Von Scheme zu DeepProbLog

Die ARS als methodologischer Bauplan für moderne neuro-symbolische Programmierung

Paul Koop

1994–2026

Zusammenfassung

Dieser Beitrag zeichnet die methodologische Kontinuität von frühen Implementierungen der Algorithmisch Rekursiven Sequenzanalyse (ARS) in Scheme, Pascal und Lisp (1992–1994) zu zeitgenössischen neuro-symbolischen Programmierframeworks wie DeepProbLog (2018) nach. Ich argumentiere, dass die ARS bereits die Kernprinzipien neuro-symbolischer Integration – Mustererkennung (System 1), regelbasiertes Schließen (System 2), probabilistische Unsicherheitsquantifizierung und Erklärbarkeit durch Design – Jahrzehnte vor der Prägung des Begriffs "neuro-symbolische KI"verkörperte. Der Beitrag rekonstruiert zunächst die proto-neuro-symbolische Architektur der ARS, stellt dann DeepProbLog als modernes Framework vor, das ähnliche Prinzipien mit größerer Skalierbarkeit implementiert, und demonstriert schließlich eine DeepProbLog-Implementierung des klassischen ARS-Verkaufsgesprächs-Korpus. Die Synthese zeigt, dass die ARS einen methodologischen Bauplan liefert, den DeepProbLog technisch instanziieren kann. Ich schließe mit einer Forschungsagenda für die Integration der methodologischen Strenge der ARS mit der Rechenleistung von DeepProbLog.

Inhaltsverzeichnis

1 Einleitung: Von Lisp zu DeepProbLog

Die Algorithmisch Rekursive Sequenzanalyse (ARS), dokumentiert in den frühen Jupyter-Notebooks und Codedateien von 1992–1994, stellt einen der frühesten systematischen Versuche dar, Mustererkennung mit regelbasiertem Schließen in der Analyse sequenzieller sozialer Interaktionen zu integrieren. Die drei Kernimplementierungen –

- **Induktor in Scheme**: Induktion probabilistischer kontextfreier Grammatiken (PCFG) aus Terminalzeichenketten durch Übergangszählung,
- **Parser in Pascal**: Validierung der Wohlgeformtheit von Sequenzen mittels eines Chart-Parsers,
- **Transduktor in Lisp**: Generierung neuer Sequenzen aus der induzierten Grammatik,

– verkörpern gemeinsam das, was heute als **neuro-symbolische KI** bezeichnet wird. Sie kombinieren datengetriebene Musterentdeckung (der Induktor) mit symbolischer Regelanwendung (der Parser und Transduktor) und quantifizieren Unsicherheit durch probabilistische Gewichte.

In den drei dazwischenliegenden Jahrzehnten hat das Feld anspruchsvollere Frameworks für neuro-symbolische Integration entwickelt. Eines der prominentesten ist **DeepProbLog** (Manhaeve et al., 2018), das die probabilistische Logikprogrammiersprache ProbLog um neuronale Prädikate erweitert, die von tiefen Netzwerken gelernt werden. DeepProbLog erlaubt es Benutzern, symbolische Regeln mit Wahrscheinlichkeiten zu definieren, während neuronale Netze die Wahrscheinlichkeiten von Grundtatsachen aus Daten lernen.

Dieser Beitrag leistet drei Dinge:

1. Er rekonstruiert die ARS-Architektur als **proto-neuro-symbolisches** System und bildet ihre Komponenten auf zeitgenössische neuro-symbolische Konzepte ab.
2. Er stellt DeepProbLog als modernes Framework vor, das dieselben Prinzipien mit größerer Skalierbarkeit und neuronaler Integration implementiert.
3. Er präsentiert eine **DeepProbLog-Implementierung** des klassischen ARS-Verkaufsgesprächs-Korpus und demonstriert, wie die ARS-Methodologie in ein modernes neuro-symbolisches Framework portiert werden kann.

Die übergreifende These ist, dass **die ARS einen methodologischen Bauplan liefert, den DeepProbLog technisch instanziieren kann**. Die beiden Ansätze sind keine Konkurrenten, sondern Komplemente: Die ARS trägt methodologische Strenge und interpretative Verankerung bei; DeepProbLog trägt Skalierbarkeit und neuronales Lernen bei.

