Torsten Seiler

Konstruktion eines massiv parallelen Planers

Transformation von Systembeschreibungen in $\mathcal{A}_{\mathcal{C}}$
in Neuronale Netze

Technische Universität Dresden
Fakultät Informatik
Wissensverarbeitung

Großer Beleg

zum Thema

Konstruktion eines massiv parallelen Planers

Transformation von Systembeschreibungen in $\mathcal{A}_\mathcal{C}$
in Neuronale Netze

von
Torsten Seiler
geb. am 14. Januar 1969 in Berlin

Technische Universität Dresden
Fakultät Informatik

Eingereicht am 22. Dezember 1995

Betreuer: Sven-Erik Bornscheuer
Verantwortlicher Hochschullehrer: Prof. Dr. Steffen Hölldobler

Inhaltsverzeichnis

Einleitung

In den letzten Jahren wurde viel Arbeit in die Konstruktion paralleler Hardware investiert. Als Vertreter von paralleler Hardware seien da die Connection Machine 5 oder die Transputersysteme von Parsytec genannt. Die Entwicklung von effizienten Algorithmen für parallele Maschinen wurde ebenso intensiv vorangetrieben. Beispielsweise wurden auf dem Gebiet der Logikprogrammierung verschiedene Varianten von parallelen Prolog-Systemen untersucht und weiterentwickelt, siehe hierzu die Arbeiten ([Kar92], [AK93],[Gre93],[Sin93]).

Natürlich steigt der Bedarf an leistungsfähiger Hardware und geeigneten Algorithmen in vielen Bereichen von Wirtschaft und Forschung weiter an. Man denke hier nur an die Simulation und Prognose von meteorologischen Abläufen, die Steuerung von autonomen Systemen und die Bildverarbeitung. Eins der erwähnten parallelen Prolog-Systeme (AURORA) wurde für die Lösung von molekularbiologischen Problemen mit Erfolg eingesetzt ([LMOS93]). Dies sind einige Beispiele, die auf konventionellen (sequentiellen) Maschinen nur unbefriedigend lösbar sind. In zunehmendem Maße kommen hier parallele Rechnerarchitekturen zum Einsatz.

Der Preisverfall der Hardware erlaubt es nicht nur grobgranulare, sondern zunehmend auch massiv parallele Systeme zu bauen. Nun besteht allerdings die Notwendigkeit, die Lösungen für bestimmte Probleme auf diese Hardwarearchitekturen anzupassen. Ohne eine optimale Anpassung der Algorithmen an die Architektur werden die Vorteile einer parallelen Maschine nicht oder nur gering zum Tragen kommen ([Brä93]).

Betrachtet man die natürliche Informationsverarbeitung, so fällt auf, daß praktisch alle informationsverarbeitenden Prozesse parallelisiert sind. Insbesondere stellt man fest, daß trotz der verhältnismäßig langsamen Komponenten biologischer Informationssysteme sehr kurze Antwortzeiten auf die für das System relevanten Fragestellungen erreicht werden, obwohl die durchgeführten Informationsverarbeitungsprozesse von komplexer Natur sind. Die Nachbildung von Teilen dieser Prozesse, beispielsweise der Bild- und Sprachverarbeitung, hat sich auf herkömmlichen sequentiellen Systemen als äußerst schwierig herausgestellt ([Gö95]).

Natürliche Systeme sind offensichtlich auch in der Lage, sich in einer veränderlichen Welt zurechtzufinden. Höhere Organismen (zumindest vom Menschen glauben wir es zu wissen) bilden mentale Modelle ihrer Umwelt, in der sie leben. Sie sind ebenfalls in der Lage, ihren Einfluß auf ihre Umwelt in gewissen Grenzen vorherzusehen. Dies ist eine entscheidende Voraussetzung für ein planvolles, zielgerichtetes Handeln.

Für technische Systeme, die in irgendeiner Form mit dynamischen Systemen umgehen müssen, ist es ebenfalls notwendig, ein Modell dieser veränderlichen Welt zu besitzen. Diese Modelle können erlernt und somit intern konstruiert sein oder aber durch den Konstrukteur des technischen Systems in dieses implementiert sein. Um ein solches Modell zu erzeugen, muß das zu betrachtende dynamische System in irgendeiner Weise formalisiert werden.

In dieser Arbeit wird, aufbauend auf einer solchen Formalisierung für dynamische Systeme, ein massiv paralleles System vorgestellt, das ein dynamisches System repräsentieren kann.

Die hier benutzte Formalisierung nennt sich $\mathcal{A}_C$ und wurde von Baral und Gelfond in [BG] vorgestellt. Bei $\mathcal{A}_C$ handelt es sich um eine Erweiterung von $\mathcal{A}$. $\mathcal{A}$ wurde von Gelfond und Lifschitz in [GL92] veröffentlicht.

Bei der Beschreibung von dynamischen Systemen mittels $\mathcal{A}$ wird davon ausgegangen, daß das dynamische System durch stabile Zustände charakterisiert ist, über die dann Aussagen getroffen werden können. Solche stabilen Zustände können nur durch die Ausführung von Aktionen verlassen werden. Das dynamische System bewegt sich durch die Ausführung einer Aktion von einem stabilen Zustand in einen anderen.

Nun ist $\mathcal{A}$ in seiner Ausdruckskraft recht beschränkt, so daß verschiedene Erweiterungen von $\mathcal{A}$ veröffentlich wurden (beispielsweise [KL93]). Eine dieser Erweiterungen ist die gleichzeitige Ausführung mehrerer Aktionen. Die so entstandene Sprache wurde $\mathcal{A}_C$ genannt ([BG]).

Sowohl in $\mathcal{A}$ als auch in $\mathcal{A}_C$ ist es möglich, die Gleichzeitigkeit von Aktionen zum Ausdruck zu bringen. Da die Semantik von $\mathcal{A}$ diese Gleichzeitigkeit nicht unterstützt, müßte für jede Kombination von Aktionen, die gleichzeitig ausgeführt werden sollen eine neue Aktion eingeführt werden. Aus dem Alltag ist uns jedoch bekannt, daß viele Aktionen unabhängig voneinander ausgeführt werden können und der Effekt die Summe der Teileffekte ist. Allerdings treten hiervon auch Ausnahmen auf. Das Öffnen einer Tür, die durch einen Türöffner verriegelt ist, erfordert zumindest das Drücken des Öffners und das gleichzeitige Ziehen an der Tür. Werden beide Aktionen nacheinander ausgeführt, wird sich die Tür nicht öffnen. In $\mathcal{A}_C$ lassen sich solche Szenarien effizient formalisieren.

Diese Arbeit befaßt sich mit der Abbildung von in $\mathcal{A}_C$ formulierten Systembeschreibungen auf neuronale Netze und somit auf eine parallele Architektur. Ein durch diese Abbildung konstruiertes Netz wird durch ein spezielles Lernverfahren derart angepaßt, daß es die Zustandsüberführung des modellierten dynamischen Systems entsprechend der Semantik von $\mathcal{A}_C$ berechnet.

Mit diesem Netz ist es allerdings erst möglich, die Überführung des Systemzustandes bei Ausführung einer Aktion zu modellieren. Für einen Plan, welcher eine Sequenz von Aktionen darstellt, die eine gegebene Start- in eine Zielsituation überführt, sind jedoch noch weitere Komponenten notwendig.

Eine mögliche Lösung ist, mehrere solcher Transitionsnetze hintereinander zu koppeln und somit die iterative Anwendung der Überführungsfunktion zu simulieren. Die Auswahl der in jedem Schritt anzuwendenden Aktion muß durch einen externen Steueralgorithmus durchgeführt werden. Eine geeignete Lösung ist, diesen Steueralgorithmus in die Dynamik des Netzes mit zu implementieren.

Genau dies ist der in dieser Arbeit beschrittene Weg. Die durch das Lernverfahren zeitlich effizient gewonnenen Informationen über die Struktur des beschriebenen Systems, gehen in einem weiteren Schritt in die Konstruktion eines Hopfieldnetzes ein. Das aus der Systembeschreibung und dem Transitionsnetz generierte Hopfieldnetz löst nun Planungsprobleme bis zu einer vor der Konstruktion festgelegten Länge. Ein Plan wird dabei als Optimierungsproblem betrachtet und jede nicht korrekt durchgeführte Transition als Fehler.

Ein anderer Ansatz ist, das Transitionsnetz als Lehrer in einem überwachten Lernverfahren zu benutzen. Die Aktionssequenzen würden dann von einem zu trainierenden Netz generiert werden. Die Ausführbarkeit solcher Aktionen ist anhand des Transitionsnetzes überprüfbar.

Das Kapitel 1 führt in die Sprache $\mathcal{A}_C$, deren Syntax und Semantik ein.

Im Kapitel 2 wird der formale Apparat für die Beschreibung von Neuronalen Netzen betrachtet.

Das oben beschriebene Transitionsnetz wird im Kapitel 3 konstruiert. Anhand des Beipiels aus Kapitel 1 werden die Konstruktionsschritte erläutert. Weiterhin befindet sich auch das Lernverfahren in diesem Kapitel.

Auf die Anwendung des in Kapitel 3 konstruierten Netzes in einem massiv parallelen Planer geht das Kapitel 4 ein. In diesem Kapitel wird auch kurz auf die Transformation von aussagenlogischen Formeln in Energiefunktionen eingegangen, die dann mittels Hopfieldnetzen minimiert werden können.

Kapitel 1

$\mathcal{A}_C$

$\mathcal{A}_C$ ist eine Repräsentationsform für dynamische Systeme, in der davon ausgegangen wird, daß diese dynamischen Systeme eine endlich Menge von stabilen Zuständen besitzen. Ein Zustand ist genau dann stabil, wenn er nur durch die Ausführung von Aktionen verlassen werden kann. Das dynamische System bewegt sich durch die Ausführung einer Aktion von einem stabilen Zustand zu einem anderen. Über solche stabilen Zustände können Aussagen getroffen werden. In $\mathcal{A}_C$ werden solche Aussagen durch sogenannte *Fluenten* repräsentiert.

Beispielsweise kann die Aussage $\langle$Die Tür ist offen$\rangle$ durch den Fluenten *open* symbolisiert werden.

Ein *Zustand* im Modell des dynamischen Systems wird durch eine *Menge von Fluenten* beschrieben.

Zwei Zustände sind gleich, wenn die sie beschreibenden Aussagenmengen gleich sind.

Es soll nun die Wirkung von Aktionen betrachtet werden. Die Aussage $\langle$Die Tür ist offen$\rangle$ soll im Zustand i gelten. Durch die Ausführung der Aktion *„Schließe die Tür“* ist die Aussage im Folgezustand nicht mehr gültig, dies wird im weiteren auch als Effekt der Aktion bezeichnet.

Aktionen sind selbst wieder aus sogenannten Elementaraktionen zusammengesetzt. Im folgenden bezeichnet Aktion sowohl die Elementaraktion als auch die komposite Aktion.

In der Sprache $\mathcal{A}_C$ ist nicht nur die Ausführung von Elementaraktionen formalisiert, sondern auch die Veränderung der Effekte bei der gleichzeitigen Ausführung von Aktionen. Existieren beispielsweise zwei Aktionen *„Tisch anheben links“* und *„Tisch anheben rechts“*, dann ist die Folge der Ausführung nur einer Aktion, daß die Suppenschüssel, die auf dem Tisch steht, auskippt. Nur wenn beide Aktionen gleichzeitig ausgeführt werden, wird die Suppe nicht verschüttet.

Ein veränderliches System kann in $\mathcal{A}_C$ mit folgenden Ausdrucksmitteln beschrieben werden:

- Σ_1 nicht leere, endliche Menge aller *Fluenten.*
- Σ_2 nicht leere, endliche Menge aller *Elementaraktionsbezeichner*[1].
- Eine *Überführungsfunktion* $\Phi : 2^{\Sigma_1} \times 2^{\Sigma_2} \mapsto 2^{\Sigma_1}$, die zu jedem Paar von Zustand und Aktion einen Folgezustand definiert.
- Eine Menge von *Effekt-Propositionen* zur Definition der Überführungsfunktion.
- Eine Menge von *Wert-Propositionen* zur Einschränkung der möglichen Initialzustände des Systems.

Definition 1.1 (Systembeschreibung) Eine *Systembeschreibung* ist ein Tupel $(\Sigma_1, \Sigma_2, \mathcal{E}, \mathcal{W})$, wobei Σ_1 die Menge der Fluent-Bezeichner, Σ_2 die Menge der Aktionsbezeichner, $\mathcal{E}$ die Menge der Effekt-Propositionen und $\mathcal{W}$ die Menge der Wert-Propositionen ist. ∎

Definition 1.2 (Fluent-Literal) Sei $f \in \Sigma_1$ ein Fluent, dann ist ein Fluent-Literal ein Fluent oder ein Fluent-Literal mit einem davorgeschriebenen $\neg$ Symbol. Dabei gilt, daß $\neg\neg fl = fl$ ist, wenn fl ein Fluent-Literal ist. ∎

Wert-Propositionen (V-Proposition) haben in einer Systembeschreibung folgende Form

$$f \quad \textbf{after} \quad [a_1, ..., a_m], \tag{1.1}$$

wobei $[a_1, ..., a_m]$ ($m \geq 0$) eine Sequenz von Aktionen ist und f ein Fluent-Literal.

Die Bedeutung einer solchen V-Proposition ist: „*f gilt nach der im initialen Zustand des Systems beginnenden Ausführung der Folge von Aktionen $[a_1, ..., a_m]$*“. Für $m = 0$ wird (1.1) als

$$\textbf{initially} \quad f \tag{1.2}$$

geschrieben.

Effekt-Propositionen (E-Propositionen) haben die Form

$$a_1, ..., a_m \quad \textbf{causes} \quad f \ \textbf{if} \ c_1, ..., c_n, \tag{1.3}$$

[1] Im folgenden wird verkürzend Elementaraktion benutzt.

wobei $\{a_1, ..., a_m\}$ eine Aktion ist und f sowie $c_1, ..., c_n (n \geq 0)$ Fluent-Literale sind.

Die Bedeutung einer Effekt-Proposition ist: „*Die Ausführung der Aktion* $\{a_1, ..., a_m\}$ *verursacht die Gültigkeit von* f *in dem resultierenden Folgezustand, falls die Bedingungen* $c_1, ..., c_n$ *im aktuellen Zustand gelten.*“

Im folgenden wird $\{a_1, ..., a_m\}$ auch als *Aktionsmenge*, $\{c_1, ..., c_n\}$ als die *Vorbedingungsmenge* und f als *Effekt* der Effekt-Proposition bezeichnet.

In $\mathcal{A}_C$ wird als Effekt einer Aktion die Summe[2] der Effekte ihrer explizit beschriebenen Teilaktionen angenommen. Treten hiervon Ausnahmen auf, wenn also der Effekt der gleichzeitigen Ausführung von Aktionen nicht die Summe der Teilaktionen ist, so muß diese Ausnahme ebenfalls explizit angegeben werden. Dies ist sinnvoll, da die meisten Aktionen voneinander unhabhängig bezüglich ihrer Effekte sind.

Im folgenden Beispiel wird gezeigt, wie in $\mathcal{A}_\mathcal{C}$ eine Welt modelliert werden kann. Insbesondere werden die Auswirkungen von gleichzeitigen Aktionen berücksichtigt. Dieses Beispiel wurde [Bor94] entnommen.

Beispiel 1 Läuft man gegen eine geschlossene Tür, so verletzt man sich. Wird gleichzeitig der Türöffner betätigt, öffnet sich die Tür und man gelangt in das Haus. Durch betätigen des Türöffners wird der neben der Tür liegende Hund wach. Die Tür kann durch ziehen wieder geschlossen werden.

In $\mathcal{A}_C$ kann dieses Szenario durch eine Systembeschreibung D_1, wie folgt modelliert werden. Zuerst werden die beiden Mengen

$$\begin{aligned} \Sigma_1 &= \{open, sleeps, hurt\} \\ \Sigma_2 &= \{activate, pull, run\_into\} \end{aligned}$$

definiert. Der initiale Zustand wird partiell durch die Wert-Proposition

$$\textbf{initially} \quad sleeps$$

festgelegt. Die Wirkungen der Aktionen werden durch die Effekt-Propositionen beschrieben:

$$\begin{array}{lrl} (1) & \{activate\} & \textbf{causes} \ \neg sleeps \\ (2) & \{run\_into\} & \textbf{causes} \ hurt \ \textbf{if} \ \neg open \\ (3) & \{pull\} & \textbf{causes} \ \neg open \\ (4) & \{activate, run\_into\} & \textbf{causes} \ open \\ (5) & \{activate, run\_into\} & \textbf{causes} \ \neg hurt \ \textbf{if} \ \neg hurt \end{array}$$

[2] Summe ist hier im Sinne von Vereinigung der einzelnen Effekte zu verstehen.

Der Effekt der gleichzeitigen Ausführung von $\{activate\}$ und $\{run\_into\}$ kommt in den Effekt-Propositionen eins, zwei, vier und fünf zum Ausdruck. Die Effekt-Proposition fünf sollte zudem auch die zweite „überstimmen“ (overrule)[3] da sie die Aktionen $\{activate, run\_into\}$ genauer beschreibt. □

Was ist unter einer eine Aktion genauer beschreibenden Effekt-Proposition zu verstehen?

