Übersetzung zwischen dem Linearzeit-μ-Kalkül und schwachen alternierenden Paritätsautomaten

Karl Mehltretter
LMU München

23. Juni 2006

Betreuer: Martin Lange

Zusammenfassung

Im Rahmen des Musabre-Projektes wurde eine Übersetzung von Formeln des Linearzeit-μ-Kalküls in schwache alternierende Paritätsautomaten implementiert. Dieser Automaten-Typ ist wegen seiner speziellen Struktur algorithmisch einfach zu handhaben. Die Rückübersetzung liefert eine Formel ohne alternierende Fixpunkt-Operatoren.

1 Grundlagen

1.1 Linearzeit-μ-Kalkül

Der Linearzeit-μ-Kalkül (Ltμ) ist eine Temporallogik zur Spezifikation von regulären Eigenschaften auf unendlichen Wörtern. Sei $\mathcal{P}$ eine Menge von atomaren Propositionen und $\mathcal{V}$ eine Menge von Variablensymbolen.

Definition 1 Formeln des Linearzeit-μ-Kalküls in positiver Normalform sind gegeben durch folgende Grammatik:

$$\varphi ::= p \mid \neg p \mid X \mid \phi \vee \phi \mid \phi \wedge \phi \mid \bigcirc \phi \mid \mu X.\varphi \mid \nu X.\varphi$$

wobei $a \in \mathcal{P}$ und $X \in \mathcal{V}$.

Im Folgenden steht σ für μ oder ν. Es wird davon ausgegegangen, dass jede Formel *wohlbenannt* ist. Dies bedeutet, dass eine Variable X nur einmal durch

eine Unterformel $\sigma X.\varphi$ gebunden wird. Damit kann man der Variable X eindeutig ihren *Fixpunkttyp* σ zuordnen. Außerdem wird angenommen, dass die Formeln *streng bewacht* ist: jeder Variable geht unmittelbar ein $\bigcirc$ voraus. Jede Ltμ-Formel kann in diese Form gebracht werden (für den Beweis siehe [4]).
Formeln von Ltμ werden interpretiert auf unendlichen Wörtern über dem Alphabet $\Sigma = 2^{\mathcal{P}}$. Für die induktive Definition der Semantik wird eine Belegung $\rho : \mathcal{V} \to \mathbb{N}_0$ benötigt.
Damit ergibt sich $[\![\varphi]\!]_\rho$ in Bezug auf ein Wort $w = w_0 w_1 w_2 \ldots$ durch:

- $[\![p]\!]_\rho := \{i \in \mathbb{N} \mid p \in w_i\}$
- $[\![\neg p]\!]_\rho := \{i \in \mathbb{N} \mid p \notin w_i\}$
- $[\![X]\!]_\rho := \rho(X)$
- $[\![\varphi_1 \vee \varphi_2]\!]_\rho := [\![\varphi_1]\!]_\rho \cup [\![\varphi_2]\!]_\rho$
- $[\![\varphi_1 \wedge \varphi_2]\!]_\rho := [\![\varphi_1]\!]_\rho \cap [\![\varphi_2]\!]_\rho$
- $[\![\bigcirc\varphi]\!]_\rho := \{i \in \mathbb{N} \mid i+1 \in [\![\varphi]\!]_\rho\}$
- $[\![\mu X.\varphi]\!]_\rho := \bigcap\{M \subseteq \mathbb{N} \mid [\![\varphi]\!]_{\rho[X \mapsto M]} \subseteq M\}$
- $[\![\nu X.\varphi]\!]_\rho := \bigcup\{M \subseteq \mathbb{N} \mid M \subseteq [\![\varphi]\!]_{\rho[X \mapsto M]}\}$

Für eine geschlossene Formel hängt die Semantik damit nicht von ρ ab und kann daher in der Notation weggelassen werden. Wir schreiben $w \vDash \varphi$ gdw. $0 \in [\![\varphi]\!]$. Die Sprache $L(\varphi)$ ist die Menge $\{w \in \Sigma^\omega \mid w \vDash \varphi\}$. Zwei Formeln sind äquivalent, wenn ihre Sprachen gleich sind.

