EINE MASCHINENSEMIOTISCHE PETRINETZ-ARCHITEKTUR FÜR EIN MENSCHENZENTRIERTES USER-INTERFACE

Peter Klimczak[1]*, Markus Huber*[1]*, Peter beim Graben*[2]*, Günther Wirsching*[3]

[1]*Brandenburgische Technische Universität Cottbus–Senftenberg*
[2]*Bernstein Center for Computational Neuroscience, Berlin*
[3]*Katholische Universität Eichstätt–Ingolstadt*
peter.klimczak@b-tu.de

Kurzfassung: Zunächst möchten wir in das Thema einführen, indem wir kurz auf gegenwärtige Probleme und zukünftige Möglichkeiten von Sprachassistenten eingehen. Es folgen grundlegende Überlegungen zur Semantik und zwei fundamentale Erkenntnisse aus der Biosemiotik und dem Konstruktivismus. Danach fokussieren wir uns auf den Spracherwerbsprozess. Hierbei kommt ein Petrinetz zum Einsatz, das einem elementaren, bereits in früheren Arbeiten vorgestellten, Verstärkungsalgorithmus folgt. Dem schließt sich eine Demonstration des Spracherwerbs anhand eines Beispiels aus dem Smart Car-Bereich an.

1 Einführung

Am 6. August 2020 verlautete eine Schlagzeile des Bayerischen Rundfunks, dass ein Gericht die Bedienung von Touchscreen-Monitoren verboten habe. In seiner Radikalität stimmt das so zwar nicht, aber die Entscheidung des Oberlandesgerichts Karlsruhe hatte der Nutzung von Touchscreens in Autos tatsächlich erneut die Grenzen aufgezeigt: Nach § 23 Abs. 1a der Straßenverkehrsordnung ist die Benutzung eines elektronischen Geräts nämlich nur dann zulässig, wenn das Gerät entweder mittels Sprachsteuerung bedient werden kann oder die Benutzung nur kurze Blicke erforderlich macht. Die letzte Bedingung schränkt den Anwendungsbereich von Touchscreens erheblich ein. Es ist daher zu erwarten, dass mittelfristig Sprachassistenten im Smart Car-Bereich eine dominierende Rolle einnehmen werden.

Allerdings: Trotz zufriedenstellender Spracherkennungsfähigkeiten [1, 2] fehlt aktuellen Sprachassistenzsystemen noch eine geeignete automatische Semantik-Analyse [3]. Das Gleiche gilt für die sinnvolle Repräsentation von pragmatischem Weltwissen [4, 5]. Stattdessen verlangen aktuelle Technologien von ihrem Nutzer, dass er Schlüsselwörter lernt, die für die effektive Bedienung und Arbeit mit der Maschine notwendig sind [6]. Eine solche maschinenzentrierte Herangehensweise kann für den Nutzer jedoch sehr frustrierend sein. Und Frustration wiederum senkt erfahrungsgemäß die Akzeptanz einer Technologie.

Der im Folgenden vorgestellte Ansatz erfordert hingegen gerade nicht, dass der Nutzer bestimmte Schlüsselwörter auswendig lernt, auf die das Gerät mit Hilfe einer Slot-Filling-Semantik reagiert [7]. Stattdessen passt die Dynamische Maschinensemiotik ihr linguistisches Wissen während des Spracherwerbs flexibel an den Nutzer an.

2 Grundüberlegungen

Der deskriptive Sinn einer Äußerung wie „Ich hole gleich Oma ab!“ ergibt sich kompositionell aus der Bedeutung ihrer sprachlichen Konstituenten, Phrasen und Einzelwörter. Der pragmatische Sinn einer Äußerung hängt jedoch von ihrem jeweiligen Kontext und ihrem jeweiligen

Fokus ab. Nehmen wir an, dass der Empfänger der Äußerung ein Radio in einer sprachgestützten Smart-Car-Umgebung sei. In diesem Fall könnte diese Äußerung einfach bedeuten, dass in der unmittelbaren Zukunft keine weitere Radiowiedergabe benötigt wird. Zum Beispiel, da die Großmutter ungerne Musik hört.

Im Rahmen der kompositionalen Semantik würde sich die Bedeutung „Nicht spielen!“ unseres Beispiels aus der Bedeutung der Konstituenten der Äußerung ergeben, zusammen mit pragmatischem Weltwissen sowie individuellem Wissen [8]. Letzteres erlaubt Inferenzen aus der Tatsache, dass der Sprecher bald von seiner Großmutter begleitet wird und dass danach keine weitere Radiowiedergabe mehr nötig sein wird. Beim gegenwärtigen Stand der Sprech- und Sprachtechnologie ist dies jedoch nicht machbar. Trotz erstaunlicher Fortschritte in der Spracherkennungstechnologie sind derzeit weder eine kompositionelle semantische Analyse noch sinnvolle pragmatische Inferenzen durch KI-Methoden möglich (bspw. [9, 10, 11, 5, 12]).