Der Wert eines Fluenten f kann nur durch die Ausführung einer Aktion a geändert werden, wenn eine Effekt-Proposition e existiert, in dieser der Fluent f vorkommt, die Aktion a in e beschrieben wird und die Vorbedingungen $c_1, ...c_n$ im aktuellen Zustand erfüllt sind. Eine solche Effekt-Proposition soll potentiell anwendbar genannt werden. Die Anwendung einer solchen Effekt-Proposition kann verhindert werden[4], wenn es eine anwendbare Effekt-Proposition e' gibt, die einen Effekt $\neg f$ hat und die Aktion a Teilmenge der Aktionsmenge der Effekt-Proposition e' ist.

Im Beispiel wäre die zweite Effekt-Proposition bei der Ausführung der Aktion $\{activate, run\_into\}$ eine potentiell anwendbare Effekt-Proposition. Sie wird jedoch durch die fünfte Effekt-Proposition überstimmt, da auch diese potentiell anwendbar ist und die Aktionsmenge der zweiten als Teilmenge enthält. Die fünfte ihrerseits wird nicht überstimmt, ist somit nicht nur potentiell anwendbar, sondern (effektiv) anwendbar.

Im folgenden soll der bisher intuitiv eingeführte Begriff der Anwendbarkeit von Effekt-Propositionen definiert werden.

Definition 1.3 (Anwendbarkeit einer E-Proposition) Sei D eine Systembeschreibung in $\mathcal{A}_C$, $a \in 2^{\Sigma_2}$ eine Aktion, f ein Literal eines Fluenten in Σ_1 und $\sigma \in 2^{\Sigma_1}$ ein Zustand, dann gilt:
a *causes* f *in* σ gdw. eine Aktion $b \in 2^{\Sigma_2}$ existiert, so daß a **causes** f **by** b **in** σ gilt.
Dabei gilt a **causes** f **by** b **in** σ gdw.

1. $b \subseteq a$
2. D enthält eine E-Proposition e mit b **causes** f **if** $c_1, ..., c_n$, so daß alle $c_1, ..., c_n$ in σ gelten.
3. D enthält keine Aktion a' mit $b \subset a' \subseteq a$ und a **causes** $\neg f$ **by** a' **in** σ

[3] Diese Beziehung von Effekt-Propositionen zueinander wird im folgenden auch als Overrule-Relation bezeichnet, d.h. $R_O(e_1, e_2)$ ist genau dann erfüllt, wenn ein Zustand existiert, in der die Effekt-Proposition e_1 die Effekt-Proposition e_2 überstimmen kann.

[4] Dann wird diese Effekt-Proposition ignoriert.

Sind die Punkte 1) und 2) erfüllt, so wird eine solche Effekt-Proposition als *potentiell anwendbar* bezeichnet. Ist zusätzlich der Punkt 3) erfüllt, so heißt eine solche Effekt-Proposition *anwendbar*. ■

Definition 1.4 (Ignorieren einer E-Proposition) Sei D eine Systembeschreibung, $e \in \mathcal{E}$ eine Effekt-Proposition aus D. Die Effekt-Proposition e wird genau dann ignoriert, wenn sie nicht anwendbar ist. ■

Die Effekte der anwendbaren Effekt-Propositionen werden in zwei Mengen zusammengefaßt.

Definition 1.5 (Effektmengen) Sei D eine Systembeschreibung in $\mathcal{A}_C$, $a \in 2^{\Sigma_2}$ eine Aktion und $\sigma \in 2^{\Sigma_1}$ ein Zustand, dann sind die Mengen B_f und B'_f definiert als

$$\begin{aligned} B_f(a,\sigma) &:= \{f \in \Sigma_1 \mid a \ causes\ f\ in\ \sigma\} \\ B'_f(a,\sigma) &:= \{f \in \Sigma_1 \mid a\ causes\ \neg f\ in\ \sigma\}. \end{aligned}$$

B_f enthält die Fluenten, die durch die Ausführung von a in σ, im Folgezustand als wahr bestimmt wurden. B'_f enthält genau die Fluenten, die durch die Ausführung von a in σ, im Folgezustand als falsch bestimmt wurden. ■

Die durch eine Systembeschreibung in $\mathcal{A}_C$ determinierte Überführungsfunktion Φ wird durch die Mengen B_f, B'_f und die zugrundegelegte Persistenzannahme definiert. Die Persistenzannahme lautet: „Wenn keine Effekt-Proposition eine Änderung eines Fluenten stützt, so bleibt der Fluent unverändert“.

Definition 1.6 (Überführungsfunktion Φ) Sei D eine Systembeschreibung in $\mathcal{A}_C$, $a \in 2^{\Sigma_2}$ eine Aktion und $\sigma \in 2^{\Sigma_1}$ ein Zustand, dann ist die durch die Systembeschreibung D determinierte Überführungsfunktion $\Phi_D(a,\sigma)$ definiert durch:

1. $\Phi_D(a,\sigma)$ ist definiert für alle Paare (a,σ), für die gilt: $B_f(a,\sigma) \cap B'_f(a,\sigma) = \{\}$.
2. Ist $\Phi_D(a,\sigma)$ definiert, so ist $\Phi_D(a,\sigma) = (\sigma \cup B_f(a,\sigma)) \setminus B'_f(a,\sigma)$.

■

Die Überführung des Zustands σ in den Folgezustand σ' erfolgt also durch Entnahme der als falsch und dem Hinzufügen der als wahr bestimmten Fluenten. Die Überführungsfunktion ist nicht definiert, falls ein Fluent, resultierend aus der Ausführung der Aktion a, im Folgezustand sowohl als wahr, wie auch als falsch bestimmt wurde. Ein solcher Widerspruch wird derart interpretiert, daß die Aktion a im aktuellen Zustand nicht ausführbar ist. Andere Ansätze zur Behandlung solcher Widersprüche wurden z.B. in [BT94] untersucht.

Eine Überführungsfunktion definiert nur das Verhalten eines dynamischen Systems in Bezug auf Aktionen. Es wird jedoch nichts über den Zustand des Systems zu irgendeinem Zeitpunkt ausgesagt.

Da jede Überführungsfunktion Φ deterministisch ist, genügt zur Beschreibung des Zustandes neben Φ die Betrachtung des initialen Systemzustandes und der bisher ausgeführten Aktionen.

Definition 1.7 (Struktur M) Eine *Struktur M* ist ein Paar (σ_0, Φ) , wobei σ_0 ein Zustand ist und Φ eine Überführungsfunktion. Die Struktur $M = (\sigma_0, \Phi)$ beschreibt das im initialen Zustand σ_0 befindliche System mit dem durch Φ bestimmten Verhalten. ■

Eine Wert-Proposition (siehe Def. 1.1) ist eine Aussage über den resultierenden Zustand nach der Ausführung einer Aktionsfolge $[a_1, ..., a_m]$, beginnend im initialen Zustand σ_0 des Systems.

Definition 1.8 (result) Sei D eine Systembeschreibung in $\mathcal{A}_C$, $a \in 2^{\Sigma_2}$ eine Aktion, $\sigma \in 2^{\Sigma_1}$ ein Zustand und $result(M, [a_1, ..., a_k])$ eine Abkürzung für $\Phi(a_k, \Phi(a_{k-1}, ..., \Phi(a_1, \sigma_0)..))$ mit $M = (\sigma_0, \Phi)$, dann ist die Wert-Proposition *wahr* in M gdw.

$$\forall\ 1 \leq k \leq m. result(M, [a_1, ..., a_k])$$

definiert ist und f in $result(M, [a_1, ..., a_m])$ gilt. ■

Das Verhalten eines dynamischen Systems wird durch die Effekt-Propositionen einer Systembeschreibung D in $\mathcal{A}_C$ beschrieben. Die möglichen initialen Zustände des Systems sind dadurch noch nicht bestimmt. Nimmt man nun die Wert-Propositionen als Fakten hinzu, d.h. als wahre Aussagen, so kann das zu einer Einschränkung der initialen Systemzustände führen. D beschreibt dann die möglicherweise leere Menge der Strukturen M, in denen das durch die Effekt-Propositionen definierte Verhalten existiert und alle durch die Wert-Propositionen in D beschriebenen Aussagen wahr sind.

Definition 1.9 (Modell) Eine Struktur M, in der alle Wert-Propositionen einer Systembeschreibung wahr sind, ist *Modell* dieser Systembeschreibung. Hat eine Systembeschreibung mindestens ein Modell, heißt sie *konsistent*. Eine Systembeschreibung D *stützt* eine Wert-Proposition v ($D \models v$) gdw. v in jedem Modell von D wahr ist. ■

Aussagen über das durch D beschriebene System, können aus dessen Beschreibung gefolgert werden, indem weitere Wert-Propositionen zu der Systembeschreibung D hinzugefügt werden und die Konsistenz des so erweiterten Systems nachgewiesen wird. Somit entspricht der Modellbegriff der Folgerbarkeitsbeziehung.

Zur Veranschaulichung der Wirkungsweise der Effekt- und Wert-Propositionen kommen wir nochmals auf das Beispiel 1 und die dort definierte Systembeschreibung D_1 zurück.

Beispiel 2 Sei $\sigma \in 2^{\Sigma_1}$ ein beliebiger Zustand in dem durch D_1 beschriebenen System, dann hat die durch die Effekt-Proposition in der Systembeschreibung D_1 bestimmte Überführungsfunktion folgendes Abbildungsverhalten:

$$\begin{array}{lcl}
\Phi(\{\}, \sigma) & = & \sigma \\
\Phi(\{run_into\}, \sigma \cup \{open\}) & = & \sigma \cup \{open\} \\
\Phi(\{run_into\}, \sigma \setminus \{open\}) & = & \sigma \setminus \{open\} \cup \{hurt\} \\
\Phi(\{pull\}, \sigma) & = & \sigma \setminus \{open\} \\
\Phi(\{activate\}, \sigma) & = & \sigma \setminus \{sleeps\} \\
\Phi(\{activate, pull\}, \sigma) & = & \sigma \setminus \{sleeps, open\} \\
\Phi(\{run_into, pull\}, \sigma \cup \{open\}) & = & \sigma \setminus \{open\} \\
\Phi(\{run_into, pull\}, \sigma \setminus \{open\}) & = & \sigma \setminus \{open\} \cup \{hurt\} \\
\Phi(\{activate, run_into\}, \sigma) & = & \sigma \cup \{open\} \\
\Phi(\{activate, run_into, pull\}, \sigma) & & \text{ist nicht definiert}
\end{array}$$

Gemäß der Definition 1.5 resultiert die Undefiniertheit von $\Phi(\{activate, run\_into, pull\}, \sigma)$ aus:

$$\begin{array}{lcl}
\{open\} & \subseteq & B_f(\{activate, run_into, pull\}, \sigma) \\
\{open\} & \subseteq & B'_f(\{activate, run_into, pull\}, \sigma)
\end{array}$$

Die Systembeschreibung D_1 hat die folgenden vier Modelle:

$$\begin{array}{cc}
(\{sleeps\}, \Phi) & (\{open, sleeps\}, \Phi) \\
(\{sleeps, hurt\}, \Phi) & (\{open, sleeps, hurt\}, \Phi)
\end{array}$$

Das sind jene Zustände, in denen *sleeps* gilt, da **initially** *sleeps* die einzige Wert-Proposition ist. Durch Hinzunahme der Wert-Proposition

$$\neg hurt \ \textbf{after} \ \{run\_into\}$$

zu D_1 ist das einzige noch verbleibende Modell

$$(\{open, sleeps\}, \Phi),$$

da für alle anderen Strukturen $hurt \in \Phi(\{run\_into\}, \sigma_0)$ gilt. Die Wert-Proposition **initially** *open* wird von dieser erweiterten Systembeschreibung gestützt. □

Für die Korrektheit der in Kapitel 3 definierten Abbildung muß die Freiheit in der Wahl der Effekt-Propositionen ein wenig eingeschränkt werden. Hierbei handelt es sich um eine Einschränkung, die in $\mathcal{A}_C$ normalerweise nicht enthalten ist.

Definition 1.10 (Transformierbarkeit) D ist eine transformierbare Systembeschreibung, wenn in der Menge von Effekt-Propositionen $\mathcal{E}$ keine zwei Effekt-Propositionen mit genau gleicher Aktionsmenge und negiertem Effekt enthalten sind. ■

Wenn also

$$a \text{ causes } f \text{ if } C_1 \quad \text{und}$$
$$a \text{ causes } \neg f \text{ if } C_2$$

in $\mathcal{E}$ enthalten sind, ist D eine nicht transformierbare Systembeschreibung.

In den folgenden Kapiteln wird unter der Semantik von $\mathcal{A}_C$ immer die durch die Definition 1.10 eingeschränkte Version verstanden.

Kapitel 2

Neuronale Netze

In diesem Kapitel wird in den formalen Apparat eingeführt, der zur weiteren Beschreibung der Neuronalen Netze in dieser Arbeit benutzt wird. Die Darstellung wurde weitgehend [NKK94] entnommen.

Da hier keine Vollständigkeit der Beschreibung Neuronaler Netze angstrebt wird, soll nur so weit in die Thematik der Neuronalen Netze eingeführt werden, wie es für das Verständnis der Arbeit notwendig ist. Für eine umfangreichere Einführung in die Thematik der Neuronalen Netze sind z.B. [NKK94] und [Roj93] geeignet.

2.1 Die Struktur Neuronaler Netze

Ein Neuronales Netz läßt sich leicht als eine Struktur formalisieren, die durch eine Menge und einige Abbildungen beschreibbar ist.

Definition 2.1 (Neuronales Netz) Ein Neuronales Netz kann als Tupel (U, W, A, O, NET, ex) aufgefaßt werden.

1. **U** ist eine Menge von Verarbeitungseinheiten[1].
2. **W**, die Netzwerkstruktur, ist eine Abbildung $(U \times U \to \mathbb{R})$,
3. **A** ist eine Abbildung, die jedem $u \in U$ eine Aktivierungsfunktion $A_u : \mathbb{R}^3 \to \mathbb{R}$ zuordnet,
4. **O** ist eine Abbildung, die jedem $u \in U$ eine Ausgabefunktion $O_u : \mathbb{R} \to \mathbb{R}$ zuordnet,
5. **NET** ist eine Abbildung, die jedem $u \in U$ eine Netzeingabefunktion $NET_u : (\mathbb{R} \times \mathbb{R})^{cardU} \to \mathbb{R}$ zuordnet und

[1]synonym werden auch die Begriffe Neuron bzw. Unit benutzt

6. **ex** ist eine externe Eingabefunktion $ex : U \to \mathbb{R}$, die jedem $u \in U$ eine externe Eingabe in Form einer rellen Zahl $ex_u = ex(u) \in \mathbb{R}$ zuordnet.

■

Die Definition 2.1 beschreibt nur die Struktur und die statischen Eigenschaften Neuronaler Netze, sie sagt nichts über ihre Dynamik aus. Die einzelnen Bestandteile der Definition sollen nun näher betrachtet werden.

Verarbeitungseinheiten

Die Verarbeitungseinheiten in einem Neuronalen Netz sind i.a. einfache Automaten oder Prozessoren, die eine sehr einfache Berechnung[2] durchführen. Sie generieren eine Ausgabe aufgrund ihres aktuellen Zustandes (ihrer Aktivierung) und der aktuellen Eingabe. Durch die Möglichkeit, in die Berechnung der neuen Ausgabe auch die aktuelle Aktivierung mit einfließen zu lassen, können Speichereffekte nachgebildet werden. Die Verarbeitungseinheiten führen ihre Berechnungen vollständig parallel und unabhängig durch. Eine feinere Unterscheidung der Verarbeitungseinheiten bzgl. ihrer Funktion und Lage im Netz ist in einigen Netzwerkmodellen von Vorteil. So spricht man z.B. von Ein-/Ausgabe-Einheiten, wenn diese eine Kommunikation mit der Umwelt durchführen. In geschichteten Netzwerkmodellen, wie z.B. dem Multi-Layer-Perceptron (MLP), unterscheidet man zusätzlich zwischen Ein- und Ausgabe-Einheiten sowie verdeckten (inneren) Einheiten.

Die Netzwerkstruktur

Die Abbildung W wird im allgemeinen als Netzwerkstruktur bezeichnet. Sie läßt sich als bewerteter gerichteter Graph darstellen, dessen Knoten die Verarbeitungseinheiten sind. Die bewerteten Kanten repräsentieren dann die gerichteten Kommunikationsverbindungen zwischen den Verarbeitungseinheiten. W wird in Matrixform auch als *Konnektionsmatrix* bezeichnet.

$w(u, u')$ wird (synaptisches) Gewicht, (synaptische) Verbindung oder Kopplung genannt. Die Netzwerkstruktur bildet die Grundlage der Kommunikation innerhalb eines Neuronalen Netzes. Die Ausgabe einer Verarbeitungseinheit ist die Eingabe einer anderen Verarbeitungseinheit, wenn zwischen ihnen eine Kopplung existiert.

Die Gewichte werden bezüglich ihrer Größe in drei Kategorien eingeteilt:

$w(u, u') = 0$: es existiert keine Verbindung von u nach u'

[2] z.B. Vektorprodukt

$w(u,u') < 0$: inhibitorische (hemmende) Verbindung von u nach u'
$w(u,u') > 0$: excitatorische (anregende) Verbindung von u nach u'

Die Abbildung W ist während eines Lernprozesses veränderbar. Das Wissen, welches ein Neuronales Netz während des Lernvorganges erworben hat, bzw. welches ihm während der Konstruktion mitgegeben wurde, wird innerhalb der Netzstruktur durch die Gewichte verteilt repräsentiert. Für eine Manipulation von W wird eine geeignete Lernregel[3] benötigt, die W so anpaßt, daß eine bestimmte Aufgabe bezüglich eines zu definierenden Fehlermaßes besser oder vollständig gelöst wird.