2 Die ARS-Architektur als proto-neuro-symbolisches System

2.1 Drei Komponenten, drei kognitive Funktionen

Die drei ARS-Implementierungen lassen sich auf die von Kahneman (Kahneman, 2011) popularisierte und von der neuro-symbolischen KI-Forschung übernommene Unterscheidung zwischen System 1 und System 2 (Marcus, 2020) abbilden:

Tabelle 1: ARS-Komponenten als kognitive Systeme

Komponente	**Kognitive Funktion**	**Neuro-symbolische Abbildung**
Induktor (Scheme)	Mustererkennung, Übergangszählung	System 1 (Lernen aus Daten)
Parser (Pascal)	Strukturelle Validierung, Wohlgeformtheitsprüfung	System 2 (Regelanwendung)
Transduktor (Lisp)	Generative Regelanwendung	System 2 (symbolische Generierung)
Multiagent (Python)	Rollenzuweisung, Interaktion	Hybrid (Entscheidungsbaum + Grammatik)

2.2 Die probabilistische Grammatik als neuro-symbolische Schnittstelle

Die induzierte probabilistische kontextfreie Grammatik (PCFG) dient als zentrale neuro-symbolische Schnittstelle:

```
(KBG -> . VBG)
(VBG -> . KBBd)
```

```
(KBBd -> . VBBd)
(VBBd -> . KBA)
(KBA -> . VBA)
(VBA -> . KBBd) (VBA -> . KAE)
(KAE -> . VAE)
(VAE -> . KAE) (VAE -> . KAA)
(KAA -> . VAA)
(VAA -> . KAV)
(KAV -> . VAV)
```

Jede Produktionsregel hat eine Wahrscheinlichkeit (in dieser vereinfachten Grammatik implizit 1.0, in der vollständigen Implementierung gewichtet nach empirischen Häufigkeiten). Die Grammatik ist gleichzeitig:

- **Symbolisch**: Regeln sind explizit, inspizierbar und falsifizierbar.
- **Probabilistisch**: Regelanwendungen haben Wahrscheinlichkeiten basierend auf empirischen Häufigkeiten.
- **Generativ**: Neue Sequenzen können durch Anwendung der Regeln erzeugt werden.
- **Verifizierbar**: Der Parser kann prüfen, ob eine Sequenz wohlgeformt ist.

Diese vier Eigenschaften sind genau das, was zeitgenössische neuro-symbolische Frameworks anstreben.

2.3 Das Multiagentensystem als neuro-symbolischer Prototyp

Das Python-Multiagentensystem (Zellen 29–33 im Notebook) ist besonders aufschlussreich:

```
# Entscheidung  ber  die Rollenverteilung basierend auf Ware
   und Zahlungsmittel
if agent_k_ware > agent_v_ware:
    agent_k_role = 'K ufer'
    agent_v_role = 'Verk ufer'
else:
    agent_k_role = 'Verk ufer'
    agent_v_role = 'K ufer'
```

Listing 1: Multiagenten-Rollenzuweisung

Dieser Entscheidungsbaum ist eine **symbolische Regel** (System 2), die Agentenrollen basierend auf einem einfachen Muster (System 1: Vergleich zweier Zahlen) bestimmt. Die anschließende Interaktion folgt der probabilistischen Grammatik. Dies ist eine hybride Architektur: Die Rollenzuweisung ist deterministisch und regelbasiert; die Dialoggenerierung ist probabilistisch und grammatikbasiert.

Die ARS antizipiert damit das **Neural | Symbolic**-Muster in Kautz' Taxonomie (Kautz, 2020): neuronale (oder heuristische) Wahrnehmung bestimmt symbolische Rollen; symbolisches Schließen (die Grammatik) steuert das anschließende Verhalten.

