapsto \top] \models D$. □

Aus dem Beweis ziehen wir folgendes Lemma:

Lemma 6. *Eine CNF ϕ ist genau dann erfüllbar, wenn mindestens eine der CNFs ϕ/A oder $\phi/\neg A$ erfüllbar ist.*

Dieses Lemma ist bereits ein rekursiver Algorithmus, der *Backtracking-Algorithmus*.

1.3.1 Optimierungen des Backtracking-Algorithmus (DPLL)

Der bekannte *DPLL-Algorithmus* (benannt nach Davis, Putnam, Loge und Loveland) besteht einfach im Backtracking-Algorithmus mit folgenden Optimierungen:

Unit Propagation Wenn $\{L\} \in \phi$ für ein Literal L, dann ersetze ϕ durch ϕ/L (korrekt per Lemma 6, da $\square \in \phi/\neg L$).

Pure literal elimination Wenn L ein Literal ist, so dass $L \notin C$ für alle $C \in \phi$, dann ersetze ϕ durch $\phi/\neg L$. Für die Korrektheit dieser Umformung ist zu zeigen, dass ϕ genau dann erfüllbar ist, wenn $\phi/\neg L$ erfüllbar ist:

„$\Leftarrow$“: per Lemma 6.

„$\Rightarrow$“: Sei $\kappa \models \phi$. Dann gilt $\kappa \models \phi/\neg L$: Sei $C \in \phi/\neg L$, also $C = \tilde{C} \setminus \{L\}$ für ein $\tilde{C} \in \phi$ mit $\neg L \notin \tilde{C}$. Nach Voraussetzung gilt $\tilde{C} \setminus \{L\} = \tilde{C}$, also $C = \tilde{C} \in \phi$ und damit $\kappa \models C$.

(Moderne SAT-Solving-Algorithmen bestehen in DPLL mit weiteren Optimierungen, z.B. *Backjumping* und *Clause Recording*.)

Beispiel

$\phi : \{A, B, \neg C\}, \{A, C, B\}, \{B, D\}, \{\neg D, \neg A\}, \{\neg A, B, C\}, \{\neg D, A\}, \{\neg A, \neg B\}$
$\phi/A : \{B, D\}, \{\neg D\}, \{B, C\}, \{\neg B\}$
Unit Prop. mit $\neg B$: $\{D\}, \{\neg D\}, \{C\}$
Unit Prop. mit D: $\square, \{C\}$ $\lightning$
$\phi/\neg A = \{B, \neg C\}, \{C, B\}, \{B, D\}, \{\neg D\}$
Unit Prop. mit $\neg D$: $\{B, \neg C\}, \{C, B\}, \{B\}$
Unit Prop. mit B: leere CNF, also erfüllbar

1.4 Tableaux

Tableaumethode: Modellbau längs der Formelstruktur.

1.4.1 Flache Tableaux

Definition Ein *flaches Tableau* (für ϕ) ist eine endliche Menge $\mathcal{T}$ von Formeln (mit $\phi \in \mathcal{T}$), so dass

$$\begin{aligned} \bot &\notin \mathcal{T} \\ \phi_1 \wedge \phi_2 \in \mathcal{T} &\Rightarrow \phi_1, \phi_2 \in \mathcal{T} \\ \neg(\phi_1 \wedge \phi_2) \in \mathcal{T} &\Rightarrow \neg\phi_1 \in \mathcal{T} \text{ oder } \neg\phi_2 \in \mathcal{T} \\ \neg\neg\phi \in \mathcal{T} &\Rightarrow \phi \in \mathcal{T} \\ \neg A \in \mathcal{T} &\Rightarrow A \notin \mathcal{T} \end{aligned}$$

Satz 7. *Eine Formel ϕ ist genau dann erfüllbar, wenn ein flaches Tableau für ϕ existiert.*

Beweis. „⇒“ Sei $\kappa \models \phi$. Sei S die Menge der Unterformeln von ϕ; wir setzen dann

$$\begin{aligned} \Sigma &= S \cup \{\neg\psi \mid \psi \in S\} \\ \mathcal{T} &= \{\psi \in \Sigma \mid \kappa \models \psi\} \end{aligned}$$

Dann ist $\mathcal{T}$ ein flaches Tableau für ϕ:

$$\begin{aligned} \kappa \not\models \bot &\Rightarrow \bot \notin \mathcal{T} \\ \psi_1 \wedge \psi_2 \in \mathcal{T} &\Rightarrow \kappa \models \psi_1 \wedge \psi_2 \Rightarrow \kappa \models \psi_1 \text{ und } \kappa \models \psi_2 \Rightarrow \psi_1, \psi_2 \in \mathcal{T} \\ \neg(\psi_1 \wedge \psi_2) \in \mathcal{T} &\Rightarrow \kappa \models \neg\psi_1 \text{ oder } \kappa \models \neg\psi_2; \text{ ferner } \neg\psi_1, \neg\psi_2 \in \Sigma, \text{ also } \neg\psi_1 \in \mathcal{T} \text{ oder } \neg\psi_2 \in \mathcal{T} \\ \neg\neg\phi \in \mathcal{T} &\Rightarrow \kappa \models \neg\neg\phi \Rightarrow \kappa \models \phi; \text{ ferner } \phi \in \Sigma, \text{ also } \phi \in \mathcal{T} \\ \neg A \in \mathcal{T} &\Rightarrow \kappa \models \neg A \Rightarrow \kappa \not\models A \Rightarrow A \notin \mathcal{T} \end{aligned}$$

„⇐“ (*Modellexistenzlemma*) Sei $\mathcal{T}$ flaches Tableau für ϕ. Setze

$$\kappa(A) = \top \iff A \in \mathcal{T}.$$

Dann gilt:

Lemma 8 (Wahrheitslemma)**.** *Für alle $\psi \in \mathcal{T}$ gilt $\kappa \models \psi$.*

Beweis. Induktion über die Größe (!) von ψ. Die Struktur der Induktion folgt der Struktur der Definition von Tableaux. Wir unterscheiden also folgende Fälle:

- $\psi = \bot$: dann $\psi \notin \mathcal{T}$.
- $\psi = \psi_1 \wedge \psi_2$: $\psi_1 \wedge \psi_2 \in \mathcal{T} \Rightarrow \psi_1, \psi_2 \in \mathcal{T} \overset{\text{IV}}{\Rightarrow} \kappa \models \psi_1, \kappa \models \psi_2 \Rightarrow \kappa \models \psi_1 \wedge \psi_2$.
- $\psi = \neg(\psi_1 \wedge \psi_2)$: Sei $\neg(\psi_1 \wedge \psi_2) \in \mathcal{T}$, dann ohne Einschränkung $\neg\psi_1 \in \mathcal{T} \overset{\text{IV}}{\Rightarrow} \kappa \models \neg\psi_1 \Rightarrow \kappa \models \neg(\psi_1 \wedge \psi_2)$.
- $\psi = \neg\neg\psi_0$: $\neg\neg\psi_0 \in \mathcal{T} \Rightarrow \psi_0 \in \mathcal{T} \overset{\text{IV}}{\Rightarrow} \kappa \models \psi_0 \Rightarrow \kappa \models \neg\neg\psi_0$.
- $\psi = \neg\bot$: $\kappa \models \psi$ ✓
- $\psi = \neg A$: $\neg A \in \mathcal{T} \Rightarrow A \notin \mathcal{T} \Rightarrow \kappa \not\models A \Rightarrow \kappa \models \neg A$
- $\psi = A$: OK nach Konstruktion von κ.

□

Aus dem Wahrheitslemma folgt dann $\kappa \models \mathcal{T}$, insbesondere $\kappa \models \phi$. □

Beispiel 9. Für die Formel

$$\phi(\neg(\neg(A \wedge \neg\neg(\neg A \wedge \neg B)) \wedge C)) \wedge \neg\neg(\neg C \wedge \neg\neg(\neg A \wedge \neg B))$$

haben wir das flache Tableau

$$\begin{aligned}\mathcal{T} = \{&\phi, \neg(\neg(A \wedge \neg\neg(\neg A \wedge \neg B)) \wedge C), \neg\neg(\neg C \wedge \neg\neg(\neg A \wedge \neg B)),\\ &\neg C \wedge \neg\neg(\neg A \wedge \neg B), \neg C, \neg\neg(\neg A \wedge \neg B), \neg A \wedge \neg B, \neg A, \neg B\}\end{aligned}$$

1.4.2 (Echte) Tableaux

Mit Blick auf die Erweiterung des Formalismus auf Beschreibungslogiken führen wir nun strukturiertere *baumförmige* Tableaux mit gelabelten Knoten ein; die *Label* sind dabei endliche Mengen $\Gamma, \Delta, \ldots$ von Formeln, konjunktiv gelesen. Der Einfachheit halber schließen wir ab jetzt $\bot$ als Formel aus (ggf. muss es dann als $A \wedge \neg A$ codiert werden); stattdessen lassen wir $\bot$ als Label zu, zur Markierung sogenannter *Clashs*. Solche Bäume werden, ausgehend von einem typischerweise mit der Zielformel gelabelten Wurzelknoten, durch erschöpfende Anwendung der folgenden Regeln erzeugt:

$$(\wedge)\ \frac{\Gamma, \phi \wedge \psi}{\Gamma, \phi, \psi} \qquad (\neg\neg)\ \frac{\Gamma, \neg\neg\phi}{\Gamma, \phi}$$

$$(\neg\wedge)\ \frac{\Gamma, \neg(\phi \wedge \psi)}{\Gamma, \neg\phi \mid \Gamma, \neg\psi} \qquad (\text{Ax})\ \frac{\Gamma, A, \neg A}{\bot}$$

Schreibweise:

„, “ $\widehat{=}$ Vereinigung
$\psi \widehat{=} \{\psi\}$

Dabei deutet der senkrechte Strich | in Regel $(\neg\wedge)$ an, dass der Baum dort verzweigt. Wir bezeichnen den Label eines Knotens n im so entstehenden Baum mit $l(n)$. Die Blätter n des vollständig aufgebauten Baums zerfallen in zwei Typen:

- *Clash:* $l(n) = \bot$
- *Saturierter Knoten:* auf $l(n) \neq \bot$ ist keine Regel anwendbar.

Definition 10. Wir definieren *erfolgreiche* Knoten rekursiv per

$$n \text{ erfolgreich} \iff \begin{cases} n \text{ saturiert} & n \text{ Blatt} \\ n \text{ hat erfolgreichen Nachfolger} & n \text{ innerer Knoten} \end{cases}$$

Diese Definition erfasst die grundsätzliche Funktionsweise des Tableau-Algorithmus: Der Algorithmus antwortet auf Eingabe ϕ 'erfüllbar', wenn ein mit ϕ gelabelter Wurzelknoten eines Tableaus erfolgreich ist. Der Algorithmus bietet beim Aufbau des Tableus Wahlfreiheit hinsichtlich der Reihenfolge der Regelanwendungen; ferner muss man nicht alle Zweige des Tableaus verfolgen, wenn andere Zweige bereits zum Erfolg führen. Dies bedeutet unter anderem, dass man einen Clash (A vs. $\neg A$) zunächst ignorieren kann; das folgende Lemma hält die (relativ offensichtliche) Tatsache fest, dass das auf Dauer nichts bringt:

Lemma 11 (Clash-Lemma)**.** *Sei n ein Knoten und A ein Atom, so dass $A, \neg A \in l(n)$. Dann ist n nicht erfolgreich.*

Beweis. Induktion über n. Der Induktionsanfang (n Blatt) kommt nicht vor, weil auf n die Axiomenregel anwendbar ist. Sei also n ein innerer Knoten und m ein Kind von n; wir müssen zeigen, dass m nicht erfolgreich ist. Wenn $l(m) = \bot$, sind wir fertig. Andernfalls entsteht m durch eine der Regeln ($\wedge$), ($\neg\neg$) und ($\neg\wedge$); in all diesen Regeln werden A und $\neg A$ von n an m vererbt, so dass m per Induktionsvoraussetzung nicht erfolgreich ist. □

Wir dehnen den Begriff 'erfolgreich' auf Label aus:

Definition 12. Ein Label $\Gamma \neq \bot$ ist *erfolgreich*, wenn jede auf Γ anwendbare Regelinstanz eine erfolgreiche Konklusion hat.

(Warum ist das tatsächlich eine Definition?)

Definition 13. Ein Label Γ *hat ein erfolgreiches Tableau*, wenn ein erfolgreicher Knoten n in einem Tableau existiert mit $\Gamma \subseteq l(n)$. Ferner *hat* Γ *ein flaches Tableau*, wenn es ein flaches Tableau $\mathcal{T}$ mit $\Gamma \subseteq \mathcal{T}$ gibt.

Satz 14. *Sei Γ ein Label. Äquivalent sind*

(i) Γ *ist erfolgreich;*

(ii) Γ *hat ein erfolgreiches Tableau;*

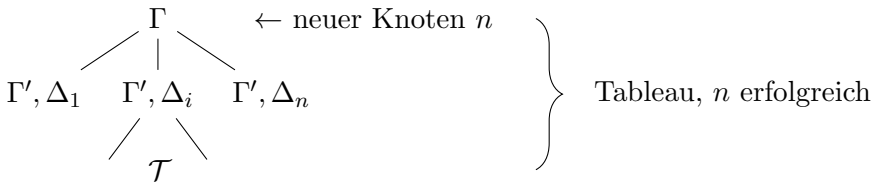
(iii) Γ *hat ein flaches Tableau;*

(iv) Γ *ist erfüllbar*

Beweis (iii) $\Leftrightarrow$ (iv): Per Satz 7.
(i) $\Rightarrow$ (ii):

Fall 1: Auf Γ ist keine Regel anwendbar. Dann ist das Tableau aus nur einem Knoten n mit $l(n) = \Gamma$ erfolgreich.

Fall 2: Wir haben $\Gamma = \Gamma', \psi$ und eine Regelinstanz $\dfrac{\Gamma', \psi}{\Gamma', \Delta_1 \mid \ldots \mid \Gamma', \Delta_n}$ (wobei wir den Fall $n = 0$ als Clash $\bot$ verstehen). Nach Voraussetzung existiert ein i, so dass Γ', Δ_i erfolgreich ist. Da Δ_i kleiner ist (genauer gesagt weniger logische Konnektive enthält) als ψ, hat Γ', Δ_i nach Induktionsvoraussetzung (in einer hier spontan losgetretenen Induktion) ein erfolgreiches Tableau $\mathcal{T}$. Wir bauen also ein Tableau der folgenden Form:



In diesem Tableau ist Γ ein erfolgreicher Knoten.

(ii) $\Rightarrow$ (iii): Es reicht der Fall $\Gamma = l(n)$. Sei also n erfolgreich; wir zeigen per Induktion über n, dass $l(n)$ ein flaches Tableau hat:

Induktionsanfang (n Blatt): Dann ist keine Regel auf $l(n)$ anwendbar und $l(n) \neq \bot$, also ist $l(n)$ ein flaches Tableau.

Induktionsschritt (von Kindern auf Eltern): n hat ein erfolgreiches Kind m. Nach Induktionsvoraussetzung hat $l(m)$ ein flaches Tableau $\mathcal{T}$; man zeigt durch Fallunterscheidung über die bei n angewendete Regel, dass dann $l(n) \cup \mathcal{T}$ ein flaches Tableau für Γ ist. Da n ein Kind hat, wird bei n eine der Regeln $(\wedge)$, $(\neg\neg)$ oder $(\wedge\neg)$ angewendet. Wir führen hier nur den Fall für $(\wedge\neg)$ durch. Dann hat $l(n)$ die Form $l(n) = \Gamma', \neg(\phi \wedge \psi)$, und o.E. $l(m) = \Gamma', \neg\phi$. Wir prüfen die Bedingungen für flache Tableaux:

- Sei $\neg(\chi \wedge \rho) \in l(n) \cup \mathcal{T}$. Es entstehen wiederum zwei Unterfälle:
  - $\neg(\chi \wedge \rho) \in \mathcal{T}$. Dann o.E. $\neg\chi \in \mathcal{T} \subseteq l(n) \cup \mathcal{T}$.
  - $\neg(\chi \wedge \rho) \in l(n) \setminus \mathcal{T}$. Da beim Übergang von $l(n)$ nach $l(m)$ nur eine Formel entfernt wird, gilt dann notwendigerweise $\neg(\chi \wedge \rho) = \neg(\phi \wedge \psi)$, und damit $\neg\chi = \neg\phi \in l(m) \subseteq \mathcal{T} \subseteq l(n) \cup \mathcal{T}$.
- Die Fälle für $\neg\neg\chi$ und $\chi \wedge \rho$ sind ähnlich, aber einfacher, da der zweite Unterfall dann nicht vorkommt. Der Fall für $\bot$ entfällt, da $\bot$ im Moment keine Formel mehr ist.
- Set $\neg A \in l(n) \cup \mathcal{T}$. Wir müssen zeigen, dass $A \notin l(n) \cup \mathcal{T}$. Wir unterscheiden wiederum zwei Unterfälle:
  - $\neg A \in l(n)$. Dann gilt auch $\neg A \in l(m)$ (da $\neg A$ aufgrund seines syntaktischen Formats verschieden von der einzigen Formel ist, die beim Übergang von $l(n)$ nach $l(m)$ entfernt wird), also $\neg A \in \mathcal{T}$. Nach dem Clash-Lemma (Lemma 11) gilt $A \notin l(n)$, da n erfolgreich ist, und weil $\mathcal{T}$ ein flaches Tableau ist, gilt $A \notin \mathcal{T}$, insgesamt also $A \notin l(n) \cup \mathcal{T}$.

– $\neg A \in \mathcal{T} \setminus l(n)$. Da $\mathcal{T}$ ein flaches Tableau ist, gilt $A \notin \mathcal{T}$; zu zeigen bleibt $A \notin l(n)$. Beweis per Widerspruch: Wenn $A \in l(n)$, dann auch $A \in l(m)$ (keine der Regeln entfernt Atome), also $A \in \mathcal{T}$, Widerspruch.

(iii) $\Rightarrow$ (i) Sei $\Gamma \subseteq \mathcal{T}$, $\mathcal{T}$ flaches Tableau. Sei $\Gamma = \Gamma', \psi$ und $\dfrac{\Gamma', \psi}{\Gamma', \Delta_1 \mid \ldots \mid \Gamma', \Delta_k}$ Regelinstanz. Da $\mathcal{T}$ ein flaches Tableau ist, gibt es ein i mit $\Gamma', \Delta_i \subseteq \mathcal{T}$. Nach (wiederum spontaner) Induktionsvoraussetzung ist Γ', Δ_i erfolgreich, wie verlangt.

Beispiel 15.

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg(A \wedge \neg B) \wedge \neg(B \wedge \neg C) \wedge A}{\neg(A \wedge \neg B), \neg(B \wedge \neg C), A}\,(\wedge)^*}{\neg A, \neg(B \wedge \neg C), A}\,(\neg\wedge)}{\bot} \qquad \dfrac{\dfrac{\dfrac{\neg\neg B, \neg(B \wedge \neg C), A}{B, \neg(B \wedge \neg C), A}\,(\neg\neg)}{\dfrac{B, \neg\neg C, A}{B, C, A}\,(\neg\neg)} \qquad \dfrac{B, \neg B, \ldots}{\bot}}{}\,(\neg\wedge)$$

Remark 16. Man entnimmt obigem Satz insbesondere die Aussage, dass die Reihenfolge, in der ich die Regeln anwende, um ein (baumförmiges) Tableau zu bauen, egal ist: Der baumförmige Tableaubegriff, der eine Reihenfolge der Regelanwendungen beinhaltet, erweist sich als äquivalent zur rekursiven Definition von erfolgreichen Labeln sowie zum flachen Tableaubegriff, die beide effektiv alle Regeln simultan abhandeln.