Im Gegensatz dazu ist unser Ansatz von grundlegenden Erkenntnissen der Biosemiotik und des Konstruktivismus (bspw. [13]) inspiriert. Zur Biosemiotik: Überträgt man die ökologische Bedeutungstheorie von von Uexküll [14] auf eine Maschine, so wird die Bedeutung einer (menschlichen) Äußerung allein durch den Handlungsspielraum der Maschine definiert. Diese müssen also nicht die Bedeutungen einzelner Wörter verstehen. Vor allem aber: Maschinen müssen die Bedeutung von Phrasen und Sätzen, die einzelne Wortbedeutungen mit zusätzlichem implizitem Weltwissen kombinieren, nicht verstehen. Zum Konstruktivismus: Nach [13] zielt das verbale Verhalten eines Senders darauf ab, das Verhalten eines Empfängers zu verändern. Skinner [15] zufolge lässt sich die Bedeutung des verbalen Verhaltens des Nutzers als Abbildung von antecedenten auf consequente Handlungszustände interpretieren. Dieses sog. Antecedent-Behaviour-Consequent (ABC)-Schema [15] des verbalen Verhaltens bildet die Grundlage der dynamischen Semantik (bspw. [16, 17, 18]).

Entsprechend haben wir einen einfachen Verstärkungsalgorithmus vorgeschlagen [7, 19], der das Verhalten der Maschine bei jeder nicht-leeren und nicht-akzeptierten Äußerung ändert. Die Bedeutungen von Äußerungen werden während der Semiose gelernt. Bleibt der Nutzer der Maschine dagegen stumm, interpretiert unser Algorithmus diese Art von verbalem Verhalten als Zustimmung zum aktuellen Betriebsmodus. Für das Radio bedeutet dies nur, dass jeder möglichen Äußerung, wie z. B. „Ich hole gleich Oma ab!“ ein Betriebsartwechsel (hier: Radio spielen oder Radio nicht spielen) zuzuordnen ist – wobei der „Wechsel“ auch darin bestehen kann, dass derselbe Zustand wiederholt wird. Diesen Spracherwerbsprozess wollen wir im Folgenden erstmals mithilfe von Petrinetzen abbilden.

3 Spracherwerbsprozess

Der zuvor angedeutete Spracherwerbsprozess kann – basierend auf der dynamischen Semantik – als Lernen von Äußerungs-Bedeutungspaaren (Utterance Meaning-Pairs) formalisiert werden. Ein Äußerungs-Bedeutungspaar ist ein geordnetes Paar aus einer transkribierten Sprachäußerung u und seiner Bedeutung $[\![u]\!]$: $(u, [\![u]\!])$.

Nach Skinner [15] sind Antecedent a und Consequent c epistemische Zustände, wobei sich c als Folge von a und der Anwendung verbalen Verhaltens ergibt. Die Bedeutung einer Äußerung u, d. h. des verbalen Verhaltens, ist dann eine Menge geordneter Antecedent-Consequent-Paare. Das Ziel unseres Algorithmus ist die Konstruktion eines mentalen Lexikons von Äußerungs-Bedeutungspaaren.

Für die Implementierung verwenden wir verallgemeinerte Petrinetze, die nicht nur einfache Kanten besitzen, über die Marken fließen, sondern auch Kanten, die auf Anwesenheit von Marken prüfen, die das Schalten einer Transition verhindern oder alle Marken einer Stelle auf einmal abziehen können (bspw. [20, 21]). Darüber hinaus nutzen wir Multimengen von Ter-

men aus einer algebraischen Spezifikation (bspw. [22, 23]). Sie dienen uns sowohl als Marken als auch als Kantenbeschriftungen [23]. Eine Spezifikation besteht aus einer vielsortigen Struktur aus Sorten, Konstanten- und Operationssymbolen gemeinsam mit einer Variablenmenge und einer Menge von Axiomen. Letztere können von einfachen Gleichungen bis zu booleschen Ausdrücken (bspw. [24, 25]) reichen. Algebraische Spezifikationen sind berechenbar und können je nach eingesetzter Programmiersprache direkt als abstrakte Datentypen umgesetzt werden. Aufgrund des begrenzten Platzes führen wir nur die nötigsten Konzepte formal ein und beschränken uns auf unbeschriftete Petrinetze, um eine Intuition für deren Verhalten zu liefern.

Definition 1 (Petrinetze) *Ein* 6*-Tupel* (P,T,F,I,R,C) *mit einer endlichen Menge* $P \neq \emptyset$ *von* Stellen*, einer endlichen Menge* $T \neq \emptyset$ *von* Transitionen*, mit* $P \cap T = \emptyset$*, einer Menge* $F \subseteq (P \times T) \cup (T \times P)$ *von* Kanten*, einer Menge* $I \subseteq P \times T$ *von* Inhibitorkanten*, einer Menge* $R \subseteq P \times T$ *von* Lesekanten *und einer Menge* $C \subseteq P \times T$ *von* Resetkanten*, mit* $C \cap F = \emptyset$*, heißt* Petrinetz*.*

In graphischen Darstellungen (vgl. Abb. 1) werden Stellen als Kreise, Transitionen als Quadrate, Kanten als Linien mit einer Pfeilspitze, Inhibitorkanten als Linien mit einem Kreis als Kopf, Lesekanten als Linien und Resetkanten als Linien mit einer Raute als Kopf dargestellt.