Aktivierungsfunktion

Die Abbildung A ordnet jeder Verarbeitungseinheit eine Aktivierungsfunktion zu. Diese Funktion berechnet die aktuelle Aktivierung a_u der Verarbeitungseinheit anhand der vorangegangenen Aktivierung $a_u^{(neu)}$, der aktuellen Netzeingabe net_u und der aktuellen externen Eingabe ex_u.

$$a_u = A_u({a_u}^{(alt)}, net_u, ex_u) \tag{2.1}$$

In vielen Fällen werden alle Verarbeitungseinheiten die gleiche Berechnung durchführen. Sollte es jedoch notwendig sein, daß einzelne Einheiten eine andere Berechnung durchführen müssen, so ist das durch die Abbildung A auch möglich.

Die meisten konnektionistischen Modelle verwenden lieare Funktionen, Treppen-, Stufen- oder sigmoide Funktionen.

Geht in diese Funktionen nur die Netzeingabe ein und wird auf eine vereinfachte Form der Aktivierungsfunktion zurückgegriffen, $A_u : \mathbb{R} \to \mathbb{R}$.

Die Wahl der Aktivierungsfunktion ist sehr stark modellabhängig. Für einige Lernverfahren (z.B. Error-Backpropagation) sind differenzierbare Aktivierungsfunktionen notwendig.

Ausgabefunktion

Die Ausgabefunktion dient der Anpassung unterschiedlicher Wertebereiche der Eingaben und der Ausgaben verschiedener Verarbeitungseinheiten. In den üblichen Netzmodellen sind die Aktivierung und die Ausgabe einer Verarbeitungseinheit jedoch gleich, so daß die Ausgabefunktion die Identitätsfunktion ist.

[3] Im Falle eines MLP ist dies für gewöhnlich eine Variante des Error-Backpropagation-Algorithmus.

Netzeingabe

Die Abbildung NET ordnet jeder Einheit eine Netzeingabefunktion NET_u zu. Üblicherweise wird auch hier für alle Einheiten eine Funktion verwendet. In vielen konnektionistischen Systemen kommt die gewichtete Summe über die Ausgaben aller Einheiten zum Einsatz:

$$net_u = \sum_{u' \in U} w(u, u') \cdot o_{u'} - \theta_u. \tag{2.2}$$

Der Schwellenwert θ_u kann über einen zusätzlichen Eingang mit dem Gewicht $-\theta_u$ und einer konstanten Eingabe von eins nachgebildet werden. Somit läßt sich die obige Formel immer in eine Form ohne θ bringen.

Die externe Eingabefunktion

Durch die externe Eingabefunktion ex steht das Neuronale Netz mit seiner Umwelt (Außenwelt) in Verbindung und erhält über diesen Weg seine Eingaben. Das Netz reagiert auf diese externen Eingaben mit einer Veränderung seiner Aktivierungen und erzeugt dadurch eine Ausgabe. In vielen Neuronalen Netzen ist nur eine Teilmenge der Verarbeitungseinheiten über eine externe Eingabe zu erreichen, d.h. ex ist nur über einer Teilmenge von Einheiten $U_I \subseteq U$, den Eingabeeinheiten definiert.

2.2 Die Arbeitsweise Neuronaler Netze

Bei der nun folgenden Betrachtung wird davon ausgegangen, daß die Wichtungsmatrix des Neuronalen Netzes konstant bleibt. Die Vorgänge während des Lernens bleiben somit vorerst unberücksichtigt.

Für die Betrachtung der zeitlichen Abläufe innerhalb eines Neuronalen Netzes werden drei Phasen definiert:

1. die Ruhephase,
2. die Eingabephase und
3. die Arbeitsphase.

In der *Ruhephase* bleiben die Aktivierungen aller Einheiten konstant. Dieser Zustand wird erreicht, wenn sich die Aktivierung keiner Verarbeitungseinheit, trotz Neuberechnung, ändert. Bei rückgekoppelten Netzen, ist es durchaus möglich, daß es eine solche Ruhephase nicht geben muß. In einem solchen Falle wird davon ausgegangen, daß das Netz „eingefroren“ und so künstlich in einen Ruhezustand versetzt werden kann. Befindet sich ein Netz in einer Ruhephase, so bezeichnet man diesen Zustand auch als *stabilen Zustand.*

Die *Eingabephase* schließt sich direkt an die Ruhephase an. In dieser Phase werden die Aktivierungen der Eingabeeinheiten berechnet. Als Eingabeeinheiten werden genau die Einheiten bezeichnet, für die die Abbildung ex definiert ist.

In der *Arbeitsphase* werden die Aktivierungen aller Einheiten berechnet und über die durch die Wichtungsmatrix definierten Verbindungen im Netz propagiert.

Die Berechnung der Aktivierung einer Einheit kann *asynchron* bzw. *synchron* bezüglich aller anderen Einheiten durchgeführt werden. Asynchron bedeutet, daß jede Einheit für sich zu einem beliebigen Zeitpunkt die Eingänge auswertet und ihre Aktivität anpaßt. Bei einer synchronen Berechnung werten alle Einheiten gleichzeitig ihre Eingänge aus und propagieren die Berechnungsergebnisse ebenfalls gleichzeitig.

2.3 Strukturveränderung – Lernen

Ein wesentlicher Aspekt Neuronaler Netze ist die Fähigkeit mittels eines Lernverfahrens die Struktur des Netzes[4] so zu adaptieren, daß eine gegebene Lernaufgabe[5] erfüllt wird.

Definition 2.2 (Eingabe-,Ausgabemuster) Sei N ein beliebiges Neuronales Netz mit einer Menge von Verarbeitungseinheiten U, der externen Eingabefunktion ex und der für eine Ausgabeeinheit v gültigen Ausgabefunktion O_v.

$$U_I = \{u_1, ..., u_n\} \subseteq U$$

sei die Menge der Eingabeeinheiten und

$$U_O = \{v_1, ..., v_m\} \subseteq U$$

die Menge der Ausgabeeinheiten. Weiterhin bezeichne *dom* den Bildbereich (Domäne) einer Funktion.

1. Eine Eingabe (Eingabemuster) i von N ist ein Element der Menge

$$\mathcal{I} = \{(x_1, ..., x_n) | x_i \in dom(ex)\} \subseteq \mathbb{R}^{U_I}.$$

2. Eine Ausgabe (Ausgabemuster) t von N ist ein Element der Menge

$$\mathcal{I} = \{(y_{v_1}, ..., y_{v_n}) | y_i \in dom(O_{v_i})\} \subseteq \mathbb{R}^{U_O}.$$

[4] also die Abbildung W

[5] Es wird u.U. auch nur verlangt, daß eine gewisse Fehlerschranke nicht überschritten wird.

■

Für die Durchführung des Lernvorganges ist neben einem geeigneten *Lernalgorithmus* die Vorgabe einer *Lernaufgabe* erforderlich. Man kann zwischen *freien* und *festen* Lernaufgaben unterscheiden.

Definition 2.3 (freie Lernaufgabe) Eine freie Lernaufgabe $\mathcal{L} \subseteq \mathcal{I}$ eines Neuronalen Netzes ist eine Menge von Eingabemustern. Das Neuronale Netz soll auf jede Eingabe $i \in \mathcal{L}$ der Lernaufgabe eine Ausgabe $t \in \mathcal{T}$ bestimmen. Ein Neuronales Netz erfüllt eine freie Lernaufgabe, wenn es zu jeder Eingabe eine Ausgabe bestimmt hat, so daß im Sinne eines geeigneten Abstandsmaßes[6] ähnliche Eingaben auch ähnliche Ausgaben hervorrufen. ■

Definition 2.4 (feste Lernaufgabe) Eine feste Lernaufgabe $\tilde{\mathcal{L}} \subseteq \mathcal{I}$ eines Neuronalen Netzes ist eine Menge von Ein-/Ausgabepaaren. Das Neuronale Netz soll auf jede Eingabe i eines Paares $(i,t) \in \tilde{\mathcal{L}}$ mit der dazugehörigen Ausgabe t reagieren. Ein Neuronales Netz erfüllt die feste Lernaufgabe, wenn es zu jeder Eingabe die laut Lernaufgabe dazugehörige Ausgabe erzeugt. ■

Definition 2.5 (Lernalgorithmus) Ein Lernalgorithmus ist ein Verfahren, das anhand einer Lernaufgabe die Netzwerkstruktur W eines Neuronalen Netzes verändert. Erfüllt das Netz diese Lernaufgabe nach Anwendung des Verfahrens, bzw. wird eine vorher festgelegte Fehlerschranke unterschritten, so war der Lernalgorithmus erfolgreich, andernfalls erfolglos. Ein Lernalgorithmus, der eine freie Lernaufgabe verwendet, heißt nichtüberwachter (unsupervised) Lernalgorithmus. Verwendet er eine feste Lernaufgabe, so heißt er überwachter (supervised) Lernalgorithmus. ■

Definition 2.6 (Fehlermaß) Sei $\tilde{\mathcal{L}}$ eine feste Lernaufgabe und $l \in \tilde{\mathcal{L}}$ ein zu lernendes Musterpaar.

1. Ein Fehlermaß für einen überwachten Lernalgorithmus ist eine Abbildung
$$e : \mathbb{R}^{U_O} \times \mathbb{R}^{U_O} \to \mathbb{R}_0^+, \tag{2.3}$$
so daß für alle $a, b \in \mathbb{R}^{U_O}$
$$e(a,b) = 0 \iff a = b \tag{2.4}$$
gilt. Der Fehler $e^{(l)}$, den das Neuronale Netz bei der Verarbeitung des Musterpaares $l \in \tilde{\mathcal{L}}$ macht, ist durch $e^{(l)} = e(t^{(l)}, o^{(l)})$ gegeben. Dabei ist $o^{(l)}$ die Ausgabe des Netzes auf das Eingabemuster $i^{(l)}$ und $t^{(l)}$ die erwünschte, durch die Lernaufgabe $\tilde{\mathcal{L}}$ vorgegebene Ausgabe (target).

[6]Ein Abstandsmaß definiert, wie ähnlich sich zwei Vektoren sind.

2. Der Fehler $e_u^{(l)}$ einer Ausgabeeinheit $u \in U_O$ bei der Verarbeitung des Musterpaares $l \in \tilde{\mathcal{L}}$ ist die Differenz zwischen dem durch die Lernaufgabe vorgegebenen Ausgabewert $t_u^{(l)}$ und des tatsächlichen Ausgabewertes $o_u^{(l)}$:

$$e_u^{(l)} = t_u^{(l)} - o_u^{(l)}. \tag{2.5}$$

■

Kapitel 3

Konstruktion von $\mathcal{N}_T$

3.1 Konstruktion $\mathcal{N}_T$ aus D

In diesem Abschnitt wird eine Konstruktionsvorschrift für das Transitionsnetz $\mathcal{N}_T$ angegeben. Anhand der Systembeschreibung D_1 (Beispiel 1) werden die durchgeführten Konstruktionschritte verdeutlicht.

3.1.1 Verwendete Berechnungseinheiten

Definition 3.1 (Transitionsnetz)
Ein Transitionsnetz $\mathcal{N}_T$ ist ein Neuronales Netz und wird beschrieben durch das Tupel (U, W, A, O, NET, ex). ■

Definition 3.2 (Unitmengen) Sei $\mathcal{N}_T$ ein Transitionsnetz und D eine Systembeschreibung in $\mathcal{A}_C$. Die Menge der Units U wird entsprechend der zu lösenden Aufgabe in Teilmengen untergliedert.

$$\begin{aligned} U &= U_F \cup U_E \cup U_A \cup U_V \\ U_V &= U_{V_a} \cup U_{V_b} \cup U_{V_d} \cup U_{V_o} \end{aligned}$$

Die Teilmengen zeichnen sich durch die Eigenschaften

$$\begin{aligned} \forall i, j \in \{F, E, A, V_a, V_b, V_d, V_o\}.(i \neq j) &\rightarrow (U_i \cap U_j = \emptyset) \\ |U_F| = |U_{v_a}| = |U_{v_b}| = |U_{v_d}| = |U_{v_o}| &= |\Sigma_1| \\ |U_A| &= |\Sigma_2| \\ |U_E| &= |\mathcal{E}| \end{aligned}$$

aus. Der Operator $|\cdot|$ ist definiert als $|Menge| := card Menge$. ■

Eine Unit wird innerhalb ihrer Menge durch den ihr zugeordneten Bezeichner angesprochen.

Definition 3.3 (Unitbezeichnung) Sei $\mathcal{N}_T$ ein Transitionsnetz und D eine Systembeschreibung, dann werden folgende Units eingeführt:

- für jedes $f \in \Sigma_1$,
  $u_f \in U_F$ Fluent-Unit (im Zustand σ)
  $u_{f,v_t} \in U_V$ Vote-Unit vom Typ t des Fluenten f für jedes $t \in \{a, b, d, o\}$
- für jede Aktion $a \in \Sigma_2$,
  $u_a \in U_A$ Aktions-Unit
- für jede E-Proposition $e \in \mathcal{E}$,
  $u_e \in U_E$ E-Prop-Unit.

■

Die genaue Funktion der einzelnen Unit-Klassen kann der nun folgenden Beschreibung entnommen werden.

Definition 3.4 (Unit-Funktionen) Sei $\mathcal{N}_T$ ein Transitionsnetz und D eine Systembeschreibung in $\mathcal{A}_C$, dann werden folgende Unit-Klassen in $\mathcal{N}_T$ verwendet:
(die Abbildungen, die nicht aufgeführt sind, sind auch nicht definiert)

Fluent-Units U_F

EX : externe Eingabe ($ex = 0$ beim Lernen, sonst $ex \in \{+1, -1\}$)
$+1$ erhält die Bedeutung logisch wahr und -1 logisch falsch.

O : Identitätsfunktion

A : $f_{act} = \begin{cases} +1 & : \quad ex = +1 \\ 0 & : \quad ex = 0 \\ -1 & : \quad ex = -1 \end{cases}$

Aktions-Units U_A

EX : externe Eingabe $ex \in \{0, 1\}$. ($ex = 1$ zur Unit korrespondierende Aktion soll ausgeführt werden, $ex = 0$ keine Ausführung)

NET : $net = \sum_{u \in U_E} w(u, u')o_u$

O : Identitätsfunktion

A : $f_{act} = \begin{cases} +1 & : \quad (net > 0) \vee (ex = 1) \\ 0 & : \quad sonst \end{cases}$

E-Prop-Units U_E

EX : externe Eingabe $ex \in \{0,1\}$.

NET : $net = \sum_{u \in U_F \cup U_A \cup U_E} w(u,u')o_u - \theta$ mit $\theta := \theta_l$ während des Lernvorganges und $\theta := \theta_\Phi$ sonst.

O : Identitätsfunktion

A : $f_{act} = \begin{cases} +1 & : (net > 0) \vee (ex = 1) \\ 0 & : sonst \end{cases}$

Vote-a-Units U_{V_a}

NET : $net = \sum_{u \in U_E} [w(u,u')o_u]^2$

O : Identitätsfunktion

A : $f_{act} = \begin{cases} +1 & : net > 0 \\ 0 & : sonst \end{cases}$

Vote-b-Units U_{V_b}

NET : $net = \begin{cases} \frac{net_a}{net_b} & : net_b > 0 \\ 0 & : sonst \end{cases}$, wobei

$$\begin{aligned} net_a &= \sum_{u \in U_E} w(u,u')o_u \\ net_b &= \sum_{u \in U_E} [w(u,u')o_u]^2 \end{aligned}$$

O : Identitätsfunktion

A : $f_{act} = \begin{cases} +1 & : net = 1 \\ -1 & : net = -1 \\ 0 & : sonst \end{cases}$

Vote-d-Units U_{V_d}

NET : $net = \sum_{u \in U_F \cup U_{v_a}} w(u,u')o_u$

O : Identitätsfunktion

A : $f_{act} = \begin{cases} +1 & : 0.5 \le net \le 1.5 \\ -1 & : -1.5 \le net \le -0.5 \\ 0 & : sonst \end{cases}$

Vote-o-Units U_{V_o}

NET : $net = \sum_{u \in U_{V_b} \cup U_{V_d}} w(u,u')o_u$

O : Identitätsfunktion

A : $f_{act} = \begin{cases} +1 & : net > 0.5 \\ -1 & : net < -0.5 \\ 0 & : sonst \end{cases}$

■

Die obige Definition soll im folgenden erläutert werden. Insbesondere wird auf die funktionalen Zusammenhänge der einzelnen Unitklassen eingegangen.

Fluent-Units U_F

Die Fluent-Units kodieren den aktuellen Zustand des durch D beschriebenen Systems. Jede Unit repräsentiert genau einen Fluenten. Gilt der Fluent in der aktuellen Situation, so ist die externe Eingabe in die korrespondierende Fluent-Unit $ex = +1$, gilt er nicht, dann ist $ex = -1$. Die externe Eingabe $ex = 0$ wird nur während des Lernvorganges benutzt, um den Einfluß der Vorbedingungen auf die E-Prop-Units zu unterdrücken.

Aktions-Units U_A

Die Aktions-Units kodieren die in der aktuellen Situation auszuführende Aktion. Eine externe Eingabe von $ex = 1$ bedeutet eine Ausführung der zur Aktions-Unit korrespondierenden Elementaraktion $a \in \Sigma_2$. Bei $ex = 0$ wird diese Elementaraktion nicht ausgeführt.

