# 2 Hennessy-Milner-Logik

## 2.1 Syntax und Semantik

*Hennessy-Milner-Logik* (HML), auch als *multi-modale Logik K* bekannt, erweitert die Aussagenlogik um zwei Konstrukte ("*diamond*" und "*box*"), mit denen man über Nachfolger eines Zustandes in einem Transitionssystem quantifiziert.

**Definition 2.1**
Seien $\mathcal{P}$ und $\Sigma$ Mengen von Propositionen bzw. Aktionennamen. Die Syntax von HML ist gegeben durch die folgende Grammatik.

$$\varphi \ ::= \ q \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \neg\varphi \mid \langle a \rangle \varphi \mid [a]\varphi$$

wobei $q \in \mathcal{P}$, $a \in \Sigma$. Für $|\Sigma| = 1$ heißt diese Logik auch *mono-modale Logik* oder einfach nur *Modallogik*. In diesem Fall schreiben wir nur $\Diamond\varphi$ und $\Box\varphi$. Als weitere Abkürzungen definieren wir $\mathtt{tt} := q \vee \neg q$ für ein $q \in \mathcal{P}$, $\mathtt{ff} := \neg\mathtt{tt}$, $\varphi \rightarrow \psi := \neg\varphi \vee \psi$.

Wir definieren die Semantik von HML über einem Transitionssystem $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \Sigma\}, \lambda, s_0)$ mit $\lambda : \mathcal{S} \rightarrow 2^{\mathcal{P}}$ wie folgt. Sei $s \in \mathcal{S}$.

$$\begin{array}{rll}
\mathcal{T}, s \models q & \text{gdw.} & q \in \lambda(s) \\
\mathcal{T}, s \models \varphi \vee \psi & \text{gdw.} & \mathcal{T}, s \models \varphi \text{ oder } \mathcal{T}, s \models \psi \\
\mathcal{T}, s \models \varphi \wedge \psi & \text{gdw.} & \mathcal{T}, s \models \varphi \text{ und } \mathcal{T}, s \models \psi \\
\mathcal{T}, s \models \neg\varphi & \text{gdw.} & \mathcal{T}, s \not\models \varphi \\
\mathcal{T}, s \models \langle a \rangle \varphi & \text{gdw.} & \exists t \in \mathcal{S}, s \xrightarrow{a} t \text{ und } \mathcal{T}, t \models \varphi \\
\mathcal{T}, s \models [a]\varphi & \text{gdw.} & \forall t \in \mathcal{S} : s \xrightarrow{a} t \text{ impliziert } \mathcal{T}, t \models \varphi
\end{array}$$

Ist klar, auf welches Transitionssystem Bezug genommen wird, dann schreiben wir auch einfach nur $s \models \varphi$.

**Definition 2.2**
Die Menge aller *Unterformeln* einer HML-Formel $\varphi$ ist $Sub(\varphi)$, definiert als

$$\begin{array}{rcl}
Sub(q) & := & \{q\} \\
Sub(\varphi \vee \psi) & := & \{\varphi \vee \psi\} \cup Sub(\varphi) \cup Sub(\psi) \\
Sub(\varphi \wedge \psi) & := & \{\varphi \wedge \psi\} \cup Sub(\varphi) \cup Sub(\psi) \\
Sub(\neg\varphi) & := & \{\neg\varphi\} \cup Sub(\varphi) \\
Sub(\langle a \rangle \varphi) & := & \{\langle a \rangle \varphi\} \cup Sub(\varphi) \\
Sub([a]\varphi) & := & \{[a]\varphi\} \cup Sub(\varphi)
\end{array}$$

Wir definieren außerdem die Größe einer Formel $\varphi$ durch die Anzahl ihrer Unterformeln: $|\varphi| := |Sub(\varphi)|$.

**Definition 2.3**
Eine HML-Formel $\varphi$ ist in *positiver Normalform*, wenn das Negationssymbol nur noch direkt vor Propositionen vorkommt, also wenn für alle $\psi \in Sub(\varphi)$ gilt: $\psi = \neg\psi'$ für ein $\psi'$ impliziert $\psi' \in \mathcal{P}$.

Sei $\text{HML}^+ := \{\varphi \in \text{HML} \mid \varphi \text{ ist in positiver Normalform } \}$.

Definiere außerdem $\text{HML}^-$ als die Menge aller HML-Formeln, die nur aus Propositionen, Disjunktionen, Negationen und dem Diamond-Operator bestehen.

**Lemma 2.1**
$\text{HML}^+ \equiv \text{HML} \equiv \text{HML}^-$

**Beweis** Übung. ∎

Es gilt sogar nicht nur, dass zu jeder Formel $\varphi$ aus HML es Formeln $\varphi^+ \in \text{HML}^+$ und $\varphi^- \in \text{HML}^-$ gibt, so dass $\varphi \equiv \varphi^+ \equiv \varphi^-$, sondern darüberhinaus auch noch $|\varphi^-| = O(|\varphi|)$ und $|\varphi^+| = O(|\varphi|)$.