Definition 2 In einer Formel hängt Variable X von Y ab (geschrieben $X \prec Y$), wenn Y in den freien Variablen der zu X gehörenden Fixpunkt-Formel vorkommt. Sei $<$ die transitive Hülle von $\prec$. Die *Alternierungstiefe* einer Formel ist die Länge n in einer maximalen Kette $X_0 < X_1 ... < X_n$, wobei der Fixpunkt-Typ der Variablen in der Kette jeweils wechselt.

Eine Formel heißt *alternierungsfrei*, wenn sie Alternierungstiefe 0 hat.
Zur Logik lassen sich auch simultane Fixpunktoperatoren $\sigma X.\Phi$ hinzufügen, wobei Φ dann eine Liste von Paaren aus Variablen und Formeln ist. Damit wird die Ausdrucksstärke nicht erhöht, aber Eigenschaften lassen sich damit oft einfacher formulieren.

Beispiele

1. $\mu X.a \vee \bigcirc X$
   diese Formel besagt, dass mindestens ein a auftreten muss.
2. $\nu X.\begin{pmatrix} X & . & a \wedge \bigcirc Y \\ Y & . & b \wedge \bigcirc X \end{pmatrix}$
   diese simultane Fixpunktformel beschreibt das Wort *abababab...*
3. $\nu X.(\mu Y.a \vee \bigcirc Y) \wedge \bigcirc X$
   die Formel ist alternierungsfrei und besagt, dass unendlich oft ein a auftritt.
4. $\nu X.\mu Y.(a \wedge \bigcirc X) \vee \bigcirc Y$
   die Formel ist äquivalent zur vorherigen Formel, hat jedoch Alternierungstiefe 1.

1.2 Alternierende Automaten

Automaten sind eine weitere Möglichkeit zur Beschreibung von Eigenschaften auf unendlichen Wörtern: Die Sprache $L(\mathcal{A})$ eines Automaten $\mathcal{A}$ ist die Menge der Wörter, die er akzeptiert.
Alternierende Automaten [5] sind eine Erweiterung nichtdeterministischer Automaten: Neben der existenziellen Auswahl bei nichtdeterministischen Automaten gibt es bei alternierenden Automaten noch die universelle Auswahl.

Definition 3 Sei Q eine Menge. Die Menge $B^+(Q)$ der *positiv booleschen Formeln* über Q ist die kleinste Menge, so dass gilt

- $Q \subseteq B^+(Q)$
- $\varphi_1, \varphi_2 \in B^+(Q) \Rightarrow \varphi_1 \vee \varphi_2$
- $\varphi_1, \varphi_2 \in B^+(Q) \Rightarrow \varphi_1 \wedge \varphi_2$

Positiv boolesche Formeln lassen sich damit zur Modellierung der existenziellen bzw. universellen Auswahl bei alternierenden Automaten verwenden. Zum Beispiel soll die Formel $q_1 \vee (q_2 \wedge q_3)$ bedeuten, dass der Automat in den Zustand q_1 übergeht oder gleichzeitig in die Zustände q_2 und q_3.
Wegen der existenziellen Auswahl gibt es mehrere Läufe des Automaten auf einem Wort, und wegen der universellen Auswahl ist ein Lauf keine Folge, sondern ein unendlicher Baum.

Definition 4 Ein *alternierender Büchi Automat* (ABA)
ist ein Tupel $(Q, \Sigma, q_0, \delta, F)$. Dabei ist

- Q eine endliche Menge von Zuständen.
- Σ ein endliches Alphabet.
- $q_0 \in Q$ der Anfangszustand.
- $\delta : Q \times \Sigma \to B^+(Q)$ die Übergangsfunktion.
- $F \subseteq Q$ die Menge der akzeptiernden Zustände.