E-Prop-Units U_E

Die Effekt-Units stellen eine zentrale Komponente des Transitionsnetzes $\mathcal{N}_T$ dar. Jede Unit kodiert genau eine Effekt-Proposition. Die Unit ist genau dann aktiv, wenn die korrespondierende Effekt-Proposition aus D anwendbar ist.

Das Lernverfahren benutzt einen verminderten Schwellwert. Diese zwei verschiedene Schwellwerte lassen sich genauso wie ein einzelner Schwellwert über zusätzliche, gewichtete Eingänge simulieren. In der Abbildung 3.7 im Abschnitt 3.2.1 ist diese Überführung nochmals dargestellt. Der Einfachheit halber wird in den folgenden Betrachtungen mit zwei unterschiedlich benannten Schwellwerten gearbeitet. Der Schwellwert θ_l wird nur beim Lernen benutzt, um den Einfluß der Fluent-Units (also der Vorbedingnung einer Effekt-Proposition) zu unterdrücken. Die Fluent-Units erhalten dazu eine externe Eingabe $ex = 0$. Im „normalen Betrieb“ des Netzes wird θ_Φ als Schwellwert verwendet.

Vote-a-Units

Diese Units ermitteln, ob es wenigstens eine E-Prop-Unit gibt, die den zugehörigen Fluenten beeinflussen könnte.

Vote-b-Units

Die Vote-b-Units ermitteln, ob alle E-Prop-Units, die eine Verbindung zu derselben Vote-b-Unit haben und aktiv sind, die gleiche Fluentenänderung propagieren. Dies ist mit dem Test der Schnittmenge von B_f und B'_f zu vergleichen. net_a wird dem Betrage nach nur dann gleich net_b, wenn alle aktiven E-Prop-Units die gleiche Fluentenänderung fordern, es also zu keinem Konflikt kommt.

Vote-d-Units

Die Vote-d-Units stellen einen Schalter dar. Ihre Aktivierung ist identisch mit der Ausgabe der Fluent-Unit des gleichen Fluenten, wenn auf die Vote-a-Unit keine aktive E-Prop-Unit wirkt. Es greift die Persistenzannahme, die alle Fluenten in den Folgezustand übernimmt, für die keine Effekt-Proposition eine Änderung postuliert. Ist die Aktivierung der zu dem Fluenten gehörigen Vote-a-Unit von Null verschieden, so verschiebt sich die Grunderregung der Vote-d-Unit so stark, daß unabhängig vom Ausgabewerte der Fluent-Unit, die Aktivierung stets Null ist und somit keinen Einfluß auf die nachgeschaltete Vote-d-Unit ausübt.

Vote-o-Units

Die Vote-o-Units sind einerseits eine Darstellung des Folgezustandes, der durch die Ausführung einer Aktion im aktuellen Zustand erreicht wird, und andererseits eine Instanz zur Erkennung von Konflikten. Die Vote-d-Unit führt die Ausgaben der Vote-b-Unit und der Vote-d-Unit des gleichen Fluenten zusammen. Durch die Konstruktion des Netzes ist es ausgeschlossen, daß die Ausgabe der Vote-b-Unit und der Vote-d-Unit gleichzeitig von Null verschieden sind. Es ist nur möglich, daß entweder eine von beiden Ausgaben Null ist, oder beide zusammen. Im zweiten Fall ist ein Konflikt in der Bestimmung des Fluenten für den Folgezustand aufgetreten. Die Ausgabe der Vote-d-Unit wird nur dann Null, wenn wenigstens eine E-Prop-Unit den Fluenten beeinflußt. Die der Vote-b-Unit nur dann, wenn gar keine oder aber wenigstens zwei sich widersprechende E-Prop-Units aktiv sind.

3.1.2 Verbindungsstruktur

Die bisherigen Definitionen haben nur die Verarbeitungseinheiten und deren Funktionalität beschrieben. In den nun folgenden Definitionen wird die Verbindungsstruktur des Transitionsnetzes, also die Wichtungsmatrix, festgelegt.

Der Aufbau des Transitionsnetze wird anhand der Systembeschreibung D_1 aus dem Beispiel 1 gezeigt. In den Abbildungen 3.2 bis 3.5 sind die durchzuführenden Operationen nochmals erläutert und dargestellt.

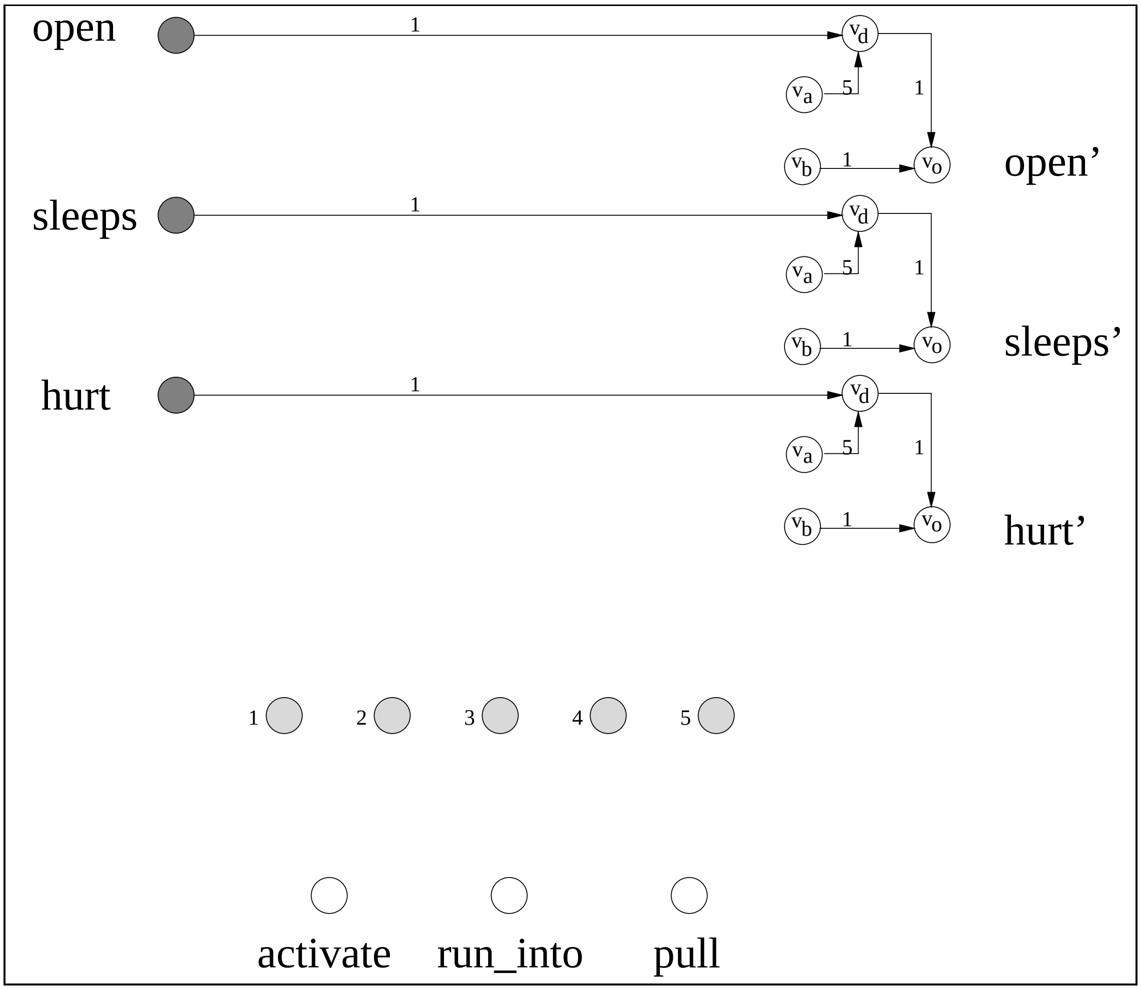


Abbildung 3.1: Verwendete Units im Transitionsnetz: Entsprechend der Systembeschreibung D_1, wurden die Fluent-Units, die Vote-Units bezüglich der Fluenten und die Units für die Effekt-Propositionen und Aktionen erzeugt. Desweiteren wurden die Verbindungen, die für die Persistenzannahme und die Konflikterkennung notwendig sind, ergänzt. Das Netz berechnet im jetzigen Zustand die Identitätsfunktion. Dies trifft auf Φ_D im entarteten Fall, mit $\mathcal{E} = \emptyset$, ebenfalls zu.

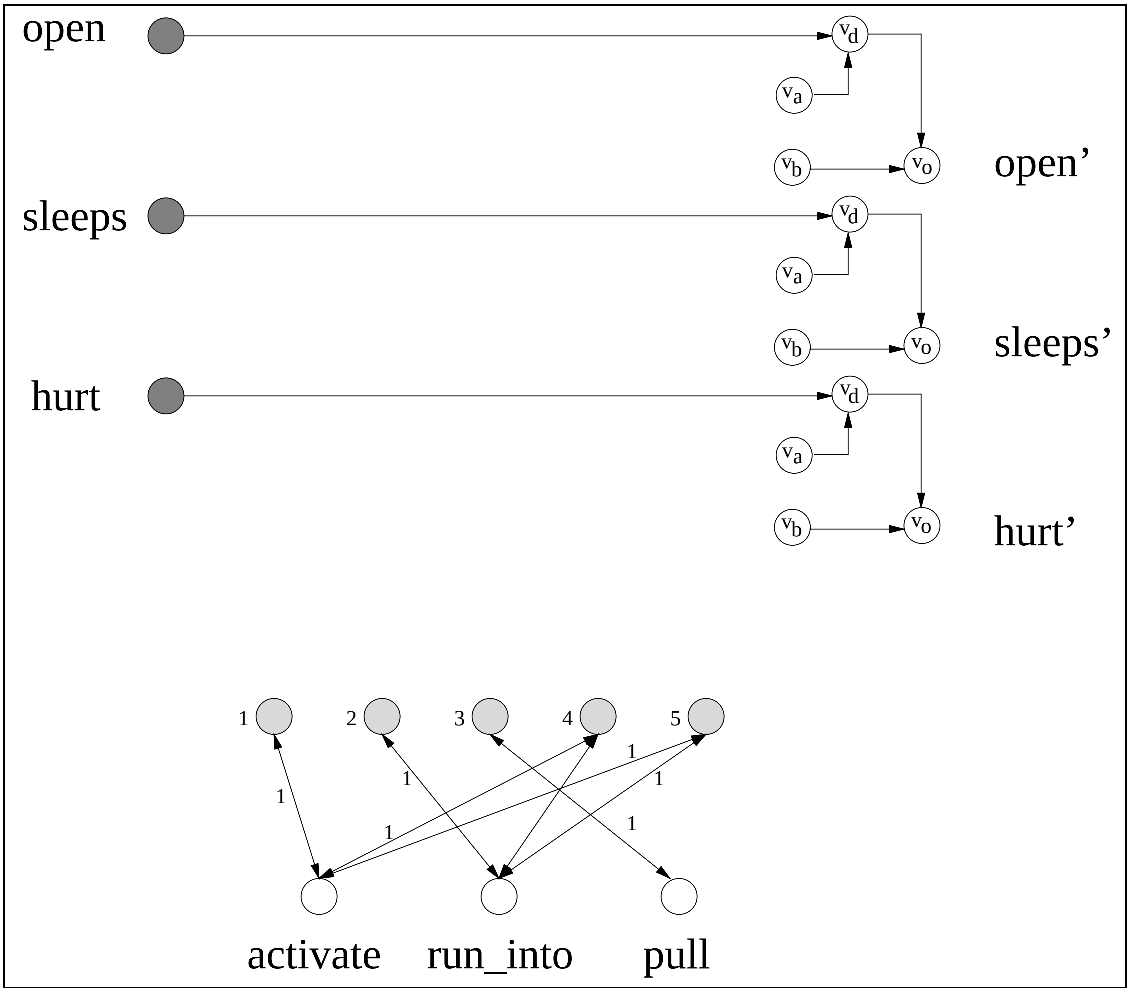


Abbildung 3.2: Verbindung der Aktions-Units mit den E-Prop-Units: Jede E-Prop-Unit wird entsprechend der korrespondierenden Effekt-Proposition mit den Aktions-Units verbunden. Die erste Effekt-Proposition aus D_1 enthält in ihrer Aktionsmenge nur die Aktionsbezeichnung für „activate". Daher werden zwei Verbindungen zwischen der Unit $u_{activate}$ und u_{e_1}, jeweils mit der Wichtung eins, eingefügt.

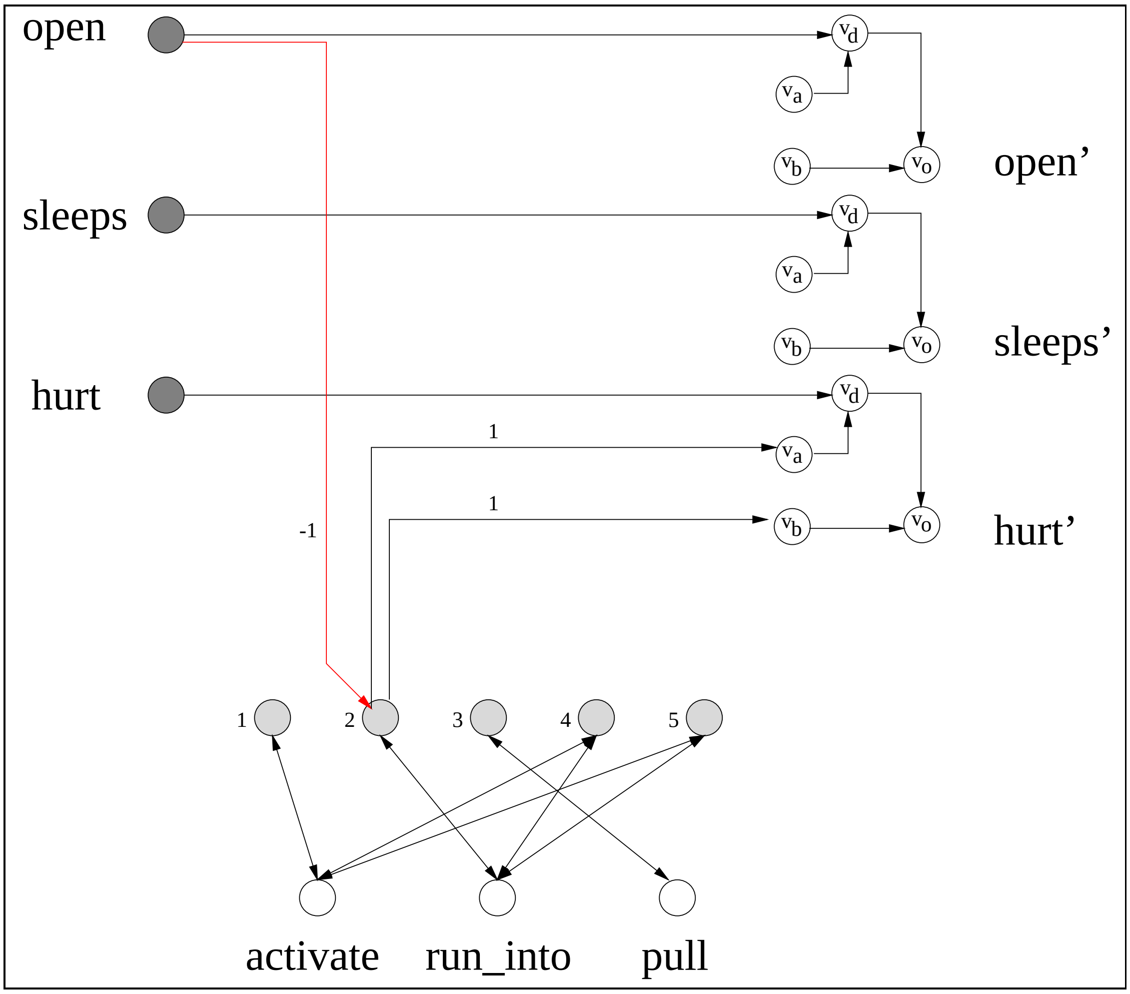


Abbildung 3.3: Realisierung der Effekt-Proposition zwei: Für jedes Vorbedingungsliteral in der Effekt-Proposition zwei wird eine Verbindung von der entsprechenden Fluent-Unit zur E-Prop-Unit, hier u_{e_2}, erzeugt. In diesem Fall handelt es sich um ein negierten Fluent ($\neg open$), somit ist das Gewicht gleich minus eins. Der Effekt (hier positiv) wird zu den dem Fluent zugeordneten Vote-Units ($u_{hurt,v_a}, u_{hurt,v_b}$) propagiert. Die Gewichte sind jeweils eins. Die Schwellwerte sind: $\theta_l = 0.5$ und $\theta_\Phi = 1.5$.