## 2.2 Bisimulationsinvarianz

**Satz 2.1**
Für alle Transitionssysteme $\mathcal{T}$ mit Zuständen $s, t \in \mathcal{S}$ gilt: Falls $s \sim t$ dann gilt für alle $\varphi \in \text{HML}$: $s \models \varphi$ gdw. $t \models \varphi$.

**Beweis** Per Induktion über den Formelaufbau. Wegen Lemma 2.1 beschränken wir uns auf Propositionen und die Konstrukte $\vee$, $\neg$ und $\langle a \rangle$.

Angenommen, $s \sim t$. Also existiert eine Bisimulation $B$ mit $(s, t) \in B$. Offensichtlich gilt für alle $(s', t') \in B$: $s' \sim t'$.

**Fall $\varphi = q$.** Aus $s \sim t$ folgt $\lambda(s) = \lambda(t)$. Somit auch $s \models q$ gdw. $q \in \lambda(s)$ gdw. $q \in \lambda(t)$ gdw. $t \models q$.

**Fall $\varphi = \psi_1 \vee \psi_2$.** $s \models \varphi$ gdw. es ein $i \in \{0, 1\}$ gibt, so dass $s \models \psi_i$. Nach Voraussetzung ist dies der Fall, gdw. es ein $i$ gibt, so dass $t \models \psi_i$. Dies ist der Fall, gdw. $t \models \varphi$.

**Fall $\varphi = \neg\psi$.** $s \models \varphi$ gdw. $s \not\models \psi$ gdw. (nach der Hypothese) $t \not\models \psi$ gdw. $t \models \varphi$.

**Fall $\varphi = \langle a \rangle \psi$.** Angenommen $s \models \varphi$. D.h. es gibt ein $s' \in \mathcal{S}$, so dass $s \xrightarrow{a} s'$ und $s' \models \psi$. Wegen $s \sim t$ gibt es auch ein $t' \in \mathcal{S}$ mit $t \xrightarrow{a} t'$ und es gilt $s' \sim t'$. Nach Induktionsvoraussetzung gilt also $s' \models \psi$ gdw. $t' \models \psi$. Somit also auch $t \models \varphi$. Die Umkehrung wird auf dieselbe Art bewiesen. ∎

Die Umkehrung von Satz 2.1 gilt nicht.

**Korollar 2.1**
HML hat die Baummodelleigenschaft.

**Beweis** Sei $\varphi \in$ HML erfüllbar. Also existiert ein Transitionssystem $\mathcal{T}$ mit einem Zustand $s$, so dass $\mathcal{T}, s \models \varphi$ gilt. Betrachte nun die Baumabwicklung $\mathcal{R}_{\mathcal{T}}(s)$ von $\mathcal{T}$ bzgl. $s$. Laut Lemma 1.2 gilt: $\mathcal{T}, s \sim \mathcal{R}_{\mathcal{T}}(s), s$. Laut Satz 2.1 gilt somit auch $\mathcal{R}_{\mathcal{T}}(s), s \models \varphi$. Somit hat $\varphi$ ein Baummodell. ∎

## 2.3 Entscheidungsverfahren und Komplexität

### 2.3.1 Das Model Checking Problem

Wir geben einen rekursiven Algorithmus an, der zu einem gegebenen Transitionssystem und einer gegebenen HML-Formel die Menge aller Zustände im Transitionssystem berechnet, die die Formel erfüllen. Sei $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \Sigma\}, \lambda)$ festgelegt.

**Algorithmus MC-HML**$(\varphi)$
`case` $\varphi$ `of`

| | |
|---|---|
| $q$ : | $\{s \in \mathcal{S} \mid q \in \lambda(s)\}$ |
| $\neg\psi$ : | $\mathcal{S} \setminus$ MC-HML$(\psi)$ |
| $\psi_1 \vee \psi_2$ : | MC-HML$(\psi_1) \cup$ MC-HML$(\psi_2)$ |
| $\langle a \rangle \psi$ : | `let` $T :=$ MC-HML$(\psi)$ `in` |
| | $\{s \in \mathcal{S} \mid \exists t \in T$ s.t. $s \xrightarrow{a} t\}$ |

**Satz 2.2**
Das Model Checking Problem für HML ist in P.

**Beweis** Zuerst überlegen wir uns die Laufzeit von Algorithmus MC-HML. Diese ist, wenn man ihn naiv implementiert, beschränkt durch $\mathcal{O}(m \cdot |\mathcal{T}|)$, wobei $m$ die Länge der Eingabeformel $\varphi$ ist. Man beachte, dass MC-HML für jedes Auftreten einer Teilformel in $\varphi$ höchstens einmal aufgerufen wird und jeder Aufruf in Zeit $\mathcal{O}(|\mathcal{T}|)$ bearbeitet werden kann. Wenn man dann noch die Ergebnisse der rekursiven Aufrufe wie beim dynamischen Programmieren in einer Tabelle abspeichert und bei nochmaliger Verwendung ausliest statt wieder zu berechnen, so ist die Laufzeit durch $\mathcal{O}(|\varphi| \cdot |\mathcal{T}|)$ beschränkt.