Etwas anders ausgedrückt liegt dies daran, dass die propositionalen Tableauregeln miteinander *kommutieren*, d.h. zwei verschiedene Regelanwendungen lassen sich stets wieder zusammenführen: Wenn sowohl Γ' als auch Γ'' Konklusionen von Regelanwendungen auf Γ sind, dann ist jede Konklusion Γ''' einer Regelanwendung auf Γ' auch Konklusion nach eventuell wiederholter Regelanwendung auf Γ'' (in welchem Fall braucht man mehrere Regeln?) – das beruht darauf, dass jede Regelanwendung immer nur eine einzelne Formel zerlegt. Diese Eigenschaft setzt sich induktiv auf Ketten von Regelanwendungen fort. Wir können also die Reihenfolge der Anwendung der propositionalen Tableauregeln willkürlich festlegen.

Kapitel 2

Logik erster Stufe

2.1 Erinnerung: Syntax, Semantik und Deduktion

(Siehe hierzu auch Skript „Grundlagen der Logik in der Informatik“.)

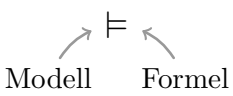
Die Syntax der *Prädikatenlogik erster Stufe* ist parametrisiert über eine *Signatur* $\Sigma = (FS, PS)$, bestehend aus Mengen

$$FS \text{ von Funktionssymbolen}$$
$$PS \text{ von Prädikatssymbolen.}$$

Jedes Symbol hat eine (nichtnegativ ganzzahlige) *Stelligkeit*. Wir verwenden allgemein Großbuchstaben für Prädikatensymbole und Kleinbuchstaben für Funktionssymbole, und schreiben kurz $P/n \in \Sigma$ für '$P \in PS$ hat Stelligkeit n', entsprechend für Funktionssymbole. 0-stellige Funktionssymbole bezeichnen wir auch als *Konstanten*. Wir unterstellen ferner einen abzählber unendlichen Vorrat Var von *Variablen*. Die Syntax von *Formeln* $\phi, \psi, \ldots$ und *Termen* $E, E_1, \ldots$ ist dann definiert durch die Grammatik

$$\begin{aligned} \phi, \psi &::= E = D \mid p(E_1, \ldots, E_n) \mid \neg\phi \mid \phi \wedge \psi \mid \forall x(\phi) && (x \in \mathsf{Var}, P/n \in \Sigma) \\ E &::= x \mid f(E_1, \ldots, E_n) && (f/n \in \Sigma) \end{aligned}$$

Semantik = Modellbegriff + Erfüllbarkeitsrelation



Ein Σ-*Modell* $\mathfrak{M}$ besteht aus:

- einem *Grundbereich* M
- für $f/n \in \Sigma$: $\mathfrak{M}[\![f]\!] : M^n \to M$
- für $P/n \in \Sigma$: $\mathfrak{M}[\![P]\!] \subseteq M^n$

Eine *Umgebung* ist eine Abbildung $\eta : \mathsf{Var} \to M$. Die Erfülltheitsrelation $\vDash$ hängt (natürlich) auch von der Wahl einer Umgebung ab, und wird wie folgt rekursiv definiert:

$$\begin{aligned}
&\mathfrak{M}, \eta \vDash \neg\phi \Leftrightarrow \mathfrak{M}, \eta \nvDash \phi \\
&\mathfrak{M}, \eta \vDash \phi_1 \wedge \phi_2 \Leftrightarrow \mathfrak{M}, \eta \vDash \phi_1 \text{ und } \mathfrak{M}, \eta \vDash \phi_2 \\
&\mathfrak{M}, \eta \vDash \forall x(\phi) \Leftrightarrow \text{für alle } x \in M \text{ gilt } \mathfrak{M}, \eta[x \to x] \vDash \phi \\
&\mathfrak{M}, \eta \vDash E = D \Leftrightarrow \mathfrak{M}[\![E]\!]\eta = \mathfrak{M}[\![D]\!]\eta \\
&\mathfrak{M}, \eta \vDash P(E_1, \ldots, E_n) \Leftrightarrow (\mathfrak{M}[\![E_1]\!]\eta, \ldots, \mathfrak{M}[\![E_n]\!]\eta) \in \mathfrak{M}[\![P]\!].
\end{aligned}$$

Dies verwendet bereits die Auswertung $\mathfrak{M}[\![E]\!]\eta$ eines Terms E, die in der offensichtlichen Weise rekursiv definiert ist:

$$\begin{aligned}
&\mathfrak{M}[\![x]\!]\eta = \eta(x) \\
&\mathfrak{M}[\![f(E_1, \ldots, E_n)]\!]\eta = \mathfrak{M}[\![f]\!](\mathfrak{M}[\![E_1]\!]\eta, \ldots, \mathfrak{M}[\![E_n]\!]\eta).
\end{aligned}$$

Fast alle sinnvollen Verwendungen der beiden Quantoren $\forall$ und $\exists$ halten die klassischen *Aristotelischen Formen* ein:

$$\begin{aligned}
\text{Alle } Q \text{ sind } P \qquad & \forall x.(Q(x) \to P(x)) \\
\text{Einige } Q \text{ sind } P \qquad & \exists x.(Q(x) \wedge P(x))
\end{aligned}$$

2.2 Unentscheidbarkeit

Wir erinnern an einige Fakten und Begriffe aus der Berechenbarkeitstheorie:

Das PCP (Post Correspondence Problem)

Sei $P \subseteq (\Sigma^*)^2$ endlich. P heißt *lösbar*, wenn es $(u_1, v_1), \ldots, (u_n, v_n) \in P$ mit $u_1 \ldots u_n = v_1 \ldots v_n$ gibt. (Achtung: Die (u_i, v_i) müssen nicht paarweise verschieden sein.) P nennt man dann *PCP-Instanz*. Zum Beispiel:

$$P = \left\{ \boxed{\begin{matrix} 1 \\ 10 \end{matrix}}, \boxed{\begin{matrix} 0 \\ 10 \end{matrix}}, \boxed{\begin{matrix} 010 \\ 01 \end{matrix}} \right\}.$$

Das Problem

$$PCP = \{P \subseteq_{\text{fin}} (\Sigma^*)^2 \mid P \text{ lösbar}\}$$

ist unentscheidbar (für $|\Sigma| > 1$), genauer nicht co-r.e., wie man durch Reduktion des Halteproblems zeigt.

Hierzu Erinnerungen (an BFS):

- Rekursive Aufzählbarkeit: $A \subseteq \Sigma^*$ ist per Definition r.e. (recursively enumerable), wenn ein Algorithmus existiert, der im Lauf der Zeit alle Elemente von A auflistet, was wiederum äquivalent dazu ist (warum?), dass ein *Halbentscheidungsverfahren* für A existiert, also ein Algorithmus, der auf Eingabe x genau dann mit Antwort „ja“ terminiert, wenn $x \in A$ (wenn $x \notin A$, kann der der Algorithmus also entweder mit einer anderen Antwort oder gar nicht terminieren).
- A ist *co-r.e.* wenn $\Sigma^* \setminus A$ r.e. ist.
- A ist genau dann entscheidbar, wenn A sowohl r.e. als auch co-r.e. ist.

Reduktionen

Seien $A, B \subseteq \Sigma^*$ (d.h. A, B sind *Probleme*) und $f : \Sigma^* \to \Sigma^*$ berechenbar. Die Funktion f *reduziert* A auf B, wenn $\forall u \in \Sigma^*. f(u) \in B \Leftrightarrow u \in A$, d.h. wenn $A = f^{-1}[B]$.

Dann gilt offenbar:

- B entscheidbar $\Rightarrow$ A entscheidbar (Reduction to)
- A unentscheidbar $\Rightarrow$ B unentscheidbar (Reduction from).

Satz 17. *Gültigkeit in FOL ist unentscheidbar, genauer nicht co-r.e.*

Wir erinnern zum Verständnis der Formulierung des Satzes daran, dass eine Formel ϕ gültig ist (Notation: $\models \phi$), wenn $\mathfrak{M}, \eta \models \phi$ für alle Modelle $\mathfrak{M}$ und alle Valuationen η (z.B. $\models x = x$).

Wir erinnern ferner an die Vollständigkeit von FOL, derzufolge insbesondere ϕ genau dann gültig ist ($\models \psi$), wenn ϕ herleitbar ist ($\vdash \psi$; Erinnerung an das Deduktionssystem siehe nachfolgende Box). Damit ist Gültigkeit in FOL r.e.: Erzeuge alle Beweise und gib jeweils die letzte Formel aus (British Museum Algorithm).

Eine per Dualisierung äquivalente Formulierung des Satzes lautet: Erfüllbarkeit in FOL ist unentscheidbar, genauer nicht r.e.

Erinnerung: Deduktion in Prädikatenlogik erster Stufe

$\Phi \models \psi \Leftrightarrow \forall \mathfrak{M}, \eta. (\mathfrak{M}, \eta \models \Phi \Rightarrow \mathfrak{M}, \eta \models \psi)$

$\Phi \vdash \psi \Leftrightarrow$ es existiert ein Beweis von ψ mit Annahmen aus Φ.

Beispiel:

$$\exists y. \forall x. P(x, y) \models \forall x. \exists y. P(x, y).$$

Natürlichsprachlicher Beweis der logischen Folgerung: Sei $\mathfrak{M} \models \exists y. \forall x. P(x, y)$. Zu zeigen ist: $\mathfrak{M}, \eta \models \forall x. \exists y. P(x, y)$. Sei also $x \in M$. Nach Voraussetzung existiert y mit $\mathfrak{M}, \eta[y \mapsto y] \models \forall x. P(x, y)$, also $\mathfrak{M}, \eta[x \mapsto x, y \mapsto y] \models P(x, y)$, also $\mathfrak{M}, \eta[x \mapsto x] \models \exists y. P(x, y)$. Es folgt $\mathfrak{M}, \eta \models \forall x. \exists y. P(x, y)$. □

1	$\exists y. \forall x. P(x, y)$			
2	c	d	$\forall x. P(x, d)$	
3			$P(c, d)$	$\forall$E, 2
4			$\exists y. P(c, y)$	$\exists$I, 3
5		$\exists y. P(c, y)$		
6	$\forall x. \exists y. P(x, y)$			

Beweis (Unentscheidbarkeitssatz). Reduction from PCP; d.h zu $P \subseteq (\{0, 1\}*)^2$ konstruieren wir (berechenbar!) eine Formel ϕ_P, so dass

$$P \text{ lösbar} \Leftrightarrow \phi_P \text{ gültig.}$$

Zum Verständnis der Konstruktion stelle man sich eine Definition des Datentyps von Bitstrings in Haskell vor, wie etwa

```
data Bitstring = Epsilon | Null Bitstring | One Bitstring
```

Dieser Datentyp ist in FOL nicht eindeutig (*monomorph*) definierbar, da man durch FO-Axiome nicht sicherstellen kann, dass tatsächlich alle Bitstrings durch `Null` und `One` aus `Epsilon` erzeugt werden (z.B. würde man im einfacher gelagerten Fall der natürlichen Zahlen, als des Datentyps

```
data Nat = Zero | Suc Nat
```

für eine eindeutige Festlegung z.B. das Induktionsaxiom $\forall P.((\forall n.(P(n) \rightarrow P(\mathsf{Suc}\ n))) \wedge P(\mathsf{Zero}) \rightarrow \forall n.P(n))$ benötigen, das wegen des Quantors $\forall P$, der über Teilmengen statt über Individuen quantifiziert, gerade nicht in FOL ausdrückbar ist). Wir werden sehen, dass die Reduktion trotzdem gelingt. Wir verwenden folgende Signatur:

ϵ Konstante	„leerer Bitstring“
f_0, f_1 unär	„$f_0(x) = 0x, f_1(x) = 1x$“
C binäres Prädikat	„konstruierbar gemäß PCP-Instanz“,

wobei wir ein Paar von Bitstrings *konstruierbar* nennen, wenn es durch Aneinanderlegen von Bausteinen (mindestens einem) aus der PCP-Instanz hergestellt werden kann. Notation: $f_{b_1 \ldots b_n}(x) = f_{b_1} \cdots f_{b_n}(x)$ für $b_1, \ldots, b_n \in \{0,1\}^*$.

Setze

$$\psi_P = \bigwedge_{(u,v)\in P} (\underbrace{C(f_u(\epsilon), f_v(\epsilon))}_{(i)} \wedge \underbrace{\forall x, y.\, (C(x,y) \rightarrow C(f_u(x), f_v(y)))}_{(ii)})$$

$$\phi_P = \psi_P \rightarrow \exists x.C(x,x)$$

Wir zeigen: P lösbar $\iff$ ϕ_P gültig.

„$\Rightarrow$“: Seien $(u_1, v_1), \ldots, (u_n, v_n) \in P$. Wir werden zeigen, dass

$$\models \psi_P \rightarrow C(f_{u_1 \ldots u_n}(\epsilon), f_{v_1 \ldots v_n}(\epsilon)). \tag{2.1}$$

Damit folgt sofort die Behauptung; wenn nämlich $u_1 \ldots u_n = v_1 \ldots v_n =: w$, dann liefert der Term $f_w(\epsilon)$ per (2.1) einen Zeugen für $\exists x.\, C(x,x)$.

Beweis von (2.1)*:* Induktion über n.

Indutionsanfang ($n = 1$) : per Teil (i) von ψ_P.

Induktionsschritt ($n - 1 \rightarrow n$): Nach IV gilt $\models \psi_P \rightarrow C(f_{u_2 \ldots u_n}(\epsilon), f_{v_2 \ldots v_n}(\epsilon))$. Ferner per ($ii$):

$$\models \psi_P \rightarrow (C(f_{u_2 \ldots u_n}(\epsilon), f_{v_2 \ldots v_n}(\epsilon)) \rightarrow C(f_{u_1}(f_{u_2 \ldots u_n}(\epsilon)), f_{v_1}(f_{v_2 \ldots v_n}(\epsilon)))),$$

$$\text{also } \models \psi_P \rightarrow C(f_{u_1 \ldots u_n}(\epsilon), f_{v_1 \ldots v_n}(\epsilon))$$

„$\Leftarrow$“: Sei ϕ_P gültig. Definiere das Modell $\mathfrak{M}$ durch

$$M = \{0,1\}^*$$
$$\mathfrak{M}[\![\epsilon]\!] = \epsilon$$
$$\mathfrak{M}[\![f_0]\!](x) = 0x$$
$$\mathfrak{M}[\![f_1]\!](x) = 1x$$
$$\mathfrak{M}[\![C]\!] = \{(u_1 \cdots u_n, v_1 \cdots v_n) \mid (u_1, v_1), \ldots, (u_n, v_n) \in P, n \geq 1\}$$

Dann haben wir nach Voraussetzung $\mathfrak{M} \vDash \psi_P \to \exists x.C(x,x)$. Ferner gilt $\mathfrak{M} \vDash \psi_P$; man beachte dazu, dass

$$\mathfrak{M}[\![f_u(x)]\!]\eta = u\eta(x)$$
$$\mathfrak{M}[\![f_u(\epsilon)]\!]\eta = u.$$

Es folgt $\mathfrak{M} \vDash \exists x.C(x,x)$. Das heißt, es existiert $(w,w) \in \mathfrak{M}[\![C]\!]$; somit ist P lösbar. □

Remark 18. Der obige Unentscheidbarkeitsbeweis per Reduktion von PCP importiert natürlich die Reduktion des Halteproblems auf PCP. Eine direkte Reduktion des Halteproblems auf das Gültigkeitsproblem ist nicht grundsätzlich schwieriger als die Reduktion von PCP, nur insgesamt komplizierter, da komplexere Objekte zu kodieren sind (eben Turingmaschinen statt PCP-Instanzen).

2.3 Ausdrucksstärke

Nachdem wir im vorhergehenden Abschnitt gesehen haben, dass FOL als Logik 'zu' ausdrucksstark ist (eben so ausdrucksstark, dass sie unentscheidbar ist), machen wir uns nunmehr Gedanken über *obere* Schranken für die Ausdrucksstärke.

Erreichbarkeit

Ein klassisches Beispiel einer in FOL *nicht* ausdrückbaren Eigenschaft ist *Erreichbarkeit* in Graphen. Wir betrachten zunächst die gerichtete Variante:

Signatur: I unär (initiale Knoten), R binär (gerichtete Kanten zwischen den Knoten).

$$\begin{aligned} & x \in M \text{ } \textit{erreichbar} \\ \Leftrightarrow \quad & \text{es existiert } \mathfrak{M}[\![I]\!] \ni x_0 \text{ und } (x_0,x_1),(x_1,x_2),\ldots,(x_{n-1},x_n) \in \mathfrak{M}[\![R]\!], n \geq 0 \end{aligned}$$

$$\mathfrak{M} \textit{ erreichbar} \Leftrightarrow \text{ jedes } x \in M \text{ erreichbar.}$$

Satz 19. *Erreichbarkeit ist in FOL nicht ausdrückbar.*

Zum Beweis benötigen wir eine weitere Erinnerung:

Kompaktheit

Aus der Vollständigkeit von FOL (in der 'starken' Form, also mit unendlichen Mengen von Annahmen) folgt unmittelbar

Satz 20 (Kompaktheit)**.** *Jede endlich erfüllbare Formelmenge ist erfüllbar.*

Dabei heißt eine (ggf. unendliche) Formelmenge *endlich erfüllbar*, wenn jede ihrer endlichen Teilmengen erfüllbar ist. Kompaktheit folgt daraus, dass die entsprechende Eigenschaft für Konsistenz statt Erfüllbarkeit klar ist, und per Korrektheit und Vollständigkeit Konsistenz und Erfüllbarkeit dasselbe sind.

Damit ist der Beweis von Satz 19 jetzt einfach:

Beweis (Satz 19). Wir können offenbar die Eigenschaft 'x ist nicht in i Schritten von I aus erreichbar' durch eine FO-Formel σ_i ausdrücken. Man nehme nun an, die Eigenschaft 'x ist (von I aus) erreichbar' würde durch eine Menge Φ von Formeln ausgedrückt. Dann ist die Menge

$$\Phi \cup \{\sigma_i \mid i \geq 0\}$$

nicht erfüllbar, aber offenbar endlich erfüllbar, im Widerspruch zur Kompaktheit. □

Über endlichen Modellen ist dieses Argument nicht anwendbar, da dort Kompaktheit nicht gilt (siehe Übungen). Wir zeigen aber im folgenden, dass auch über endlichen Modellen Erreichbarkeit nicht FO-ausdrückbar ist.