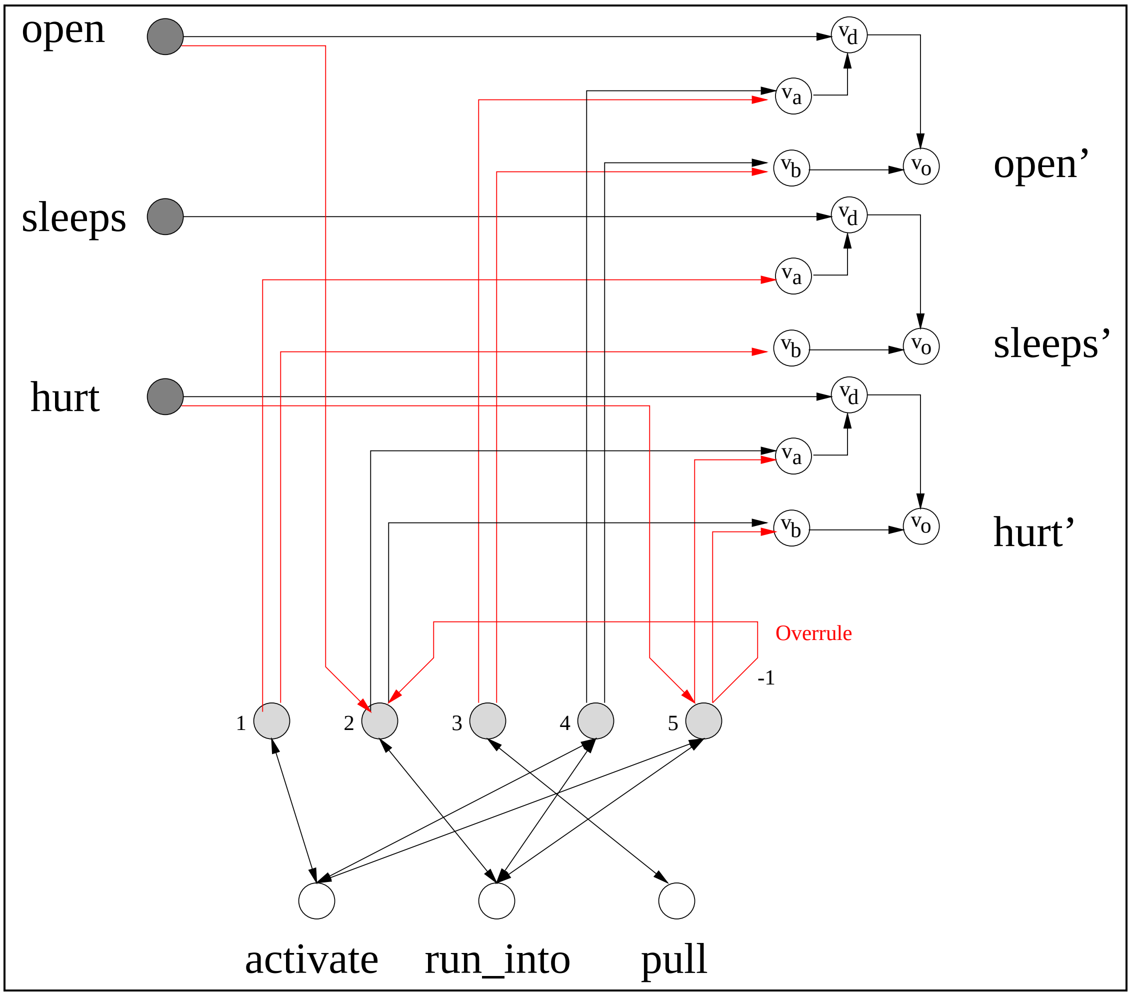


Abbildung 3.4: Vollständiges Transitionsnetz: Das dargestellte Netz enthält sowohl die durch die Effekt-Propositionen bestimmten Verbindungen, also auch die durch den Lernvorgang hinzugefügten Overrule-Verbindungen. Da es sich dabei um inhibitorische Verbindungen handelt, ist die Unit u_{e_2} nicht in der Lage aktiv zu werden, wenn die Unit u_{e_5} aktiv ist.

Definition 3.5 Sei $\mathcal{N}_T$ ein Transitionsnetz und D eine Systembeschreibung in $\mathcal{A}_C$. Für jede E-Proposition $e \in \mathcal{E}$ aus D mit der Form

$$A_e \ \textbf{causes} \ f_e^* \ \textbf{if} \ C_e^*, \tag{3.1}$$

wobei $A_e \in 2^{\Sigma_2}$, f_e^* ein Fluent-Literal und C_e^* eine Menge von Fluent-Literalen ist, werden folgende Verbindungen eingeführt:

1. Für den Effekt der E-Proposition, also das Fluent-Literal f_e^*,

$$f_e^* = \begin{cases} f & : \ w(u_e \in U_E, u_{f,v_{a,b}} \in U_V) = 1 \\ \neg f & : \ w(u_e \in U_E, u_{f,v_{a,b}} \in U_V) = -1 \end{cases} \tag{3.2}$$

2. Die Schwellwerte der E-Prop-Unit u_e sind:

$$\begin{aligned} \theta_l &= |A_e| \text{ und} \\ \theta_\Phi &= |A_e| + |C_e^*|. \end{aligned}$$

3. Für jede Aktion $a \in A_e \subseteq \Sigma_2$

$$w(u_a \in U_A, u_e \in U_E) = 1, \tag{3.3}$$
$$w(u_e \in U_E, u_a \in U_A) = 1. \tag{3.4}$$

4. Für jede Vorbedingung $c_e^* \in C_e^*$ mit $c \in \Sigma_1$

$$c_e^* = \begin{cases} c & : \ w(u_c \in U_F, u_e \in U_E) = 1 \\ \neg c & : \ w(u_c \in U_F, u_e \in U_E) = -1 \end{cases} \tag{3.5}$$

■

Die von der Menge der Effekt-Propositionen unabhängige Verbindungsstruktur entsteht durch die Persistenzannahme.

Definition 3.6 (Basisstruktur) Sei $\mathcal{N}_T$ ein Transisitionsnetz und D eine Systembeschreibung in $\mathcal{A}_C$, so sind für jeden Fluenten $f \in \Sigma_1$ folgende Verbindungen in $\mathcal{N}_T$ zu ergänzen (siehe Bild 3.1):

$$w(u_f \in U_F, u_{f,v_d} \in U_V) = 1 \tag{3.6}$$
$$w(u_{f,v_a} \in U_V, u_{f,v_d} \in U_V) = 5 \tag{3.7}$$
$$w(u_{f,v_d} \in U_V, u_{f,v_o} \in U_V) = 1 \tag{3.8}$$
$$w(u_{f,v_b} \in U_V, u_{f,v_o} \in U_V) = 1 \tag{3.9}$$

■

3.2 Lernen der Overrule-Relation

Das bisher konstruierte Netz $\mathcal{N}_T$ berechnet die Überführungsfunktion Φ_D noch nicht korrekt, da die Semantik von $\mathcal{A}_C$ bezüglich der Overrule-Relation (siehe Definition 1.3) zwischen den Effekt-Propositionen bisher nicht berücksichtigt wurde. Die Overrule-Relation läßt sich in der bisherigen Netzwerkstruktur sehr leicht erlernen. Vorteilhaft ist, daß durch die Struktur des Netzes die Teilmengenbeziehungen der Aktionen von zwei Effekt-Propositionen in konstanter Zeit berechnet wird. Somit ist es möglich, die Overrule-Relation in linearer Zeit bezüglich der Anzahl der Effekt-Propositionen zu berechnen[1].

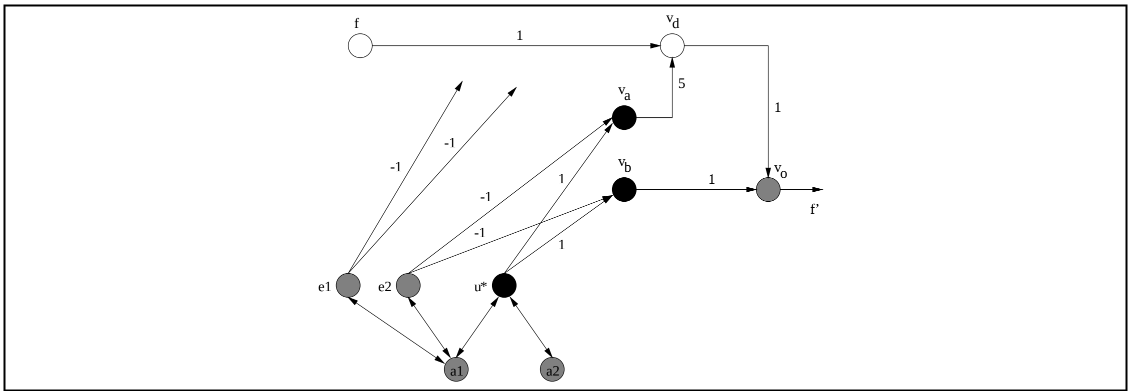


Abbildung 3.5: Lernen der Overrule-Relation: Es wird das Prinzip des Erlernens der Overrule-Relation dargestellt. Die Unit u^* wurde durch die externe Eingabe $ex(u^*) = 1$ aktiviert. Diese Aktivierung breitet sich über die Aktions-Units auf die E-Prop-Units e_1 und e_2 aus. Außerdem ist die Vote-b-Unit aktiv. Da Verbindung $w(u_{v_b}, u_{e_1}) = 0$ ist, wird keine inhibitorische Verbindung zu u_{e_1} eingeführt. Die Unit u_{e_2} ist jedoch auch aktiv. Da jedoch das Produkt $o(u_{e_2}) \cdot o(u_{v_b}) \cdot w(u_{v_b}, u_{e_2}) = -1$ ist, wird die Verbindung $w(u^*, u_{e_2}) = -1$ eingefügt.

3.2.1 Lernalgorithmus

Das hier vorgestellte Lernverfahren verändert die Netztopologie derart, daß die Aktivierung von Effekt-Propositions-Units (E-Prop-Units) eingeschränkt wird. Diese Einschränkung entspricht genau der Overrule-Relation in der Semantik von $\mathcal{A}_C$.

Vorausgesetzt wird ein globales, für alle E-Prop-Units gleichermaßen zugängliches Register, in dem die Adresse einer ausgewählten E-Prop-Unit gespeichert ist.

[1]gezeigt wird das im folgenden Kapitel

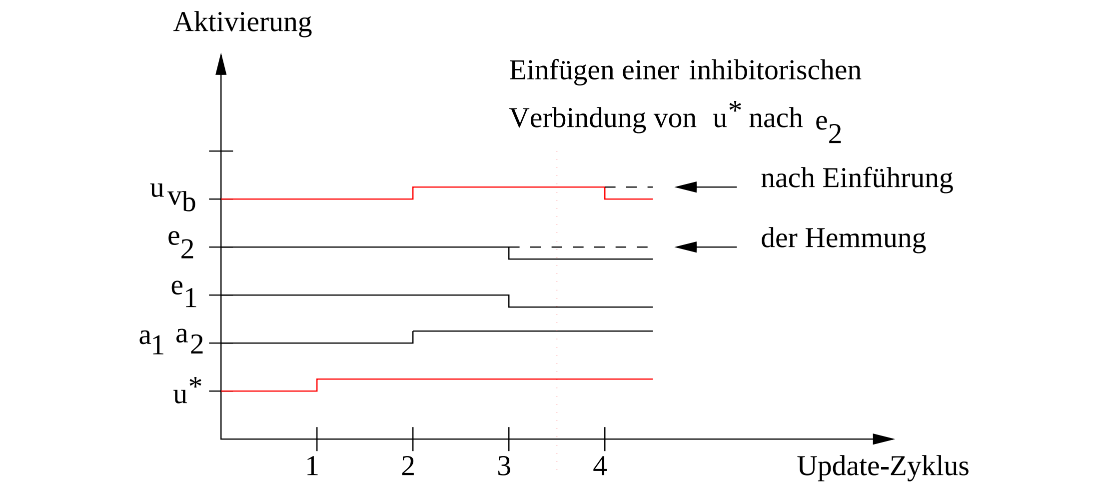


Abbildung 3.6: Zeitlicher Ablauf beim Lernen: Es sind nochmals die Aktivierungen der Units aus dem Bild 3.5 dargestellt. Gleichzeitig ist im Takt 3 die Wirkung einer schon existierenden inhibitorischen Verbindung sichtbar.

Während des Lernvorganges erfolgt eine synchrone Berechnung der Aktivitäten der einzelnen Units. Daraus folgt, daß die Aktualisierungsergebnisse immer erst im nächsten Update-Zyklus für die anderen Units zur Verfügung stehen. Diese Tatsache wird im Lernalgorithmus ausgenutzt.

Definition 3.7 Die Funktion $\Phi_{\mathcal{N}_T}$ sei definiert durch die Berechnung, die das Netz $\mathcal{N}_T$ durchführt. ■

Definition 3.8 Sei $\mathcal{N}_T$ ein entsprechend der Systembeschreibung D und den Definitionen 3.1 bis 3.6 konstruiertes Transitionsnetz. So ist die feste Lernaufgabe $\tilde{\mathcal{L}}$ indirekt durch die Überführungsfunktion Φ_D und eine Transformation auf Aktivierungen in $\mathcal{N}_T$ definiert. ■

Definition 3.9 (Lernalgorithmus) Sei $\mathcal{N}_T$ ein entsprechend der Systembeschreibung D konstruiertes Transitionsnetz, so wird die die Wichtungsmatrix W aus $\mathcal{N}_T$ nach folgendem Algorithmus angepaßt:

1. Vorbereitung des Netzes:
   - Für alle $u \in U_E$ Schwellwert auf $\theta_u := \theta_{u,l}$ absenken.
   - Für alle $u \in U_F$ wird die externe Eingabe $ex(u) = 0$ gesetzt.
2. Adaptionszyklus:
   Für jede E-Prop-Unit $u^* \in U_E$:

(a) Rücksetzen des Netzes (alle $u \in U$ werden deaktiviert, $o(u) = 0$).

(b) Die Unit u^* wird aktiviert, $ex(u^*) = 1$, $ex(u \neq u^*) = 0$.

(c) Es werden drei synchrone Update-Zyklen durchgeführt, so daß alle Units parallel ihre neue Aktivierung berechnen.

(d) $\forall u_e \in U_E, \forall u_{v_b} \in U_{V_b}$:

$$(o(u_e) \cdot o(u_{v_b}) \cdot w(u_e, u_{v_b}) = -1) \Longrightarrow w(u^*, u_e) := -1. \tag{3.10}$$

3. Rücksetzen:
   Für alle $u \in U_E : \theta_u := \theta_{u,\Phi}$ setzen.

■

3.2.2 Bemerkungen zum Lernalgorithmus

Die konkreten Werte für $\theta_{u,l}$ sind schon während des Konstruktionsprozesses direkt aus der Effekt-Proposition ableitbar, siehe Definition 3.5. Eine solche Konstruktion mit zwei verschieden Schwellwerten ist leicht über zwei zusätzliche Eingänge zu realisieren, siehe Abbildung 3.7.

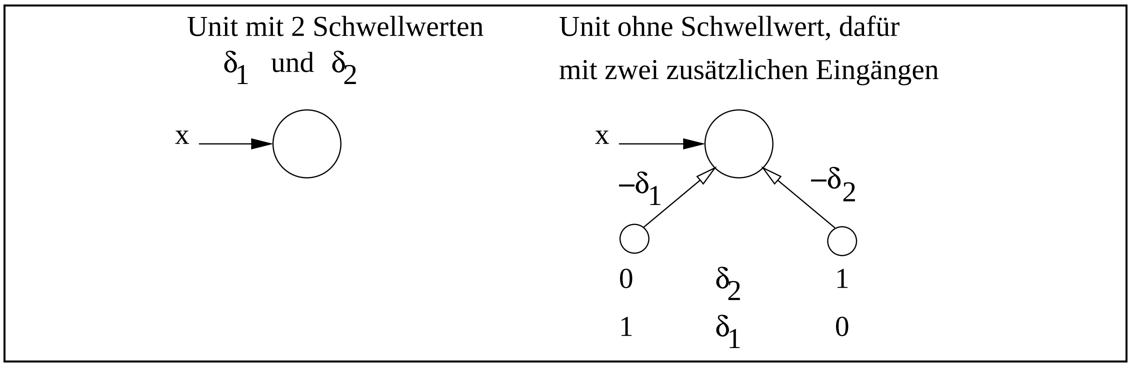


Abbildung 3.7: Ersetzen von Schwellwerten: Das linke Bild zeigt eine Unit mit zwei verschiedenen Schwellwerten, welcher benutzt wird muß durch eine externe Steuerung festgelegt werden. Im rechten Bild wurde diese Unit in eine andere Struktur überführt. Diese besitzt zwei zusätzliche Eingänge, die mit den negierten Schwellwerten gewichtet sind. Welcher Schwellwert zum Einsatz kommt, kann durch Aktivierung des einen oder des anderen Einganges entschieden werden.

Die Fluent-Units erhalten während des Lernvorganges eine externe Eingabe von Null ($ex(u \in U_F) = 0$). Durch diese Eingabe, der im „normalen Betrieb" des Netzes keine Bedeutung zugeordnet ist, wird der Einfluß der Vorbedingung auf die Anwendbarkeit einer Effekt-Proposition unterbunden. Die Absenkung des Schwellwertes auf die Anzahl der Aktionsbezeichner in der Aktionsmenge (θ_l) ist dann natürlich notwendig, damit die E-Prop-Unit überhaupt aktiv werden kann.

Es soll jetzt anhand einer ausgewählten E-Prop-Unit gezeigt werden, wie der Lernalgorithmus funktioniert. Zuerst wird die ausgewählte Unit u^* durch eine externe Eingabe $ex(u^*) = 1$ aktiviert. Diese Aktivierung liegt nach dem ersten synchronen Update am Ausgang der Unit u^* an (primäre Aktivierung). Durch den zweiten synchronen Update breitet sich diese Aktivierung entsprechend der Verbindungen in die Menge der Aktions-Units und auf genau zwei Vote-Units des gleichen Fluentenbezeichners f (u_{f,v_b} und u_{f,v_a}) aus. Zu diesem Zeitpunkt ist die durch u^* direkt erreichbare Vote-b-Unit die einzige Vote-b-Unit, die aktiv ist. Der dritte synchrone Update aktiviert weitere E-Prop-Units (sekundäre Aktivierung). Die im zweiten Update aktivierte Vote-b-Unit, ist immer noch die einzige aktive Vote-b-Unit.