Ein Pfad eines Laufes ist akzeptierend, wenn ein Element aus F unendlich oft darin vorkommt. Ein Lauf ist akzeptierend, wenn alle Pfade des Laufes akzeptierend sind. Ein ABA akzeptiert ein Wort, wenn es einen akzeptierenden Lauf gibt.
Bei einem *co-ABA* ist ein Pfad eines Laufes akzeptierend, wenn *kein* Element aus F unendlich oft vorkommt.
Eine Verallgemeinerung von Büchi-Automaten sind Paritätsautomaten:

Definition 5 Ein *alternierender Paritätsautamat* (APA) ist ein Tupel $(Q, \Sigma, q_0, \delta, \Omega)$. Hierbei ist im Vergleich zum ABA

- $\Omega : Q \to \mathbb{N}$. Ω ordnet jedem Zustand eine *Priorität* zu.

Ein Pfad im Lauf eines APA (bzw. co-APA) ist akzeptierend, wenn die kleinste Priorität, die unendlich oft vorkommt, gerade (bzw. ungerade) ist.
Die folgenden *schwachen* Automaten haben im Vergleich zu obigen (*starken*) Automaten eine eingeschränkte Struktur. Sie sind damit das Pendant zu alternierungsfreien Formeln im Linearzeit-μ-Kalkül.

Definition 6 Ein *normalisierter schwacher alternierender Paritätsautomat* (WAPA) ist ein APA $(Q, \Sigma, q_0, \delta, P)$ mit

- Ω ist *normalisiert*: Von Zustand mit Priorität p sind nur Zustände mit Priorität $p' \leq p$ erreichbar.
- ein Pfad eines Laufes ist akzeptierend, wenn die kleinste Priorität, die überhaupt vorkommt, gerade ist.

Da Ω normalisiert ist, macht es für die Akzeptanz keinen Unterschied, ob eine Priorität unendlich oft oder mindestens einmal auf einem Pfad vorkommt.

Definition 7 Ein *schwacher alternierender Büchi Automat* (WABA)
ist ein ABA $A = (Q, \Sigma, q_0, \delta, F)$ und es gilt: es gibt eine Partition $\mathbb{P}$ von Q und eine Ordnung $\preceq$ darauf, so dass gilt:

- für $P \in \mathbb{P} : P \cap F = \emptyset$ oder $P \cap F = P$
- wenn q' von q aus erreichbar ist, dann $[q] \preceq [q']$. Dabei bezeichnet $[q]$ die Komponente von $\mathbb{P}$, in der sich q befindet.

Bei einem WAPA ist also die spezielle Struktur eines schwachen Automaten durch die Normalitätsbedingung in den Prioritäten gespeichert, während bei einem WABA nur die Existenz einer Partition, die die Schwäche bezeugt, gefordert wird.

1.3 Implementierung der Automaten

Bei der Verwendung von Automaten bei der Verifikation von Systemläufen ist das Alphabet $\Sigma = 2^{\mathcal{P}}$ (mit $\mathcal{P}$ einer Menge an atomaren Propositionen) in der Regel sehr groß. Daher kann aus Effizienzgründen die Zustandsübergangsfunktion nicht direkt z.B. als Tabelle abgespeichert werden.
Deswegen hat in der Implementierung die Übergangsfunktion der altenierenden Automaten den Typ $Q \rightarrow \mathit{Delta}(Q, \mathcal{P})$. ein Element von $\mathit{Delta}(Q, \mathcal{P})$ ist dabei

- ein Zustand $q \in Q$.
- ein Literal p bzw. $\neg p$ ($p \in \mathcal{P}$) als *Guard*
- eine Konjunktion bzw. Disjunktion von Elementen des Typs $\mathit{Delta}(Q, \mathcal{P})$. Mit der leeren Menge kann damit auch *true* bzw. *false* definiert werden.