Insbesondere ist damit dann Terminierung gezeigt. Dadurch lassen sich Korrektheit und Vollständigkeit leicht durch Induktion über die Anzahl der Rekursionsschritte, bzw. über den Formelaufbau zeigen. D.h. man zeigt, dass MC-HML$(\psi) = \{s \mid \mathcal{T}, s \models \psi\}$ für jedes $\psi$ ist. ∎

Als nächstes zeigen wir, dass dies bereits optimal ist. Von folgendem Problem ist bekannt, dass es P-hart ist: Gegeben ein gerichteter Graph $G = (V, E)$ und zwei Knoten $s, t \in V$, so dass $V$ disjunkt zerlegt ist in $V_I \uplus V_{II}$. Zwei Spieler (I und II) verschieben nun einen Spielstein, der zu Beginn auf $s$ liegt, entlang der Kanten $E$ durch $G$. Liegt der Spielstein auf einem $v \in V_p$, dann darf Spieler $p$ ihn weiterschieben. Spieler II gewinnt, wenn der Spielstein $t$ erreicht. Spieler I gewinnt, wenn die Partie unendlich lang ist und

dabei nie $t$ erreicht, oder eine Sackgasse erreicht ist, die nicht $t$ ist. Es ist P-hart zu entscheiden, ob Spieler II dieses Spiel gewinnen kann, d.h. immer $t$ erreichen kann, egal wie Spieler I zieht. Dieses Problem wird auch *Alternating Directed Graph Reachability* (ADGR) genannt.

**Satz 2.3**
Das Model Checking Problem für HML ist P-hart.

**Beweis** Wir reduzieren nun ADGR auf das Model Checking Problem für HML. Sei $G = (V, E)$ ein gerichteter Graph mit $s, t \in V$ und $V = V_I \uplus V_{II}$. Fasse diesen als knotenbeschriftetes Transitionssystem $\mathcal{T}_G = (V, \to, \lambda, s)$ über der Menge $\mathcal{P} = \{q_I, q_{II}, t\}$ auf, wobei für alle $v \in V$:

- $t \in \lambda(v)$ gdw. $v = t$,
- $q_I \in \lambda(v)$ gdw. $v \in V_I$,
- $q_{II} \in \lambda(v)$ gdw. $v \in V_{II}$.

Dann gilt: Spieler II gewinnt das Spiel auf $G$ gdw. $\mathcal{T}_G, s \models \psi_{|V|}$, wobei

$$\begin{aligned} \psi_1 &:= q_t \\ \psi_{n+1} &:= q_t \vee \big((q_I \to \Box\psi_n) \wedge (q_{II} \to \Diamond\psi_n)\big) \end{aligned}$$

Dies basiert darauf, dass Spieler II jede Partie bereits mit höchstens $|V|$ vielen Zügen gewinnen kann, wenn er überhaupt gewinnen kann.

Da $\mathcal{T}_G$ und $\psi_{|V|}$ mit zusätzlichem Platz, der höchstens $O(\log |G|)$ ist, konstruiert werden können, ist das Model Checking Problem für HML ebenfalls P-hart. ∎

**Korollar 2.2**
Das Model Checking Problem für HML ist P-vollständig.

### 2.3.2 Das Erfüllbarkeitsproblem

**Lemma 2.2**
Für alle $\varphi, \psi_1, \psi_2 \in$ HML gilt: $(\psi_1 \vee \psi_2) \wedge \varphi$ erfüllbar gdw. es ein $i \in \{1, 2\}$ gibt, so dass $\psi_i \wedge \varphi$ erfüllbar ist.

**Beweis** Dies folgt sofort aus der Äquivalenz $(\psi_1 \vee \psi_2) \wedge \varphi \equiv (\psi_1 \wedge \varphi) \vee (\psi_2 \wedge \varphi)$. ∎

**Definition 2.4**
Ein HML-Literal ist jede Formel der Form $q$ oder $\neg q$, wobei $q \in \mathcal{P}$. Eine Menge $L$ von Literalen heißt *konsistent*, wenn es kein $q \in \mathcal{P}$ gibt, so dass $\{q, \neg q\} \subseteq L$.