2.4 Ehrenfeucht-Fraïssé-Spiele

Wir führen nun einen semantischen Äquivalenzbegriff ein, *Ehrenfeucht-Fraïssé-Äquivalenz*, unter dem FOL *invariant* ist in dem Sinne, dass äquivalente Strukturen dieselben Formeln erfüllen. Ehrenfeucht-Fraïssé-Äquivalenz lässt sich sowohl in spieltheoretischen Begriffen als auch in einer etwas statischeren Sicht formulieren; wir führen beide Formulierungen ein und zeigen ihre Äquivalenz. Zentral ist der folgende Begriff von lokaler Ununterscheidbarkeit:

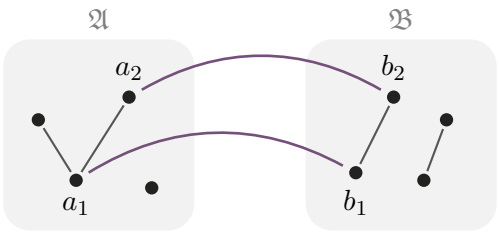
Definition 21 (Partieller Isomorphismus)**.** Ein *partieller Isomorphismus* zwischen Σ-Modellen (ohne Funktionssymbole) $\mathfrak{A}$ und $\mathfrak{B}$ ist ein Paar

$$(\bar{a}, \bar{b}) = ((a_1 \dots a_k), (b_1 \dots b_k)) \in A^k \times B^k,$$

so dass folgendes gilt:

- Für alle i, j gilt $a_i = a_j \iff b_i = b_j$; d.h. die Abbildung $a_i \mapsto b_i$, $i = 1, \dots, k$, ist wohldefiniert und injektiv.
- Für alle $P/k \in \Sigma$ gilt $(a_{i_1}, \dots, a_{i_k}) \in \mathfrak{A}[\![P]\!] \iff (b_{i_1}, \dots, b_{i_k}) \in \mathfrak{B}[\![P]\!]$.

Beispiel 22 (Partieller Isomorphismus)**.** In den Modellen



ist $((a_1, a_2), (b_1, b_2))$ ein partieller Isomorphismus, obwohl die Modelle nicht global isomorph sind.

Notation 23. Für einen Vektor $\bar{x} = (x_1, \dots, x_k)$ von Variablen bezeichnet $(\mathfrak{A}, \bar{a})$ mit Vektor $\bar{a} = (a_1, \dots, a_k) \in A^k$ das Paar aus dem Modell $\mathfrak{A}$ und der Umgebung η mit $\eta(x_i) = a_i \quad (i = 1, \dots, k)$. Wir bezeichnen so ein Paar $(\mathfrak{A}, \bar{a})$ als eine *Struktur*.

Definition 24 (Quantorenrang)**.** Der *Quantorenrang* $\mathsf{QR}(\phi)$ bezeichnet die Schachtelungstiefe von Quantoren in einer Formel ϕ:

$$\begin{aligned}\mathsf{QR}(\phi) &= 0, \text{wenn } \phi \text{ atomar}\\ \mathsf{QR}(\neg\phi) &= \mathsf{QR}(\phi)\\ \mathsf{QR}(\phi \wedge \psi) &= \max(\mathsf{QR}(\phi), \mathsf{QR}(\psi))\\ \mathsf{QR}(\forall x(\phi)) &= 1 + \mathsf{QR}(\phi)\end{aligned}$$

Beispiel 25. Nach obiger Definition haben wir $\mathsf{QR}(\forall x(x = y) \wedge \exists x \forall y(P(x, y))) = 2$ – die Formel enthält zwar drei Quantoren, die aber nicht alle ineinander geschachtelt sind.

Wir stratifizieren logische Ununterscheidbarkeit von Strukturen nach dem Quantorenrang:

Definition 26 (m-Äquivalenz)**.** Für Modelle $\mathfrak{A}$, $\mathfrak{B}$ und Wertevektoren $\bar{a} \in A^k$, $\bar{b} \in B^k$ schreiben wir

$$(\mathfrak{A}, \bar{a}) \cong_m (\mathfrak{B}, \bar{b})$$

(lies: $(\mathfrak{A}, \bar{a})$ und $(\mathfrak{B}, \bar{b})$ sind *m-äquivalent*), wenn für jede Formel ϕ mit Quantorenrang $\mathsf{QR}(\phi) \leq m$ und mit freien Variablen $\mathsf{FV}(\phi) \subseteq \{x_1, \ldots, x_k\}$

$$(\mathfrak{A}, \bar{a}) \vDash \phi \Leftrightarrow (\mathfrak{B}, \bar{b}) \vDash \phi$$

gilt.

Wir werden m-Äquivalenz mittels des folgenden Spiels semantisch charakterisieren:

Definition 27 (Ehrenfeucht-Fraïssé-Spiel)**.** Das *m-Runden-Ehrenfeucht-Fraïssé-Spiel* $G_m((\mathfrak{A}, \bar{a}), (\mathfrak{B}, \bar{b}))$ hat

Konfigurationen: $(\bar{a}\bar{a}', \bar{b}\bar{b}') \in A^{k+i} \times B^{k+i}$ mit $i \geq 0$, $\bar{a}' \in A^i$, $\bar{b}' \in B^i$ („potentielle partielle Isomorphismen“)

Initialkonfiguration: $(\bar{a}, \bar{b})$

Spieler: *Spoiler* (S), *Duplicator* (D)

Züge in der i-ten Runde ($i = 1, \ldots, m$):

- S wählt $a_{k+i} \in A$ (oder $b_{k+i} \in B$)
- D wählt dann $b_{k+i} \in B$ (oder $a_{k+i} \in A$)

Die dann von Konfiguration $(\bar{a}\bar{a}', \bar{b}\bar{b}')$ aus erreichte neue Konfiguration ist $(\bar{a}\bar{a}' a_{k+i}, \bar{b}\bar{b}' b_{k+i})$

Nach m Runden gewinnt Duplicator, wenn die dann erreichte Konfiguration ein partieller Iso ist. (Dann sind automatisch auch alle zwischendurch erreichten Konfigurationen partielle Isomorphismen.)

Satz 28 (Ehrenfeucht-Fraïssé)**.** *Wenn D eine Gewinnstrategie in $G_m((\mathfrak{A}, \bar{a}), (\mathfrak{B}, \bar{b}))$ hat, dann gilt $(\mathfrak{A}, \bar{a}) \cong_m (\mathfrak{B}, \bar{b})$. Wenn Σ endlich ist, dann gilt auch die umgekehrte Implikation.*

Der Satz kann zum Beweis der Nichtausdrückbarkeit gewisser Eigenschaften, wie etwa Erreichbarkeit, verwendet werden:

Man nehme an, ϕ drücke etwa Erreichbarkeit aus. Wenn man dann Modelle $\mathfrak{A}, \mathfrak{B}$ findet, so dass

- $\mathfrak{A} \cong_{\mathsf{QR}(\phi)} \mathfrak{B}$ (per Ehrenfeucht-Fraïssé)
- $\mathfrak{A}$ erreichbar, $\mathfrak{B}$ nicht,

dann folgt sofort ein Widerspruch, da nach Voraussetzung nunmehr $\mathfrak{A} \models \phi$, aber $\mathfrak{B} \not\models \phi$.

Beweis (Satz von Ehrenfeucht-Fraïssé). „$\Rightarrow$“ (für beliebiges Σ): Wir zeigen $(\mathfrak{A}, \bar{a}) \vDash \phi \Leftrightarrow (\mathfrak{B}, \bar{b}) \vDash \phi$ für $\mathsf{QR}(\phi) \leq m, \mathsf{FV}(\phi) \subseteq \{x_1, \ldots, x_n\}$. per Induktion über ϕ.

Die Booleschen Schritte sind trivial; z.B. Negation:

$$(\mathfrak{A}, \bar{a}) \vDash \neg\phi \Leftrightarrow (\mathfrak{A}, \bar{a}) \nvDash \phi \overset{\text{IV}}{\Longleftrightarrow} (\mathfrak{B}, \bar{b}) \nvDash \phi \Leftrightarrow (\mathfrak{B}, \bar{b}) \vDash \neg\phi$$

Atomare Formeln: Weil D gewinnt, ist $(\bar{a}, \bar{b})$ ein partieller Isomorphismus. Dies verwenden wir in den beiden folgenden Äquivalenzumformungen jeweils an der mit (1) markierten Stelle:

- $$(\mathfrak{A}, \bar{a}) \vDash P(x_{i_1}, \ldots, x_{i_n}) \Leftrightarrow (a_{i_1}, \ldots, a_{i_n}) \in \mathfrak{A}[\![P]\!]$$
  $$\overset{(1)}{\Longleftrightarrow} (b_{i_1}, \ldots, b_{i_n}) \in \mathfrak{B}[\![P]\!] \Leftrightarrow (\mathfrak{B}, \bar{b}) \vDash P(x_{i_1}, \ldots, x_{i_n})$$
- $(\mathfrak{A}, \bar{a}) \vDash x_i = x_j \Leftrightarrow a_i = a_j \overset{(1)}{\Longleftrightarrow} b_i = b_j \Leftrightarrow (\mathfrak{B}, \bar{b}) \vDash x_i = x_j$

Es bleibt der interessanteste Fall, $\exists x_{k+1}.\phi$. Sei also $(\mathfrak{A}, \bar{a}) \vDash \exists x_{k+1}.\phi$. Dann existiert a_{k+1} mit $(\mathfrak{A}, \bar{a}a_{k+1}) \vDash \phi$. Nun ist a_{k+1} ein möglicher Zug für S im Ehrenfeucht-Fraïssé-Spiel (mit m Runden), das ja von D gewonnen wird; sei also $b_{k+1} \in B$ die Gewinnantwort von D auf den Zug a_{k+1}, d.h. D gewinnt das $(m-1)$-Runden-Ehrenfeucht-Fraïssé-Spiel auf $(\mathfrak{A}, \bar{a}a_{k+1})$ und $(\mathfrak{B}, \bar{b}b_{k+1})$. Ferner gilt $\mathsf{QR}(\phi) \leq m - 1$ und $\mathsf{FV}(\phi) \subseteq \{x_1, \ldots, x_{k+1}\}$. Per Induktion folgt also $(\mathfrak{B}, \bar{b}b_{k+1}) \vDash \phi$, so dass $(\mathfrak{B}, \bar{b}) \vDash \exists x_{k+1}.\phi$. Die umgekehrte Implikation ist völlig symmetrisch.

Wir haben nun „$\Rightarrow$“ gezeigt. Sei ab jetzt Σ endlich; wir zeigen „$\Leftarrow$“. Dazu halten wir zunächst folgende Tatsache fest:

Lemma 29. *Sei Σ endlich. Dann gibt es bis auf logische Äquivalenz nur endlich viele ϕ mit $\mathsf{QR}(\phi) \leq m, \mathsf{FV}(\phi) \subseteq \{x_1, \ldots, x_n\}$.*

Beweis (Lemma 29). Wir halten zunächst fest, dass die entsprechende Aussage in der Aussagenlogik gilt: Es existieren bis auf logische Äquivalenz nur endlich viele aussagenlogische ψ mit $\mathsf{At}(\psi) \subseteq \{A_1, \ldots, A_r\}$ (nämlich so viele wie Wahrheitstafeln, also $2^{(2^r)}$).

Damit zeigen wir nun die Behauptung per Induktion über m:

$m = 0 : \phi$ ist eine aussagenlogische Formel über Atomen der Form $P(x_{i_1}, \ldots, x_{i_n})$ oder $x_i = x_j$, von beiden Typen gibt es nur endlich viele. (Hier geht ein, dass Σ endlich ist.)

$m \to m+1 : \phi$ ist bis auf α-Äquivalenz (d.h. Umbenennen quantifizierter Variablen) eine aussagenlogische Formel über Atomen wie bei $m = 0$ oder $\exists x_{n+1}.\phi$ mit $\mathsf{QR}(\phi) \leq m, \mathsf{FV}(\phi) \subseteq \{x_1, \ldots, x_{n+1}\}$ bei fest gewählter Variable x_{n+1}. Nach Induktionsvoraussetzung gibt es bis auf logische Äquivalenz nur endlich viele solche ϕ, also auch nur endlich viele Atome $\exists x_{n+1}.\phi$. $\square$

Damit läuft nun der Beweis von „⇐“ wie folgt. Wir beschreiben eine Gewinnstrategie für D durch eine *Invariante*: D muss sicherstellen, dass jeweils nach der i-ten Runde

$$(\mathfrak{A}, \bar{a}\bar{a}') \cong_{m-i} (\mathfrak{B}, \bar{b}\bar{b}')$$

gilt, für $i = 0, \ldots, m$. Wenn D dies gelingt, gewinnt sie, da dann nach der letzten Runde $(\mathfrak{A}, \bar{a}\bar{a}') \cong_0 (\mathfrak{B}, \bar{b}\bar{b}')$ gilt, insbesondere also $(\mathfrak{A}, \bar{a}\bar{a}')$ und $(\mathfrak{B}, \bar{b}\bar{b}')$ dieselben atomaren Formeln erfüllen, was (nach im wesentlichen derselben Rechnung wie im Fall für atomare Formeln im Beweis der ersten Implikation) bedeutet, dass $(\bar{a}\bar{a}', \bar{b}\bar{b}')$ ein partieller Isomorphismus ist.

Wir müssen noch zeigen, dass D die Invariante in der Tat spielen kann, d.h. dass a) sie am Anfang gilt, und b) D sie in jeder Runde durchsetzen kann. Ersteres ist gerade die Voraussetzung des Satzes; wir zeigen b):

Nehmen wir also an, das Spiel sei nach Runde $i < m$ gemäß Invariante in einer Konfiguration $(\bar{a}\bar{a}', \bar{b}\bar{b}')$ mit $(\mathfrak{A}, \bar{a}) \cong_{m-i} (\mathfrak{B}, \bar{b}\bar{b}')$ angekommen. O.E. spielt S jetzt $a_{k+i+1} \in A$; wir suchen also für D nach einer Antwort $b_{k+i+1} \in B$, so dass $(\mathfrak{A}, \bar{a}\bar{a}' a_{k+i+1}) \cong_{m-i-1} (\mathfrak{B}, \bar{b}\bar{b}' b_{k+i+1})$.

Da Σ endlich ist, gibt es nach Lemma 29 bis auf Äquivalenz nur endlich viele $\psi_1, \ldots, \psi_N$ mit $\mathsf{QR}(\psi_j) \leq m - (i+1)$ und $\mathsf{FV}(\psi_j) \subseteq \{x_1, \ldots, x_{k+i+1}\}$. Setze

$$\bar{\psi_j} = \begin{cases} \psi_j & \text{falls } \mathfrak{A}, (a_1, \ldots, a_{k+i+1}) \vDash \psi_j \\ \neg\psi_j & \text{sonst} \end{cases}$$

Dann

$$\begin{aligned}
&\mathfrak{A}, \bar{a}\bar{a}' a_{k+i+1} \vDash \bigwedge_{j=1}^{N} \bar{\psi_j} \\
&\implies \mathfrak{A}, \bar{a}\bar{a}' \vDash \exists x_{k+i+1} \bigwedge_{j=1}^{N} \bar{\psi_j} \\
&\implies \mathfrak{B}, \bar{b}\bar{b}' \vDash \exists x_{k+i+1} \bigwedge_{j=1}^{N} \bar{\psi_j} && ((\mathfrak{A}, \bar{a}\bar{a}') \cong_{m-i} (\mathfrak{B}, \bar{b}\bar{b}')) \\
&\implies \text{es existiert } b_{k+i+1} \text{ mit } \mathfrak{B}, \bar{b}\bar{b}' b_{k+i+1} \vDash \bigwedge_{j=1}^{N} \bar{\psi_j}.
\end{aligned}$$

Dieses b_{k+i+1} wählen wir als den gesuchten Antwortzug für D. Nach Definition der $\bar{\psi_j}$ gilt dann wie verlangt unsere Invariante $(\mathfrak{A}, \bar{a}\bar{a}' a_{k+i+1}) \cong_{m-(i+1)} (\mathfrak{B}, \bar{b}\bar{b}' b_{k+i+1})$. □

2.5 FO-Definierbarkeit

Wir kommen nunmehr auf unser Ausgangsproblem, Definierbarkeit von Eigenschaften in FOL, zurück, und nutzen die bisher entwickelte Ehrenfeucht-Fraïssè-Theorie aus, um Beispiele nicht FO-definierbarer Eigenschaften zu erhalten. Wir definieren den Begriff der FO-Definierbarkeit zunächst formal:

Definition 30. Seien $\mathcal{C} \subseteq \mathcal{D}$ Klassen von Σ-Strukturen.

- $\mathcal{C}$ heißt FO-definierbar in $\mathcal{D}$, wenn eine Formel ϕ existiert mit $\mathcal{C} = \{(\mathfrak{A}, \bar{a}) \in \mathcal{D} \mid (\mathfrak{A}, \bar{a}) \vDash \phi\}$

- $\mathcal{C}$ heißt FO-definierbar, wenn $\mathcal{C}$ FO-definierbar in der Klasse aller Σ-Strukturen ist.

Beispiel 31. $\Sigma = \{\leq\}$

$$ORD = \{(\mathfrak{A}, a_0 a_1) \mid \mathfrak{A} \text{ ist endliche lineare Ordnung mit Minumum } a_0 \text{ und Maximum } a_1\}$$
$$EVEN = \{(\mathfrak{A}, \bar{a}) \in ORD \mid |A| \text{ gerade}\}$$

Satz 32. *EVEN ist nicht FO-definierbar in ORD.*

Der Beweis benötigt folgendes Lemma, das besagt, dass hinreichend große Strukturen in *ORD* unter Formeln beschränkten Quantorenrangs nicht unterscheidbar sind:

Lemma 33. *Seien* $(\mathfrak{A}, \bar{a})$, $(\mathfrak{B}, \bar{b}) \in ORD$ *mit* $|A|, |B| > 2^m$. *Dann* $\mathfrak{A}, \bar{a} \cong_m \mathfrak{B}, \bar{b}$.

Beweis. Wir geben eine Gewinnstrategie für D im Ehrenfeucht-Fraïssé-Spiel an. Wir definieren die offensichtliche Distanzfunktion d auf A formal durch

$$d(a, a') = |\{c \in A \mid a < c \leq a'\}| \in \mathbb{N}$$

für $a \leq a'$, analog für $a' \leq a$, entsprechend auf B.

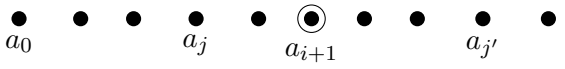
Wir beschreiben die Gewinnstrategie dann wieder durch eine *Invariante*, die jeweils nach Runde i gilt:

1. Die Konfiguration ist ein partieller Isomorphismus
2. Für $j, j' \in \{0, \ldots, i+1\}$ gilt $d(a_j, a_{j'}) = d(b_j, b_{j'})$ oder $d(a_j, a_{j'}), d(b_j, b_{j'}) \geq 2^{m-i}$

Damit gewinnt D nach Konstruktion, falls sie die Invariante durchhält. Dazu:

- Die Invariante gilt anfangs, d.h. für $i = 0$, da per Annahme $|A|, |B| > 2^m$, also $d(a_0, a_1), d(b_0, b_1) \geq 2^m$.
- Die Invariante lässt sich in Runde i durchhalten (so dass sie also nach Runde i gilt): Ohne Einschränkung zieht Spoiler a_{i+1}. Setze

$$j = \arg\max\{a_k \mid a_k < a_{i+1}\}$$
$$j' = \arg\min\{a_k \mid a_k > a_{i+1}\}$$



(Da es für S nicht sinnvoll ist, schon gespielte Elemente zu wiederholen, nehmen wir durchweg an, dass alle gespielten Werte paarweise verschieden sind. Insbesondere spielt S nicht noch einmal die Randelemente, so dass j und j' in der Tat existieren.) Dann gilt $a_j < a_{j'}$, also noch Teil 1. der Invarianten auch $b_j < b_{j'}$. D muss nun b_{i+1} wählen, so dass $b_j < b_{i+1} < b'_j$ (dies garantiert dann Teil 1. der Invarianten) und ferner:

- Wenn $d(a_j, a_{i+1}) < 2^{m-i}$, dann $d(b_j, b_{i+1}) = d(a_j, a_{i+1})$;
- andernfalls $d(b_j, b_{i+1}) \geq 2^{m-i}$;

– entsprechend für j'.