2 Übersetzung von Ltμ in einen WAPA

2.1 Ltμ nach APA

Satz 1 Zu jeder Ltμ-Formel gibt es einen äquivalenten APA bzw. co-APA. Die Grösse des Automaten ist linear in der Größe der Formel.

Ein APA ist im Grunde der Syntaxbaum einer Formel, wobei durch die Fixpunkt-Operatoren Zykel im Automaten entstehen. Die Prioritäten ergeben sich aus den Fixpunkt-Typen und den Abhängigkeiten der Variablen.
Die Unterscheidung zwischen APA und co-APA erfolgt, um sicherzustellen, dass die kleinste Priorität 1 auch tatsächlich im Automaten vorkommt.
Zustände, die zu einem μ gehören, erhalten in einem APA ungerade Priorität, und diejenigen, die zu einem ν gehöhren, erhalten eine gerade Priorität. Im Falle eines co-APAs verhält es sich gerade umgekehrt.

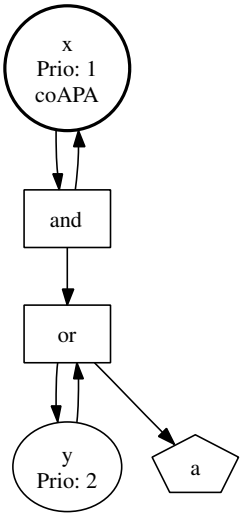

Abbildung 1: co-APA zu $\nu X.(\mu Y.a \vee \bigcirc Y) \wedge \bigcirc X$

Beispiele Abbildungen 1 und 2 zeigen die entstehenden Automaten zu den 2 äquivalenten Formeln aus Abschnitt 1.1. Zustände sind durch Kreise dargestellt, Fünfecke stehen für die Guards der Zustandsübergänge. Zu beachten ist, dass sich die Alternierung in der zweiten Formel im entsprechenden Automaten dadurch bemerkbar macht, dass es einen Zykel zwischen gerader und ungerade Priorität gibt. Gerade dies ist bei schwachen Paritätsautomaten nicht möglich.

2.2 Zu Läufen von Büchi-Automaten

Um die Idee zu verdeutlichen, wie aus einem starken Automaten ein äquivalenter schwacher Automat gewonnen werden kann, muss auf einige Eigenschaften von Läufen eingegangen werden. Zur Erinnerung sei darauf verwiesen, dass ein Lauf eines alternierenden Automaten ein unendlicher Baum ist.

Definition 8 Ein Lauf heist *gedächtnislos*, wenn für alle Knoten des Laufes gilt:
Sind Knoten mit den gleichen Zuständen beschriftet und liegen auf demselben Level, so sind deren Unterbäume gleich.

Damit können gleichbeschriftete Knoten auf demselben Level verschmolzen werden. Der Baum wird damit zu einem DAG (directed acyclic graph), dessen Breite durch die Anzahl der Zustände beschränkt ist. Man kann zeigen,

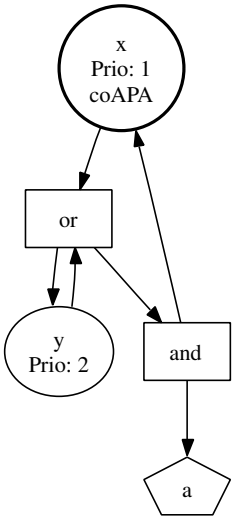

Abbildung 2: co-APA zu $\nu X.\mu Y.(a \wedge \bigcirc X) \vee \bigcirc Y$

dass es, wenn es einen akzeptierenden Lauf gibt, auch einen gedächtnislosen akzeptierenden Lauf gibt.

Definition 9 Sei ein Lauf eines co-ABAs als DAG gegeben. Ein Knoten k des DAGs heißt

- *eventually safe* wenn von k aus nur endlich viele Knoten erreichbar sind.
- *currently safe*, wenn k selbst und alle von k aus erreichbaren Knoten nicht in F sind.