**Lemma 2.3**
Sei $l_1, \ldots, l_k$ eine konsistente Menge von HML-Literalen und $n, m, k \in \mathbb{N}$. Für alle $\varphi_i, \psi_i \in$ HML gilt:

$$\langle a_1\rangle\varphi_1 \wedge \ldots \langle a_n\rangle\varphi_n \wedge [b_1]\psi_1 \wedge \ldots [b_m]\psi_m \wedge l_1 \ldots l_k$$

ist erfüllbar gdw. $\varphi_i \wedge \bigwedge\{\psi_j \mid b_j = a_i\}$ für alle $i = 1, \ldots, n$ erfüllbar ist.

**Beweis** Sei $\Phi := \langle a_1\rangle\varphi_1 \wedge \ldots [b_1]\psi_1 \wedge \ldots l_1 \ldots$, $L = \{l_1, \ldots, l_k\}$ und $\Phi_i := \varphi_i \wedge \bigwedge\{\psi_j \mid b_j = a_i\}$.

($\Rightarrow$). Angenommen $\Phi$ is erfüllbar, hat also ein Modell $\mathcal{T}, s$. Laut der Semantik von HML gibt es dann $t_1, \ldots, t_n$, so dass für alle $i = 1, \ldots, n$:

- $s \xrightarrow{a_i} t_i$,
- $\mathcal{T}, t_i \models \varphi_i$,
- $\mathcal{T}, t_i \models \psi_j$ für alle $j = 1, \ldots, m$ mit $b_j = a_i$.

Dann gilt aber auch $\mathcal{T}, t_i \models \Phi_i$ für alle $i = 1, \ldots, n$, was zu zeigen ist.

($\Leftarrow$). Sei nun $\Phi_i$ erfüllbar für alle $i = 1, \ldots, n$. D.h. es gibt Transitionssysteme $\mathcal{T}_i = (\mathcal{S}_i, \{\xrightarrow{a}_i \mid a \in \Sigma\}, \lambda_i, t_i)$, so dass $\mathcal{T}, t_i \models \Phi_i$ für alle $i = 1, \ldots, n$ gilt. Konstruiere nun ein Transitionssystem $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \Sigma\}, \lambda, s)$ wie folgt.

- $\mathcal{S} = \{s\} \uplus \biguplus_{i=1}^{n} \mathcal{S}_i$,
- $\xrightarrow{a} = \bigcup_{i=1}^{n} \xrightarrow{a}_i \ \cup \ \{(s, t_i) \mid a = a_i\}$,
- $\lambda(v) = \lambda_i(v)$, falls $v \in \mathcal{S}_i$, und $\lambda(s) = L \cap \mathcal{P}$.

Man sieht leicht, dass $\mathcal{T}, s \models \Phi$ gilt. Dies benutzt die Tatsache, dass $L$ konsistent ist. ■

Wir geben nun einen rekursiven Algorithmus an, der zu einer gegebenen Formel $\varphi$ entscheidet, ob diese erfüllbar ist. Dazu generalisieren wir das Problem und entscheiden zu einer gegebenen Formel*menge* $\Phi$ ob diese (als Konjunktion verstanden) erfüllbar ist. Wir gehen o.B.d.A. davon aus, dass die Eingabe $\varphi$ in positiver Normalform vorliegt. Beachte, dass also Literale die einzig negierten Unterformel von $\varphi$ sein können.

**Algorithmus SAT-HML**($\Phi$)
```
if Φ enthält Formel der Form ψ1 ∧ ψ2 then
    SAT-HML((Φ \ {ψ1 ∧ ψ2}) ∪ {ψ1, ψ2})
else if Φ enthält Formel der Form ψ1 ∨ ψ2 then
        if  SAT-HML((Φ \ {ψ1 ∨ ψ2}) ∪ {ψ1}) = true then
            true
        else
            SAT-HML((Φ \ {ψ1 ∨ ψ2}) ∪ {ψ2})
else
    sei Φ = {⟨a1⟩φ1, ..., ⟨an⟩φn, [b1]ψ1, ..., [bm]ψm, l1, ..., lk}
    if {l1, ..., lk} inkonsistent then false
    else
        for i = 1, ..., n do
            if SAT-HML({φi} ∪ {ψj | bj = ai}) = false then false
        done
        true
```

Es ist nicht schwer zu sehen, dass SAT-HML($\Phi$) immer terminiert. Sei $\|\Phi\| := \sum_{\psi\in\Phi} \|\psi\|$, wobei

$$\begin{array}{rcl} \|q\| = \|\neg q\| & := & 0 \qquad \text{für alle } q \in \mathcal{P} \\ \|\psi_1 \vee \psi_2\| = \|\psi_1 \wedge \psi_2\| & := & 1 + \|\psi_1\| + \|\psi_2\| \\ \|\langle a\rangle\psi\| = \|[a]\psi\| & := & 1 + \|\psi\| \end{array}$$

Dann gilt: Wenn $\|\Phi\| = 0$, dann trifft der letzte Fall zu, und darin wird die `for`-Schleife nicht durchlaufen, d.h. es finden keine weiteren rekursiven Aufrufe statt. In den anderen Fällen verringert sich $\|\Phi\|$ jeweils in den rekursiven Aufrufen um mindestens 1.