Für eine Menge X heißt ein Element $m \in \mathbb{N}_0{}^X$, also eine Funktion $m\colon X \to \mathbb{N}_0$ eine *Multimenge (über X)* (vgl. [23]). Mit ϑ wird die *leere Multimenge* notiert. Für ein Petrinetz $N = (P,T,F,I,R,C)$ wird eine Funktion $M \in \mathbb{N}_0{}^P$ *Markierung* genannt und mit M_0 wird die *Anfangsmarkierung* von N bezeichnet. Eine Stelle $p \in P$ heißt *markiert* in einer Markierung M, wenn $M(p) > 0$ gilt. Für eine Transition $t \in T$ bezeichnet ${}^\bullet t = \{p \in P \mid p\ F\ t\}$ ihren *Vorbereich*, ${}^\bullet_\circ t = \{p \in P \mid p\ I\ t\}$ ihren *blockierenden Vorbereich*, ${}^\bullet_- t = \{p \in P \mid p\ R\ t\}$ ihren *aktivierenden Vorbereich* und ${}^\bullet_\diamond t = \{p \in P \mid p\ C\ t\}$ ihren *Lösch-Vorbereich*. Mit $t^\bullet = \{p \in P \mid t\ F\ p\}$ wird ihr *Nachbereich* bezeichnet. Eine Transition $t \in T$ ist in einer Markierung M *aktiviert*, gdw. $M \geq {}^\bullet t \wedge M|_{{}^\bullet_\circ t} = \vartheta \wedge M \geq {}^\bullet_- t$ gilt. Die Folgemarkierung M', die durch *Schalten* der Transition entsteht ist durch $M' = M - M|_{{}^\bullet_\diamond t} - {}^\bullet t + t^\bullet$ gegeben. Da algebraische Petrinetze [23] Multimengen von Termen verarbeiten, ändern sich Verhalten und Formeln im Wesentlichen nicht.

Wir führen nun zunächst die im Algorithmus verwendeten Begriffe formal ein.

Definition 2 (Äußerungs-Bedeutungspaare) *Ein Tupel* $(u, [\![u]\!])$ *mit einer Äußerung* u *aus einer nicht-leeren Menge* U *und deren Bedeutung* $[\![u]\!]$ *– eine partielle Funktion der Antecedents (Vorbedingungen) in die Consequents (Nachbedingungen) –, heißt* Äußerungs-Bedeutungspaar*. Wir nehmen an, dass die endlich vielen Antecedent-Consequent-Paare explizit aufgelistet werden können und, dass die leere Äußerung* ε*, die* Schweigen *repräsentiert, in* U *existiert.*

Wir schreiben ein Äußerungs-Bedeutungspaar $(u, [\![u]\!])$ mit $[\![u]\!] = \{(a_1,c_1),\ldots,(a_n,c_n)\}$ auch als Menge $\{(u,(a_1,c_1)),\ldots,(u,(a_n,c_n))\}$ von *partiellen Äußerungs-Bedeutungspaaren* (pUMPs).

Definition 3 (Maschinenzustände) *Sei* S *eine maximal rekursiv aufzählbare Menge beobachtbarer Zustände der Maschine, mit* $|S| \geq 2$*. Es existiert* $s_0 \in S$ *als Erzeuger von* S *bzgl. der Funktion* $n\colon S \to S$*, d. h. für jedes* $s \in S$ *existiert* $i \in \mathbb{N}_0$ *mit* $n^i(s_0) = s$*, wobei* $n^0 = \mathrm{id}_S$ *gilt. Als dynamisches System betrachtet [26], ist eine solche Maschine* ergodisch*.*

Weiterhin nutzen wir eine Funktion $N\colon S \times S \to S$, die zu einem gegebenen Zustandspaar (a,z) vermöge $i = \min\{j \in \mathbb{N}_0 \mid n^j(z) \neq a\}$ von z aus den nächsten Zustand $n^i(z) \neq a$ berechnet. Wir verwenden die Elemente von S sowohl als Antecedents als auch als Consequents.

Definition 4 (Maschinenbedeutung) *Für eine Äußerung* $u \in U$ *heißt eine partielle Funktion* $[\![u]\!]\colon S \to S$*, mit der Interpretation* $[\![u]\!](a) = c$ *für den aktuellen Zustand der Maschine* $a \in S$ *und dem herzustellenden Zustand der Maschine* $c \in S$*, die* Bedeutung *von* u*. Im Fall* $x = [\![u]\!](x)$ *führt die Äußerung* u *im Zustand* x *nicht zu einem Zustandswechsel; sie* akzeptiert *den aktuellen Antecedent als neuen Consequent.*

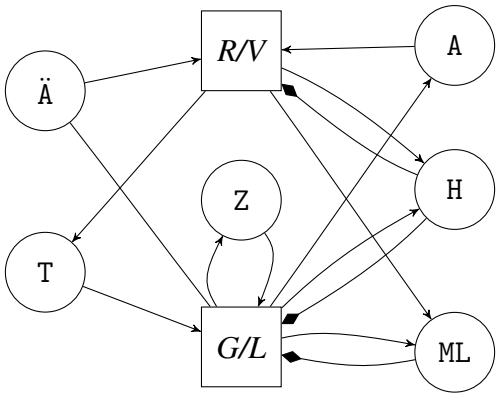

Abbildung 1 – Das Maschinennetz mit dem zweigeteilten Kanal.