Wir zeigen, dass b_{i+1} mit den verlangten Eigenschaften nach Teil 2. der Invarianten vor Runde i existiert; im Detail unterscheiden wir gemäß der Invarianten zwei Fälle:

– $d(a_j, a_{j'}) = d(b_j, b_{j'})$. Dann existiert offenbar b_{i+1} mit $d(b_j, b_{i+1}) = d(a_j, a_{i+1})$ und $d(b_{i+1}, b_{j'}) = d(a_{i+1}, a_{j'})$, womit dann alle o.g. Bedingungen an b_{i+1} erfüllt sind.
– $d(a_j, a_{j'}), d(b_j, b_{j'}) \geq 2^{m-i+1}$. Wir unterscheiden weitere Fälle nach den Abständen $d(a_j, a_{i+1})$ und $d(a_{i+1}, a_{j'})$:
  * $d(a_j, a_{i+1}) < 2^{m-i}$. Wir wählen dann $b_{i+1} > b_j$ mit $d(b_j, b_{i+1}) = d(a_j, a_{i+1})$. Dann gilt $d(a_{i+1}, a_{j'}), d(b_{i+1}, b_{j'}) \geq 2^{m-i}$.
  * Der Fall $d(a_{i+1}, a_{j'}) < 2^{m-i}$ ist analog.
  * Ansonsten gilt $d(a_j, a_{i+1}), d(a_{i+1}, a_{j'}) \geq 2^{m-i}$. Da $d(b_j, b_{j'}) \geq 2^{m-i+1}$, können wir $b_j < b_{i+1} < b_{j'}$ wählen, so dass $d(b_j, b_{i+1}), d(b_{i+1}, b_{j'}) \geq 2^{m-i}$.

Es bleibt zu zeigen, dass weiterhin Teil 2. der Invarianten gilt; es reicht natürlich, die Bedingung für den neuen Index $i+1$ und einen weiteren Index $k \in \{0, \ldots, i+1\}$ zu zeigen. Die Fälle $k = j$, $k = j'$ sind nach Konstruktion erledigt, und der für $k = i+1$ ist trivial. O.E. gilt ansonsten $a_k > a_{j'}$, und dann auch $b_k > b_{j'}$, wegen Teil 1. der Invariante. Wir unterscheiden zwei Fälle, wiederum nach der Invariante vor der i-ten Runde:

1. $d(a_{j'}, a_k) = d(b_{j'}, b_k)$: Wir unterscheiden zwei Unterfälle:
   (a) $d(a_{i+1}, a_{j'}) = d(b_{i+1}, b_{j'})$: Dann gilt auch $d(a_{i+1}, a_k) = d(b_{i+1}, b_k)$.
   (b) $d(a_{i+1}, a_{j'}), d(b_{i+1}, b_{j'}) \geq 2^{m-i}$: Dann gilt erst recht $d(a_{i+1}, a_k), d(b_{i+1}, b) \geq 2^{m-i}$.
2. $d(a_{j'}, a_k), d(b_{j'}, b_k) \geq 2^{m-(i-1)}$: Dann auch $d(a_{i+1}, a_k), d(b_{i+1}, b_k) \geq 2^{m-(i-1)} > 2^{m-i}$.

□

Beweis (Satz 32). Man nehme jetzt an, ϕ mit $\mathsf{QR}(\phi) = m$ definiere die Klasse *EVEN* in *ORD*. Wähle dann $(\mathfrak{A}, \bar{a}), (\mathfrak{B}, \bar{b}) \in ORD$ mit $|A| > 2^m$ und $\mathfrak{B}$ mit $|B| = |A| + 1$. Nach obigem Lemma sind dann $(\mathfrak{A}, \bar{a})$ und $(\mathfrak{B}, \bar{b})$ m-äquivalent, im Widerspruch dazu, dass nach Wahl von $|A|$ und $|B|$ nur genau eine der Strukturen ϕ erfüllt. □

Wir haben damit das erste Beispiel einer in FOL nicht definierbaren Eigenschaft. Weitere (interessantere) Beispiele gewinnen wir durch das Prinzip der *logischen Reduktion*, das gewisse Anklänge an die Reduktionsprinzipien der Berechenbarkeits- und Komplexitätstheorie hat.

Im allgemeinen verläuft die *logische Reduktion* eines Definierbarkeitsproblems $\mathcal{C} \subseteq \mathcal{D}$ auf ein anderes $\mathcal{C}' \subseteq \mathcal{D}'$ wie folgt:

- Nimm an, eine FO-Formel ϕ definiere $\mathcal{C}'$ in $\mathcal{D}'$.
- Konstruiere aus ϕ eine FO-Formel ϕ', die $\mathcal{C}$ in $\mathcal{D}$ definiert.
- Wenn $\mathcal{C}$ in $\mathcal{D}$ nicht FO-definierbar ist, haben wir einen Widerspruch, so dass also $\mathcal{C}'$ in $\mathcal{D}'$ nicht FO-definierbar ist.

Beispiel 34 (Zusammenhang ungerichteter Graphen)**.** Wir zeigen, dass Zusammenhang ungerichteter Graphen nicht FO-definierbar ist.

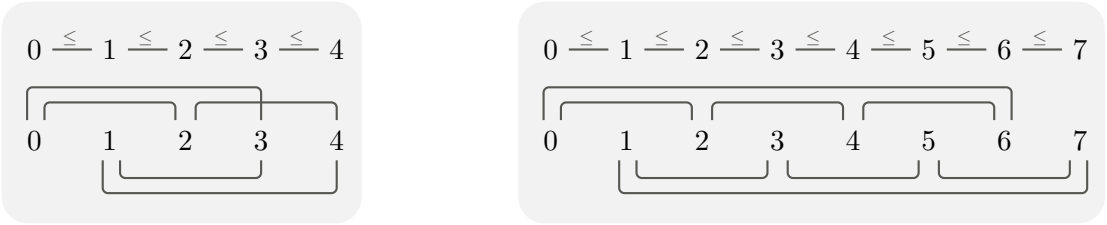
Ungerichtete Graphen sind Σ-Modelle für $\Sigma = \{R/2\}$, in denen R durch eine symmetrische Relation interpretiert wird. Wir notieren diese der Einfachheit halber in der Form $\mathfrak{A} = (A, R)$. Gegeben so ein Modell $\mathfrak{A}$ sagen wir, $\mathfrak{A}$ sei *zusammenhängend*, wenn es für alle $a, a' \in A$ in $\mathfrak{A}$ einen R-Pfad von a nach a' gibt, d.h. wenn $a_0, \ldots, a_n \in A$ existieren mit $a_0 = a$, $a_n = a'$, und $a_i R a_{i+1}$ für $i = 1, \ldots, n-1$ (bei $n = 0$ heißt dies, dass $a = a'$).

Wir nehmen nun also an, ϕ definiere Zusammenhang, und zeigen, dass dann auch *EVEN* in *ORD* FO-definierbar wäre. Wir gehen nach folgender Idee vor:

1. Wir konstruieren aus $(\mathfrak{A}, \bar{a}) \in \mathit{ORD}$ einen ungerichteten Graphen (A, R) mit

   $$(\mathfrak{A}, \bar{a}) \notin \mathit{EVEN} \Leftrightarrow (A, R) \text{ zusammenhängend},$$

   Wir deuten die Konstruktion hier nur bildlich an:



2. Wir drücken $R(x, y)$ durch eine Formel $\psi(x, y)$ über $\leq$ aus; siehe Übung.

3. Ersetze dann in ϕ jedes Atom $R(x, y)$ durch $\psi(x, y)$. Für die so erhaltene Formel ϕ' gilt dann

   $$(\mathfrak{A}, \bar{a}) \vDash \neg\phi' \iff (A, R) \text{ nicht zusammenhängend} \iff (\mathfrak{A}, \bar{a}) \in \mathit{EVEN},$$

   im Widerspruch zur Nicht-Definierbarkeit von *EVEN*.

Kapitel 3

Beschreibungslogik

Beschreibungslogiken sind der aktuell erfolgreichste logikbasierte Wissensrepräsentationsformalismus. Wir führen zunächst die Beschreibungslogik $\mathcal{ALC}$ ein, eine Erweiterung der Aussagenlogik, die sich als Fragment in FOL einbetten lässt. Beschreibungslogiken bieten eine „objektorientierte“ Sicht auf Individuen, indem sie diese *Klassen* zuordnen und andererseits mit Attributen bzw. Eigenschaften versehen, die sie zu anderen Individuen in Beziehung setzen; die Sichtweise ist hierbei immer *lokal*, d.h. Beziehungen werden immer von einem Referenzindividuum aus gesehen (das sich allerdings während der Auswertung einer Formel ändern kann). Hierbei entsprechen Klassen letztlich unären Prädikaten und Eigenschaften binären Prädikaten.

Syntaktisches Material Eine *Signatur* in Beschreibungslogik besteht aus Mengen

- N_I von *Individuennamen* (diesen Aspekt lassen wir allerdings zunächst einmal weg)
- N_C von *Klassennamen* / atomaren Konzepten
- N_R *Eigenschaften* / Rollen(namen).

Syntax Konzepte („Klassenausdrücke / class expressions“) werden durch die Grammatik

$$C, D ::= \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.\, C \qquad (A \in N_C, R \in N_R)$$

erzuegt.

Interpretationen Eine Interpretation $\mathcal{I}$ besteht aus

- einem (Grund-)Bereich (*domain*), d.h. einer Menge $\Delta^{\mathcal{I}} \neq \emptyset$
- einer Teilmenge $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ für jedes $A \in N_C$; und
- einer Relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ für jedes $R \in N_R$.

Direkte Semantik Wir interpretieren Konzepte C als Teilmengen $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, ihre *Extension*, und schreiben $\mathcal{I}, d \models C$ für $d \in C^{\mathcal{I}}$. Extensionen sind rekursiv definiert:

$$\bot^{\mathcal{I}} = \emptyset \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} - C^{\mathcal{I}} \qquad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(\exists R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}}.\, (d, e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$$

Beispiel: $\exists$ hasChild. gingerHaired.

Wir führen neben *existentiellen Restriktionen* $\exists R.\, C$ noch *universelle Restriktionen* $\forall R.\, C :\equiv \neg\exists R.\, \neg C$ ein. Als Semantik ergibt sich dann

$$(\forall R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}}.(d, e) \in R^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\}$$

Beispiel: $\forall$ hasAccount. overDrawn

3.1 Modallogik

Es hat sich herausgestellt, dass $\mathcal{ALC}$ in der oben eingeführten *German-DL*-Syntax äquivalent ist zu einer klassischen Modallogik, nämlich zu *multimodalem K* bzw. zu K_m, wobei m die Anzahl der Rollen ist ($m = |N_R|$).

Formeln in K_m werden erzeugt durch die Grammatik

$$\phi ::= \bot \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Diamond_i \phi \quad (p \in P,\ i \in I),$$

wobei I eine m-elementige Indexmenge ist. Wir definieren ferner $\Box_i \phi :\equiv \neg\Diamond_i\neg\phi$. Diese Formeln entsprechen den Konzepten in $\mathcal{ALC}$; wenn $N_R = \{R_1, \ldots, R_m\}$, entspricht hierbei $\Diamond_i$ der existentiellen Restriktion $\exists R_i$, und somit $\Box_i$ der universellen Restriktion $\forall R_i$.

Den Interpretationen entsprechen dann *Kripke-Modelle* $\mathfrak{M} = (\mathcal{F}, V)$, wobei $\mathcal{F} = (X, (R_i)_{i \in I})$ ein (multimodaler) *Kripkerahmen* bestehend aus

- einer nichtleeren Menge X von *Zuständen* oder *Welten* und
- Relationen $R_i \subseteq X \times X$ für $i = 1, \ldots, m$ ('Erreichbarkeit', 'Transition', 'Übergang')

und V eine *Valuation* $V : P \to \mathcal{P}(X)$ ist. Die Semantik (Erfülltheit, Extensionen) ist dann mutatis mutandis gleich.

Beispiel: Die Formel $\Diamond_i$ gingerHaired entspricht dem Konzept $\exists$ hasChild. gingerHaired, wenn wir $R_i =$ hasChild unterstellen.

Wörterbuch

Modallogik	Beschreibungslogik
Formel	Konzept
(Kripke-)Modell	Interpretation
Modalität ($\Diamond, \Box$)	Restriktion ($\exists R$, $\forall R$)
Relation	Rolle
Erfüllbarkeit	Erfüllbarkeit oder Konsistenz

Indirekte Semantik Alternativ zur oben diskutierten direkten Semantik können wir eine Semantik äquivalenterweise auch mittels einer Übersetzung ST_y (der *standard translation*) in Logik erster Stufe angeben, mit

$$\underbrace{\mathfrak{M}, w \models \phi}_{\text{gemäß direkter Semantik}} \Leftrightarrow \mathfrak{M}, w \models \mathsf{ST}_y(\phi) \qquad \mathfrak{M} = ((X, (R_i)), V), w \in X$$

und $\mathsf{FV}(\mathsf{ST}_y(\phi)) \subseteq \{y\}$:

$$\begin{aligned} &\mathsf{ST}_y(\bot) = \bot && \mathsf{ST}_y(p) = p(y) \\ &\mathsf{ST}_y(\neg\phi) = \neg\mathsf{ST}_y(\phi) && \mathsf{ST}_y(\phi \wedge \psi) = \mathsf{ST}_y(\phi) \wedge \mathsf{ST}_y(\psi) \\ &\mathsf{ST}_y(\Diamond_i\phi) = \exists z.R_i(y,z) \wedge \mathsf{ST}_z(\phi) && (z \text{ frisch}) \end{aligned}$$

In der Tat genügen zwei Variablen x_0, x_1:

$$\mathsf{ST}_{x_j}(\Diamond_i\phi) = \exists x_{1-j}.R_i(x_j, x_{1-j}) \wedge \mathsf{ST}_{x_{1-j}}(\phi).$$

Somit können wir K_m sogar ins sogenannte Zweivariablenfragment von FOL übersetzen, womit nach bekannten Resultaten bereits die Entscheidbarkeit der Modallogik folgt. Wir werden letztere hier aber direkt zeigen.

3.2 Terminologien

Beschreibungslogische Wissensbasen formalisieren einerseits Zusammenhänge zwischen Konzepten („Zu jedem Topf gehört ein Deckel“) und andererseits aus Aussagen über konkrete Individuen („John Doe hat Masern“). Wir konzentrieren uns hier auf den ersteren Aspekt, also die Repräsentation sogenannten *terminologischen* Wissens (den zweiten Typ bezeichnet man als *assertionales* Wissen). Terminologisches Wissen wird in Form einer *TBox* organisiert (und assertionales in einer *ABox*), die im allgemeinsten Fall aus *general concept inclusions (gcis)*, d.h. Axiomen der Form $C \sqsubseteq D$ für Konzepte C, D besteht. Erfülltheit solcher gcis wird für Interpretationen als ganzes definiert:

$$\mathcal{I} \models C \sqsubseteq D :\Leftrightarrow C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$$

Wir definieren ferner Axiome $C = D$ als Abkürzungen für $C \sqsubseteq D \wedge D \sqsubseteq C$, d.h.

$$\mathcal{I} \models C = D \Leftrightarrow C^{\mathcal{I}} = D^{\mathcal{I}}$$

Beispiel 35.

$$\begin{aligned} \mathsf{Grandfather} &= \mathsf{Male} \sqcap (\exists\, \mathsf{hasChild}.\, \mathsf{Mother} \sqcup \exists \mathsf{hasChild}.\, \mathsf{Father}) \\ \mathsf{Grandfather} &= \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\, \exists \mathsf{hasChild}.\, \top \\ \top &= \mathsf{Person} \rightarrow (\mathsf{Male} \sqcup \mathsf{Female}) \\ \top &= \forall \mathsf{hasChild}.\, \mathsf{Person} \end{aligned}$$

Achtung: Für die Erfüllbarkeit von Axiomen ist es relevant, dass der Bereich per Definition nichtleer ist; damit ist z.B. die gci $\top \sqsubseteq \bot$ unerfüllbar.

Definition 36. Wie angekündigt definieren wir eine TBox einfach als eine endliche Menge $\mathcal{T}$ von gcis. Ein $\mathcal{T}$-*Modell* ist dann eine Interpretation $\mathcal{I}$ mit $\mathcal{I} \models \mathcal{T}$, d.h. $\mathcal{I} \models \phi$ für alle gcis $\phi \in \mathcal{T}$. Ein Konzept C ist *erfüllbar über* $\mathcal{T}$, wenn ein $\mathcal{T}$-Modell $\mathcal{I}$ existiert mit $C^{\mathcal{I}} \neq \emptyset$ (insbesondere ist C *erfüllbar* schlechthin, wenn eine Interpretation $\mathcal{I}$ mit $C^{\mathcal{I}} \neq \emptyset$ existiert).

Wir definieren *lokale Konsequenz* $\models$ per

$$C \models D :\Leftrightarrow \forall \mathcal{I}.\, \mathcal{I} \models C \sqsubseteq D;$$

d.h. $C \models D$, wenn für alle $\mathcal{I}$ und alle $d \in \Delta^{\mathcal{I}}$ mit $\mathcal{I}, d \models C$ auch $\mathcal{I}, d \models D$ gilt. Wir sagen dann auch, die gci $C \sqsubseteq D$ sei *gültig*. Ein Konzept C ist *gültig*, wenn $\top \sqsubseteq C$ gültig ist. (Damit ist $C \sqsubseteq D$ genau dann gültig, wenn $\neg C \sqcup D$ gültig ist.)

Schließlich definieren wir *globale Konsequenz* oder *TBox-Konsequenz* $\models_g$ per

$$\mathcal{T} \models_g \phi :\Leftrightarrow \forall \mathcal{I}.\, \mathcal{I} \models \mathcal{T} \Rightarrow \mathcal{I} \models \phi$$

für TBoxen $\mathcal{T}$ und gcis ϕ.

Wir können auch eine globale Konsequenzrelation zwischen Konzepten einführen per $C \models_g D :\Longleftrightarrow \{\top \sqsubseteq C\} \models_g \top \sqsubseteq D$, d.h. $C \models_g D$ wenn für jede Interpretation $\mathcal{I}$ mit $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ auch $D^{\mathcal{I}} = \Delta^{\mathcal{I}}$ gilt. Klar ist, dass $C \models D$ impliziert, dass auch $C \models_g D$. Die Umkehrung gilt nicht: Z.B. gilt $A \models_g \forall R.\, A$, aber nicht $A \models \forall R.\, A$.

Die zentralen *Deduktionsprobleme* (*Reasoning Tasks*) sind

1. *TBox-Konsistenz*: existiert ein $\mathcal{T}$-Modell?
2. *Konzepterfüllbarkeit*: Ist C erfüllbar über $\mathcal{T}$?
3. *Subsumption*: gilt $\mathcal{T} \models_g C \sqsubseteq D$?