Somit lassen sich Korrektheit und Vollständigkeit durch Induktion über die Rekursionstiefe zeigen. Genauer: Man zeigt leicht, dass SAT-HML($\Phi$) = *true* gdw. $\Phi$ erfüllbar. Beachte, dass SAT-HML($\Phi$) keine rekursiven Aufrufe durchläuft gdw. $\Phi$ vor der Form $\{[b_1]\psi_1, \ldots, [b_m]\psi_m, l_1, \ldots, l_k\}$ für $m \geq 0$ ist und dass solch eine Formelmenge genau dann erfüllbar ist, wenn $l_1, \ldots, l_k$ konsistent ist. Korrektheit und Vollständigkeit folgt dann sofort aus der Induktionshypothese, und zwar trivialerweise im ersten Fall einer auftretenden Konjunktion, aus Lemma 2.2 im Fall einer auftretenden Disjunktion und aus Lemma 2.3 in dem letzten Fall.

**Satz 2.4**
Das Erfüllbarkeitsproblem für HML ist in PSPACE.

**Beweis** Algorithmus SAT-HML kann so implementiert werden kann, dass man für jedes Argument eines rekursiven Aufrufs höchstens Platz $\mathcal{O}(|\varphi|)$ braucht, wenn $\varphi$ die Eingabeformel ist. Man beachte, dass dabei SAT-HML immer nur mit einer Teilmenge der Unterformelmenge von $\varphi$ aufgerufen wird. Außerdem ist die Rekursionstiefe durch $|\varphi|$ beschränkt, da in jedem rekursiven Aufruf mindestens eine Unterformel aus dem aktuellen Argument entfernt und höchstens durch kleinere Unterformeln ersetzt wird. Somit braucht der Rekursionsstack höchstens $|\varphi|$ viele Einträge, und der Gesamtplatzverbrauch ist damit durch $\mathcal{O}(|\varphi|^2)$ beschränkt. ■

Abschließend zeigen wir wieder, dass obiges Komplexitätsresultat optimal ist.

**Definition 2.5**
Eine *quantifizierte, boolesche Formel* in Normalform über einer Menge $x_1, \ldots, x_n$ von Variablen ist ein

$$\Phi := Q_n x_n. \ldots \exists x_3. \forall x_2. \exists x_1. \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{i,j}$$

wobei die $l_{i,j}$ Literale über den Variablen $x_1, \ldots, x_n$ sind und $Q_n = \exists$, falls $n$ ungerade, $Q_n = \forall$ sonst.

Die Gültigkeit eines solches $\Phi$ ist induktiv definiert durch:

0 und $\neg 1$ sind nicht gültig,

1 und $\neg 0$ sind gültig,

$\bigvee l_i$ ist gültig, wenn ein $l_i$ gültig ist,

$\bigwedge \Phi_i$ ist gültig, wenn alle $\Phi_i$ gültig sind,

$\exists x_i.\Phi$ ist gültig, wenn $\Phi[0/x_i]$ oder $\Phi[1/x_i]$ gültig ist,

$\forall x_i.\Phi$ ist gültig, wenn $\Phi[0/x_i]$ und $\Phi[1/x_i]$ gültig sind.

Dabei bezeichnet $\Phi[b/x_i]$ die Formel, die aus $\Phi$ entsteht, wenn jedes Vorkommen von $x_i$ in ihr durch $b$ ersetzt wird.

Sei QBF := $\{\Phi \mid \Phi \text{ ist gültig } \}$.

**Satz 2.5 (ohne Beweis)**
Die Sprache QBF ist PSPACE-hart.

**Satz 2.6**
Das Erfüllbarkeitsproblem für HML ist PSPACE-hart.

**Beweis** Wir zeigen dies durch Reduktion auf QBF. Sei $\Phi$ eine quantifizierte, boolesche Formel

$$\Phi \;:=\; Q_n x_n. \ldots \exists x_3. \forall x_2. \exists x_1. \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{i,j}$$