3 DeepProbLog: Ein modernes neuro-symbolisches Framework

3.1 Was DeepProbLog ist

DeepProbLog (Manhaeve et al., 2018) erweitert die probabilistische Logikprogrammiersprache ProbLog um **neuronale Prädikate**. Ein neuronales Prädikat ist ein Prädikat, dessen Wahrheitswahrscheinlichkeit von einem neuronalen Netz berechnet wird. Ein neuronales Prädikat 'digit(image, d)' könnte beispielsweise die Wahrscheinlichkeit repräsentieren, dass ein Bild die Ziffer 'd' zeigt.

DeepProbLog-Programme bestehen aus:

- **Fakten**: Grundatome mit Wahrscheinlichkeiten (z.B. '0.5::edge(a,b)').
- **Regeln**: Logische Implikationen (z.B. 'path(X,Y) :- edge(X,Y)').
- **Neuronale Prädikate**: Durch neuronale Netze definierte Prädikate.
- **Anfragen**: Zu beantwortende probabilistische Fragen.

Die Inferenz in DeepProbLog berechnet die Wahrscheinlichkeit einer Anfrage gegeben das Programm und die Ausgaben des neuronalen Netzes. Lernen aktualisiert die neuronalen Netzgewichte, um die Likelihood der beobachteten Daten zu maximieren.

3.2 Abbildung von ARS-Konzepten auf DeepProbLog

Tabelle 2: Abbildung von ARS auf DeepProbLog

ARS-Konzept	**DeepProbLog-Konzept**	**Erklärung**
Terminalzeichen	Grundfakten	‘KBG‘, ‘VBG‘, ‘KBBd‘, etc.
Produktionsregeln	Logische Regeln	‘next(X,Y) :- transition(X,Y)‘
Übergangswahrscheinlichkeiten	Fakten wahrscheinlichkeiten	‘0.8::next(KBG, VBG)‘
Induktor (Übergangszählung)	Neuronales Prädikatenlernen	Aus Daten gelernt
Parser (Wohlgeformtheit)	Beweissuche	Anfrage ‘next(KBG, VBG)‘
Transduktor (Generierung)	Sampling aus Verteilung	‘sample(next(Start, X))‘
Multiagentenrollen	Probabilistische Entscheidungsregeln	Rollenzuweisung mit Wahrscheinlichkeit

3.3 Warum DeepProbLog ein natürlicher Nachfolger der ARS ist

DeepProbLog bewahrt die wichtigsten methodologischen Tugenden der ARS:

1. **Erklärbarkeit**: Regeln sind explizit und inspizierbar.
2. **Probabilistische Unsicherheit**: Wahrscheinlichkeiten quantifizieren Unsicherheit.
3. **Generative Kapazität**: Neue Sequenzen können erzeugt werden.
4. **Verifizierbarkeit**: Anfragen können geprüft werden.

Aber es fügt Fähigkeiten hinzu, die der ARS fehlen:

1. **Neuronale Integration**: Neuronale Netze können Wahrscheinlichkeiten aus Rohdaten (Bildern, Text, Audio) lernen, nicht nur aus vorkodierten Kategorien.

2. **Skalierbarkeit**: DeepProbLog kann große Datensätze durch stochastischen Gradientenabstieg verarbeiten.

3. **Kontinuierliches Lernen**: Das neuronale Netz kann inkrementell aktualisiert werden, wenn neue Daten eintreffen.

4. **Tiefes Merkmalslernen**: Neuronale Netze können automatisch relevante Merkmale entdecken und reduzieren so den Bedarf an manueller Kategorienbildung.