Das Modell der Maschine als Petrinetz ist in Abb. 1 zu sehen. Die Namen der Netzelemente sind in deren Innerem notiert. Für die Kommunikation zwischen Benutzer und Maschine wird ein Kanal verwendet, der explizit zweiteilig modelliert ist: ein Äther Ä überträgt die Äußerung vom Benutzer an die Maschine und eine Tafel T überträgt den beobachtbaren Zustand von der Maschine an den Benutzer. Die Transition *G/L* ist für das Generieren bzw. Laden einer Reaktion auf eine Äußerung des Benutzers zuständig (Regeln 1 und 2). Über die Transition *R/V* werden Revision (Regel 3) und Verschieben (Regel 4) bewerkstelligt.

In der Stelle ML, dem mentalen Lexikon, liegen die gelernten partiellen Äußerungs-Bedeutungspaare. Solange sich ein partielles Äußerungs-Bedeutungspaar nicht auf ML befindet, aber von der Maschine benutzt wird, wird es *Hypothese* genannt. Die noch nicht akzeptierten – und evtl. zu revidierenden – Hypothesen liegen auf H, das als *Historie* bezeichnet wird. Die neuste (oder aktuelle) Hypothese befindet sich im Speicher A. Die Tafel T zeigt $a_0 \in S$ als Startzustand der Maschine an, im Speicher Z liegt $s_0 \in S$. Die Stelle A wird auch zur internen Kommunikation zwischen *R/V* und *G/L* verwendet und dient zusammen mit T dazu, dass die beiden Transitionen nur abwechselnd schalten können. Die Synchronisation mit einem evtl. Nutzernetz würde über die Stelle Ä abgewickelt. Auf ihr liegt auch die erste Äußerung $u \in U$ des Nutzers.

1. Es existiert kein passendes $(u,(a,c))$:
   (a) $u = \varepsilon$: Generieren einer akzeptierenden ε-Hypothese $(u,(a,a))$ auf A und Ablage von s_0 auf Z.
   (b) $u \neq \varepsilon$: Sei z der Zustand aus Z und $c = N(a,z)$. Generieren einer modifizierenden Hypothese $(u,(a,c))$ auf A und Ablage von c auf Z.
2. Es existiert ein passendes $(u,(a,c))$: Verschieben des passenden partiellen Äußerungs-Bedeutungspaares $(u,(a,c))$ aus dem mentalen Lexikon ML oder der Historie H als aktuelle Hypothese nach A.
   (a) $(u,(a,c))$ kam aus ML: Ablage von s_0 auf Z.
   (b) $(u,(a,c))$ kam aus H: Ablage von c auf Z.
3. Revidieren der Hypothesen auf H. Sei $(u,(a,c))$ die aktuelle Hypothese auf A.
   (a) $a = c$: Offensichtlich ist eine Revision nicht erforderlich.
   (b) $a \neq c$: Tausche bei jeder Hypothese $(u',(a',c'))$ von H den Zustand c' gegen den Zustand c aus.
4. Verschieben der Hypothesen, Setzen des neuen Maschinenzustands c auf T und Entfernen der Äußerung von Ä. Sei wieder $(u,(a,c))$ die aktuelle Hypothese auf A.

(a) $a = c$: Verschiebe alle Hypothesen von A und H nach ML.

(b) $a \neq c$: Verschiebe die Hypothese von A nach H.

Je nachdem, ob der Nutzer mit dem neuen Zustand c auf der Tafel T zufrieden ist oder nicht, wird er Schweigen oder eine nicht-leere Äußerung über den Äther Ä schicken.

Unser Algorithmus prüft kontinuierlich, ob eine Äußerung des Nutzers gegenüber der Maschine schon zuvor im aktuellen Zustand geäußert wurde. Dazu durchsucht *G/L* das mentale Lexikon ML und die Historie H nach einem passenden partiellen Äußerungs-Bedeutungspaar mit der Funktion $s_p\colon {\mathbb{N}_0}^{U\times(S\times S)} \times U \times S \to {\mathbb{N}_0}^{U\times(S\times S)}$, gegeben durch

$$s_p(m,u,a)((v,(s,s'))) \quad = \quad \begin{cases} m((v,(s,s'))) & \text{für } v = u \wedge s = a \\ 0 & \text{sonst,} \end{cases} \tag{1}$$

mit der alle Einträge auf Übereinstimmung mit der aktuellen Äußerung u und dem Antecedent a überprüft werden. Aufgrund der Algorithmenkonstruktion liefert s_p entweder eine einelementige Menge oder ϑ. Das Gegenstück zu s_p bildet $s_n\colon {\mathbb{N}_0}^{U\times(S\times S)} \times U \times S \to {\mathbb{N}_0}^{U\times(S\times S)}$, gegeben durch

$$s_n(m,u,a)((v,(s,s'))) \quad = \quad \begin{cases} m((v,(s,s'))) & \text{für } v \neq u \vee s \neq a \\ 0 & \text{sonst,} \end{cases} \tag{2}$$

das alle unpassenden Einträge liefert.