In allen drei Schritten kann jede E-Prop-Unit lokal entscheiden, ob eine inhibitorische Verbindung von u^* aus notwendig ist. In den Schritten eins und zwei ist die Ausgabe aller E-Prop-Units u_e ($u_e \neq u^*$) gleich Null ($o(u) = 0, u \in U_E$). Die Voraussetzung 2.(d) in Definition 3.9 ist nicht erfüllt. Für die Unit u^* ist diese Voraussetzung auch nie erfüllt, da die Ausgabe der Vote-b-Unit der von u^* geforderte Wert ist. Erst im dritten Schritt können andere E-Prop-Units aktiv geworden sein. Durch den Einfluß der Wichtungsmatrix kommen nur die E-Prop-Units für eine inhibitorische Verbindung in Betracht, die die gleiche Vote-b-Unit, wie auch u^* beeinflussen. Weiterhin muß der durch die E-Prop-Unit u propagierte Wert im Widerspruch zu dem durch u^* propagierten stehen ($o(u) \cdot w(u, u_{f,v_b}) \neq o(u_{f,v_b})$). Sind diese Vorraussetzungen erfüllt, so wird eine inhibitorische Verbindung von u^* nach u eingeführt.

Es werden für jede Unit u^* nur genau drei Update-Schritte durchgeführt. Bei einem vierten Schritt würde die Wirkung der indirekt aktivierten E-Prop-Units auf die Vote-b-Units einsetzen.

Die Anzahl der benötigten Update-Schritte hängt nur linear von der Anzahl der Effekt-Propositionen in der Systembeschreibung ab. Es wird dabei jedoch ein vollständig paralleles System vorausgesetzt, welches über vollständig parallelen Lesezugriff auf Systemressourcen (auf die Adresse von u^*) verfügt. Weiterhin müssen natürlich auch die Manipulationen der Wichtungsmatrix vollständig parallel durchführbar sein. Dies ist ebenfalls möglich, da jede E-Prop-Unit nur maximal eine Verbindung in jedem Lernschritt einführen muß.

Das Lernverfahren garantiert auch, daß nur die E-Prop-Units korrigiert werden, die den gleichen Fluenten, wie die Unit u^*, beeinflussen.

In der erlernten Overrule-Relation stehen nach Abschluß des Lernens nur die E-Prop-Units, die wenigstens über eine Teilmengenbeziehung ihrer Aktionsmengen verfügen. Das es sich bei der erlernten Relation wirklich um die Overrule-Relation entsprechend der Semantik von $\mathcal{A}_C$ handelt, wird im nächsten Abschnitt gezeigt.

Die Menge der E-Prop-Units zerfällt bezüglich jedes Fluenten in zwei Teilmengen. Sei P die Menge der Units, die eine positive Fluentenänderung stützen und N die Menge der Units, die eine negative Änderung stützen. Im ungünstigsten Fall steht jedes Element von P mit jedem Element von N in Beziehung. Somit ist $k \leq |P \times N|$. Es werden $O(n^2)$ mit $n = |P| + |N|$ Verbindungen benötigt. Inwieweit dieses Ergebnis einen Einfluß auf reale Systembeschreibungen besitzt, bleibt zu untersuchen.

Der Lernalgorithmus erfordert eine Einschränkung (siehe Def. 1.10) bezüglich der erlaubten Effekt-Propositionen. Es dürfen keine widersprüchlichen Effekt-Propositionen mit genau gleicher Aktionsmenge in $\mathcal{E}$ enthalten sein. Sei $a \in 2^{\Sigma_2}$, C_1, C_2 Mengen von Fluent-Literalen und f ein Fluent-Literal, dann dürfen folgende Effekt-Propositionen nicht gleichzeitig in $\mathcal{E}$ enthalten sein:

$$a \text{ causes } f \text{ if } C_1,$$
$$a \text{ causes } \neg f \text{ if } C_2.$$

Wenn diese Einschränkung vernachlässigt wird, fängt das Netz zu schwingen an, da sich die widersprüchlichen E-Prop-Units gegenseitig hemmen und im nächsten Zyklus wieder aktiv würden.

3.3 Korrektheit von $\mathcal{N}_T$

Für den Korrektheitsbeweis der Abbildung $\Phi_{\mathcal{N}_T}$, die durch $\mathcal{N}_T$ berechnet wird, ist die Definition einiger Hilfsabbildungen notwendig.

Definition 3.10 Zur Beschreibung von aktiven Units werden folgenden Mengen definiert:

$$\begin{aligned}
\mathcal{A}^+(U) &:= \{u \in U | o(u) > 0\}, \\
\mathcal{A}^-(U) &:= \{u \in U | o(u) < 0\}, \\
\mathcal{A}(U) &:= \mathcal{A}^+(U) \cup \mathcal{A}^-(U).
\end{aligned}$$

■

Definition 3.11 Sei D eine Systembeschreibung und $\mathcal{N}_T$ ein Transitionsnetz entsprechend D, dann sind die Abbildungen τ eineindeutig und wie folgt definiert:

$$\begin{aligned}
&\tau_\sigma : 2^{\Sigma_1} \to 2^{U_F}, \quad \text{so daß} \quad \forall \sigma \in 2^{\Sigma_1}, \forall \mathcal{A}^+(U_F) \in 2^{U_F}. f \in \sigma \leftrightarrow u_f \in \mathcal{A}^+(U_F), \\
&\tau_\alpha : 2^{\Sigma_2} \to 2^{U_A}, \quad \text{so daß} \quad \forall \alpha \in 2^{\Sigma_2}, \forall \mathcal{A}^+(U_A) \in 2^{U_A}. a \in \alpha \leftrightarrow u_a \in \mathcal{A}^+(U_A), \\
&\tau_{\sigma'} : 2^{\Sigma_1} \to 2^{U_V}, \quad \text{so daß} \quad \forall \sigma' \in 2^{\Sigma_1}, \forall \mathcal{A}^+(U_{V_o}) \in 2^{U_{V_o}}. f \in \sigma' \leftrightarrow u_{f,v_o} \in \mathcal{A}^+(U_{V_o}).
\end{aligned}$$

Die Existenz dieser Abbildungen wird durch die Konstruktion von $\mathcal{N}_T$ sichergestellt. (Definition 3.3) τ^{-1} bezeichne die Umkehrabbildung. ■

Satz 3.1 (Anwendbarkeit von Effekt-Propositionen) Sei D eine Systembeschreibung in $\mathcal{A}_C$ und $\mathcal{N}_T$ ein Transitionsnetz, das entsprechend D konstruiert und angelernt wurde. Dann gilt $\forall \sigma \in 2^{\Sigma_1}, \forall a \in 2^{\Sigma_2}, \forall e \in \mathcal{E}$:

$$(e \in \mathcal{E} \text{ anwendbar} \Longleftrightarrow u_e \in \mathcal{A}^+(U_E)) \tag{3.11}$$

□

Beweis 3.1 (Anwendbarkeit) Sei D eine Systembeschreibung in $\mathcal{A}_C$ und $\mathcal{N}_T$ ein Transitionsnetz, das entsprechend D konstruiert und angelernt wurde. Zusätzlich sei $\sigma \in 2^{\Sigma_1}$ ein beliebiger Zustand und $a \in 2^{\Sigma_2}$ eine beliebige, auszuführende Aktion.

Wenn die Effekt-Proposition $e \in \mathcal{E}$ anwendbar ist[2], dann sind alle Vorbedingungen von e erfüllt, die Aktionsmenge der Effekt-Proposition e ist Teilmenge der auszuführenden Aktion und es exsitiert keine anwendbare Effekt-Proposition $e' \in \mathcal{E}$, die e laut Definition 1.3 überstimmt. Durch die Konstruktionsvorschriften für $\mathcal{N}_T$ (Definitionen 3.1 bis 3.9) folgt daraus, daß die Netzeingabe für die zu e korrespondierende E-Prop-Unit u_e maximal wird und den Schwellwert θ_Φ der Unit erreicht. Somit wird die Unit u_e aktiv (Definition 3.4:E-Prop-Units).

Angenommen es existiert eine weitere aktive E-Prop-Unit $u_{e'}$ mit einer inhibitorische Verbindung zu u_e. Daraus folgt, daß ebenfalls eine anwendbare Effekt-Proposition e' existieren muß. Dies steht jedoch im Widerspruch zur Anwendbarkeit von e. Also kann eine solche aktive E-Prop-Unit $u_{e'}$ nicht existieren.

Wenn die E-Prop-Unit u_e aktiv ist, dann sind alle mit dieser Unit verbundenen Aktions-Units ebenfalls aktiv und alle Vorbedingungen (Fluent-Units) erfüllt. Weiterhin kann keine E-Prop-Unit $u_{e'}$ mit einer inhibitorische Verbindung zu u_e aktiv sein, da sonst u_e nicht aktiv wäre. Also ist auch die zu u_e korrespondierende Effekt-Proposition anwendbar und es existiert keine e überstimmende Effekt-Proposition e'. □

Definition 3.12 Sei $\mathcal{N}_T$ ein Transitionsnetz. Die Funktion $\Phi_{\mathcal{N}_T}$ ist genau dann definiert, wenn für eine konstante Netzeingabe alle Vote-o-Units $u \in U_{V_o}$ eine Ausgabe $o(u) \neq 0$ nach spätestens vier synchronen Berechnungszyklen produzieren.

■

[2] e ist dann auch potentiell anwendbar.

Satz 3.2 Sei D eine Systembeschreibung in $\mathcal{A}_C$ und $\mathcal{N}_T$ ein Transitionsnetz, das entsprechend D konstruiert und angelernt wurde. $\forall \sigma \in 2^{\Sigma_1}, \forall a \in 2^{\Sigma_2}$:

$$(\Phi_D(a,\sigma) \text{ ist definiert} \Longleftrightarrow \Phi_{\mathcal{N}_T}(\tau_\alpha(a), \tau_\sigma(\sigma)) \text{ ist definiert}) \tag{3.12}$$

□

Beweis 3.2 Sei D eine Systembeschreibung in $\mathcal{A}_C$ und $\mathcal{N}_T$ ein Transitionsnetz, das entsprechend D konstruiert und angelernt wurde. Seien $\sigma \in 2^{\Sigma_1}$ ein beliebiger Zustand und $a \in 2^{\Sigma_2}$ eine beliebige, auszuführende Aktion.

Φ_D ist genau dann nicht definiert, wenn $\exists f \in \Sigma_1 : f \in (B_f \cap B'_f)$ (siehe Definition 1.6). Es gibt wenigstens zwei anwendbare Effekt-Propositionen e_1 (mit dem Effekt f) und e_2 (mit dem Effekt $\neg f$). Nach Satz 3.1 existierend dann auch zwei aktive E-Prop-Units u_{e_1} und u_{e_2}. Diese wirken ebenfalls auf die zum Fluenten f korrespondierenden Vote-a-, und Vote-b-Units mit negiertem Gewicht. Nach Definition 3.4 erzeugen die Vote-a-Unit und die Vote-b-Unit die Ausgabe $o(u_{f,v_a}) = 1$ und $o(u_{f,v_b}) = 0$. Aus Definition 3.6 folgt nun, daß die Ausgaben der zugehörigen Vote-d- und Vote-o-Units $o(u_{f,v_d}) = o(u_{f,v_o}) = 0$ sind.

Nach Defintion 3.12 ist die durch $\mathcal{N}_T$ berechnete Überführungsfunktion $\Phi_{\mathcal{N}_T}$ nicht definiert, wenn es wenigstens eine Vote-o-Unit mit einer Ausgabe $(o(u \in U_{V_o}) = 0)$ gibt.

Ist Φ_D definiert, dann ist $B_f \cap B'_f = \emptyset$. Dann sind die Ausgaben der bzgl. eines Fluenten zusammengehörenden Vote-a-Units und Vote-b-Units gleich Null und der Wert des Fluenten im Folgezustand wird über die Vote-d-Unit von der Fluent-Unit übernommen oder die Ausgabe der Vote-b-Unit ist von Null verschieden, dann wird der Wert des Fluenten im Folgezustand durch die Ausgabe der Vote-b-Unit bestimmt. In beiden Fällen ist die Ausgabe der Vote-o-Unit von Null verschieden und damit ist auch die Funktion $\Phi_{\mathcal{N}_T}$ definiert. □

Satz 3.3 (Äquivalenz Φ_D und $\Phi_{\mathcal{N}_T}$) Sei D eine Systembeschreibung in $\mathcal{A}_C$ und $\mathcal{N}_T$ ein Transitionsnetz, das entsprechend D konstruiert und angelernt wurde. $\forall \sigma \in 2^{\Sigma_1}, \forall a \in 2^{\Sigma_2}$ gilt:

$$\Phi_D(a,\sigma) \text{ ist definiert} \Longrightarrow \tau_{\sigma'}(\Phi_D(a,\sigma)) = \Phi_{\mathcal{N}_T}(\tau_\alpha(a), \tau_\sigma(\sigma)) \tag{3.13}$$

□

Beweis 3.3 Sei D eine Systembeschreibung in $\mathcal{A}_C$ und $\mathcal{N}_T$ ein Transitionsnetz, das entsprechend D konstruiert und angelernt wurde. Seien $\sigma \in 2^{\Sigma_1}$ ein beliebiger Zustand und $a \in 2^{\Sigma_2}$ eine beliebige, auszuführende Aktion. Weiterhin sei $\Phi_D(a,\sigma)$ definiert.

Also gilt $B_f \cap B'_f = \emptyset$. Sei $f \in \Sigma_1$, dann soll zwischen den folgenden Fällen unterschieden werden.

1. $f \in B_f \cap B'_f$ ist Widerspruch zur Voraussetzung, da Φ_D nicht definiert ist.

2. Wenn $f \in B_f$, dann existiert eine anwendbare Effekt-Proposition $e \in \mathcal{E}$ mit dem Effekt f. Aus Satz 3.1 folgt, daß dann auch eine aktive E-Prop-Unit u_e existiert. Wegen der Voraussetzung $B_f \cap B'_f = \emptyset$ existiert dann keine anwendbare Effekt-Proposition e' mit dem Effekt $\neg f$ und somit auch keine aktive E-Prop-Unit $u_{e'}$ mit Einfluß auf die gleiche Vote-b-Unit wie u_e.

   Aus den Definitionen 3.4 bis 3.6 folgt nun

   $$o(u_e) \cdot w(u_e, u_{f,v_b}) = o(u_{f,v_a}) = o(u_{f,v_b}) = 1. \tag{3.14}$$

   Daraus folgt

   $$o(u_{f,v_d}) = 0 \tag{3.15}$$

   und aus (3.14) und (3.15) folgt

   $$o(u_{f,v_o}) = 1. \tag{3.16}$$

   Somit ist $f \in \mathcal{A}^+_f(U_{V_o}) := \{f | u_{f,vo} \in \mathcal{A}^+(U_{V_o})\}$.

3. Wenn $f \in B'_f$, dann existiert eine anwendbare Effekt-Proposition e mit dem Effekt $\neg f$. Aus Satz 3.1 folgt, daß dann auch eine aktive E-Prop-Unit u_e existiert. Wegen der Voraussetzung $B_f \cap B'_f = \emptyset$ existiert dann keine anwendbare Effekt-Proposition e' mit dem Effekt f und somit auch keine aktive E-Prop-Unit $u_{e'}$ mit Einfluß auf die gleiche Vote-b-Unit wie u_e.

   Aus den Definitionen 3.4 bis 3.6 folgt nun

   $$o(u_e) \cdot w(u_e, u_{f,v_b}) = o(u_{f,v_a}) = o(u_{f,v_b}) = -1. \tag{3.17}$$

   Daraus folgt

   $$o(u_{f,v_d}) = 0 \tag{3.18}$$

   und aus (3.17) und (3.18) folgt

   $$o(u_{f,v_o}) = -1. \tag{3.19}$$

   Somit ist $f \in \mathcal{A}^-_f(U_{V_o}) := \{f | u_{f,vo} \in \mathcal{A}^-(U_{V_o})\}$.

4. Wenn $f \notin (B_f \cup B'_f)$, dann existiert keine anwendbare Effekt-Proposition e mit einem Effekt f bzw. $\neg f$. Daraus folgt, daß auch keine E-Prop-Unit aktiv ist, die auf die Vote-a-Unit und Vote-b-Unit dieses Fluenten einen Einfluß hat.

   Aus den Definitionen 3.4 bis 3.6 folgt nun

   $$o(u_{f,v_a}) = o(u_{f,v_b}) = 0. \tag{3.20}$$

Das hat die Aktivität der Vote-d-Unit zur Folge

$$o(u_{f,v_d}) = o(u_f), \tag{3.21}$$

woraus schließlich der Ausgabewert der Vote-o-Unit

$$o(u_{f,v_d}) = o(u_{f,v_o}) \tag{3.22}$$

folgt.