Dann konstruieren wir daraus eine HML-Formel $\varphi$ über der Menge $\mathcal{P} = \{x_1, \ldots, x_n\}$ von Propositionen. Setze $\varphi := \psi_n \wedge \alpha$, wobei

$$\psi_i \;:=\; \begin{cases} \Diamond\psi_{i-1} \;, & \text{falls } i \text{ ungerade} \\ \Diamond(x_i \wedge \psi_{i-1}) \wedge \Diamond(\neg x_i \wedge \psi_{i-1}) \;, & \text{falls } i \text{ gerade} \end{cases} \quad \text{für } i = 1, \ldots, n$$

$$\psi_0 \;:=\; \bigwedge_{i=1}^{m} \Diamond\psi_i'$$

$$\psi_i' \;:=\; \bigvee_{j=1}^{k_i} l_{i,j} \qquad \text{für alle } i = 1, \ldots, m$$

$$\alpha \;:=\; \bigwedge_{i=1}^{n} \bigwedge_{j=i}^{n+1} \Box^j \big((x_i \to \Box x_i) \wedge (\neg x_i \to \Box\neg x_i)\big)$$

$$\Box^0\beta \;:=\; \beta$$

$$\Box^{k+1}\beta \;:=\; \Box\Box^k\beta$$

Zuerst sei bemerkt, dass die syntaktische Länge von $\varphi$ zwar exponentiell in $|\Phi|$ ist, aber $\varphi$ nur aus polynomiell in $|\Phi|$ vielen Unterformeln besteht.

Ein *Zeuge* für solch ein $\Phi$ ist ein Baum $\mathcal{T}$ der Tiefe $n+2$, dessen Knoten mit Mengen von Propositionen beschriftet sind. Wir schreiben $\lambda(s)$ für die Beschriftung eines Knotens $s$. Der Einfachheit halber gehen wir davon aus, dass der Baum die Ebenen $n, \ldots, -1$ hat mit der Wurzel auf der Ebene $n$ und den Blättern auf der Ebene $-1$. Folgendes gilt:

1. Für alle ungeraden $i = 1, 3 \ldots, n$: Ein Knoten $s$ auf Ebene $i$ hat genau einen Nachfolger $t$ und entweder $x_i \in \lambda(t)$ oder $x_i \notin \lambda(t)$ (was natürlich selbstverständlich ist).
2. Für alle geraden $i = 2, 4 \ldots, n$: Ein Knoten $s$ auf Ebene $i$ hat genau zwei Nachfolger $t_1$ und $t_2$ und $x_i \in \lambda(t_1)$ und $x_i \notin \lambda(t_2)$.
3. Ein Knoten auf Ebene 0 hat genau $m$ Nachfolger $t_1, \ldots, t_m$. Für diese gilt: $\lambda(t_i) \supseteq \{x_h \mid x_h = l_{i,j}$ für ein $j \in \{1, \ldots, k_i\}\}$.
4. Für alle $i = 1, \ldots, n$ und alle $j = 0, \ldots, n - i$ und alle Knoten $s$ auf Ebene $j$ gilt: wenn $x_i \in \lambda(s)$, dann $x_i \in \lambda(t)$, und wenn $x_i \notin \lambda(s)$, dann $x_i \notin \lambda(t)$, für alle Nachfolger $t$ von $s$.

Man überzeugt sich leicht, dass gilt: Wenn $\Phi$ gültig ist dann gibt es einen Zeugen von $\Phi$. Ein solcher Zeuge $\mathcal{T}$ mit Wurzel $s$ ist ein Transitionssystem und $\mathcal{T}, s \models \varphi$. Also ist $\varphi$ erfüllbar.

Umgekehrt: Wenn $\varphi$ erfüllbar ist, dann hat es ein Modell $\mathcal{T}, s$. Laut Korollar 2.1 kann man annehmen, dass $\mathcal{T}$ ein Baum mit Wurzel $s$ ist. Aus diesem lässt sich dann ein Zeuge für $\Phi$ gewinnen, womit $\Phi$ gültig sein muss. ■

## 2.4 Ausdrucksstärke

Wir haben gesehen, dass HML bereits einige gute Eigenschaften hat: Bisimulationsinvarianz, polynomielles Model Checking Problem, nicht allzu schweres Erfüllbarkeitsproblem, etc. Dennoch hat HML einen entschiedenen Nachteil: Die Logik ist viel zu ausdrucksschwach. Wir werden zeigen, dass bereits sehr elementare, aber wichtige Eigenschaften nicht in HML ausdrückbar sind. Der Grund dafür ist, dass eine HML-Formel nur "beschränkt tief in ein Transitionssystem schauen kann".