4 DeepProbLog-Implementierung des ARS-Korpus

4.1 Die Terminalzeichen als probabilistische Fakten

Der erste Schritt ist die Kodierung der ARS-Terminalzeichen als probabilistische Fakten. Die Übergangswahrscheinlichkeiten werden aus dem Korpus gelernt:

```
% DeepProbLog-Implementierung der ARS-Grammatik f r Verkaufsgespr che
% Basierend auf dem Aachener Markt-Transkript (1994)

% Terminalzeichen als Pr dikate
predicate(kbg/0). predicate(vbg/0). predicate(kbbd/0). predicate(vbbd/0).
predicate(kba/0). predicate(vba/0). predicate(kae/0). predicate(vae/0).
predicate(kaa/0). predicate(vaa/0). predicate(kav/0). predicate(vav/0).

% Neuronale Pr dikate f r bergangswahrscheinlichkeiten
nn(transition, [in:symbol, out:symbol]) :: neural_network.

% Regeln: Wohlgeformte Sequenzen folgen bergngen
% Startsymbol ist KBG (Kundenbegr ung)
next(S) :- transition(start, S).

% Rekursive Regel f r Sequenzen der L nge > 1
next([A,B|Rest]) :-
    transition(A, B),
    next([B|Rest]).

```

```
% Anfrage: Wahrscheinlichkeit, dass eine gegebene Sequenz
    wohlgeformt ist
query(well_formed(Sequence)) :- next(Sequence).

% Generierung: Sample einer wohlgeformten Sequenz
sample(well_formed(S)) :- next(S).
```

Listing 2: DeepProbLog-Kodierung der ARS-Grammatik

4.2 Lernen der Übergangswahrscheinlichkeiten aus Daten

Das neuronale Netz für Übergangswahrscheinlichkeiten kann auf den aus dem ARS-Korpus extrahierten Terminalzeichensequenzen trainiert werden. Das Korpus lautet:

`KBG VBG KBBd VBBd KBA VBA KBBd VBBd KBA VBA KAE VAE KAE VAE KAA VAA KAV VAV`

In DeepProbLog können wir dies als Trainingsdaten kodieren:

```
% Trainingsbeispiele: beobachtete    bergnge
train(transition(kbg, vbg), true).
train(transition(vbg, kbbd), true).
train(transition(kbbd, vbbd), true).
train(transition(vbbd, kba), true).
train(transition(kba, vba), true).
train(transition(vba, kbbd), true).
train(transition(vba, kae), true).
train(transition(kae, vae), true).
train(transition(vae, kae), true).
train(transition(vae, kaa), true).
train(transition(kaa, vaa), true).
train(transition(vaa, kav), true).
train(transition(kav, vav), true).

% Negative Beispiele (optional)
train(transition(kbg, kbbd), false).
train(transition(vbg, vbg), false).
```

Listing 3: Kodierung der Trainingsdaten

Das neuronale Netz lernt, beobachteten Übergängen hohe Wahrscheinlichkeiten und unbeobachteten niedrige Wahrscheinlichkeiten zuzuweisen. Nach dem Training approximiert das Netz die empirischen Übergangshäufigkeiten.

4.3 Das Multiagentensystem in DeepProbLog

Das Multiagentensystem kann als eine Menge probabilistischer Regeln mit Rollenzuweisung implementiert werden:

```
% Agentenrollen basierend auf Waren- und Geldausstattung
% Diese k nnten durch neuronale Netze aus Daten gelernt
   werden
nn(role, [in:goods, in:money, out:role]) :: role_network.

% Rollenzuweisungsregel
agent_role(A, buyer) :- goods(A, G), money(A, M), role(G, M,
   buyer).
agent_role(A, seller) :- goods(A, G), money(A, M), role(G, M,
    seller).

% Interaktionsregeln basierend auf Rollen
utterance(A, kb) :- agent_role(A, buyer), start_turn.
utterance(A, vg) :- agent_role(A, seller), previous_utterance
   (_, kb).

% Grammatikbasierte Dialogfortsetzung
next_utterance(A, Sym) :-
    previous_utterance(_, PrevSym),
    transition(PrevSym, Sym),
    agent_role(A, _).

% Anfrage: Wahrscheinlichkeitsverteilung der n chsten
     uerung
query(next_utterance(seller, Sym)).
```