Somit ist

$$\begin{array}{ll} f \in \mathcal{A}_f^+(U_{V_o}), wenn & f \in \mathcal{A}_f^+(U_F) := \{f|u_f \in \mathcal{A}^+(U_F)\} \\ f \in \mathcal{A}_f^-(U_{V_o}), wenn & f \in \mathcal{A}_f^-(U_F) := \{f|u_f \in \mathcal{A}^-(U_F)\} \end{array} \tag{3.23}$$

5. Durch die Voraussetzung, daß Φ_D und somit auch $\Phi_{\mathcal{N}_T}$ definiert sind, gilt:

$$\forall f \in \Sigma_1 : f \in (\mathcal{A}_f^+(U_{V_o}) \cup \mathcal{A}_f^-(U_{V_o})). \tag{3.24}$$

Nach der Definition 1.6 der Überführungsfunktion Φ_D berechnet sich der Folgezustand σ' wie folgt:

$$\sigma' = (\sigma \cup B_f) \setminus B'_f \tag{3.25}$$

läßt sich wegen $B_f \cap B'_f = \emptyset$ auch darstellen als

$$\sigma' = (\sigma \setminus B'_f) \cup B_f. \tag{3.26}$$

Dies läßt sich zu

$$\sigma' = (\sigma \setminus (B'_f \cup B_f)) \cup B_f \tag{3.27}$$

umformen. Die Menge $B'_f \cup B_f$ entspricht genau den Fluenten, die durch die Ausführung der Aktion betroffen sind. Sei die Menge Γ definiert als

$$\Gamma := \sigma \setminus (B'_f \cup B_f). \tag{3.28}$$

Diese Menge repräsentiert genau jene Fluenten, die durch die Ausführung der Aktion nicht betroffen sind. Sei nun noch

$$\Delta := B_f, \tag{3.29}$$

dann läßt sich 3.26 auch als

$$\sigma' = \Gamma \cup \Delta \tag{3.30}$$

schreiben. Nun ist nur noch zu zeigen, daß wenn $f \in \mathcal{A}_f^+(U_{V_o})$ ist, auch $f \in \sigma'$ gilt und umgekehrt.

6. Wenn $f \in \Delta$ ist, dann ist auch $f \in \mathcal{A}_f^+(U_{V_o})$ (lt. Pkt. 2).

7. Wenn $f \in \Gamma$ ist, dann kann muß $f \notin (B'_f \cup B_f)$ gelten, dann ist $f \in \mathcal{A}_f^+(U_{V_o})$ (lt. Pkt. 4).

8. Wenn $f \in \mathcal{A}_f^+(U_{V_o})$, dann ist $f \in B_f$ oder $f \notin (B'_f \cup B_f)$ (lt. Pktn. 2 und 4).

Also gilt:

$$\sigma' = \Gamma \cup \Delta = \mathcal{A}_f^+(U_{V_o}). \tag{3.31}$$

$\square$

Kapitel 4

Anwendung von $\mathcal{N}_T$

4.1 Aktionsfolgen

Dieses Kapitel behandelt das Problem der Lösung von Planungsaufgaben. Mit dem im vorherigen Kapitel konstruierten Netz ist die Berechnung eines Überführungsschrittes sehr effizient möglich. Ein Plan besteht aus einer Folge von solchen Überführungsschritten, die einen Startzustand in einen Zielzustand überführen sollen.

Es besteht das Problem, daß die Auswahl der anzuwendenden Aktionen in einem bestimmten Schritt falsch gewesen ist und für eine erfolgreiche Lösung des Planungsproblems revidiert werden muß. Folglich ist es notwendig eine Steuerung zu entwickeln, die die Aktionsauswahl „geeignet" durchführt und eigene Entscheidungen rückgängig machen kann.

Eine mögliche Variante wird im nächsten Abschnitt behandelt. Dort wird die eben erwähnte Steuerung in das Berechnungsmodell integriert. Die Grundidee ist, daß ein Plan als Optimierungsproblem dargestellt wird und jeder korrekte Überführungsschritt ein Schritt zum Optimum bedeutet. Das Optimum wird nur dann erreicht, wenn der Start- in den Zielzustand durch eine Folge von anwendbaren Aktionen überführt wurde.

Ein anderer, hier nur kurz angedachter Weg, ist die Konstruktion eines rekurrenten Netzes. Dieses Netz liefert in jedem Schritt eine anwendbare Aktion und generiert somit über der Zeit die gesuchte Aktionsfolge. Das Netz muß zuvor jedoch den Zustandsraum geeignet lernen. Hierzu bietet sich wiederum das Transitionsnetz an. Es erlaubt in konstanter Zeit zu überprüfen, ob eine Aktion in einer gegebenen Situation anwendbar ist. Das rekurrente Netz erhält neben dem aktuellen Zustand zusätzlich den Zielzustand als Eingabe. Damit ist es möglich, aus einer Vielzahl möglicher Aktionen, die im aktuellen Zustand anwendbar wären, die zur Lösung des Problems geeignete auszuwählen.

4.2 Planen als Energieminimierung

Betrachtet man einen Plan als ein Optimierungsproblem, so liegt es nahe, für einen Plan eine Energiefunktion zu definieren, die zu minimieren ist. Das hier betrachtete Planungsverfahren läßt sich in die Aussagenlogik abbilden. Das Erfüllbarkeitsproblem der Aussagenlogik, ist jedoch eine Problemklasse, welche sich mit Hilfe der Energieminimierung lösen läßt. In [Pin90] und [Pin91] zeigte Pinkas die Äquivalenz zwischen der Energieminimierung und dem Erfüllbarkeitsproblem in der Aussagenlogik. Läßt sich ein Problem in Form einer quadratischen Energiefunktion darstellen, so kann man diese mit Hilfe eines Hopfieldnetzes minimieren und so das ursprüngliche Problem lösen. Das Hopfieldnetz läßt sich praktisch direkt aus der Energiefunktion ableiten.

4.2.1 Aussagenlogik und Energiefunktionen

Aussagenlogik

(Wohlgeformte) aussagenlogische Formeln werden mit den Variablen X_i und den Junktoren $\neg$[1] und $\wedge$ wie üblich gebildet (siehe [Sch92]). Zusätzlich seien zur notationellen Vereinfachung auch die Junktoren $\rightarrow$, $\vee$ und $\leftrightarrow$ als Abkürzungen für die entsprechenden Formeln zugelassen.

Im weiteren bezeichne $\mathcal{F}(\bar{X})$ eine aussagenlogische Formel über dem Vektor $\bar{X}$ von Variablen und der Vektor $\bar{x}$ eine Variablenbelegung mit $x_i \in \{0,1\}$. Die Auswertung einer solchen aussagenlogische Formel bezüglich $\bar{x}$, sei induktiv wie folgt definiert:

1. $X_i[\bar{x}] = x_i$
2. $\neg\mathcal{F}[\bar{x}] = 1 - \mathcal{F}[\bar{x}]$
3. $(\mathcal{F}_1 \wedge \mathcal{F}_2)[\bar{x}] = \mathcal{F}_1[\bar{x}] \cdot \mathcal{F}_2[\bar{x}]$
4. $(\mathcal{F}_1 \rightarrow \mathcal{F}_2)[\bar{x}] = 1 - \mathcal{F}_1[\bar{x}] + \mathcal{F}_1[\bar{x}] \cdot \mathcal{F}_2[\bar{x}]$
5. $(\mathcal{F}_1 \vee \mathcal{F}_2)[\bar{x}] = \mathcal{F}_1[\bar{x}] + \mathcal{F}_2[\bar{x}] - \mathcal{F}_1[\bar{x}] \cdot \mathcal{F}_2[\bar{x}]$
6. $(\mathcal{F}_1 \leftrightarrow \mathcal{F}_2)[\bar{x}] = [(\mathcal{F}_1 \rightarrow \mathcal{F}_2) \wedge (\mathcal{F}_2 \rightarrow \mathcal{F}_1)][\bar{x}]$

Die Wahrheitswerte sind: „0“ für falsch und „1“ für wahr.

Die Punkte 4 bis 6 stellen notarielle Vereinfachungen dar.

[1] Das Symbol $\neg$ ist nicht identisch mit dem Negationszeichen für Fluent-Literale.

Beispiel 3 Die folgende Formel sei unter Nutzung der Variablenbelegung [111] an die Variablen $X_1, .., X_3$ nach obiger Vorschrift ausgewertet:

$$(\neg(X_1 \wedge X_2) \vee X_3)[111] = 1. \tag{4.1}$$

□

Im folgenden wird zwischen zwei Klassen von Variablen unterschieden, den sichtbaren Variablen $\bar{X}$ und den internen Variablen $\bar{T}$. Eine Formel wird jetzt mit $\mathcal{F}(\bar{X}, \bar{T})$ bezeichnet.

Definition 4.1 (sichtbares Modell) Sei $\mathcal{F}(\bar{X}, \bar{T})$ eine aussagelogische Formel, $\bar{X}$ die Menge der sichtbaren und $\bar{T}$ der internen Variablen. Weiterhin sei $\bar{x}$ eine Belegung der Variablen in $\bar{X}$.
$\bar{x}$ heißt sichtbares Modell von $\mathcal{F}$ gdw. es ein $\bar{t}$ gibt mit $\mathcal{F}[\bar{x}, \bar{t}\,] = 1$. ■

Definition 4.2 Seien $\mathcal{F}_1$ und $\mathcal{F}_2$ zwei aussagenlogische Formel mit internen Variablen. Dann sind $\mathcal{F}_1$ und $\mathcal{F}_2$ äquivalent bezüglich $\approx_1$, wenn gilt

$$\mathcal{F}_1(\bar{X}, \bar{T_1}) \approx_1 \mathcal{F}_2(\bar{X}, \bar{T_2}) \text{ gdw. } \forall \bar{x} \exists \bar{t_1}, \bar{t_2} : \mathcal{F}_1[\bar{x}, \bar{t_1}] = \mathcal{F}_2[\bar{x}, \bar{t_2}]. \tag{4.2}$$

■

Neben den bekannten konjunktiven, bzw. disjunktiven Normalformen in der Aussagenlogik wird hier eine weitere Normalform definiert.

Definition 4.3 (CTF) CTF (conjunction of triples) sei eine Normalform mit der Struktur

$$\mathcal{F}(\bar{X}) = \bigwedge_{i=1}^{m} \mathcal{F}_i(\bar{X}_i) \text{ mit } \bar{X}_i \subseteq \bar{X}, |\bar{X}_i| \leq 3. \tag{4.3}$$

■

Satz 4.1 Jede aussagenlogische Formel $\mathcal{F}$ läßt sich unter Einführung neuer interner Variablen in eine Formel $\mathcal{F}$' in CTF transformieren, so daß $\mathcal{F} \approx_1 \mathcal{F}'$ gilt. (ohne Beweis, siehe [Sch92]) □

Es läßt sich zeigen, daß diese Transformation in linearer Zeit bzgl. der Länge von $\mathcal{F}$ durchführen ist. Außerdem werden nicht mehr als $O(Länge(\mathcal{F}))$ neue Variablen eingeführt. (siehe [Pin90])

Energiefunktion

Eine Energiefunktion ist eine Abbildung eines n-dimensionalen Raumes auf die reellen Zahlen. Im Fall der Aussagenlogik bedeutet dies:

$$E : \{0,1\}^n \rightarrow \mathbb{R}. \tag{4.4}$$

Eine Energiefunktion E besitzt den Grad k und wird mit $E^k(\bar{X})$ bezeichnet, wenn sie als Summe von Produkttermen dargestellt werden kann, die jeweils aus maximal k Variablen bestehen (z.B. $(\alpha_i x_{i,1} x_{i,2} ... x_{xi,k})$). Es soll wiederum zwischen sichtbaren und internen Variablen unterschieden werden.

Definition 4.4 Seien $\bar{X}$ die sichtbaren, $\bar{T}$ die internen Variablen in einer Energiefunktion $E^k(\bar{X}, \bar{T})$ vom Grade k, so ist $\bar{x}$ eine *sichtbare Lösung* von E^k gdw. es eine Belegung $\bar{t}$ gibt mit

$$E^k[\bar{x}, \bar{t}] = min_{\bar{x'}, \bar{t'}}\{E^k[\bar{x'}, \bar{t'}]\}. \tag{4.5}$$

■

Beispiel 4 Für die Formel

$$E = X_1X_2 - 2X_1T - 2X_2T - 2X_3T + 5T \tag{4.6}$$

existieren u.a. die sichtbaren Lösungen [000], [001] und [111]. □

Ähnlich wie bei den aussagenlogischen Formeln, wird auch über Energiefunktionen eine Äquivalenz definiert.

Definition 4.5 Seien $E_1^{k_1}(\bar{X}, \bar{T_1})$ und $E_2^{k_2}(\bar{X}, \bar{T_2})$ Energiefunktionen. Dann ist

$$E_1^{k_1}(\bar{X}, \bar{T_1}) \approx_2 E_2^{k_2}(\bar{X}, \bar{T_2}) \tag{4.7}$$

gdw. die Mengen der sichtbaren Lösungen identisch sind. ■

Jede Energiefunktion läßt sich in eine Energiefunktion mit einem anderen Grad transformieren, so daß beide Energiefunktionen äquivalent bezüglich $\approx_2$ sind. In dieser Arbeit wird nur eine Transformation benötigt, die eine Verringerung des Grades der Energiefunktion zur Folge hat. Daher wird auch nur dieser Teil betrachtet. Die Vollständige Transformation ist in [Pin90] zu finden.

Definition 4.6 Sei $E^k(\bar{X}, \bar{T})$ eine Energiefunktion. Für die Reduktion des Grades k zu $k-1$ (für $k \geq 3$) sind folgende Substitutionen durchzuführen:
Sei $t = \alpha \prod_{i=1}^{k} X_i$ ein Produkt aus k Variablen. Falls

$\alpha < 0$: setze für
$t := \sum_{i=1}^{k} 2\alpha X_i T - (2k-1)\alpha T$,
$\alpha > 0$: setze für
$t := \alpha \prod_{i=1}^{k-1} X_i - \sum_{i=1}^{k-1} 2\alpha X_i T + 2\alpha X_k T + (2k-3)\alpha T$.

■

Aussagenlogik und Energieminimierung

Definition 4.7 Eine aussagenlogische Formel $\mathcal{F}$ ist zu einer Energiefunktion E^k vom Grad k äquivalent,

$$\mathcal{F}(\bar{X}, \bar{T}) \approx_3 E(\bar{X}, \bar{T}') \tag{4.8}$$

gdw. die Menge der sichtbaren Modelle von $\mathcal{F}$ gleich der Menge der sichtbaren Lösungen von E ist. ■

Für die Konstruktion einer Energiefunktion aus einer aussagenlogischen Formel wird eine Penaltyfunktion $P_{\mathcal{F}(\bar{X})}$ eingeführt. Diese wird genau dann Null, wenn alle Teilkonjunktionen wahr sind. Für jede unerfüllte Konjunktion steigt der Wert der Penaltyfunktion an.

Definition 4.8 Sei $\mathcal{F} = \bigwedge_{i=1}^{m} \mathcal{F}_i$, eine aussagenlogischen Formel in CTF, dann ist die zugehörige Penaltyfunktion

$$P_{\mathcal{F}(\bar{X})} = \sum_{i=1}^{m}(1 - \mathcal{F}_i(\bar{X})). \tag{4.9}$$

■

4.2.2 Energiefunktion und Hopfieldnetz

Zum Abschluß dieses Abschnittes soll anhand eines kleinen Beispiels die Vorgehensweise zur Konstruktion eines Hopfieldnetzes aus einer gegeben aussagenlogischen Formel dargelegt werden.

Beispiel 5 Gegeben sei die Formel $\mathcal{F} = (A \wedge \neg B) \vee (\neg A \wedge B)$.

Zuerst wird $\mathcal{F}$ in die CTF-Form überführt. Die Beispielformel ist schon in CTF-Form, da insgesamt nur zwei Variablen vorkommen.

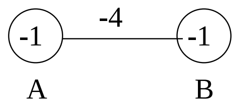
Im nächsten Schritt wird die Energiefunktion konstruiert. Hierzu wird die Penaltyfunktion $E_{\mathcal{F}}$ konstruiert.

$$E_{\mathcal{F}} = 1 - (A + B - 2AB)$$

Ausgehend von der Struktur der Energiefunktion eines Hopfieldnetzes mit den Aktivierungszuständen $\{0, 1\}$, sieht die resultierende Energiefunktion wie folgt aus:

$$\begin{aligned} E_{\mathcal{F}} &= -1/2 \sum_{i \neq j} w_{i,j} x_i x_j + \sum_i \theta_i x_i \\ &= 2AB - A - B + 1 \end{aligned}$$

Die Wichtungsmatrix des zu konstruierenden Hopfieldnetzes kann direkt abgelesen werden, wenn jeder Variablen eine Unit im Netz zugeordnet wird. Der konstante Summand kann weggelassen werden, da er auf die Lage des Energieminimums keinen Einfluß hat.



□

4.2.3 Abbildung von Φ_D auf die Aussagenlogik

Im Abschnitt 4.2.1 wurde gezeigt, daß eine Äquivalenz bezüglich der Erfüllbarkeit von aussagenlogischen Formeln und den Minima von Energiefunktionen besteht. In diesem Abschnitt soll eine aussagenlogische Formel konstruiert werden, die eine Abbildung der Überführungsfunktion Φ_D in $\mathcal{A}_C$ darstellt. Aus dieser Formel wird im folgenden eine Energiefunktion und daraus ein Hopfieldnetz abgeleitet. Auch in diesem Abschnitt sind alle Formeln, wenn nichts anderes erwähnt wird, allquantifiziert.