**Definition 2.6**
Die *modale Tiefe* $md(\varphi)$ einer HML-Formel $\varphi$ ist induktiv definiert als:

$$\begin{aligned} md(q) &:= 0 \\ md(\varphi \vee \psi) = md(\varphi \wedge \psi) &:= \max\{md(\varphi), md(\psi)\} \\ md(\neg\varphi) &:= md(\varphi) \\ md(\langle a \rangle \varphi) = md([a]\varphi) &:= 1 + md(\varphi) \end{aligned}$$

**Satz 2.7**
Sei $q \in \mathcal{P}$ und $M := \{(\mathcal{T}, s) \mid \mathcal{T} = (\mathcal{S}, \rightarrow, \lambda, s_0)$ und es gibt einen Pfad $s_0, s_1, \ldots$ durch $\mathcal{T}$ mit $s_0 = s$ und ein $n \in \mathbb{N}$, so dass $q \in \lambda(s_n)\}$. Es gibt keine HML-Formel $\varphi$, so dass $[\![\varphi]\!] = M$.

**Beweis** Wir definieren zuerst zwei Familien $\mathcal{T}_i, \mathcal{T}_i'$, $i \in \mathbb{N}$, von Transitionssystemen. $\mathcal{T}_i = (\{0, \ldots, i{+}1\}, \rightarrow, \lambda, 0)$, wobei $j \rightarrow j{+}1$ für alle $j \leq i$ und $\lambda(i{+}1) = \{q\}$, $\lambda(j) = \emptyset$ für alle $j \leq i$. $\mathcal{T}_i'$ ist definiert wie $\mathcal{T}_i$ mit dem einzigen Unterschied, dass dort gilt $\lambda'(i{+}1) = \emptyset$.

Offensichtlich gelten die folgenden beiden Aussagen für alle $i \in \mathbb{N}$ und alle $j \leq i$.

1. $(\mathcal{T}_i, 0) \in M$ und $(\mathcal{T}_i', 0) \notin M$,

2. $\mathcal{T}_i, 0 \sim \mathcal{T}_{i+1}, 1$ und $\mathcal{T}_i', 0 \sim \mathcal{T}_{i+1}', 1$.

Als nächstes zeigen durch Induktion über den Formelaufbaeu von HML: Für alle $i \in \mathbb{N}$ und alle $\varphi \in$ HML mit $md(\varphi) = i$ gilt: $\mathcal{T}_i, 0 \models \varphi$ gdw. $\mathcal{T}_i', 0 \models \varphi$.

Fall $\varphi = p$ für ein $p \in \mathcal{P}$. Offensichtlich ist $md(p) = 0$. Dann gilt $\mathcal{T}_0, 0 \not\models p$ und $\mathcal{T}_0', 0 \not\models p$ wegen $\lambda(0) = \lambda'(0) = \emptyset$.

Fall $\varphi = \psi_1 \vee \psi_2$. Sei $i := md(\varphi)$. Dann gilt: $\mathcal{T}_i, 0 \models \varphi$ gdw. $\mathcal{T}_i, 0 \models \psi_1$ oder $\mathcal{T}_i, 0 \models \psi_2$ gdw. $\mathcal{T}_i', 0 \models \psi_1$ oder $\mathcal{T}_i', 0 \models \psi_2$ gdw. $\mathcal{T}_i', 0 \models \varphi$.

Die Fälle $\varphi = \psi_1 \wedge \psi_2$ und $\varphi = \neg\psi$ werden genauso gezeigt.

Fall $\varphi = \Diamond\psi$. Sei $i = md(\varphi)$, also $md(\psi) = i - 1$. Angenommen, $\mathcal{T}_i, 0 \models \varphi$. Da $0 \to 1$ in $\mathcal{T}_i$ gilt: $\mathcal{T}_i, 1 \models \psi$. Aber $\mathcal{T}_i, 1 \sim \mathcal{T}_{i-1}, 0$ laut Aussage (2) oben. Nach Satz 2.1 gilt somit: $\mathcal{T}_{i-1}, 0 \models \psi$. Die Induktionshypothese liefert $\mathcal{T}_{i-1}', 0 \models \psi$, und wiederum wegen (2) gilt $\mathcal{T}_i', 1 \models \psi$. Dann gilt aber auch $\mathcal{T}_i', 0 \models \psi$. Die Rückrichtung wird genauso gezeigt.

Der Fall wird $\varphi = [a]\psi$ wird genauso gezeigt wie der vorige Fall.

Die Aussage des Satzes ist dann einfach zu beweisen. Angenommen, es gäbe eine HML-Formel $\varphi$ mit $[\![\varphi]\!] = M$. Dann hätte diese eine feste, modale Tiefe, z.B. $md(\varphi) = i$. Also gilt $\mathcal{T}_i, 0 \models \varphi$ und $\mathcal{T}_i', 0 \not\models \varphi$. Dies wiederspricht aber der eben bewiesenen Aussage, dass $\varphi$ nicht $\mathcal{T}_i, 0$ und $\mathcal{T}_i', 0$ unterscheiden kann. ∎

## 2.5 Infinitäre Modallogik

Zur Erinnerung: Die Modellklasse einer HML-Formel ist unter Bisimulation abgeschlossen, aber nicht alle Modelle einer solchen Klasse sind bisimilar. Laut der Bemerkung nach Satz 2.1 und dem Beweis von Satz 2.7 ist das Problem, dass eine Formel nur "beschränkt tief in ein Transitionssystem schauen kann". Diese Beschränkung ließe sich leicht mithilfe von unendlich großen Formeln umgehen. So wird "es gibt einen Pfad, auf dem irgendwann einmal $q$ gilt" z.B. durch die Formel $\bigvee_{i \in \mathbb{N}} \Diamond^i q$ ausgedrückt.