Wird nichts gefunden, gilt Regel 1, andernfalls schaltet das System in den Modus 2. Im ersten Fall gehen wir davon aus, dass eine leere Äußerung (1a) die Bedeutung hat, dass der Nutzer den aktuellen Aktionszustand der Maschine befürwortet. Ist die Äußerung jedoch nicht leer, ist der Nutzer mit dem Betriebszustand der Maschine nicht einverstanden (1b) und es muss in einen neuen Zustand gewechselt werden. Regel 2 gilt dann, wenn die Äußerung im erworbenen Lexikon enthalten ist und das Antecedent ihrer Bedeutung mit einem bekannten Consequent verbunden ist. Dann soll der vorgeschriebenen Zustandsübergang ausgeführt werden. Die Transition *G/L* kann an die passende Hypothese mittels $h\colon {\mathbb{N}_0}^{U\times(S\times S)} \times {\mathbb{N}_0}^{U\times(S\times S)} \times U \times S \times S \to U \times (S \times S)$, gegeben durch

$$h(\mathtt{ML},\mathtt{H},u,a,z) \quad = \quad \begin{cases} \begin{cases} (u,(a,a)) & \text{für } u = \varepsilon \\ (u,(a,N(a,z))) & \text{für } u \neq \varepsilon \end{cases} & \text{für } s_p(\mathtt{ML}+\mathtt{H},u,a) = \vartheta \\ (u,(a,c)) & \text{für } s_p(\mathtt{ML}+\mathtt{H},u,a) = (u,(a,c)), \end{cases} \tag{3}$$

gelangen und sie auf A ablegen. Über die Anwendung von s_n können die jeweils unpassenden Einträge wieder nach ML und H zurückgelegt werden. In Z muss der nächste Startzustand für die evtl. Anwendung von N in der nächsten Iteration beiseite gelegt werden. Die Regeln 1a, 1b, 2a und 2b werden über die Funktion $g\colon {\mathbb{N}_0}^{U\times(S\times S)} \times {\mathbb{N}_0}^{U\times(S\times S)} \times U \times S \times S \to S$, gegeben durch

$$g(\mathtt{ML},\mathtt{H},u,a,z) \quad = \quad \begin{cases} \begin{cases} s_0 & \text{für } u = \varepsilon \\ N(a,z) & \text{sonst} \end{cases} & \text{für } s_p(\mathtt{ML}+\mathtt{H},u,a) = \vartheta \\ \begin{cases} c & \text{für } s_p(\mathtt{H},u,a) = (u,(a,c)) \\ s_0 & \text{sonst} \end{cases} & \text{sonst,} \end{cases} \tag{4}$$

abgedeckt, wobei man den ersten und den letzten Fall noch zu einem zusammenfassen könnte.

Regel 3 revidiert eine gesamte Historie nicht akzeptierter Äußerungen. Die Revision beginnt mit der frühesten nicht akzeptierten Äußerung, die unmittelbar auf die jüngste akzeptierte Äußerung folgte. Die Revisionsregel beruht auf der impliziten Annahme, dass der durch die

aktuelle Äußerung induzierte Zustandsübergang der vom Nutzer gewünschte ist. Dabei werden alle „falsch“ gelernten Folgerungen in der Historie durch die aktuelle ersetzt. Dafür bedient sich *R/V* der Funktion $a_r\colon \mathbb{N}_0{}^{U\times(S\times S)} \times (U \times (S \times S)) \to \mathbb{N}_0{}^{U\times(S\times S)}$, gegeben durch

$$a_r(\mathtt{H},(u,(a,c))) \quad = \quad \begin{cases} \vartheta & \text{für } a = c \\ \sum_{(v,(s,s'))\in U\times(S\times S)} \mathtt{H}((v,(s,s')))(v,(s,c)) + (u,(a,c)) & \text{sonst,} \end{cases} \tag{5}$$

die die Regeln 3a und 3b, aber auch schon Regel 4b umsetzt.

Nach einer akzeptierten Äußerung wird die gesamte Historie gemäß Regel 4a ins Lexikon befördert; alle neuen/revidierten Folgerungen wurden erworben. Dazu wird von *R/V* die letzte Funktion $a_s\colon \mathbb{N}_0{}^{U\times(S\times S)} \times (U \times (S \times S)) \to \mathbb{N}_0{}^{U\times(S\times S)}$, gegeben durch

$$a_s(\mathtt{H},(u,(a,c))) \quad = \quad \begin{cases} (u,(a,c)) + \mathtt{H} & \text{für } a = c \\ \vartheta & \text{sonst,} \end{cases} \tag{6}$$

eingesetzt.

4 Demonstration

Ein mögliches Szenario würde wie in Tabelle 1 aussehen, die wir über unsere Implementierung generiert haben. Wir nehmen der Einfachheit halber an, dass der initiale Arbeitszustand des Radios ein eingeschaltetes Radio darstellt. Der passende erste Antecendet P ist in der Spalte a unter Iteration 1 (Spalte #) zu finden. Die Spalten a bis c stellen den Inhalt der Stelle `A` des Petrinetzes dar, nachdem *G/L* geschaltet hat und bevor *R/V* schaltet. Die Spalten `H` und `ML` sind der Inhalt der entsprechenden Stellen, nachdem auch *R/V* geschaltet hat.

Iteration 1: Der Fahrer genehmigt den anfänglichen Betriebszustand durch eine leere Äußerung. Das heißt, der Fahrer schweigt, sodass die initiale Äußerung ε lautet. Regel 1a kommt zum Tragen, das System ändert seinen Betriebszustand nicht. Als Consequent erhalten wir den Arbeitszustand P. Das erste gelernte pUMP ist $(\varepsilon,(P,P))$.