Diese Probleme sind teilweise aufeinander reduzierbar:

- $1 \to 2$: $\mathcal{T}$ konsistent $\Leftrightarrow$ $\top$ erfüllbar über $\mathcal{T}$
- $2 \to 3$: C unerfüllbar über $\mathcal{T} \Leftrightarrow \mathcal{T} \models_g C \sqsubseteq \bot$
- $3 \to 2$: $\mathcal{T} \models_g C \sqsubseteq D \Leftrightarrow C \sqcap \neg$D unerfüllbar über $\mathcal{T}$.

Für manche Zwecke muss das Format der TBox in einem der folgenden Sinne eingeschränkt werden:

Definition 37. Sei $\mathcal{T}$ eine TBox. $\mathcal{T}$ ist *klassisch*, wenn folgendes gilt:

1. Alle Axiome in $\mathcal{T}$ sind Konzeptgleichungen.
2. In $C = D \in \mathcal{T}$ ist C stets atomar.
3. Die linken Seiten aller $C = D \in \mathcal{T}$ sind paarweise verschieden.

(Die Einschränkung auf Gleichungen ist eigentlich keine: Eine gci $A \sqsubseteq C$ ist äquivalent zu $A = C \sqcap A$.)

Ferner ist $\mathcal{T}$ *azyklisch*, wenn $\mathcal{T}$ klassisch ist und die durch

$$A\ U\ B :\Leftrightarrow \exists (A = C) \in \mathcal{T}, B \text{ kommt in } C \text{ vor}$$

definierte Relation U (*uses*) auf atomaren Konzepten azyklisch ist.

Beispiel 38. Die nur aus dem Axiom

$$\mathsf{PopularGuy} = \forall\, \mathsf{hasFriend}.\, \mathsf{PopularGuy}$$

bestehende TBox ist klassisch, aber *nicht* azyklisch, da $\mathsf{PopularGuy}\ U\ \mathsf{PopularGuy}$.

Die Kodierung von $\sqsubseteq$ durch $=$ wie für klassische TBoxen ist natürlich nicht azyklisch. Dennoch können wir $\sqsubseteq$ auch in azyklischen TBoxen zumindest erfüllbarkeitsäquivalent durch $=$ emulieren:

Lemma 39. *Eine TBox der Form $\mathcal{T} \cup \{A \sqsubseteq C\}$ ist für Zwecke der Konzepterfüllbarkeit äquivalent (aber natürlich nicht logisch äquivalent) zu $\mathcal{T} \cup \{A = C \sqcap A^*\}$ für ein frisches atomares Konzept A^*.*

Beweis. Sei D ein Konzept; wir zeigen, dass D genau dann erfüllbar über $\mathcal{T} \cup \{A \sqsubseteq C\}$ ist, wenn D erfüllbar über $\mathcal{T} \cup \{A = C \sqcap A^*\}$ ist.

„$\Leftarrow$“: Sei $\mathcal{I}$ eine Interpretation mit $\mathcal{I} \models \mathcal{T} \cup \{A = C \sqcap A^*\}$ und $D^\mathcal{I} \neq \emptyset$. Dann gilt auch $\mathcal{I} \models \mathcal{T} \cup \{A \sqsubseteq C\}$, da $A^\mathcal{I} = C^\mathcal{I} \cap (A^*)^\mathcal{I} \subseteq C^\mathcal{I}$.

„$\Rightarrow$“: Sei $\mathcal{I} \models \mathcal{T} \cup \{A \sqsubseteq C\}$ und $D^\mathcal{I} \neq \emptyset$. Definiere $\mathcal{I}^*$ wie $\mathcal{I}$, aber mit $(A^*)^{\mathcal{I}^*} = A^\mathcal{I}$. Dann gilt weiterhin $\mathcal{I}^* \models \mathcal{T}$ sowie $D^{\mathcal{I}^*} \neq \emptyset$, da A^* in $\mathcal{T}$ und D nicht vorkommt. Ferner gilt $\mathcal{I}^* \models A = C \sqcap A^*$: da A^* nicht in C vorkommt, gilt $C^{\mathcal{I}^*} = C^\mathcal{I}$, also $C^{\mathcal{I}^*} \cap (A^*)^{\mathcal{I}^*} = C^\mathcal{I} \cap A^\mathcal{I} = A^\mathcal{I} = A^{\mathcal{I}^*}$, wobei wir im vorletzten Schritt verwenden, dass $\mathcal{I} \models A \sqsubseteq C$. □

3.3 Bisimilarität

Genau wie für Logik erster Stufe gibt es auch für Modallogik (wir verwenden ab jetzt in der technischen Entwicklung vorwiegend die Schreibweise und Terminologie der Modallogik; alle Resultate gelten aber natürlich genauso für Beschreibungslogik) einen passenden semantischen Äquivalenzbegriff, die *Bisimilarität*. Wir führen zunächst eine schwächere Relation ein, die eher den Charakter einer Ordnungsrelation hat (allerdings im allgemeinen nicht antisymmetrisch ist):

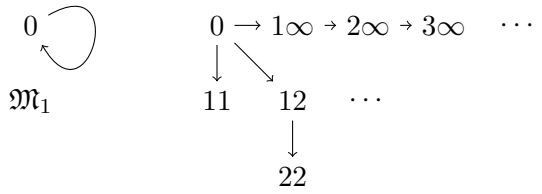
Definition 40 (Simulation)**.** Seien $\mathfrak{M}_1 = ((X_1, R_1), V_1), \mathfrak{M}_2 = ((X_2, R_2), V_2)$ Kripkemodelle. Eine Relation $S \subseteq X_1 \times X_2$ heißt *Simulation*, wenn für alle $(x, y) \in S$ gilt:

- $x \in V_1(p) \Rightarrow y \in V_2(p)$ für alle $p \in P$
- $xR_1x' \Rightarrow \exists y'.\, yR_2y' \wedge x'Sy'$:

$$\begin{array}{ccc} x & S & y \\ R_1 & \searrow & R_2 \\ x' & S & \exists y' \end{array}$$

Der Zustand x ist *similar zu* y, bzw. y simuliert x ($x \preceq y$), wenn eine Simulation S mit xSy existiert.

Beispiel 41. Man betrachte die wie folgt graphisch dargestellten Kripke-Modelle $\mathfrak{M}_1$, $\mathfrak{M}_2$:



Dann ist $S_1 = \{(0,0), (0,1\infty), (0,2\infty), \ldots\}$ eine Simulation, aber auch – in der anderen Richtung – $S_2 = X_2 \times X_1$.

Hieraus gewinnen wir wie folgt den Begriff der Bisimilarität:

Definition 42. Eine Relation S zwischen Kripkemodellen wie in Definition 40 ist eine *Bisimulation*, wenn sowohl S als auch

$$S^- = \{(y,x) \mid (x,y) \in S\}$$

Simulationen sind. Ein Zustand x ist *bisimilar zu* y ($x \approx y$), wenn eine Bisimulation S mit xSy existiert.

Beispiel 43. Wir betrachten wieder die Modelle $\mathfrak{M}_1, \mathfrak{M}_2$ aus Beispiel 41. Wir haben gesehen, dass die beiden Wurzelzustände 0 sich gegenseitig simulieren; sie sind aber nicht bisimilar. Wenn man nämlich eine Bisimulation S mit $0S0$ hätte, ergäbe sich wie folgt ein Widerspruch:

$\mathfrak{M}_1$		$\mathfrak{M}_2$	
0	S	0	
↓		↓	
0	S	11	
↓		↓	$\lightning$
0		11	

Dabei wird im ersten Schritt verwendet, dass S^- eine Simulation ist, und der Widerspruch ergibt sich im zweiten Schritt zu der Forderung, dass S eine Simulation ist.

Wir zeigen als nächstes, dass der Begriff der Bisimilarität für unsere Zwecke seinen Daseinsgrund erfüllt:

Lemma 44 (Bisimulationsinvarianz der Modallogik)**.** *Bisimilare Zustände erfüllen die gleichen modalen Formeln.*

Beweis. Sei S eine Bisimulation. Wir zeigen

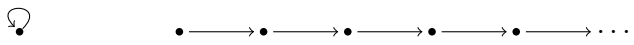
$$\forall x, y.\, xSy \Rightarrow (x \models \phi \Leftrightarrow y \models \phi)$$

per Induktion über ϕ; es reicht jeweils, „⇒“ zu zeigen (mit weiterhin „⇔“ in der Induktionsvoraussetzung). Sei also xSy. Die Booleschen Fälle ($\bot, \neg, \wedge$) sind trivial. Die verbleibenden Fälle sind:

- $\phi = p$: Sei $x \models p$; dann $x \in V_1(p)$ nach Definition der Semantik, also $y \in V_2(p)$, da S eine Simulation ist. Es folgt $y \models p$.
- $\phi = \Diamond\psi$: Sei $x \models \Diamond\psi$. Dann existiert x' mit $xR_1x', x' \models \psi$. Da S eine Simulation ist, existiert y' mit yR_2y' und $x'Sy'$. Nach Induktionsvoraussetzung folgt $y' \models \psi$, also $y \models \Diamond\psi$. □

Beispiel 45. Wir können somit anhand geeigneter modaler Formeln sehr schnell zeigen, dass die Wurzeln der beiden Modelle aus Beispiel 41 nicht bisimilar sind: Wir haben $\mathfrak{M}_2, 0 \models \Diamond\Box\bot$, aber $\mathfrak{M}_1, 0 \not\models \Diamond\Box\bot$.

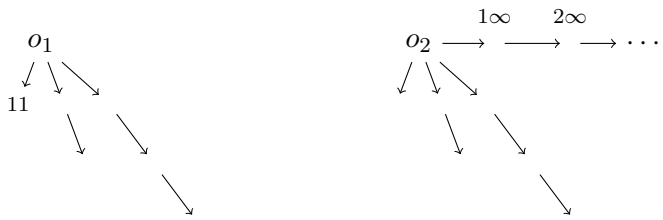
Beispiel 46. Umgekehrt können wir schließen, dass Eigenschaften von Zuständen, die nicht bisimulationsinvariant sind, nicht durch modale Formeln ausdrückbar sind. Z.B. ist die Eigenschaft „Zustand x erreicht sich selbst“ (xRx) nicht bisimulationsinvariant, also nicht durch eine modale Formel ausdrückbar: Die Wurzeln der beiden Modelle



sind bisimilar; links ist aber die Eigenschaft erfüllt, und rechts nicht.

Umgekehrt gilt ohne weitere Annahmen im Allgemeinen *nicht*, dass Zustände, die die gleichen Modalformeln erfüllen, bereits bisimilar sind:

Beispiel 47. Man betrachte die beiden Kripkemodelle



Die beiden Wurzelzustände o_1, o_2 sind *nicht* bisimilar, denn wenn S eine Bisimulation mit $o_1 S_2 o_2$ wäre, ergäbe sich wie folgt ein Widerspruch:

$$o_1 \longrightarrow 1i \longrightarrow \cdots \longrightarrow ii \not\longrightarrow$$

$$S \qquad S \qquad\qquad S$$

$$o_2 \to 1\infty \to \cdots \to i\infty \longrightarrow$$

in Worten: Da $o_2 \to 1\infty$, muss es ein i mit $o_1 \to 1i$ und $1i\, S\, 1\infty$ geben; es folgt induktiv schließlich $ii\, S\, i\infty$, und dann ergibt sich ein Widerspruch, da $i\infty$ einen Übergang hat, ii aber nicht. Die Zustände o_1 und o_2 erfüllen aber die gleichen Formeln, da für jede modale Formel mit Schachtelungstiefe n der Modaloperatoren nur Zustände bis zur Entfernung n von der Wurzel relevant sind.

Unter einer geeigneten Endlichkeitsannahme gilt die bewusste Umkehraussage aber doch:

Definition 48. Ein Modell $\mathfrak{M} = ((X, R), V)$ heißt *endlich verzweigend*, wenn für jedes $x \in X$ die Menge $\{x' \mid xRx'\}$ endlich ist.

Satz 49 (Hennessy/Milner)**.** *Seien $\mathfrak{M}_1 = ((X_1, R_1), V_1), \mathfrak{M}_2 = ((X_2, R_2), V_2)$ endlich verzweigende Kripkemodelle. Wenn $x \in X_1$, $y \in X_2$ die gleichen Modalformeln erfüllen, dann gilt $x \approx y$.*

Beweis. Setze

$$S = \{(x, y) \in X_1 \times X_2 \mid x \text{ und } y \text{ erfüllen dieselben modalen Formeln}\}$$

Wir zeigen, dass S eine Bisimulation ist; per Symmetrie reicht es zu zeigen, dass S eine Simulation ist. Sei also xSy.

- Sei $x \in V_1(p)$, d.h. $x \models p$. Dann gilt nach Voraussetzung auch $y \models p$ und somit $y \in V_2(p)$.
- Sei xR_1x'. $\mathfrak{M}_2$ ist endlich verzweigend; sei also

  $$\{y' \mid yR_2y'\} = \{y_1, \ldots, y_n\}.$$

  Wir müssen zeigen, dass $x'Sy_i$ für ein i. Wir nehmen für Zwecke eines Widerspruchs an, es gelte $\neg(x'Sy_i)$ für alle i. Da modale Formeln negiert werden können, existieren dann $\phi_1, \ldots, \phi_n$ mit $x' \models \phi_i$ und $y_i \models \neg\phi_i$ für $i = 1, \ldots, n$. Dann gilt

  $$y \models \Box(\neg\phi_1 \vee \ldots \vee \neg\phi_n),$$

  da xSy also auch

  $$x \models \Box(\neg\phi_1 \vee \ldots \vee \neg\phi_n)$$

  und somit gemäß der Semantik der Modalität $\Box$

  $$x' \models \neg\phi_1 \vee \ldots \vee \neg\phi_n,$$

  im Widerspruch zu $x' \models \phi_i$ für alle i. □

In der Tat gilt noch eine weitere Umkehrung der Bisimulationsinvarianz (Lemma 44), die wir hier nicht beweisen:

Satz 50 (van Benthem/Rosen)**.** *Sei $\phi(x)$ eine Formel in Logik erster Stufe (in der Signatur der Korrespondenzsprache, die wir auch für die Standarübersetzung verwenden, d.h. mit einem binären Prädikat und mit unären Prädikaten für die propositionalen Atome). Wenn $\phi(x)$ bisimulationsinvariant ist (d.h. für $\mathfrak{M}_1, x_1$ und $\mathfrak{M}_2, x_2$ bisimilar gilt $\mathfrak{M}_1, x_1 \models \phi(x) \Leftrightarrow \mathfrak{M}_2, x_2 \models \phi(x)$), dann existiert eine modale Formel ψ mit $\phi(x) \equiv \mathsf{ST}_x(\psi)$. Diese Aussage gilt auch über endlichen Modellen.*

3.4 Tableaux für $\mathcal{ALC}$

Wir führen nunmehr ein Tableausystem zur Erfüllbarkeitsprüfung von Konzepten in $\mathcal{ALC}$ ein; das System erweitert einfach das (echte) aussagenlogische Tableausystem um die Regel

$$\frac{\Gamma, \Box\phi_1, \ldots, \Box\phi_n, \neg\Box\phi_0}{\phi_1, \ldots, \phi_n, \neg\phi_0}\ (\neg\Box),$$

wobei wir verlangen, dass Γ keine Formel der Form $\Box\psi$ enthält (das ist gegenüber der uneingeschränkt anwendbaren Regel ohne Einschränkung; warum?)

Beispiel 51. 1. $\Diamond(\Diamond p \wedge \Diamond\neg p)$ ist erfüllbar:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg\Box\neg(\neg\Box\neg p \wedge \neg\Box p)}{\neg\neg(\neg\Box\neg p \wedge \neg\Box p)}(\neg\Box)}{\neg\Box\neg p \wedge \neg\Box p}(\neg\neg)}{\neg\Box\neg p, \neg\Box p}(\wedge)}{\neg\neg p}(\neg\Box) \qquad \neg p}{p}(\neg\neg)$$

2. $\Box(q \to \Diamond p), \Diamond q, \Box\Box\neg p$ ist nicht erfüllbar:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Box\neg(q \wedge \Box\neg p), \neg\Box\neg q, \Box\Box\neg p}{\neg(q \wedge \Box\neg p), \neg\neg q, \Box\neg p}(\neg\Box)}{\neg(q \wedge \Box\neg p), q, \Box\neg p}(\neg\neg)}{\neg q, q, \Box\neg p}(\neg\wedge) \qquad \dfrac{\neg\Box\neg p, q, \Box\neg p}{p, \neg p}(\neg\Box)}{\bot} \qquad \bot$$

Ein Label Γ ist *propositional saturiert*, wenn keine propositionale Regel auf Γ anwendbar ist, äquivalenterweise dann, wenn Γ keine Formeln $\phi \wedge \psi$ enthält, und $\neg\psi$ nur für $\psi = q \notin \Gamma$. Dann hat Γ die Form $\Gamma = \pm q_1, \ldots, \pm q_n, \pm\Box\phi_1, \ldots, \pm\Box\phi_k$, wobei $\pm \in \{\cdot, \neg\}$ jeweils für Negation oder eben keine Negation steht, wobei kein q_i mit zwei verschiedenen Vorzeichen vorkommt.

Ein Label Γ heißt *eklatant inkonsistent*, wenn $q, \neg q \in \Gamma$ existieren.

Eine formale(re) Definition von Tableaux ist dann wie folgt:

- Tableaus sind Bäume mit gelabelten Knoten; wie bisher bezeichnet $l(n)$ den Label eines Knotens n.
- Knoten werden unterschieden in
  - *AND-Knoten*: Dies sind die Knoten n, für die $l(n)$ propositional saturiert ist; die Kinder von n entstehen in diesem Fall aus allen anwendbaren Instanzen von $(\neg\Box)$.
  - *OR-Knoten*: Alle anderen Knoten; die Kinder sind in diesem Fall alle Konklusionen einer ausgewählten auf den Knoten anwendbaren propositionalen Regel (wie schon im aussagenlogischen Fall).

Ein Knoten n heißt *erfolgreich*, wenn

- n ein erfolgreiches Kind hat, wenn n ein OR-Knoten ist
- alle Kinder von n erfolgreich sind, wenn n ein AND-Knoten ist (insbesondere also dann, wenn n bzw. $l(n)$ *saturiert* ist, d.h. überhaupt keine Regeln mehr auf $l(n)$ anwendbar sind).