Definition 10 Sei G der Lauf eines co-ABAs als DAG. Definiere die Folge G_i an DAGs wie folgt:

- $G_0 = G$
- $G_{2i+1} = G_{2i} \setminus \{k \mid k \text{ eventually safe in } G_{2i}\}$
- $G_{2i+2} = G_{2i+1} \setminus \{k \mid k \text{ currently safe in } G_{2i+1}\}$

Setze nun den *Rang* eines Knotens k wie folgt: $\text{rang}(k) = i$ für k eventually safe oder currently safe in G_i.

Bei Automaten auf unendlichen Wörtern gilt, dass der Rang eines Knotens maximal $2n$ ist, wobei n die Anzahl der Zustände des Automaten ist.

Die Idee für die Umwandlung eines ABAs in einen WABA ist nun, den entstehenden Automaten die Ränge eines akzeptierenden gedächtnislosen Laufes raten zu lassen. Zu Details dieser Konstruktion siehe [2].

2.3 APA nach WABA

Die im vorherigen Abschnitt beschriebene Idee wird nun für die Umwandlung eines APA in einen WABA konkretisiert.
Die Umwandlung wird schrittweise erfolgen, wobei in jedem Schritt Ränge geraten werden. Als Datenstruktur für das Verfahren wird noch folgender Automaten-Typ benötigt:

Definition 11 Ein *weak-parity alternating automaton* (WPAA) ist ein Tupel $(S, P, \Sigma, q_0, \delta, \Omega)$ mit

- S: die Menge der einfachen Zustände.
- P: die Menge der Paritätszustände (disjunkt zu S).
- $q_0 \in S \cup P$: der Anfangszustand.
- $\delta : S \cup P \times \Sigma \to B^+(S \cup P)$ die Übergangsfunktion.
- $\Omega : P \to \mathbb{N}$ die Prioritätsfunktion.

Ein WPAA ist also ein APA, der zusätzlich noch einfache Zustände enthält, die keine Priorität erhalten und für die Akzeptanz eines Wortes keine Rolle spielen.
Ein APA $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ kann als WPAA $\mathcal{A}' = (\emptyset, Q, \Sigma, q_0, \delta, \Omega)$ aufgefasst werden. Die *Breite* von $\mathcal{A}'$ ist $|Q|$.
Außerdem werden diese abkürzenden Schreibweisen verwendet:

- $[i]$ für $\{0, \ldots, i\}$
- $[i]^{even}$ für die geraden Zahlen aus $[i]$
- $[i]^{odd}$ für die ungeraden Zahlen aus $[i]$

Satz 2 Ein WPAA $\mathcal{A}$ mit $k > 1$ Prioritäten und Breite ρ kann in einen äquivalenten co-WPAA mit $k - 1$ Prioritäten und Breite ρ umgewandelt werden.

Sei $\mathcal{A} = (S, P, \Sigma, q_0, \delta, \Omega)$ gegeben. Definiere $\mathcal{A}' = (S', P', \Sigma, q_0', \delta', \Omega')$ mit

- $S' = S \cup (P \times [2\rho]^{even})$.