Iteration 2: Bei der nächsten Iteration wird der Antecedent dynamisch mit dem Consequent-Wert aus dem vorherigen Zeitschritt initialisiert. In unserem speziellen Fall haben wir den Arbeitszustand $a = P$. Nun ändert der Nutzer seine Meinung und wünscht einen Wechsel des Betriebszustands, indem er sagt „Ich hole gleich Oma ab!“. Gemäß der Trainingsregel 1b, bei der eine nicht-leere Äußerung den Wunsch des Nutzers nach einem Wechsel des Betriebszustands anzeigt, ist der neue Folgesatz $\neg P$. Das resultierende pUMP ist dann $(\text{Ich hole gleich Oma ab!},(P,\neg P))$.

Iteration 3: Entsprechend dem letzten Wechsel ist das Antecedent nun $\neg P$. Dieser Betriebszustand wird nun durch Schweigen bestätigt. Die leere Äußerung ε muss verarbeitet werden. Da die Äußerung dem Radio zwar bereits bekannt ist, jedoch das Antecedent $\neg P$ in der bisherigen Bedeutungsdefinition von ε nicht gegeben ist, folgt wieder Regel 1a: Der Consequent ist derselbe wie der Antecedent: $\neg P$. Der Betriebszustand wird fortgesetzt und das neu gewonnene pUMP ist dann $(\varepsilon,(\neg P,\neg P))$.

Iteration 4: Es liegt weiterhin $\neg P$ als Antecedent vor. Vor lauter Freude, seine Oma bald zu sehen, ist der Fahrer leicht geistesabwesend und äußert noch einmal „Ich hole gleich Oma ab!“. Die Äußerung ist dem Radio zwar bekannt, doch da das Antecedent $\neg P$ in der bisherigen Bedeutungsdefinition der Äußerung nicht vorkommt, ändert sich der Betriebszustand des Geräts und eine zusätzliche Bedeutung wird gelernt: $(\text{Ich hole gleich Oma ab!},(\neg P,P)\star)$. Allerdings: Die neu erlernte Bedeutung ist pragmatisch nicht angemessen, da entgegen der Intention des Nutzers. Auf dieses „falsche“ Wissen verweist das Sternchen $\star$ hinter der Bedeutung.

Tabelle 1 – Mithilfe des Netzes aus Abb. 1 generiertes Beispiel.

#	a	u	c	H	ML
1	P	ε	P		$(\varepsilon, \{(P,P)\})$
2	P	Ich hole gleich Oma ab!	$\neg P$	$(\mathrm{I}\ldots, \{(P,\neg P)\})$	$(\varepsilon, \{(P,P)\})$
3	$\neg P$	ε	$\neg P$		$(\varepsilon, \{(P,P),(\neg P,\neg P)\})$ $(\mathrm{I}\ldots, \{(P,\neg P)\})$
4	$\neg P$	Ich hole gleich Oma ab!	P	$(\mathrm{I}\ldots, \{(\neg P,P)\star\})$	$(\varepsilon, \{(P,P),(\neg P,\neg P)\})$ $(\mathrm{I}\ldots, \{(P,\neg P)\})$
5	P	Nein!	$\neg P$	$(\mathrm{I}\ldots, \{(\neg P,\neg P)\})$ $(\mathrm{N}\ldots, \{(P,\neg P)\})$	$(\varepsilon, \{(P,P),(\neg P,\neg P)\})$ $(\mathrm{I}\ldots, \{(P,\neg P)\})$
6	$\neg P$	ε	$\neg P$		$(\varepsilon, \{(P,P),(\neg P,\neg P)\})$ $(\mathrm{I}\ldots, \{(P,\neg P),(\neg P,\neg P)\})$ $(\mathrm{N}\ldots, \{(P,\neg P)\})$
7	$\neg P$	Gut gemacht!	P	$(\mathrm{G}\ldots, \{(\neg P,P)\star\})$	$(\varepsilon, \{(P,P),(\neg P,\neg P)\})$ $(\mathrm{I}\ldots, \{(P,\neg P),(\neg P,\neg P)\})$ $(\mathrm{N}\ldots, \{(P,\neg P)\})$
8	P	Nein!	$\neg P$	$(\mathrm{G}\ldots, \{(\neg P,\neg P)\})$ $(\mathrm{N}\ldots, \{(P,\neg P)\})$	$(\varepsilon, \{(P,P),(\neg P,\neg P)\})$ $(\mathrm{I}\ldots, \{(P,\neg P),(\neg P,\neg P)\})$

Iteration 5: Aufgrund der ungewollten Änderung des Betriebszustands äußert der Nutzer ein „Nein!“. Da die Äußerung vorher nicht geäußert wurde, wird der Betriebszustand gemäß Regel 1b auf $\neg P$ geändert. Außerdem lernt das Gerät ein neues pUMP: $(\text{Nein!}, (P,\neg P))$. Allerdings: „Nein!“ folgt unmittelbar auf die nicht-leere Äußerung „Ich hole gleich Oma ab!“, weshalb die Revisionsregel 3 zu berücksichtigen ist. Entsprechend wird der Consequent in der zuletzt gelernten Bedeutungsdefinition von „Ich hole gleich Oma ab!“ durch den aktuell wie auch damals gewünschten Consequent $\neg P$ ersetzt.