Lemma 52 (Korrektheit)**.** *Wenn $L(n)$ erfüllbar ist, dann ist n erfolgreich.*

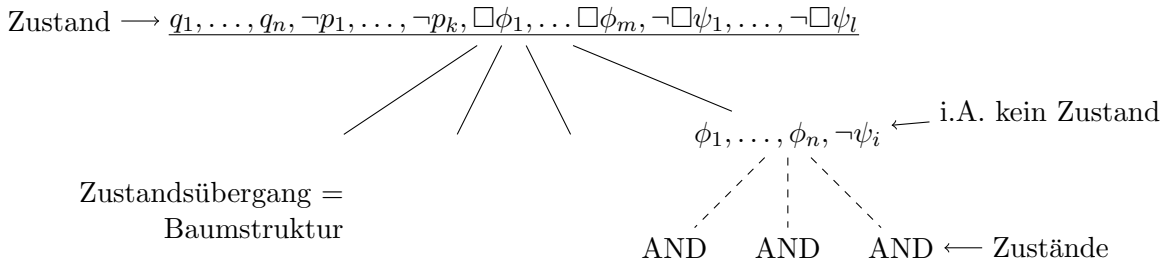
Beweis. Per Induktion über die syntaktische Größe von $L(n)$. Der Induktionsschritt für den Fall, dass n ein OR-Knoten ist, wird wie im aussagenlogischen Fall abgehandelt. Sei nun also n ein AND-Knoten. Sei $\mathfrak{M} = ((X, R), V)$, $x \in X$, $\mathfrak{M}, x \models L(n)$. Ein gegebenes Kind n' von n entsteht dann durch Anwendung von $(\neg\Box)$ auf $\Box\phi_1, \ldots, \Box\phi_n, \neg\Box\phi_0 \in L(n)$; dann $L(n') = \{\phi_1, \ldots, \phi_n, \neg\phi_0\}$. Wegen $x \models L(n)$ existiert xRx' mit $x' \models \phi_i$, $i = 1, \ldots, n$, und $x' \models \neg\phi_0$, also $x' \models L(n')$. Damit ist $L(n')$ erfüllbar, nach Induktionsvoraussetzung also n' erfolgreich. □

3.5 Vollständigkeit des Tableaux-Algorithmus

Wir haben auch die Umkehrung des Korrektheitslemmas, d.h. die Tableaumethode ist *vollständig*:

Satz 53 (Vollständigkeit)**.** *Wenn ein Tableauknoten n erfolgreich ist, dann ist $L(n)$ erfüllbar.*

Beweis. Skizze: Wir haben ein Modell von $L(n)$ zu konstruieren. Zustände des Modells sind erfolgreiche AND-Knoten; die Zustandsübergangsrelation lesen wir dann aus den Tableau ab:



Formal: Für Tableauknoten n, m schreiben wir $n \to m$, wenn m Nachfolger von n im Tableau ist, und spezieller $n \to_{\neg\Box} m$, wenn m durch Anwendung von $(\neg\Box)$ entsteht, sowie $n \to_{PL} m$, wenn m durch Anwendung einer propositionalen Regel entsteht. Wie üblich schreiben wir nR^*m, wenn m von n aus über beliebig viele (auch 0) R-Schritte gemäß einer Relation R erreichbar ist. Wir halten fest:

Definition 54. Für einen Label Γ und eine Formel ψ schreiben wir $\Gamma \vdash_{PL} \psi$, wenn ψ aus Formeln in Γ rein durch aussagenlogisches Schließen folgt, d.h. wenn $\bigwedge\Gamma \to \psi$ eine Substitutionsinstanz einer aussagenlogischen Tautologie ist. Für einen Label Δ schreiben wir $\Gamma \vdash_{PL} \Delta$, wenn $\Gamma \vdash_{PL} \bigwedge\Delta$. (Mit $\bigwedge\Gamma$ bezeichnen wir dabei die Konjunktion aller Formeln in Γ.)

Lemma 55. *Wenn $n \to_{PL}^* m$, dann $L(m) \vdash_{PL} L(n)$.*

Beweis. Es reicht offenbar, die Behauptung für den Fall $n \to_{PL} m$ zu beweisen; in diesem Fall zeigt man sie einfach durch Inspektion der propositionalen Regeln. □

Wir konstruieren nunmehr aus einem Tableau ein Kripkemodell $\mathfrak{M} = ((X, R), V)$ per

$$\begin{aligned} X &= \{n \mid n \text{ erfolgreicher AND-Knoten}\} \\ nRn' &\Leftrightarrow \exists n''.n \to_{\neg\Box} n'' \to_{PL}^* n' \\ n \in V(p) &\Leftrightarrow p \in L(n) \end{aligned}$$

Das so konstruierte Modell leistet, was es soll:

Lemma 56 (Wahrheitslemma)**.** *Für jedes $n \in X$ gilt $\mathfrak{M}, n \models L(n)$.*

Beweis. Da (X, R) ein endlicher Baum ist (diese Eigenschaft erbt (X, R) vom Tableau), können wir Induktion über n verwenden. Wir unterscheiden Fälle über die Form in $L(n)$ vorkommender Formeln, unter Berücksichtigung der Tatsache, dass $L(n)$ propositional saturiert ist:

- $p \in L(n) \Rightarrow n \in V(p) \Rightarrow n \models p$

- $\neg p \in L(n) \overset{L(n) \text{ nicht eklatant inkonsistent}}{\Rightarrow} p \notin L(n) \Rightarrow n \notin V(p) \Rightarrow n \models \neg p$
- $\neg\Box\phi \in L(n)$. Dann existiert $n \to_{\neg\Box} n'$ im Tableaux mit $\neg\phi \in L(n')$ und n' erfolgreich. Da n' erfolgreich ist, existiert $n' \to_{PL}^* n''$ mit $n'' \in X$ und nRn''. Nach Lemma 55 gilt $L(n'') \vdash_{PL} L(n')$, insbesondere $L(n'') \vdash_{PL} \neg\phi$. Nach Induktionsvoraussetzung gilt $\mathfrak{M}, n'' \models L(n'')$, also folgt $\mathfrak{M}, n'' \models \neg\phi$.
- $\Box\phi \in L(n)$. Sei nRn', d.h. wir haben $n \to_{\neg\Box} n'' \to_{PL}^* n'$. Dann gilt $\phi \in L(n'')$. Nach Lemma 55 gilt $L(n') \vdash_{PL} L(n'')$, also $L(n') \vdash_{PL} \phi$, ferner per Induktionsvoraussetzung $n' \models L(n')$, also $n' \models \phi$.

□

Aus dem Wahrheitslemma folgt sofort die Behauptung des Vollständigkeitssatzes. □

Da jede Tableauregel den Label verkleinert, ist Terminierung der Tableaumethode klar, d.h. wir haben

Korollar 57. *Erfüllbarkeit in $K/\mathcal{ALC}$ (ohne T-Box) ist entscheidbar.*

Wir erinnern nun an einige Begriffe aus der Komplexitätstheorie: Ein Problem A (also eine Teilmenge $A \subseteq \{0,1\}^*$) ist in PSPACE, wenn eine Turingmaschine M existiert, die A entschedet und nur $\mathcal{O}(p(n))$ Bandzellen besucht für ein Polynom p (wobei n wie stets die Eingabegröße bezeichnet).

Wir behaupten, dass der Tableaux-Algorithmus in PSPACE implementierbar ist. Unterstelle dazu für alle OR-Knoten die Auswahl einer propositionalen Regel. Für AND-Knoten sind alle ($\neg\Box$)-Instanzen „ausgewählt“.

Wir haben dann einen deterministischen rekursiven Algorithmus $\mathsf{SAT}(\Gamma)$, der entscheidet, ob ein Label Γ erfüllbar ist:

$$\mathsf{SAT}(\Gamma) \Leftrightarrow \mathsf{SAT}(\Gamma_1) \vee \cdots \vee \mathsf{SAT}(\Gamma_n) \text{ für alle ausgewählten Regelinstanzen } \frac{\Gamma}{\Gamma_1 \mid \ldots \mid \Gamma_n}$$

Wir analysieren den Platzverbrauch dieses Algorithmus. Wie üblich implementieren wir die Rekursion mittels eines Stacks. Da jeder rekursive Aufruf die Größe des Labels verkleinert, hat der Stack Tiefe $\mathcal{O}(n)$. Aus demselben Grund hat ein einzelner Stack Frame Größe $\mathcal{O}(n)$; insgesamt brauchen wir also Platz $\mathcal{O}(n^2)$

Achtung: Das Tableau ist insgesamt exponentiell groß, wird aber nie als Ganzes gespeichert.

Wir erinnern abschließend noch einmal an die Position von PSPACE in der Hierarchie der wichtigsten Komplexitätsklassen:

$$\text{P} \subseteq \text{NP} \subseteq \text{NPSPACE} = \text{PSPACE} \subseteq \text{EXPTIME} \neq P$$

3.6 PSPACE-Härte

Wir haben im vorigen Abschnitt gezeigt, dass das Erfüllbarkeitsproblem für K in PSPACE liegt, d.h. wir haben gezeigt, dass PSPACE eine obere Schranke der Komplexität des Problems ist. Wir zeigen nun, dass PSPACE auch eine untere Schranke ist, d.h. dass das Problem

PSPACE-hart ist. Wir verwenden Reduktion des Auswertungsproblems für quantifizierte boolesche Formeln (QBF).

Zur Erinnerung: QBF erweitern die Grammatik für einfache boolesche (d.h. aussagenlogische) Formeln um Quantifizierung über boolesche Variable X, Y, ...; die Semantik ist wie erwartet gegeben durch

$$\forall X.\phi \equiv \phi[X \mapsto \top] \wedge \phi[X \mapsto \bot]$$
$$\exists X.\phi \equiv \neg\forall X.\neg\phi \equiv \phi[X \mapsto \top] \vee \phi[X \mapsto \bot].$$

Wir haben für solche Formeln den üblichen Begriff von freien Variablen (FV). Eine *geschlossene* QBF ϕ (also eine mit $\mathsf{FV}(\phi) = \emptyset$) evaluiert unabhängig von einer vorab gegebenen Wahrheitsbelegung zu $\top$ oder $\bot$. Unser Entscheidungsproblem stellt sich also dar als

$$TQBF = \{\phi \in \text{QBF geschlossen} \mid \phi \equiv \top\}.$$

Wir haben offenbar $TQBF \in$ PSPACE: die rekursive Funktion

$$\begin{aligned}\mathsf{True}(\bot) &= \bot\\ \mathsf{True}(\neg\phi) &= \neg\mathsf{True}(\phi)\\ \mathsf{True}(\phi \wedge \psi) &= \mathsf{True}(\phi) \wedge \mathsf{True}(\psi)\\ \mathsf{True}(\forall X.\phi) &= \mathsf{True}(\phi[X \mapsto \top]) \wedge \mathsf{True}(\phi[X \mapsto \bot])\end{aligned}$$

entscheidet $TQBF$; sie benötigt einen linear tiefen Stack und linear große Stack Frames, also Platz $\mathcal{O}(n^2)$.

$TQBF$ ist aber auch PSPACE-hart. Wir zeigen dies, indem wir die Akzeptenzmenge einer in polynomiellem Platz laufenden Turingmaschine auf $TQBF$ reduzieren.

Eine Turingmaschinenkonfiguration in Platz s besteht aus folgenden Daten:

$$C = (q, p, t) \text{ mit } \begin{cases} q & \text{Zustand der Kontrolleinheit}\\ p & \text{Kopfposition } 1 \le p \le s\\ t & \text{Bandinhalt } t@i \text{ auf Positionen } i = 1 \dots s\end{cases}$$

Hierbei benötigt q Platz $\mathcal{O}(1)$, p Platz $\mathcal{O}(\log(s))$, und t Platz $\mathcal{O}(s)$

Lemma 58. *Für eine gegebene Turingmaschine M und Platzschranke s existiert eine aussagenlogische Formel ϕ^s_M mit $|\phi^s_M| \in \mathcal{O}(s^2)$, so dass für Konfigurationen C, C' in Platz s*

$$\phi^s_M(C, C') \Leftrightarrow C \text{ geht in } M \text{ in einem Schritt nach } C' \text{ über.}$$

Beweis. Sei $C = (q, p, t), C' = (q', p', t')$. Dann hat ϕ^s_M die Form

$$\phi^s_M(C, C') = \underbrace{\bigwedge_{\substack{q_0 \in Q, p_0 \in \{1,\dots,s\}\\ b_0 \in \Sigma}}}_{\mathcal{O}(s)\text{ Konjunkte}} \Big(\overbrace{(q = q_0 \wedge p = p_0 \wedge t@p = b_0)}^{\mathcal{O}(\log(s))} \rightarrow \big(\begin{array}{ll} q' = \mathsf{nextstate}(q_0, b_0) & \mathcal{O}(1)\\ \wedge\, p' = \mathsf{nextpos}(q_0, b_0, p_0) & \mathcal{O}(\log(s))\\ \wedge\, t'@p_0 = \mathsf{nextsymb}(q_0, b_0) & \mathcal{O}(1)\\ \wedge \bigwedge_{\substack{\bar{p} \neq p_0\\ \bar{p} \in \{1,\dots,s\}}} \underbrace{t'@\bar{p} = t@\bar{p}}_{\mathcal{O}(1)} \big) & \mathcal{O}(s)\end{array}$$

Die Gesamtgröße von ϕ^s_M ist also $\mathcal{O}(s^2)$. □

Satz 59. *TQBF ist PSPACE-hart.*

Beweis. Sei $M \in \mathsf{space}(p(n))$, p Polynom. Dann läuft M in Zeit $2^{q(n)}$ für ein Polynom q (q ist etwas größer als p, weil die Anzahl möglicher Konfigurationen, die M durchlaufen kann, noch die Anzahl Kontrollzustände und die Position des Schreib-/Lesekopfs beinhaltet). Wir konstruieren eine Reduktionsfunktion f von $\{x \mid M(x) = 1\}$ auf *TQBF* (d.h. $f : \{0,1\}^* \to$ QBF, $f(x) \in TQBF \Leftrightarrow M(x) = 1$). Das Grundprinzip der Reduktion ist das folgende:

$$f(x) = \phi(C_{\text{init},x}, C_{\text{acc}}) \text{ mit } C_{\text{init},x} = (q_{\text{init}}, 1, x),$$
$$C_{\text{acc}} = (q_{\text{acc}}, 1, 1)$$

(mit der letzten „1“ verstanden als Antwort „ja“), wobei

$$\phi(C, C') \Leftrightarrow M \vdash C \to^{2^{\leq s}} C' \qquad (s = q(n))$$

(mit $\to^{\leq k}$ ist gemeint „es gibt einen Übergang in höchstens k Schritten“).

Idee 1 $\phi(C, C') = \exists \underbrace{C_0, \ldots, C_{2^s}}_{\text{exponentiell viele}} . C = C_0 \wedge C' = C_{2^{p'(n)}}$
$\wedge \forall i \in \{0, \ldots, 2^s - 1\}. (\phi_M^s(C_i, C_{i+1}) \vee C_i = C_{i+1})$

Idee 2 $\phi(C, C') = \phi_s(C, C')$ mit $\phi_i(C, C') = M \vdash C \to^{\leq 2^i} C'$ rekursiv definiert:

$$\phi_0(C, C') = (C = C' \vee \phi_M^s(C, C'))$$
$$\phi_{i+1}(C, C') = \exists C''. \underbrace{\phi_i(C, C'') \wedge \phi_i(C'', C')}_{\text{Dopplung, also letztlich exponentielle Größe}}$$

Idee 3 Abkürzen per $\forall$: Äquivalent können wir $\phi_{i+1}(C, C')$ definieren als

$$\exists C''. \forall D_1, D_2. \left[\big((D_1 = C \wedge D_2 = C'') \vee (D_1 = C'' \wedge D_2 = C') \big) \to \phi_i(D_1, D_2) \right].$$

Hier wird ϕ_i nicht mehr verdoppelt, sondern nur mit einem Präfix der Länge $\mathcal{O}(s)$ versehen. Der Platzverbrauch ist somit insgesamt $|\phi_s| = \mathcal{O}(s^2)$, und die involvierten Berechnungen sind leicht, so dass die Reduktion in Polynomialzeit implementierbar ist. □

Wir erinnern an die pränexe Normalform: Eine Formel ist in *pränexer Normalform*, wenn sie die Form

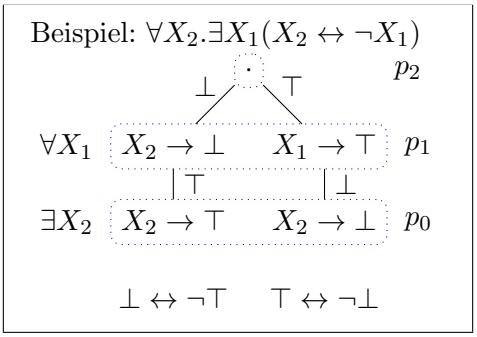
$$Q_n X_n \ldots Q_1 X_1 . \phi_0, \qquad Q_n, \ldots, Q_1 \in \{\forall, \exists\}$$

hat, mit ϕ quantorenfrei. Wie im Falle der Logik ersten Stufe zeigt man, dass zu jeder QBF eine äquivalente polymoniell große pränexe Normalform existiert. Somit bleibt TQBF auch bei Einschränkung auf pränexe Normalform PSPACE-hart .

Definition Ein *Quantorenbaum* für $\phi = Q_n X_n \ldots Q_1 X_1 . \phi_0$ wie oben ist ein 2-1-Baum uniformer Tiefe n, dessen Knoten v mit $l(v) \in \{\bot, \top\}$ beschriftet sind, so dass, wenn wir die Ebenen von der Wurzel her mit $n, \ldots, 0$ numerieren, folgende Struktur vorliegt:

- Knoten in Ebene i: $\begin{cases} \text{1-verzweigend} & \text{wenn } Q_i = \exists \\ \text{2-verzweigend} & \text{wenn } Q_i = \forall \end{cases}$

- Kinder von 2-verzweigenden Knoten haben verschiedene Label
- für Pfad $v_n \dots v_0$ gilt $[X_n \mapsto l(v_n), \dots, X_0 \mapsto l(v_0)] \models \phi_0$



(Im Bild sind zur späteren Verwendung schon propositionale Variablen zur Markierung der Ebenen angedeutet.) Äquivalenterweise werden Quantorenbäume für eine geschlossene pränexe Normalform $\phi = Q_n X_n \dots Q_1 X_1.\phi_0$ wie folgt als rekursiver Datentyp definiert:

- $n = 0$: Der Baum aus nur einem Knoten ist ein Quantorenbaum für $\phi = \phi_0$, *sofern* ϕ_0 (in diesem Fall eine rein aussagenlogische Formel ohne Boolesche Variablen, da ϕ geschlossen ist) zu $\top$ auswertet.
- $n \to n+1$: Setze $\phi' = Q_n \dots Q_1 X_1.\phi_0$. Wir unterscheiden zwei Fälle:
  - $Q_{n+1} = \forall$: Ein Quantorenbaum für ϕ besteht aus einem Wurzelknoten mit zwei Kindern, von denen eins ein Quantorenbaum für $\phi'[\top/X_n]$ und das andere ein Quantorenbaum für $\phi'[\bot/X_n]$ ist; die jeweiligen Wurzelknoten der Kinder werden zusätzlich entsprechend mit $\top$ bzw. $\bot$ gelabelt.
  - $Q_{n+1} = \exists$: Ein Quantorenbaum für ϕ besteht aus einem Wurzelknoten mit einem Kind, das ein Quantorenbaum für eine Formel der Form $\phi'[b/X_n]$ mit $b \in \{\bot, \top\}$ ist; der Wurzelknoten des Kinds wird mit b gelabelt.

Lemma 60. *Sei ϕ geschlossene QBF in pränexer NF. Dann gilt: ϕ ist wahr $\Leftrightarrow$ es existiert ein Quantorenbaum für ϕ.*

Beweis. Wir verwenden die rekursive Variante:

Induktion über n. Der Induktionsanfang $n = 0$ gilt nach der Definition von 'ϕ hat einen Quantorenbaum'. Wir führen den Schritt von n nach $n+1$ durch:

Sei $\phi' = Q_n X_n \dots Q_1 X_1 \phi_0$, $\phi = Q_{n+1} X_{n+1}.\phi'$.