Selbst wenn man Formeln dieser Form in HML zuließe, würde die Rückrichtung von Satz 2.1 immer noch nicht gelten. Man braucht in der Tat *beliebig* große Disjunktionen und Konjunktionen. So erhält man die für praktische Zwecke wertlose, aber in der Theorie nützliche, infinitäre Modallogik $\infty$-HML. Ihre Syntax ist definiert durch

$$\varphi \ ::= \ q \mid \bigvee_{i \in I} \varphi_i \mid \bigwedge_{i \in I} \varphi_i \mid \neg\varphi \mid \langle a \rangle \varphi \mid [a]\varphi$$

wobei $q \in \mathit{Prop}$, $a \in \Sigma$ und $I$ eine beliebige Indexmenge ist. Die Semantik einer $\infty$-HML-Formel $\varphi$ ist definiert wie die einer herkömmlichen HML-Formel.
Wir betrachten die Logik $\infty$-HML nur aus Gründen der Bisimulationsinvarianz. Der erste Teil – "$\infty$-HML-Formeln können bisimilare Strukturen nicht unterscheiden" wird genauso induktiv über den Formelaufbau wie Satz 2.1 bewiesen. Der trans-finite Fall der booleschen Operatoren macht dabei keine Schwierigkeiten.

**Satz 2.8**
Für alle Transitionssysteme $\mathcal{T}$ mit Zuständen $s, t \in \mathcal{S}$ gilt: $s \sim t$ gdw. für alle $\varphi \in \infty$-HML gilt: $s \models \varphi$ gdw. $t \models \varphi$.

**Beweis** ($\Rightarrow$) Wird genauso wie Satz 2.1 durch Induktion über den Formelaufbau beweisen. Die trans-finiten Fälle machen dabei keine Schwierigkeiten.

($\Leftarrow$) Angenommen, $s \models \varphi$ gdw. $t \models \varphi$ für alle $\varphi \in \infty$-HML. Wir konstruieren eine binäre Relation $B$ auf $\mathcal{S}$ wie folgt.

$$B \;:=\; \{(s', t') \mid \forall \varphi \in \infty\text{-HML} : s' \models \varphi \text{ gdw. } t' \models \varphi\}$$

Wir zeigen, dass $B$ eine Bisimulation ist. Dafür nehmen wir das Gegenteil an. Es gebe also $(s', t') \in B$, so dass es ein $a \in \Sigma$ und ein $s'' \in \mathcal{S}$ gibt mit $s' \xrightarrow{a} s''$. Jetzt gibt es zwei Möglichkeiten, die Eigenschaften einer Bisimulation zu verletzen.

(1) Angenommen, es gibt kein $t''$ mit $t' \xrightarrow{a} t''$. Aber $s' \models \langle a \rangle \mathtt{tt}$ und somit auch $t' \models \langle a \rangle \mathtt{tt}$, was ein Widerspruch ist.

(2) Sei die Menge $T := \{t'' \mid t' \xrightarrow{a} t''\}$ nicht leer, aber $\{(s'', t'') \mid t'' \in T\}$ leer. D.h. für jedes $t'' \in T$ existiert eine Formel $\varphi_{t''} \in \infty$-HML, so dass nicht gilt: $s'' \models \varphi_{t''}$ gdw. $t'' \models \varphi_{t''}$. Da $\infty$-HML unter Negation abgeschlossen ist, können wir $s'' \models \varphi_{t''}$ und $t'' \not\models \varphi_{t''}$ voraussetzen. Betrachte jetzt die Formel $\psi := \bigwedge_{t'' \in T} \varphi_{t''}$. Offensichtlich gilt $s'' \models \psi$, aber $t'' \not\models \psi$ für jedes $t'' \in T$. Somit gilt auch $s' \models \langle a \rangle \psi$, aber $t' \not\models \langle a \rangle \psi$ im Widerspruch zu der Annahme, das $(s', t') \in B$.

Somit muss $B$ eine Bismulation sein. Ausserdem gilt $(s, t) \in B$ wegen der Annahme, dass $s \models \varphi$ gdw. $t \models \varphi$ für alle $\varphi \in$ HML. Also $s \sim t$. ∎