Listing 4: Multiagentensystem in DeepProbLog

4.4 Erklärbarkeit in DeepProbLog

Ein entscheidender Vorteil von DeepProbLog gegenüber rein neuronalen Netzen ist die **Erklärbarkeit durch Design**. Für jede Anfrage kann DeepProbLog einen Beweisbaum liefern:

```
| ?- explain(well_formed([KBG, VBG, KBBd, VBBd, KBA])).

Beweis:
```

```
1. well_formed([KBG, VBG, KBBd, VBBd, KBA])
       next([KBG, VBG, KBBd, VBBd, KBA])
2. next([KBG, VBG, KBBd, VBBd, KBA])
       transition(KBG, VBG)     next([VBG, KBBd, VBBd, KBA])
3. transition(KBG, VBG)     neural_network(KBG, VBG) [p =
   0.67]
4. next([VBG, KBBd, VBBd, KBA])
       transition(VBG, KBBd)     next([KBBd, VBBd, KBA])
5. transition(VBG, KBBd)     neural_network(VBG, KBBd) [p =
   1.00]
   ... (Fortsetzung)

Wahrscheinlichkeit: 0.67    1.00    0.67    ... = 0.42
```

Listing 5: Erklärbarkeitsausgabe

Dieser Beweisbaum ist direkt interpretierbar und bildet exakt die ARS-Grammatikregeln ab. Der einzige Unterschied ist, dass die Wahrscheinlichkeiten von einem neuronalen Netz gelernt und nicht manuell gezählt werden.

4.5 Vergleich mit der ursprünglichen ARS-Implementierung

Tabelle 3: ARS vs. DeepProbLog-Implementierung

Kriterium	**ARS (Scheme/-Lisp)**	**DeepProbLog**
Wahrscheinlichkeitslernen	Manuelles Zählen	Neuronales Netzlernen
Regelrepräsentation	Assoziationslisten	Logische Prädikate
Parsing-Algorithmus	Chart-Parser (handkodiert)	Beweissuche (eingebaut)
Generierung	Benutzerdefinierter Transduktor	Sampling aus Verteilung
Erklärbarkeit	Nachvollziehbar via Code	Beweisbäume
Skalierbarkeit	Gering (n=8)	Hoch (n > 1000)
Neuronale Integration	Keine	Vollständig (neuronale Prädikate)

Die DeepProbLog-Implementierung bewahrt die methodologischen Tugenden der ARS und fügt Skalierbarkeit und neuronales Lernen hinzu. Sie ist kein Ersatz, sondern eine **technische Instanziierung** derselben methodologischen Prinzipien.

5 Auf dem Weg zu einer Synthese: ARS als Bauplan, DeepProbLog als Motor

5.1 Was die ARS zu DeepProbLog beiträgt

Die ARS-Methodologie bietet drei Lehren für DeepProbLog-Praktiker:

1. **Interpretative Verankerung**: Die Bedeutung von Symbolen muss dokumentiert sein. Ein DeepProbLog-Programm mit nicht interpretierten Symbolen ist nicht erklärend. Die ARS zeigt, wie Symbole in qualitativer Interpretation verankert werden können.

2. **Trennung von Struktur und Statistik**: Die ARS hält eine strikte Trennung zwischen strukturellen Regeln (deterministisch, logisch) und statistischen Regularitäten (probabilistisch, empirisch) ein. DeepProbLogs Mischung aus logischen Regeln und neuronalen Wahrscheinlichkeiten riskiert die Vermischung dieser Ebenen. Die ARS legt nahe, sie in der Programmstruktur getrennt zu halten.

3. **Falsifizierbarkeit als Validierung**: Die ARS besteht darauf, dass Grammatiken durch Gegenbeispiele falsifizierbar sein müssen. Die Validierung in DeepProbLog stützt sich typischerweise auf Likelihood-Maximierung. Die ARS schlägt vor, dies durch qualitative Falsifikationstests zu ergänzen.

5.2 Was DeepProbLog zur ARS beiträgt

Umgekehrt bietet DeepProbLog drei Verbesserungen für ARS-Praktiker:

1. **Skalierbares Lernen**: Das manuelle Übergangszählen der ARS skaliert nicht. DeepProbLogs neuronales Lernen kann tausende Beispiele verarbeiten.

2. **Rohdatenintegration**: Die ARS benötigt vorkodierte Terminalzeichen. DeepProbLog kann direkt aus Rohdaten (Text, Bilder, Audio) durch neuronale Prädikate lernen.