Iteration 6: Nehmen wir noch einmal an, dass der Nutzer den aktuellen Arbeitszustand durch Schweigen bestätigt. Nun ist dem Radio nicht nur die Äußerung bekannt, sondern auch deren Bedeutung. Das heißt: Das entsprechende Antecedent-Consequent-Paar ist Teil des mentalen Lexikons. Somit tritt Regel 2 in Kraft und der Arbeitszustand ändert sich entsprechend der gelernten Bedeutung nicht.

Iteration 7: Der Fahrer ist mit seinem Radio so zufrieden, dass er es mit der Äußerung „Gut gemacht!“ zu loben versucht. Da die Äußerung dem Radio noch nicht bekannt ist, kommt Regel 1b zum Tragen. Daher wird der Arbeitszustand – aus Sicht des Fahrers – ungewollt auf P geändert. Außerdem lernt das Radio fälschlicherweise ein neues pUMP: $(\text{Gut gemacht!}, (\neg P,P)\star)$.

Iteration 8: Aufgrund der unbeabsichtigten Änderung des Betriebszustands äußert der Nutzer erneut „Nein!“. Da die Äußerung sowie ihre jeweilige Bedeutung dem System bereits bekannt sind, tritt Regel 2 in Kraft. Entsprechend der bereits erworbenen Bedeutung von „Nein!“ wechselt das Radio seinen Arbeitszustand auf $\neg P$. Weiterhin: Da „Nein!“ unmittelbar auf die nicht-leere Äußerung „Gut gemacht!“ folgt, muss die Revisionsregel 3 berücksichtigt werden. Entsprechend wird der Consequent in der gelernten Bedeutungsdefinition von „Gut gemacht!“ durch den aktuell wie auch zuvor gewünschten Consequent $\neg P$ ersetzt.

Fazit: In nur acht Iterationen konnte die Bedeutung sowohl von Stille als auch von „Ich hole gleich Oma ab!“ vollständig gelernt werden. Auch ein Teilaspekt der Bedeutung von „Nein!“ wurde erworben. Außerdem wurde gezeigt, dass falsch gelernte Bedeutungen revidiert werden können. Oder mit anderen Worten: Der Algorithmus kann mit irrationalem, also menschlichem, Verhalten umgehen. Und nicht zuletzt: Der Algorithmus ist auch imstande affirmative Bedeutungen von nicht-leeren Äußerungen zu lernen.

5 Zusammenfassung und Schlussfolgerungen

Unser Trainingsalgorithmus basiert auf zwei intuitiven, jedoch entscheidenden Annahmen zum Nutzerverhalten: Erstens billigt das Schweigen des Nutzers die aktuelle Betriebsart, und zweitens zeigt verbales Verhalten an, dass der Nutzer mit einer aktuellen Betriebsart nicht einverstanden ist. Dementsprechend haben wir einen Weg vorgeschlagen, der unpassende Bedeutungen bei nachfolgenden neuen Eingaben kontinuierlich revidiert.

Der vorgeschlagene Algorithmus ist überaus robust, weil jede Benutzeräußerung als maschinenrelevant angesehen und mit einer entsprechenden Bedeutung assoziiert wird. Dabei wird indes ein größtenteils harmloser, rationaler und kooperativer Benutzer vorausgesetzt, der die Grice'schen Konversationsmaximen beachtet [4]. Wenn dies nicht der Fall ist, treten Konvergenzprobleme auf, die sich folgendermaßen illustrieren lassen. Angenommen, der Benutzer ist ein Schwätzer, der unentwegt auf die Maschine einredet und dadurch die Grice'schen Maximen der Quantität und der Relevanz verletzt [4]. Dann wird niemals ein Maschinenzustand durch eine Benutzeräußerung akzeptiert, sodass letztlich jede jemals gemachte Äußerung durch die gerade aktuell letzte revidiert werden muss. Ebenso lösen fortwährende Verletzungen der Qualitätsmaxime durch inkonsistente Nutzeräußerungen periodische Revisionen aus, sodass niemals konsistente Bedeutungen entstehen können.

Durch die Modellierung als Petrinetz stehen uns diverse mathematische Analysemethoden (bspw. [27]) für den Algorithmus zur Verfügung. Unter anderem können wir die Lösung der angesprochenen Konvergenzprobleme angehen, indem Benutzer, die die Grice'schen Konversationsmaximen unterschiedlich stark verletzen, ebenfalls durch Petrinetze modelliert werden. Dadurch kann das Verhalten eines Gesamtsystems aus Maschine und Nutzer unter diesen Gesichtspunkten eingehend studiert werden.

Auf diese Weise ist ein intuitives und universelles User-Interface geschaffen. Intuitiv ist es, weil nicht der Mensch die Sprache der Maschine erlernt, sondern umgekehrt. Universell ist das Interface, weil die Maschine in der Lage ist, die individuelle Sprache eines jeden Nutzers zu lernen. Das schließt sogar Idiolekte mit ein.

Unser Petrinetzalgorithmus beruht wesentlich auf ergodischen Markovketten (Definition 3, s. bspw. [26]). Daher ist die zukünftige Erweiterung vom einfachen Trial-and-Error-Aktualisierungsverfahren auf Markov-Entscheidungsprozesse unaufwändig zu leisten. Für den kommerziellen Einsatz könnte es zudem sinnvoll sein, das System mit einem vortrainierten mentalen Lexikon auszuliefern, das die wahrscheinlichsten Äußerungen in ihren stereotypen pragmatischen Kontexten enthält. Deren Bedeutungen müssten dennoch im individuellen Sprachprozess revidierbar sein, um die Adaptivität des Systems sicher zu stellen.