- $P' = P \times [2\rho]^{odd}$.
- Falls $q_0 \in S$, dann $q'_0 = q_0$ sonst $q'_0 = (q_0, 2\rho)$.
- $\delta' : (S' \cup P') \times \Sigma \to B^+(S' \cup P')$ mit:
  - Für $q \in S$ und $\sigma \in \Sigma$: $\delta'(q, \sigma) = \text{annotate}_\rho(\delta(q, \sigma)))$
  - Für $(q, i) \in P \times [2\rho]$ und $\sigma \in \Sigma$ ist $\delta((q, i), \sigma)$:
    - $\text{release}_\rho(\delta(q, \sigma), i, q)$ falls $q \notin F$ oder i ist gerade.
    - sonst: *false* (d.h. der Automat hat hier den falschen Rang geraten).
  - Dabei sind annotate_ρ und release_ρ wie folgt definiert.
    - $\text{annotate}_\rho(\delta)$ ersetzt in δ einen Zustand q' durch $\bigvee_{j \in [2\rho]}(q, j)$
    - $\text{release}_\rho(\delta, i, q)$: ersetzt q' durch $\bigvee_{j \in [i]}(q', j)$, falls $q \equiv q'$, sonst wie $\text{annotate}_\rho(\delta)$
    - es gilt $q \equiv q'$ gdw. $q = (\ldots((x, n_0), n_1), \ldots, n_j)$ und $q' = (\ldots((y, n_0), n_1), \ldots, n_j)$ für ein $j \in \mathbb{N}$, wobei x und y Zustände aus dem ursprünglichen APA sind und $n_i \in [2\rho]$ für $i \in \{1 \ldots j\}$. In den Zuständen q und q' sind also in jedem Schritt die gleichen Ränge geraten worden.
- $\Omega' : P' \to \mathbb{N}$ mit
  - $\Omega'((q, i)) = 1$ falls $\Omega(q) = 1$
  - $\Omega'((q, i)) = k - 1$ falls $\Omega(q) = k > 1$

Die duale Konstruktion (Ersetzen von $\bigvee$ durch $\bigwedge$) kann verwendet werden, um einen co-WPAA mit k Prioritäten in einen WPAA mit $k - 1$ Prioritäten umzuwandeln.
Zur Korrektheit der Konstruktion siehe [3].

Satz 3 Zu einem APA $\mathcal{A}$ mit n Zuständen und k Prioritäten gibt es einen äquivalenten WABA $\mathcal{A}'$ mit $O(n^k)$ Zuständen.

Der APA wird als WPAA aufgefasst und mit obigen Satz schrittweise in einen (co-)WPAA $\mathcal{A}'' = (S, P, q_0, \delta, \Omega)$ umgewandelt, in dem es nur noch eine Priorität gibt: Alle Zustände in P haben Priorität 1, Zustände in S haben per Definition keine Priorität. Setze nun $\mathcal{A}' := (S \cup P, \Sigma, q_0, \delta, F)$ mit $F = S$ falls A'' ein co-WPAA ist, $F = P$ sonst.

2.4 WABA nach WAPA

Für den Umgang mit schwachen Automaten, insbesondere der Rückübersetzung, ist es notwendig, die Struktur, die die Schwäche des Automaten belegt, zu kennen. Aus diesem Grund wird im Folgenden ein Verfahren für die Gewinnung eines WAPAs aus einem WABA angegeben.

Satz 4 Zu einem WABA $\mathcal{A}$ mit n Zuständen gibt es einen äquivalenten WAPA $\mathcal{A}'$ mit n Zuständen.

Für $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ setze A' = $(Q, \Sigma, q_0, \delta, \Omega)$ mit einem Ω, so dass gilt:

- $\Omega(q)$ gerade für $q \in F$, ungerade sonst.
- Ω ist normalisiert.

Beweis Ein entsprechendes Ω mit kleinstmöglicher Anzahl an Prioritäten kann leicht durch folgenden Algorithmus gefunden werden: Zuerst wird jedem Zustand q die Priorität 2, falls $q \in F$, und 1 sonst zugewiesen. Danach werden solange die Prioritäten erhöht, bis Ω normalisiert ist. Der Algorithmus terminiert, da die Eingabe ein WABA ist.

Beispiel In Abbildung 3 ist der WAPA dargstellt, der aus dem APA aus Abbildung 2 konstruiert wurde.

3 Zurück von WAPA nach Ltμ

In diesem Abschnitt wird davon ausgegangen, dass $\Sigma = 2^{\mathcal{P}}$ für eine Menge $\mathcal{P}$ gilt und δ den Typ $Q \to \mathit{Delta}(Q, \mathcal{P})$ hat (siehe Abschnitt 1.3).

Satz 5 [4] Zu einem WAPA $\mathcal{A}$ gibt es eine alternierungsfreie Ltμ-Formel $\varphi_{\mathcal{A}}$ mit $L(\mathcal{A}) = L(\varphi_{\mathcal{A}})$.

Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ mit m Prioritäten gegeben.

- Seien $q_1^p, q_2^p, \ldots, q_j^p$ alle Zustände aus Q mit Priorität p.
- Für $p = 1$ bis m: $\Phi_p := \begin{pmatrix} q_1^p & . & \mathrm{tr}_1(\delta(q_1^p)) \\ q_2^p & . & \mathrm{tr}_2(\delta(q_2^p)) \\ \ldots & \ldots & \ldots \\ q_j^p & . & \mathrm{tr}_j(\delta(q_j^p)) \end{pmatrix}$
- Dabei ist $\mathrm{tr}_i : \mathit{Delta}(Q, \mathcal{P}) \to$ Ltμ folgendermaßen definiert:

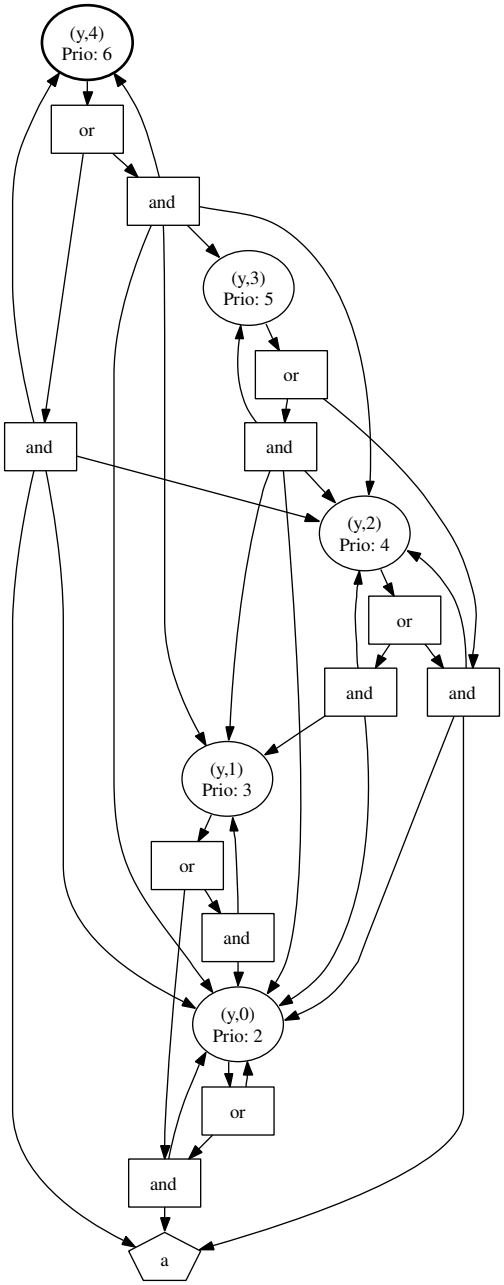

Abbildung 3: WAPA zu $\nu X.\mu Y.(a \wedge \bigcirc X) \vee \bigcirc Y$

- $\mathrm{tr}_i(q_k^j) = \bigcirc q_k^i$ für $j = i$
- $\mathrm{tr}_i(q_k^j) = \begin{cases} \bigcirc \mu q_k^j.\Phi_j & \text{falls } j \text{ ungerade} \\ \bigcirc \nu q_k^j.\Phi_j & \text{falls } j \text{ gerade} \end{cases}$ für $j < i$
- $\mathrm{tr}_i(\bigwedge_t \phi_t) = \bigwedge_t \mathrm{tr}_i(\phi_t)$
- $\mathrm{tr}_i(\bigvee_t \phi_t) = \bigvee_t \mathrm{tr}_i(\phi_t)$
- $\mathrm{tr}_i(p) = p$ und $\mathrm{tr}_i(\neg p) = \neg p$ für $p \in \mathcal{P}$

- Mit Anfangszustand $q_0 = q_k^j$ ist dann $\varphi_{\mathcal{A}} := \begin{cases} \mu q_k^j.\Phi_j & \text{falls } j \text{ ungerade} \\ \nu q_k^j.\Phi_j & \text{falls } j \text{ gerade} \end{cases}$.