3. **Kontinuierliche Aktualisierung**: ARS-Grammatiken sind statisch. DeepProbLog-

Netze können inkrementell aktualisiert werden, wenn neue Daten eintreffen.

5.3 Eine Forschungsagenda für neuro-symbolische ARS

Basierend auf dieser Synthese schlage ich eine Forschungsagenda vor:

1. **Portierung des ARS-Korpus nach DeepProbLog**: Vervollständigung der Implementierung der Verkaufsgesprächs-Grammatik in DeepProbLog, einschließlich aller 12 Terminalzeichen und ihrer Übergangswahrscheinlichkeiten.
2. **Hinzufügung neuronaler Prädikate für Rohdaten**: Training neuronaler Netze zur direkten Abbildung von Rohdaten (Audio, Transkript) auf Terminalzeichen.
3. **Implementierung des Multiagentensystems**: Aufbau eines vollständigen Multiagentensystems, in dem Agenten Rollen und Interaktionsmuster durch DeepProbLog lernen.
4. **Validierung mit den XAI-Kriterien**: Evaluation der DeepProbLog-Implementierung anhand der XAI-Kriterien der ARS (Verständlichkeit, Genauigkeit, Wissensgrenzen).
5. **Skalierung auf größere Korpora**: Anwendung der DeepProbLog-ARS auf größere Datensätze (hunderte oder tausende Gespräche) zur Testung der Skalierbarkeit.

6 Fazit

Dieser Beitrag hat die methodologische Kontinuität von frühen ARS-Implementierungen in Scheme, Pascal und Lisp zu zeitgenössischer neuro-symbolischer Programmierung in DeepProbLog nachgezeichnet. Ich habe argumentiert, dass die ARS bereits die Kernprinzipien neuro-symbolischer Integration – Mustererkennung, regelbasiertes Schließen, probabilistische Unsicherheit und Erklärbarkeit durch Design – Jahrzehnte vor der Prägung des Begriffs verkörperte.

Die Abbildung von ARS-Konzepten auf DeepProbLog ist direkt und natürlich. Die probabilistische Grammatik wird zu einer Menge logischer Regeln mit neuronalen Prädikaten; der Parser wird zur Beweissuche; der Transduktor wird zum Sampling. DeepProbLog ersetzt die ARS nicht, sondern *instanziiert* ihren methodologischen Bauplan mit modernen Rechenwerkzeugen.

Die Synthese ist kein Wettbewerb, sondern eine Komplementarität. Die ARS liefert die methodologische Strenge und interpretative Verankerung, die DeepProbLog (und der neuro-symbolischen KI allgemein) oft fehlt. DeepProbLog liefert die Skalierbarkeit und das neuronale Lernen, das der ARS fehlt. Zusammen deuten sie auf ein **methodologisch fundiertes, skalierbares neuro-symbolisches Framework** für die Analyse sequenzieller sozialer Interaktionen hin.

Die Frage für zukünftige Forschung ist nicht, ob die ARS oder DeepProbLog überlegen ist. Beide sind Werkzeuge für unterschiedliche Zwecke. Die Frage ist, wie sie integriert werden können, so dass die methodologischen Lehren der ARS die technische Entwicklung von DeepProbLog informieren und die Rechenleistung von DeepProbLog die Reichweite der ARS erweitert.

Literatur

Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Kautz, H. (2020). The third AI summer: AAAI Robert S. Engelmore Memorial Award Lecture. *AI Magazine*, 43(1), 93-104.

Koop, P. (1992). *Demo-Parser Chart-Parser Version 1.0*. Pascal-Quellcode.

Koop, P. (1994). *Grammatikinduktion empirisch gesicherter Verkaufsgespräche*. Scheme-Quellcode.

Koop, P. (1994). *Sequenzanalyse empirisch gesicherter Verkaufsgespräche*. Lisp-Quellcode.

Koop, P. (2023). *Qualitative Sozialforschung und Große Sprachmodelle*. Jupyter Notebook.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., & De Raedt, L. (2018). DeepProbLog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31.

Marcus, G. (2020). The next decade in AI: Four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.