Fall 1: $Q_{n+1} = \forall$. Dann ist ϕ genau dann wahr, wenn

$$\phi_\top = \phi'[X_{n+1} \mapsto \top] \text{ und}$$
$$\phi_\bot = \phi'[X_{n+1} \mapsto \bot]$$

beide wahr sind. Nach Induktionsvoraussetzung ist dies genau dann der Fall, wenn $\phi_\top$ und $\phi_\bot$ Quantorenbäume haben, was wiederum per Definition äquivalent dazu ist, dass ϕ einen Quantorenbaum hat. Fall 2 ($Q_{n+1} = \exists$) ist ähnlich. □

Nach diesen Vorbereitungen können wir nunmehr die eigentliche Reduktion von K auf *TQBF* in Angriff nehmen:

Idee für die Reduktion Wir kontrollieren die Baumstruktur durch propositionale Atome $p_n, \ldots, p_0$, die die Ebenen des Baumes darstellen, wobei wieder n die Ebene der Wurzel ist. Wir konstruieren zu geschlossenen pränexen Normalform $\phi = Q_n X_n \ldots Q_1 X_1 . \phi_0$ eine modale Formel $f(\phi)$ mit

$$\phi \text{ hat Quantorenbaum} \Leftrightarrow f(\phi) \text{ erfüllbar.} \tag{3.1}$$

Allgemeiner definieren wir für $i = 0, \ldots, n$ und die Unterformeln $\phi_i = Q_i X_i \ldots Q_1 X_1 . \phi_0$ von ϕ Formeln ψ_i mit folgender Eigenschaft: Für jede Substitution $\kappa \colon \{X_n, \ldots, X_{i+1}\} \to \{\bot, \top\}$ gilt

$$\phi_i\kappa \text{ hat Quantorenbaum} \Leftrightarrow \psi_i \wedge \underbrace{\bigwedge_{j=i+1}^{n} (X_i \leftrightarrow \kappa(X_i))}_{=:\rho_\kappa^i} \text{ erfüllbar.} \tag{3.2}$$

Zum Verständnis beachte man hierbei:

- Die Variablen $X_n, \ldots, X_{i+1}$ sind gerade die freien Variablen in ϕ_i.
- Diese Variablen haben in jedem Knoten in Ebene i eines Quantorenbaums bereits festgelegte Werte; diese Festlegung entspricht oben gerade κ.
- $\phi_i\kappa$ ist die durch Festlegung von Werten gemäß κ aus ϕ_i entstehende geschlossene Formel.
- ρ_κ^i ist eine rein aussagenlogische Formel, die gerade besagt, dass $X_{i+1}, \ldots, X_n$ die durch κ vorgegebenen Werte haben (im aktuellen Zustand).

Wir setzen dann $f(\phi) = f(\phi_n) = \psi_n$; da dann κ die leere Substitution und $\phi^\kappa \equiv \top$ ist, wird aus der obigen Invarianten (3.2) gerade unsere eigentliche Reduktionsbedingung (3.1).

Wir definieren ψ_i durch Rekursion über I. Im Basisfall $i = 0$ ist ϕ_0 bereits eine modale Formel (zufällig ohne Modalitäten), und wir setzen $f(\phi_0) = \phi_0$. Damit gilt die Invariante (3.2): Erfüllbarkeit der rechten Seite von (3.2) bedeutet in diesem Fall einfach, dass $\phi_0\kappa$ (eine aussagenlogische Formel ohne propositionale Variablen) wahr ist.

Im Schritt $i - 1 \to i$ definieren wir ψ_i als Konjunktion von

1. p_i: Wir sind an der Wurzel.
2. Falls $Q_i = \exists$: $\Diamond p_{i-1}$
   Wir kommen in die nächste Ebene (wo dann ein Wert für X_i festgelegt wird).
3. Falls $Q_i = \forall$: $\Diamond(p_{i-1} \wedge X_n) \wedge \Diamond(p_{i-1} \wedge \neg X_n)))$
   Wenn der zur Ebene i gehörige Quantor $Q_i = \forall$ ist, gibt es für beide möglichen Werte von X_i einen Nachfolger.
4. $(X_j \to \Box(p_{i-1} \to X_j)) \wedge (\neg X_j \to \Box(p_{i-1} \to \neg X_j)) \qquad (j = i+1, \ldots, n)$:
   die in Ebene i schon festgelegten Werte für Variablen $X_{i+1}, \ldots, X_n$ werden beibehalten.
5. $\Box(p_{i-1} \to \psi_{i-1})$:
   Unter jedem Knoten der nächsten Ebene hängt ein Quantorenbaum.

Wir halten zunächst fest, dass ψ_n polynomiell groß in der Größe $|\phi|$ der Eingabeformel ϕ ist: Es finden nur linear viele rekursive Aufrufe via Konjunkt 5 statt; bei jedem rekursiven Aufruf werden nicht-rekursiv die anderen Konjunkte sowie der im letzten Konjunkt außerhalb des rekursiven Aufrufs liegende Anteil $\Box(p_{i-1} \to (-))$ hinzugefügt. Dominiert wird die Größenabschätzung hierbei von den Konjunkten 4, die zusammengenommen Größe $\mathcal{O}(n \log n)$ haben (es gibt höchstens n solche Konjunkte, und die Indizes an den propositionalen Atomen brauchen jeweils Platz $\log n$). Insgesamt ergibt sich somit qeine Größe von $\mathcal{O}(n^2 \log n |\phi_0|)$, also polynomiell in $|\phi|$. Die involvierten Berechnungen sind leicht, die Reduktion lässt sich also in polynomieller Zeit (sogar in logarithmischem Platz durchführen.

Wir zeigen nun (3.2), genauer zeigen wir bei „$\Rightarrow$“ sogar, dass $\psi_i \wedge \rho^i_\kappa$ erfüllbar *in* $S4$, also in einem Zustand r in einem Rahmen mit reflexiver und transitiver Übergangsrelation ist. Wir verstärken noch die Induktionsbehauptung dahingehend, dass die Ebenenmarkierung q_i *nur* in r erfüllt ist.

„$\Rightarrow$“: Man unterscheidet Fälle über den ersten Quantor Q_i in ϕ_i. Wir führen hier nur den Fall $Q_i = \forall$ durch; der Fall $Q_i = \exists$ ist sehr ähnlich. Das Modell bekommt eine Wurzel r. Sei nun T ein Quantorenbaum für $\phi_i\kappa$. Wir setzen $\kappa_b = \kappa[X_i \mapsto b]$ für $b \in \{\bot, \top\}$. Nach der rekursiven Definition von Quantorenbäumen hat die Wurzel von T für $b \in \{\bot, \top\}$ je ein Kind, das mit b gelabelt ist und unter dem ein Quantorenbaum T_b für $\phi_{i-1}\kappa_b$ hängt. Damit existieren nach Voraussetzung S4-Modelle $\mathfrak{M}_b$, in denen jeweils an einem Zustand r_b die Formel $\psi_{i-1} \wedge \rho^{i-1}_{\kappa_b}$ erfüllt ist, und in denen jeweils q_{i-1} nur in r_b gilt. Das zu konstruierende Modell $\mathfrak{M}$ besteht dann aus r, $\mathfrak{M}_\bot$ und $\mathfrak{M}_\top$, ergänzt wie folgt:

- An r erfüllte propositionale Atome sind genau q_i und diejenigen X_j mit $\kappa(X_j) = \top$, für $j = i+1, \dots, n$. Damit gilt wieder die Verstärkung der Invariante bezüglich q_i, und r erfüllt ρ^i_κ.
- Nachfolger von r werden r selbst und alle Nachfolger von $r_\bot$ und $r_\top$ (inklusive $r_\bot$ und $r_\top$ selbst); somit ist die Übergangsrelation wieder reflexiv und transitiv.

Damit erfüllt r per Konstruktion Konjunkte 1 und 5, und da r_b jeweils $\rho^{i-1}_{\kappa_b}$ erfüllt, erfüllt r auch Konjunkte 3 und 4. Damit erfüllt r wie verlangt auch ψ_i.

„$\Leftarrow$“: Sei $\mathfrak{M} = ((X, R), V)$ K-Modell mit $\mathfrak{M}, r \models \phi_i \wedge \rho^i_\kappa$. Man unterscheidet wieder Fälle über den ersten Quantor Q_i von ϕ_i; wir führen hier (zur Abwechslung) nur den Fall für $Q_i = \exists$ durch. Nach 2 hat r einen Nachfolger r_b, so dass $r_b \models q_{i-1}$, wobei wir mit b den Wahrheitswert von X_i bei r_b bezeichnen. Nach 5 gilt $r_b \models \psi_{i-1}$, nach 4 gilt $r_b \models r^i_\kappa$, und damit nach Wahl von b auch $r_b \models r^{i-1}_{\kappa_b}$, mit κ_b wie oben. Nach Induktionsvoraussetzung hat also $\phi_{i-1}\kappa_b$ einen Quantorenbaum T_b; damit hat auch $\phi_i\kappa$ einen Quantorenbaum, der aus einem neuen Wurzelknoten mit einem mit b gelabelten Kind besteht, unter dem T_b als Teilbaum hängt.

Insgesamt haben wir gezeigt:

Satz 61. *Erfüllbarkeit in $\mathcal{L}$ ist PSPACE-hart für $K \subseteq \mathcal{L} \subseteq S4$.*

Korollar 62. *Das Erfüllbarkeitsproblem in K ist PSPACE-vollständig.*

3.7 Leichtgewichtige Beschreibungslogik: $\mathcal{EL}$

Für manche Zwecke, insbesondere in Verbindung mit sehr großen Ontologien, sind Logiken nützlich, die tatsächlich effizient, d.h. in Polynomialzeit, lösbare Schlussfolgerungsprobleme

besitzen. Ein Beispiel einer solchen Logik ist $\mathcal{EL}$, das aus $\mathcal{ALC}$ im wesentlichen durch Verbieten von Disjunktion und Box entsteht, d.h. in modaler Notation durch die Grammatik

$$\phi ::= \top \mid p \mid \phi \wedge \psi \mid \Diamond\phi$$

gegeben ist.

Man sieht leicht, dass jede $\mathcal{EL}$-Formel erfüllbar ist. Als Schlussfolgerungsproblem betrachten wir daher stattdessen Subsumption („gilt $\phi \sqsubseteq \psi$?“).

Für Zwecke der theoretischen Diskussion erweitern wir die $\mathcal{EL}$-Syntax wieder um Disjunktion und erhalten *positive Formeln*, definiert durch die Grammatik

$$\phi ::= \underbrace{\ldots}_{\text{wie } \mathcal{EL}} \mid \bot \mid \phi \vee \psi$$

Wir halten zwei entscheidende Eigenschaften positiver Formeln fest:

Lemma 63. *Positive Formeln sind* monoton*, d.h. wenn* $((X, R), V), x \models \phi$ *und* $V(p) \subseteq V'(p)$ *für alle* p*, dann auch* $((X, R), V'), x \models \phi$*.*

Wir erinnern daran, dass $x \preceq y$ bedeutet, dass y x simuliert, d.h. eine Simulation S mit xSy existiert.

Lemma 64 (Simulationsstabilität)**.** *Positive Formeln* ϕ *sind* simulationsstabil*, d.h. wenn* $x \preceq y$ *und* $x \models \phi$*, dann* $y \models \phi$*.*

Beweis. Sei S Simulation mit xSy. Induktion über ϕ:

- p: Wenn $x \models p$, dann $y \models p$, da S eine Simulation ist.
- $\bot, \top$: trivial
- $\phi \vee \psi$: Wenn $x \models \phi \vee \psi$, dann o.E. $x \models \phi$, per Induktionsvoraussetzung also $y \models \phi$ und somit $y \models \phi \vee \psi$.
- $\phi \wedge \psi$: analog
- $\Diamond\phi$: Sei $x \models \Diamond\phi$. Dann existiert x' mit $x \to x'$, $x' \models \phi$. Wir können also das übliche Viereck vervollständigen:

$$\begin{array}{ccc} x & S & y \\ \downarrow & \exists & \downarrow \\ x' & S & y' \end{array}$$

  d.h. es existiert y' mit $y \to y'$ und $x'Sy'$. Nach Induktionsvoraussetzung folgt $y' \models \phi$ und damit $y \models \Diamond\phi$.

□

Die entscheidende Eigenschaft von $\mathcal{EL}$ ist nunmehr die Tatsache, dass sich jede $\mathcal{EL}$-Formel durch ein zu simulierende Modell ersetzen lässt:

Definition 65. Ein punktiertes Modell $\mathfrak{M}, x$ heißt *Materialisator* von ϕ, wenn für alle $\mathfrak{N}, y$ gilt:

$$\mathfrak{N}, y \models \phi \text{ genau dann, wenn } x \preceq y.$$

Lemma 66. *Sei $\mathfrak{M}, x$ Materialisator von ϕ. Dann gilt:*

1. $\mathfrak{M}, x \models \phi$.
2. *Für jede positive Formel ψ gilt $\phi \sqsubseteq \psi$ genau dann, wenn $\mathfrak{M}, x \models \psi$.*

Beweis. 1. Folgt unmittelbar aus der Definition, da $x \preceq x$.

2. „⇒“: Unmittelbar aus Teil 1.

   „⇐“: Sei $\mathfrak{N}, y \models \phi$, also nach Definition $x \preceq y$, also folgt aus $\mathfrak{M}, x \models \psi$ per Simulationsstabilität $\mathfrak{N}, y \models \psi$. □

Teil 2. des Lemmas liefert einen Algorithmus für $\phi \sqsubseteq \psi$, der in polynomieller Zeit in der Größe von $\mathfrak{M}$ läuft; explizit: Um zu entscheiden, ob $\phi \sqsubseteq \psi$ (für ϕ in $\mathcal{EL}$ und ψ positiv) gültig ist, bilde man den Materialisator $\mathfrak{M}, x$ von ϕ (gemäß dem aus dem Beweis von Satz 68 hervorgehenden Verfahren) und prüfe dann, ob $\mathfrak{M}, x \models \psi$. Um zu sehen, dass dies wirklich in Polynomialzeit funktioniert, braucht man zweierlei: Erstens darf $\mathfrak{M}$ nur polynomiell groß sein, was Satz 68 unten denn auch in der Tat garantiert, und zweitens muss das Modellprüfungsproblem (gilt $\mathfrak{M}, x \models \psi$?) in Polynomialzeit entscheidbar sein. Das ist es auch (wie?).

Die Definition von Materialisatoren lässt sich zunächst noch etwas vereinfachen:

Lemma 67. *Sei ϕ eine positive Formel und $\mathfrak{M}, x$ ein punktiertes Modell, so dass Folgendes gilt:*

1. $\mathfrak{M}, x \models \phi$
2. *Für all $\mathfrak{N}, y$ mit $\mathfrak{N}, y \models \phi$ gilt $x \preceq y$.*

Dann ist $\mathfrak{M}, x$ ein Materialisator von ϕ.

(Die Bedingungen des Lemmas sind offenbar auch notwendig.)

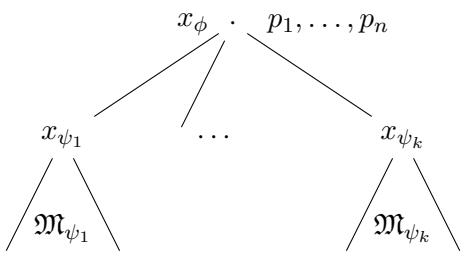
Beweis. Zu zeigen ist nur noch, dass $\mathfrak{N}, y \models \phi$, sobald $x \preceq y$; das folgt aber aus der Simulationsstabilität von ϕ. □

Satz 68. *Jede $\mathcal{EL}$-Formel hat einen Materialisator polynomieller (in der Tat linearer) Größe.*

Beweis. Wir definieren zu ϕ einen Materialisator $\mathfrak{M}_\phi, x_\phi$ per Rekursion über ϕ. Nach geeigneter Umformung hat ϕ die Form

$$\phi \equiv p_1 \wedge \ldots \wedge p_n \wedge \Diamond\psi_1 \wedge \ldots \wedge \Diamond\psi_k, \qquad (k, n \geq 0).$$

Wir konstruieren $\mathfrak{M}_\phi, x_\phi$ als



Nach Lemma 67 ist zu zeigen:

1. $\mathfrak{M}_\phi, x_\phi \models \phi$
2. Für alle $\mathfrak{N}, y$ mit $\mathfrak{N}, y \models \phi$ gilt $x \preceq y$.

Dies sieht man wie folgt:

1.: Induktion über ϕ.

2.: Sei $\mathfrak{N}$ ein Kripkemodell. Wir definieren $S \subseteq M \times N$ durch

$$S = \{(x_\psi, y) \mid x_\psi \in M, y \models \phi\}$$

und zeigen, dass S eine Simulation ist. Wir verwenden dazu der Einfachheit halber die obige Notation weiter (ϕ, p_i, ψ_i etc.); alle weiteren Zustände sind ja per Rekursion analog strukturiert. Sei also $(x_\phi, y) \in S$. Sei $x_\phi \models q$ für ein propositionales Atom q. Dann ist q eines der p_i; da $y \models \phi$, gilt insbesondere $y \models p_i$, wie verlangt. Zu zeigen bleibt die Forth-Bedingung für eine gegebene Transition $x_\phi \to x_{\psi_i}$. Da $y \models \phi$, gilt insbesondere $y \models \Diamond\psi_i$, d.h. es existiert y' mit $y' \models \psi_i$. Damit gilt per Definition $x_{\psi_1} S y'$, wie verlangt:

$$\begin{array}{ccc} x_\phi & S & y \\ \downarrow & & \downarrow \\ x_{\psi_i} & S & y'. \end{array}$$

□

3.8 TBoxen in $\mathcal{EL}$

Wir beschränken uns für Zwecke von $\mathcal{EL}$ auf klassische TBoxen (zur Erinnerung: dies sind TBoxen der Form $A_1 = C_1, A_2 = C_2, \ldots$, wobei die A_i paarweise verschiedene atomare Konzepte sind, aber beliebig in den C_i vorkommen dürfen). [Man kann in $\mathcal{EL}$ durchaus auch, unter Beibehaltung der effizienten Entscheidbarkeit, allgemeine TBoxen zulassen, das machen wir hier nur nicht.] Unsere bisherige Semantik für solche TBoxen war *deskriptiv*, d.h. wir lassen alle Interpretation der A_i zu, so lange sie nur die Gleichungen in der TBox lösen. Für Zwecke von $\mathcal{EL}$ führen wir nun eine restriktivere Semantik ein, in der die Definitionen in der TBox mittels *größter Fixpunkte* interpretiert werden; diese Semantik bezeichnen wir abkürzend als die *gfp-Semantik* (gfp=*greatest fixpoint*). Grob gesprochen unterstellen wir also die größtmögliche Lösung der durch die TBox gegebenen Gleichungen.

Beispiel 69. In gfp-Semantik definiert die TBox

$$\mathsf{Lion} = \exists\, \mathsf{hasParent}.\, \mathsf{Lion}$$

das Konzept Lion als die Menge aller Individuen, von denen aus eine unendliche hasParent-Kette existiert (also ein unendlicher Pfad $\bullet \to \bullet \to \bullet \to \ldots$, wobei $\to$ die Interpretation von hasParent bezeichnet).