Literatur

[1] BÖCK, R., O. EGOROW, J. HÖBEL-MÜLLER, A. F. REQUARDT, I. SIEGERT, und A. WENDEMUTH: *Anticipating the user: acoustic disposition recognition in intelligent interactions*, S. 203–233. Springer, Cham, 2019.

[2] MESNIL, G., Y. DAUPHIN, K. YAO, Y. BENGIO, L. DENG, D. HAKKANI-TUR, X. HE, L. HECK, G. TUR, D. YU, und G. ZWEIG: *Using recurrent neural networks for slot filling in spoken language understanding*. *IEEE Transactions on Audio, Speech and Language Processing*, 23(3), S. 530–539, 2015.

[3] ALLEN, J.: *Learning a lexicon for broad-coverage semantic parsing*. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, S. 1–6. 2014.

[4] GRICE, P.: *Studies in the Way of Words*. Havard University Press, Cambridge (MA), 1989.

[5] BENZ, A.: *On Bayesian pragmatics and categorical predictions*. *Zeitschrift für Sprachwissenschaft*, 35(1), S. 45, 2016.

[6] GUO, D., G. TUR, W. YIH, und G. ZWEIG: *Joint semantic utterance classification and slot filling with recursive neural networks*. In *Proceedings of the IEEE Spoken Language Technology Workshop (SLT)*, S. 554–559. 2014.

[7] KLIMCZAK, P. und G. WIRSCHING: *Maschinensemiotik*. *Zeitschrift für Semiotik*, 40(3-4), S. 3–19, 2018.

[8] ALLEN, J. F.: *Natural language processing*. In *Encyclopedia of Computer Science*, S. 1218–1222. Wiley, Chichester (UK), 2003.

[9] ALLEN, J. und C. M. TENG: *Putting semantics into semantic roles*. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, S. 235–244. 2018.

[10] GALESCU, L., C. M. TENG, J. ALLEN, und I. PERERA: *Cogent: a generic dialogue system shell based on a collaborative problem solving model*. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, S. 400–409. 2018.

[11] PERERA, I., J. F. ALLEN, L. GALESCU, C. M. TENG, M. BURSTEIN, S. FRIEDMAN, D. MCDONALD, und J. RYE: *Natural language dialogue for building and learning models and structures*. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[12] FRANKE, M. und G. JÄGER: *Probabilistic pragmatics, or why Bayes' rule is probably important for pragmatics*. *Zeitschrift für Sprachwissenschaft*, 35(1), S. 3, 2016.

[13] MATURANA, H. und F. VARELA: *The Tree of Knowledge*. Shambhala Press, Boston, 1998.

[14] VON UEXKÜLL, J.: *The theory of meaning*. *Semiotica*, 42(1), S. 25–79, 1982.

[15] SKINNER, B. F.: *Verbal Behavior*. Appleton-Century-Crofts, New York, 1957. Reprinted 2015.

[16] GÄRDENFORS, P.: *Knowledge in Flux. Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge (MA), 1988.

[17] KRACHT, M.: *Dynamic semantics*. *Linguistische Berichte*, Sonderheft X, S. 217–241, 2002.

[18] BEIM GRABEN, P.: *Order effects in dynamic semantics*. *Topics in Cognitive Science*, 6(1), S. 67–73, 2014.

[19] KLIMCZAK, P., G. WIRSCHING, und P. BEIM GRABEN: *Machine semiotics*. arXiv:2008.10522 [cs.CL], 2020.

[20] ARAKI, T. und T. KASAMI: *Some decision problems related to the reachability problem for petri nets*. *Theor. Comput. Sci.*, 3(1), S. 85–104, 1976.

[21] KLEIJN, H. und M. KOUTNY: *Process semantics of general inhibitor nets*. *Information and Computation*, 190(1), S. 18–69, 2004.

[22] EHRIG, H. und B. MAHR: *Fundamentals of Algebraic Specification 1*, Bd. 6 d. Reihe *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, Heidelberg, 1985.

[23] REISIG, W.: *Petri nets and algebraic specifications*. *Theor. Comput. Sci.*, 80(1), S. 1–34, 1991.

[24] BROY, M., W. DOSCH, H. PARTSCH, P. PEPPER, und M. WIRSING: *Existential quantifiers in abstract data types*. In H. A. MAURER (Hrsg.), *Colloquium on Automata, Languages and Programming 1979*, Bd. 71 d. Reihe *Lecture Notes in Computer Science*, S. 73–87. Springer, 1979.

[25] CERIOLI, M., T. MOSSAKOWSKI, und H. REICHEL: *From total equational to partial first-order logic*. In E. ASTESIANO, H.-J. KREOWSKI, und B. KRIEG-BRÜCKNER (Hrsg.), *Algebraic Foundations of Systems Specification*, IFIP State-of-the-Art Reports, Kap. 3, S. 31–104. Springer, Berlin, Heidelberg, 1999.

[26] LUNZE, J.: *Ereignisdiskrete Systeme: Modellierung und Analyse dynamischer Systeme mit Automaten, Markovketten und Petrinetzen*. De Gruyter Oldenbourg, Berlin, 2017.

[27] REISIG, W.: *Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien*. Leitfäden der Informatik. Vieweg+Teubner Verlag, 2010.