Die Konstruktion erfolgt also *bottom-up* beginnend bei den Zuständen mit der kleinsten Priorität. Aus der Definition von tr_i ist ersichtlich, dass $\varphi_{\mathcal{A}}$ alternierungsfrei ist.

Beispiel Abbildung 4 zeigt den Output der Implementation für den WA-PA aus Abbildung 3. `Mus` und `Nus` stehen für die simultanen Fixpunkt-Operatoren μ bzw. ν, `<>` für $\bigcirc$.

Abschließend ergibt sich aus der Kombination der Übersetzungen folgendes Ergebnis:

Satz 6 Für jede Ltμ-Formel gibt es eine äquivalente alternierungsfreie Ltμ-Formel.

Dies steht im Kontrast zum μ-Kalkül auf unendlichen Bäumen: Dort lässt sich (siehe [1]) für jedes $k \in \mathbb{N}$ eine Formel mit Alternierungstiefe k finden, zu der es keine äquivalente Formel mit Alternierungstiefe $k-1$ gibt.

```
 Nus Q6_1.[(Q6_1,<>Mus Q3_1.[(Q3_1,<>Nus Q2_1.[(Q2_1,<>Q2_1 or
<>Q2_1 and a)] and <>Q3_1 or <>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1
and a)] and a)] and <>Mus Q5_1.[(Q5_1,<>Nus Q2_1.[(Q2_1,<>Q2_1
or <>Q2_1 and a)] and <>Nus Q4_1.[(Q4_1,<>Mus Q3_1.[(Q3_1,<>Nus
Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Q3_1 or <>Nus
Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and a)] and <>Nus
Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Q4_1 or <>Q4_1 and
<>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and a)] and a or <>Mus
Q3_1.[(Q3_1,<>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Q3_1
or <>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and a)] and <>Nus
Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Nus Q4_1.[(Q4_1,<>Mus
Q3_1.[(Q3_1,<>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Q3_1
or <>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and a)] and <>Nus
Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Q4_1 or <>Q4_1 and <>Nus
Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and a)] and <>Q5_1)] and <>Nus
Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Nus Q4_1.[(Q4_1,<>Mus
Q3_1.[(Q3_1,<>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Q3_1
or <>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and a)] and <>Nus
Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Q4_1 or <>Q4_1 and
<>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and a)] and <>Q6_1 or
<>Q6_1 and <>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Nus
Q4_1.[(Q4_1,<>Mus Q3_1.[(Q3_1,<>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and
a)] and <>Q3_1 or <>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and a)]
and <>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and <>Q4_1 or <>Q4_1
and <>Nus Q2_1.[(Q2_1,<>Q2_1 or <>Q2_1 and a)] and a)] and a)]
```

Abbildung 4: Ltμ-Formel zu dem WAPA aus Abbildung 3

Literatur

[1] Julian C. Bradfield. The modal -calculus alternation hierarchy is strict. In *CONCUR'96: Concurrency Theory, 7th International Conference*, volume 1119 of *LNCS*, pages 233–246, 1996.

[2] O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. In *5th Israeli Symposium on Theory of Computing and Systems*, pages 147–158. IEEE Computer Society Press, 1997.

[3] O. Kupferman and M.Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proc. 30th ACM Symposium on Theory of Computing*, Dallas, 1998.

[4] M. Lange. Weak automata for the linear time μ-calculus. In R. Cousot, editor, *Proc. 6th Int. Conf. on Verification, Model Checking and Abstract Interpretation, VMCAI'05*, volume 3385 of *LNCS*, pages 267–281, 2005.

[5] David E. Muller and Paul E. Schupp. Alternating automata on infinite objects, determinacy and rabin's theorem. In *Automata on Infinite Words*, pages 100–107, 1984.