Es ergeben sich einige vielleicht etwas unerwartete Effekte; z.B. definiert unter gfp-Semantik das weitere Axiom

$$\mathsf{Tiger} = \exists\, \mathsf{hasParent}.\, \mathsf{Tiger}$$

das Konzept Tiger als dieselbe Menge wie oben Lion (während die beiden Konzepte unter deskriptiver Semantik verschiedene Extensionen haben können). Das ist aber letztlich nicht weiter verwunderlich; es wird in den beiden Axiomen eben einfach nicht genug über Tiger bzw. Löwen gesagt.

3.9 Fixpunkte

Unter einem *Fixpunkt* einer Funktion F versteht man allgemein eine Lösung x der Gleichung $F(x) = x$; diese muss (wie schon die obige Diskussion zeigt) keineswegs immer eindeutig bestimmt sein. In Gegenwart geeigneter Ordnungsstrukturen kann man aber eben in der Menge der Fixpunkte von F bestimmte Lösungen auszeichnen, etwa den kleinsten oder den größten. Im folgenden behandeln wir Bedingungen an F und die unterliegende Ordnung, unter denen solche größten bzw. kleinsten Fixpunkte existieren.

Recall: Ordnungstheorie

- $(X, \leq)$ *Ordnung* $\Leftrightarrow \leq$ ist reflexiv, transitiv und antisymmetrisch.
- x obere Schranke von $A \subseteq X$ (geschrieben $x \geq A$) $\Leftrightarrow \forall y \in A.\, x \geq y$.
- x *Supremum* von A (geschrieben $x = \bigvee A$) $\Leftrightarrow x$ kleinste obere Schranke von A.
- $(X, \leq)$ ist ein *vollständiger Verband* $\Leftrightarrow$ jedes $A \subseteq X$ hat ein Supremum.

Fakten

- In einem vollständigen Verband hat jedes $A \subseteq X$ auch ein Infimum $\bigwedge A$, nämlich $\bigwedge A = \bigvee\{x \mid x \leq A\}$.
- Jeder vollständige Verband X hat ein kleinstes Element $\bot = \bigvee \emptyset = \bigwedge X$ und ein größtes Element $\top = \bigwedge \emptyset = \bigvee X$.

Beispiel 70.
- $(\mathcal{P}(Y), \subseteq)$ ist vollständiger Verband: Für $\mathfrak{A} \subseteq \mathcal{P}(Y)$ haben wir $\bigvee \mathfrak{A} = \bigcup \mathfrak{A}$ (und $\bigwedge \mathfrak{A} = \bigcap \mathfrak{A}$).
- Wenn X und Y vollständige Verbände sind, dann, wie wir sehen werden, auch $X \times Y$ mit der komponentenweisen Ordnung $(x, y) \leq (x', y') \Leftrightarrow x \leq x' \wedge y \leq y'$.

Eine äquivalente Formulierung der Vollständigkeit ist die folgende: $(X, \subseteq)$ ist genau dann ein vollständiger Verband, wenn jede Familie $(x_i)_{i \in I} \in X^I$ ein Supremum $\bigvee_{i \in I} x_i$ hat. Die Äquivalenz sieht man wie folgt:
„$\Rightarrow$“: $\bigvee_{i \in I} x_i = \bigvee\{x_i \mid i \in I\}$.
„$\Leftarrow$“: $\bigvee A = \bigvee_{a \in A} a$.

Damit erhält man unmittelbar die Vollständigkeit von $X \times Y$ mit der komponentenweisen Ordnung, indem man nämlich $\bigvee_{i \in I}(x_i, y_i) = (\bigvee_{i \in I} x_i, \bigvee_{x \in I} y_i)$ hat. Zurückübersetzt in Begriffe von Suprema von Mengen bedeutet dies für $A \subseteq X \times Y$

$$\begin{aligned}\bigvee A &= \bigvee_{(x,y) \in A}(x, y) \\ &= \big(\bigvee_{(x,y) \in A} x, \bigvee_{(x,y) \in A} y\big) \\ &= \big(\bigvee\{x \mid \exists y.(x, y) \in A\}, \bigvee\{y \mid \exists x.(x, y) \in A\}\big) \\ &. = \big(\bigvee \pi_1[A], \bigvee \pi_2[A]\big),\end{aligned}$$

wobei $\pi_1 : X \times Y \to X$ und $\pi_2 : X \times Y \to Y$ die beiden Projektionen sind ($\pi_1(x, y) = x$, $\pi_2(x, y) = y$).

In Ordnungen kann man den Begriff des Fixpunkts noch sinnvoll abschwächen:

Definition 71. Sei $(X, \leq)$ eine Ordnung und $f : X \to X$. Ein Punkt $x \in X$ heißt *Präfixpunkt* von f, wenn $f(x) \leq x$, und *Postfixpunkt* von f, wenn $x \leq f(x)$.

(Damit ist natürlich ein Fixpunkt gerade ein Präfixpunkt, der gleichzeitig ein Postfixpunkt ist.)

Fixpunktsätze sind Aussagen über die Existenz von Fixpunkten unter bestimmten Bedingungen sowohl an die Struktur der unterliegenden Menge als auch an die Funktion, wie im folgenden:

Definition 72. Sei $(X, \leq)$ eine Ordnung. Eine Funktion $f : X \to X$ ist *monoton*, wenn aus $x \leq y$ stets $f(x) \leq f(y)$ folgt.

Satz 73 (Knaster/Tarski)**.** *Sei $(X, \leq)$ ein vollständiger Verband und $f : X \to X$ monoton. Dann hat f einen kleinsten (Prä-)Fixpunkt*

$$\mu f = \bigwedge \{x \mid f(x) \leq x\}.$$

Weil mit $(X, \leq)$ wie oben beobachtet auch $(X, \geq)$ ein vollständiger Verband ist (d.h. weil in vollständigen Verbänden auch Infima aller Teilmengen existieren), folgt per *Dualisierung* sofort: f hat auch einen größten (Post-)Fixpunkt

$$\nu f = \bigvee \{x \mid x \leq f(x)\}.$$

Beweis. Nach Konstruktion ist μf wie oben definiert kleiner oder gleich jedem Präfixpunkt von f, also auch kleiner oder gleich jedem Fixpunkt. Zu zeigen bleibt:

1. μf ist Präfixpunkt, d.h. $f(\mu f) \leq \mu f$: Sei x Präfixpunkt von f; nach Definition von μf reicht es, $f(\mu f) \leq x$ zu zeigen. Nun gilt nach Definition von μf

   $$\mu f \leq x,$$

   per Monotonie von f und der Präfixpunkteigenschaft von x also

   $$f(\mu f) \leq f(x) \leq x.$$

2. μf ist Fixpunkt, d.h. $\mu f \leq f(\mu f)$: Dazu reicht nach Konstruktion von μf, dass $f(\mu f)$ ein Präfixpunkt ist. Wir haben laut 1.

   $$f(\mu f) \leq \mu f,$$

   und damit per Monotonie von f

   $$f(f(\mu f)) \leq f(\mu f).$$

□

Zur Berechnung von Fixpunkten auf endlichen Ordnungen $(X, \leq)$ verwenden wir eine vereinfachte Version des *Kleeneschen Fixpunktsatzes*. Wir beobachten, dass wir für monotones f eine aufsteigende Kette

$$\bot \leq f(\bot) \leq f^2(\bot) \ldots$$

haben; per Endlichkeit von X muss diese Kette nach endlich vielen, nämlich höchstens $n = |X|$, Schritten *stabilisieren*, d.h. man hat $f^{n+1}(\bot) = f^n(\bot)$, und dann natürlich $f^m(\bot) = f^n(\bot)$ für alle $m \geq n$.

Satz 74 (Vereinfachter Kleenescher Fixpunktsatz)**.** *Sei $(X, \leq)$ eine endliche Ordnung mit kleinstem Element $\bot$ und $f : X \to X$ monoton. Dann hat f einen kleinsten (Prä-)fixpunkt μf, nämlich $\mu f = \bigvee f^i(\bot)$; in der Tat existiert n mit $f^{n+1}(\bot) = f^n(\bot)$, und dann $\mu f = \bigvee f^i(\bot) = f^n(\bot)$.*

Beweis. Nach Voraussetzung ist μf ein Fixpunkt. Es bleibt zu zeigen, dass $\mu f \leq x$ für einen gegebenen Präfixpunkt x von f. Wir zeigen per Induktion über i, dass

$$f^i(\bot) \leq x$$

für alle i:

$$\begin{aligned} f^0(\bot) &= \bot \\ f^{i+1}(\bot) &\overset{f \text{ monoton, IV}}{\leq} f(x) \leq x \end{aligned}$$

□

Wiederum gilt auch die duale Aussage, d.h. für X endlich und f monoton ist

$$\nu f = \bigwedge f^i(\top)$$

größter (Post-)Fixpunkt von f, und $\nu f = f^n(\top)$ spätestens für $n \geq |X|$. Dies liefert also eine Berechnungsmethode für μf bzw. νf, die nach spätestens $|X|$ Schritten terminiert.

3.9.1 $\mathcal{EL}$ und klassische TBoxen in gfp-Semantik

Wir führen nunmehr einen expliziten Begriff von *Variablen* ein: Wir haben eine Menge Var von Variablen, die wir in Formeln wie propositionale Atome verwenden können und die in Modellen $\mathfrak{M} = (X, R, V)$ durch Valuationen $\eta \colon \mathsf{Var} \to \mathcal{P}(X)$ interpretiert werden; d.h. wir haben nunmehr η zusätzlich zu $\mathfrak{M}$ und einem Zustand $x \in X$ auf der linken Seite der Erfülltheitsrelation, die dann im wesentlichen wie bisher definiert wird, mit einer zusätzlichen Klausel

$$\mathfrak{M}, \eta, x \models X \quad \iff \quad x \in \eta(X).$$

Wir schreiben $[\![\phi]\!]_\eta = \{x \in X \mid \mathfrak{M}, \eta, x \models \phi\}$. Wir verwenden dann Variablen als die linken Seiten von Konzeptdefinitionen in klassischen TBoxen; wir begreifen eine solche TBox als eine Folge gegenseitig rekursiver Definitionen

$$\begin{aligned} X_1 &= \phi_1 \\ &\vdots \\ X_n &= \phi_n. \end{aligned}$$

Wir versehen eine solche Definition nunmehr wie angekündigt mit ihrer gfp-Semantik: Eine Valuation der X_i ist im wesentlichen ein Element von $\mathcal{P}(X)^n$. Gegeben $\mathfrak{M}$ haben wir somit eine monotone Abbildung

$$\begin{aligned} F : \mathcal{P}(X)^n &\to \mathcal{P}(X)^n \\ (A_1, \ldots, A_n) &\mapsto ([\![\phi_i]\!]_{[X_1 \mapsto A_1, \ldots, X_n \mapsto A_n]})_{i=1,\ldots,n} \end{aligned}$$

und setzen dann

$$([\![X_1]\!]_{\mathfrak{M}}, \ldots, [\![X_n]\!]_{\mathfrak{M}}) := \nu F.$$

Wir dehnen diese Definition mit den üblichen Klauseln auf alle Formeln aus, d.h. $[\![\phi]\!]_{\mathfrak{M}}$ bezeichnet die Extension von ϕ in gfp-Semantik (und braucht daher keine Valuation der X_i).

Beispiel 75. Für die TBox

$$X_1 = p_1 \wedge \Diamond X_2$$
$$X_2 = p_2 \wedge \Diamond X_1$$

bekommen wir z.B. als Semantik von X_1 alle Zustände, die Anfangspunkt einer unendlichen Kette der Form

$$\overset{p_1}{\cdot} \longrightarrow \overset{p_2}{\cdot} \longrightarrow \overset{p_1}{\cdot} \longrightarrow \overset{p_2}{\cdot} \longrightarrow \cdots$$

sind (wobei die Wiederholung von Zuständen erlaubt ist).

Achtung: Die Bedingung $xSy \wedge x \models p \implies y \models p$ in der Definition von Simulationen S bezieht sich weiterhin nur auf propositionale Atome, nicht auf Variablen (sonst wäre insbesondere das folgende Lemma trivial).

Lemma 76 (Simulationsstabilität mit gfp)**.** *Sei $\mathcal{T} = \{X_i = \phi_i \mid i = 1, \ldots, n\}$ eine klassische $\mathcal{EL}$-TBox mit gfp-Semantik, und sei S eine Simulation zwischen Modellen $\mathfrak{M}$ und $\mathfrak{N}$. Dann folgt aus wSv und $\mathfrak{M}, w \models X_i$ stets $v \models X_i$.*

Beweis. Wir schreiben

$$S[A] = \{y \mid \exists x \in A, xSy\}$$

für $A \subseteq X$, und für Valuationen η

$$S[\eta](X_i) = S[\eta(X_i)].$$

Man zeigt nun per Induktion über ϕ, dass

$$S\,[[\![\phi]\!]_\eta] \subseteq [\![\phi]\!]_{S[\eta]}. \tag{3.3}$$

Hierbei sind alle Fälle wie in Lemma 64, bis auf den neuen Fall $\phi = X_i$:

$$x \in \eta(X_i), xSy \Rightarrow y \in S[\eta(X_i)] = S[\eta](X_i) \Rightarrow y \in [\![X_i]\!]_{S[\eta]}$$

In unserer neuen Schreibweise ist die Behauptung des Lemmas

$$S\,[[\![X_i]\!]_{\mathfrak{M}}] \subseteq [\![X_i]\!]_{\mathfrak{N}}. \tag{3.4}$$

Wir beweisen dies per *Koinduktion:* Weil die $[\![X_i]\!]_{\mathfrak{N}}$ einen größten Postfixpunkt bilden, reicht es zu zeigen, dass

$$(S[[\![X_i]\!]_{\mathfrak{M}}])_{i=1,\ldots,n}$$

ein Post-Fixpunkt der wie oben durch

$$F(A_1, \ldots, A_n) = ([\![\phi_i]\!]_{[X_1 \mapsto A_1, \ldots, X_n \mapsto A_n]})_{i=1,\ldots,n}.$$

definierten Funktion F ist. Zu zeigen ist also

$$S[\underbrace{[\![X_i]\!]_{\mathfrak{M}}}_{=[\![\phi_i]\!]_{[X_i \mapsto [\![X_i]\!]_{\mathfrak{M}}]_{i=1,\ldots,n}}}] \subseteq [\![\phi_i]\!]_{[X_i \mapsto S[[\![X_i]\!]_{\mathfrak{M}}]]_{i=1,\ldots,n}}.$$

Das folgt aber gerade aus (3.3). □

Normalisierung In $X_i = \phi_i$ hat ϕ_i ohne Einschränkung die Form

$$\phi_i = p_1 \wedge \ldots \wedge p_k \wedge \Diamond X_{i_1} \wedge \ldots \wedge \Diamond X_{i_m}.$$

Dies erreichen wir durch relativ offensichtliche Umformungen, z.B. Sortieren von Konjunktionen sowie durch Einführen neuer Variablen für komplexe Formeln unter Modaloperatoren (ersetze z.B. $X = \Diamond\Diamond X$ durch $X = \Diamond Y$, $Y = \Diamond X$). Der einzige nicht völlig offensichtliche Teil ist die Beseitigung von Vorkommen der X_i, die nicht unter Modaloperatoren liegen. Wir führen dies nur am Beispiel vor: in der TBox

$$X_1 = X_2 \wedge \Diamond\phi_1$$
$$X_2 = X_1 \wedge \Diamond\phi_2$$

folgt $X_1 = X_2$; wir können also überall X_2 durch X_1 ersetzen und erhalten dann durch Zusammenlegen der beiden Gleichungen für X_1

$$X_1 = X_1 \wedge \Diamond\hat{\phi}_2 \wedge \Diamond\hat{\phi}_1,$$

wobei $\hat{\phi}_i = \phi_i[X_1/X_2]$. Dies ist äquivalent zu

$$X_1 \sqsubseteq \Diamond\hat{\phi}_1 \wedge \Diamond\hat{\phi}_2,$$

und unter gfp-Semantik, weil größte Fixpunkte eben dasselbe sind wie größte Postfixpunkte, zu

$$X_1 = \Diamond\hat{\phi}_1 \wedge \Diamond\hat{\phi}_2.$$

Dies illustriert den Fall zyklischer Abhängigkeit; nicht-zyklische Anhängigkeiten können wir einfach expandieren, z.B. ersetzen wir in

$$X_1 = X_2 \wedge \Diamond\phi$$
$$X_2 = p \wedge \Diamond\psi$$

die Definition von X_1 durch

$$X_1 = p \wedge \Diamond\psi \wedge \Diamond\phi.$$

(Anders als sonst oft beim Expandieren von Definitionen müssen wir hier *keinen* exponentiellen Blowup befürchten; warum nicht?)

Materialisator Wir konstruieren nun den Materialisator $\mathfrak{M}_\mathcal{T}$ für $\mathcal{T}$:

$$X_\mathcal{T} = \{X_1, \ldots, X_n\}$$
$$V_\mathcal{T}(p) = \{X_i \mid p \text{ Konjunkt von } \phi_i\}$$
$$X_i R_\mathcal{T} X_j \Leftrightarrow \Diamond X_j \text{ Konjunkt von } \phi_i$$

Wie bereits im TBox-freien Fall haben wir zwei Eigenschaften zu überprüfen:

Lemma 77. $X_i \in [\![X_i]\!]_{\mathfrak{M}_\mathcal{T}}$

Beweis. Per Koinduktion: Es reicht zu zeigen, dass $(\{X_1\}, \ldots, \{X_n\})$ ein Post-Fixpunkt der entsprechenden Funktion ist, d.h. dass

$$X_i \in [\![\phi_i]\!]_{[X_i \mapsto \{X_i\}]_{i=1,\ldots,n}};$$

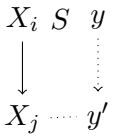
das gilt aber nach Konstruktion von $\mathfrak{M}_\mathcal{T}$. □

Lemma 78. *Wenn $\mathfrak{N}$ eine Kripke-Modell ist und $y \in [\![X_i]\!]_{\mathfrak{N}}$, dann $X_i \preceq y$ (für den Zustand X_i in $\mathfrak{M}_{\mathcal{T}}$).*

Beweis. Wir zeigen, dass

$$S = \{(X_i, y) \mid y \in [\![X_i]\!]_{\mathfrak{N}}\}$$

ein Simulation ist. Bewahrung propositionaler Atome ist klar. Sei



Wir müssen zeigen, dass y' wie im Diagramm existiert. Weil $X_i R_{\mathcal{T}} X_j$, ist $\Diamond X_j$ Konjunkt von ϕ_i; somit gilt $y \in [\![\Diamond X_j]\!]_{\mathfrak{N}}$, d.h. es existiert y' mit $y \to y'$ und $y' \models X_j$, also $X_j S y'$. □

Damit erhalten wir wieder einen Polynomialzeitalgorithmus für Subsumption, nun mit klassischen TBoxen unter gfp-Semantik: Durch Einführen abkürzender Definitionen lässt sich Subsumption reduzieren auf den Fall, dass beide Seiten definierte Variablen sind; für den Fall haben wir

$$\mathcal{T} \models X_i \sqsubseteq X_j \Leftrightarrow X_i \in [\![X_j]\!]_{\mathfrak{M}_{\mathcal{T}}}.$$

Die letztere Bedingung lässt sich per Kleeneschem Fixpunktsatz in Polynomialzeit überprüfen (warum genau?).