Aussagenlogisches Äquivalent zu Φ

Definition 4.9 Sei D eine Systembeschreibung in $\mathcal{A}_C$, dann ist die eineindeutige Funktion $\gamma : \mathcal{E} \rightarrow \mathbb{N}$ wie folgt definiert:
Jeder Effekt-Proposition wird eine natürliche Zahl derart zugewiesen, so daß die erste E-Proposition in D die Zahl 1 erhält, die zweite die Zahl 2 usw., die letze E-Proposition erhält die Zahl $|\mathcal{E}|$. ■

Definition 4.10 Seien A, C, E und F aussagelogische Variablen. Sei l_{max} die maximal zu berechnende Planlänge, D eine Systembeschreibung in $\mathcal{A}_C$ und $\mathcal{N}_T$ ein Transitionsnetz, entsprechend D konstruiert und angelernt. Eff(e) ist eine Funktion, die das Fluent-Literal (f_e^*) der Effekt-Proposition e liefert.

Die Formel $\mathcal{F}$ sei nun wie folgt definiert:

$$\mathcal{F} = \bigwedge_{0 \leq l < l_{max}} (\mathcal{F}_{E,l} \wedge \mathcal{F}_{O,l} \bigwedge_{f \in \Sigma_1} \mathcal{F}_{f,l}). \tag{4.10}$$

1. Wobei $\mathcal{F}_{f,l}$ definiert ist als:

$$\mathcal{F}_{f,l} := (\Delta_{f,l} \vee \Gamma_{f,l}) \text{ mit} \tag{4.11}$$

$$\Delta_{f,l} \;:=\; [[(F_f^{l+1} \leftrightarrow \bigvee_{e\in\mathcal{E}.\mathrm{Eff}(e)=f} E^l_{\gamma(e)}) \wedge (\neg F_f^{l+1} \leftrightarrow \bigvee_{e\in\mathcal{E}.\mathrm{Eff}(e)=\neg f} E^l_{\gamma(e)})],$$
$$\Gamma_{f,l} \;:=\; [(F_f^l \leftrightarrow F_f^{l+1}) \wedge \bigwedge_{e\in\mathcal{E}.\mathrm{Eff}(e)=f} \neg E^l_{\gamma(e)} \wedge \bigwedge_{e\in\mathcal{E}.\mathrm{Eff}(e)=\neg f} \neg E^l_{\gamma(e)}]]$$

2. Für alle $e \in \mathcal{E}$ der Form, A_e **causes** f_e^* **if** C_e, wobei $A_e \in 2^{\Sigma_2}$, f_e^* ein Fluent-Literal und C_e eine (möglicherweise leere) Menge von Fluent-Literalen ist, wird $\mathcal{F}_{E,l}$ definiert als:

$$\mathcal{F}_{E,l} := \bigwedge_{e\in\mathcal{E}} \mathcal{F}_{e,l}$$

. Wobei $\mathcal{F}_{e,l}$ eine der folgendenen Formeln ist.

- Ist f_e^* ein positiver Effekt, also f, dann ist

$$\mathcal{F}_{e,l} := [E^l_{\gamma(e)} \leftrightarrow (\bigwedge_{a\in A_e} A^l_a \wedge \bigwedge_{c^*\in C_e} \begin{cases} c^* = c & :F^l_c \\ c^* = \neg c & :\neg F^l_c \end{cases} \wedge \neg E^l_{overrule,\gamma(e)})], \tag{4.12}$$

- Ist f_e^* ein negativer Effekt, also $\neg f$, dann ist

$$\mathcal{F}_{e,l} := [E^l_{\gamma(e)} \leftrightarrow (\bigwedge_{a\in A_e} A^l_a \wedge \bigwedge_{c^*\in C_e} \begin{cases} c^* = c & :F^l_c \\ c^* = \neg c & :\neg F^l_c \end{cases} \wedge \neg E^l_{overrule,\gamma(e)})]. \tag{4.13}$$

Wobei c^* Fluent-Literale im Vorbedingungsteil der Effekt-Propositionen sind. c ist der Fluentbezeichner von c^* ohne führendes $\neg$-Symbol.

3. Die Anwendbarkeit einer potentiell anwendbaren Effekt-Proposition wird durch die Overrule-Relation geregelt. $\mathcal{F}_{O,l}$ sei definiert als:

$$\mathcal{F}_{O,l} := \bigwedge_{e\in\mathcal{E}} \mathcal{F}_{O,e,l}.$$

- Es existiert ein $u_{e'} \in U_E$ mit $w(u_{e'}, u_e) < 0$, dann

$$\mathcal{F}_{O,e,l} := (E^l_{overrule,\gamma(e)} \leftrightarrow (\bigvee_{u_{e'}\in U_E \wedge w(u_{e'},u_e)<0} E^l_{\gamma(e')}). \tag{4.14}$$

- Es existiert kein $u_{e'} \in U_E$ mit $w(u_{e'}, u_e) < 0$, dann

$$\mathcal{F}_{O,e,l} := (E^l_{overrule,\gamma(e)} \leftrightarrow \text{false}). \tag{4.15}$$

■

Für die Modellfindung interessieren nur die Modelle, die auch Plan darstellen und mit der Aufgabenstellung übereinstimmen. Demzufolge werden einige Variablen mit einem bestimmten Wahrheitswert fixiert und für die übrigen Variablen wird ein Wahrheitswert gesucht.

Wurde eine Belegung gefunden, die jede Teilkonjunktion erfüllt, dann erfüllt sie auch $\mathcal{F}$ und stellt somit einen gültigen Plan dar.

Die Variablen in $\mathcal{F}$ werden in zwei Klassen eingeteilt. Die sichtbaren Variablen, sind die Variablen, die auf die Zustände und Aktionen abgebildet werden. Alle anderen Variablen sind interne Variablen und könne während des Modellfindungsprozesses beliebig manipuliert werden.

Die sichtbaren Variablen, die die aktuelle Situation beschreiben (F_f^l), korrespondieren mit den Fluent-Units in $\mathcal{N}_T$. Jene, die die Folgesituation beschreiben (F_f^{l+1}), mit den Vote-o-Units.

Die E-Prop-Units korrespondieren mit den $E_{\gamma(e)}^l$ Variablen und die Aktions-Units mit den A_a^l Variablen.

Nun entsprechen die Aktivierungen der Fluent-Units und der Vote-o-Units für den Fall, daß Φ_D bei gegebenem Zustands-Aktions-Paar definiert ist, genau den Wahrheitswerten der korrespondierenden Variablen. Die E-Prop-Units berechnen gerade die in Definition 4.10 angegebene Konjunktion und können nur durch eine weitere aktive E-Prop-Unit an der Aktivierung gehindert werden. Diese Hinderung wird durch die Overrule-Formeln beschrieben, die ja direkt aus der Wichtungsmatrix konstruiert wurden. Es reicht eine weitere anwendbare Effekt-Proposition oder aktive E-Prop-Unit aus, damit die aktuelle E-Prop-Unit nicht aktiv werden kann.

Der Formelteil, der die Fluentenänderung bestimmt, wird insgesamt nur dann wahr, wenn entweder der Wert eines Fluenten durch die Anwendung einer oder mehrerer Effekt-Propositionen eindeutig bestimmt wird oder keine diesen Fluenten beeinflußende Effekt-Proposition anwendbar ist, dann muß jedoch der Wert des Fluenten im Folgezustand gleich dem Wert im aktuellen Zustand sein.

Auf einen formalen Beweis soll an dieser Stelle aus Gründen des Umfanges der Arbeit verzichtet werden.

4.2.4 Abschließendes Beispiel

Zur Veranschaulichung der Konstruktion eines Hopfieldnetzes aus einer Systembeschreibung werden in diesem Abschnitt die einzelnen Schritte nochmals zusammengefaßt. Das Beispiel ist allerdings nicht ganz vollständig, da aus Gründen des Umfanges darauf verzichtet wurde, die Wichtungsmatrix des Hopfieldnetzes tatsächlich auszurechnen. Es wird davon ausgegangen, daß die Transformationen ab der aussagenlogischen Formel $\mathcal{F}$ rein formal und somit nicht sonderlich instruktiv sind.

Beispiel 6 Gegeben sei folgende Systembeschreibung D_2. Es handelt sich dabei um eine gefüllte Suppenschüssel, die auf einem Tisch steht. Es können zwei Aktionen ausgeführt werden. Diese sind „links anheben" und „rechts anheben". Wird nur eine Aktion ausgeführt, wird die Suppe in der Schüssel verschüttet.

$$\begin{aligned} \Sigma_1 &= \{\mathit{spilled}\} \\ \Sigma_2 &= \{\mathit{left},\ \mathit{right}\} \end{aligned}$$

initially not spilled

(1) $\{\mathit{left}\}$ **causes** *spilled*
(2) $\{\mathit{right}\}$ **causes** *spilled*
(3) $\{\mathit{left},\ \mathit{right}\}$ **causes** $\neg \mathit{spilled}$ **if** $\neg \mathit{spilled}$

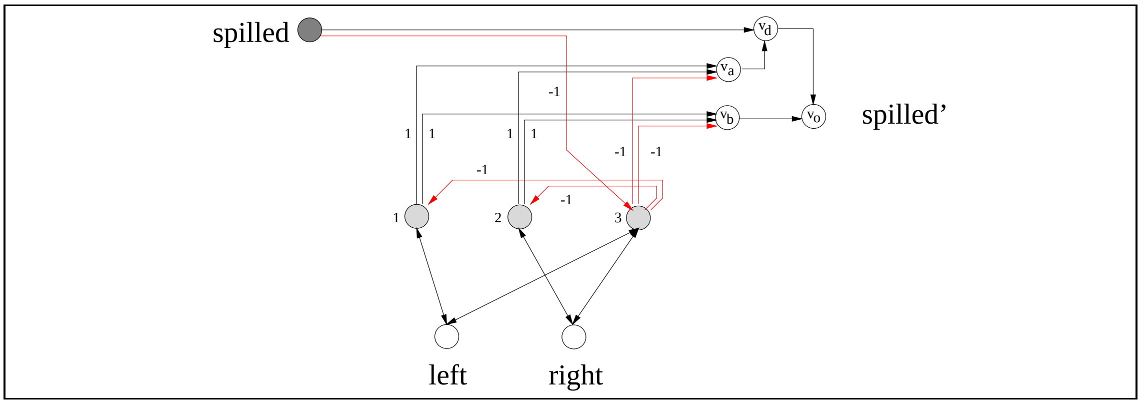


Abbildung 4.1: Das aus D_2 resultierende Transitionsnetz.

Das aus D_2 konstruierte Transitionsnetz ist in der Abbildung 4.1 zu sehen. Durch den Lernalgorithmus wurden die inhibitorischen Verbindungen von der E-Prop-Unit u_{e3} zu den E-Prop-Units u_{e1} und u_{e2} eingeführt.

Im nächsten Schritt wird die aussagenlogische Formel $\mathcal{F}$ für Pläne der Länge zwei ($l_{max} = 2$) konstruiert und daraus dann die Wichtungsmatrix für ein Hopfieldnetz abgeleitet.

1. Aufstellen der aussagenlogischen Formel entsprechend der maximal zu berechnenden Planlänge.
   $\mathcal{F}$ enthält für jeden Überführungsschritt eine Gruppe von Konjunktionen, die ihrerseits wieder strukturiert sind.

$$\mathcal{F} \quad = \quad \bigwedge_{0 \le l < 2} (\mathcal{F}_{E,l} \wedge \mathcal{F}_{O,l} \bigwedge_{f \in \Sigma_1} \mathcal{F}_{f,l})$$

Die Teilkonjunktionen für die Anwendung der Effekt-Propositionen im Schritt l besitzen die Struktur:

$$\begin{aligned}
\mathcal{F}_{E,l} &:= (E_1^l \leftrightarrow A_{left}^l \wedge \neg E_{over,1}^l) \\
&\wedge\ (E_2^l \leftrightarrow A_{right}^l \wedge \neg E_{over,2}^l) \\
&\wedge\ (E_3^l \leftrightarrow A_{left}^l A_{right}^l \wedge \neg E_{over,3}^l)
\end{aligned}$$

Die Overrule-Relation wird durch folgende Teilkonjunktionen realisiert:

$$\begin{aligned}
\mathcal{F}_{O,l} &:= (E_{over,1}^l \leftrightarrow E_3^l) \\
&\wedge\ (E_{over,2}^l \leftrightarrow E_3^l) \\
&\wedge\ (E_{over,3}^l \leftrightarrow false)
\end{aligned}$$

Zur Repräsentation der Veränderung des Wertes eines Fluenten im Folgezustand bei Anwendung einer Effekt-Proposition oder aber der Übernahme aus dem aktuellen Zustand, wenn keine Effekt-Proposition anwendbar ist, werden die nun folgenden Teilkonjunktion erzeugt:

$$\begin{aligned}
\mathcal{F}_{spilled,l} &:= [(F_{spilled}^{l+1} \leftrightarrow E_1^l \vee E_2^l) \wedge (\neg F_{spilled}^{l+1} \leftrightarrow E_3^l)] \\
&\vee\ [(F_{spilled}^l \leftrightarrow F_{spilled}^{l+1}) \wedge \neg E_1^l \wedge \neg E_2^l \wedge \neg E_3^l]
\end{aligned}$$

2. Die soeben definierten Formeln werden jetzt in die CTF-Form gebracht. Hierdurch werden neue interne Variablen eingeführt. Diese sind ebenfalls mit dem Index l zu versehen, da die gesamte Formel durch „aufrollen“ entsteht und auch die internen Variablen vervielfacht werden müssen.

3. Nachdem die Formel $\mathcal{F}$ in CTF dargestellt wurde, kann die Penalty-Funktion aufgestellt werden. Die nun entstehende Energiefunktion ist maximal vom Grad drei.

4. Wenn die Energiefunktion vom Grad drei ist, muß sie nun in eine Energiefunktion vom Grad zwei transformiert werden. Auch dies geschieht wieder unter Einführung neuer interner Variablen.

5. „Aufrollen“ der Darstellung von $\mathcal{F}$. Durch die Ersetzung des Parameters l werden die Fluent-Variablen der Folgezustände gekoppelt.

6. Aus der nun aufgerollten Darstellung wird die Wichtungsmatrix für das Hopfieldnetz abgelesen.

□

Kapitel 5

Zusammenfassung

In dieser Arbeit wurde gezeigt, daß sich eine Systembeschreibung D in $\mathcal{A}_C$ auf ein neuronales Netz $\mathcal{N}_T$ abbilden läßt und das Netz $\mathcal{N}_T$ eine zur Überführungsfunktion Φ_D äquivalente Berechnung durchführt. Diese Abbildung kann in linearer Zeit bezüglich der Elemente in der Systembeschreibung durchgeführt werden, wenn elementare Operationen zur Generierung des Neuronalen Netzes parallelisiert wurden.

Das Lernverfahren, welches auf das Netz angewendet wird, benötigt ebenfalls nur linear viele Schritte bezüglich der Anzahl der Effekt-Propositionen in D.

Das Ergebnis ist ein Neuronales Netz, das die durch die Semantik von $\mathcal{A}_C$ definierte Zustandsüberführung in konstanter Zeit berechnet. Dieser Geschwindigkeitsvorteil wurde mit dem höheren Verbrauch an Speicherplatz bezahlt. Die Anzahl der notwendigen Wichtungen zwischen den E-Prop-Units kann im ungünstigsten Fall $O(|\mathcal{E}|^2)$ werden. Es besteht jedoch die Vermutung, daß für reale Szenarien dieser Worst-Case-Fall nicht eintritt, denn in der realen Welt ist die vorherrschende Eigenschaft die Ähnlichkeit. Der Worst-Case setzt aber voraus, daß bei jeder weiteren Aktion der Effekt jedesmal invertiert wird. Die Anzahl der benötigten Units in $\mathcal{N}_T$ ist

$$N = 5 \cdot |\Sigma_1| + |\Sigma_2| + |\mathcal{E}|$$

und somit linear.

Im Kapitel 4 wurde eine Anwendung des zuvor konstruierten Transitionsnetzes in einem Planungssystem vorgestellt. Dazu war es notwendig, eine Sequenz von Zustandsübergängen zu berechnen. Da das Transitionsnetz nicht in der Lage ist die Umkehrfunktion von Φ zu berechnen (diese ist außerdem nicht eindeutig), wurde die Überführungsfunktion auf eine aussagenlogische Formel $\mathcal{F}$ abgebildet. Diese Abbildung entstammt zwei Quellen. Zum einen ist es die Systembeschreibung D und zum anderen das schon angelernte Netz $\mathcal{N}_T$. Diese Zweiteilung macht deshalb Sinn, weil die Grundstruktur in einfacher Weise aus der Syntax der Systembeschreibung abgeleitet werden kann. $\mathcal{F}$ wurde so konstruiert, daß jeder Effekt-Proposition zusätzlich eine Variable hinzugefügt wurde, welche die Anwendbarkeit der Effekt-Proposition beschreibt. Der Unterschied zwischen einer anwendbaren und eine nur potentiell anwendbaren Effekt-Proposition besteht darin, daß die zweite von einer anderen anwendbaren Effekt-Proposition überstimmt wurde. Diese Überstimmungsrelation ist jedoch sequentiell nur aufwendig zu berechnen, da viele Mengenoperationen durchzuführen sind. Das Netz $\mathcal{N}_T$ hat diese Berechnung schon parallel mit linearem zeitlichen Aufwand durchgeführt, so daß die zusätzlichen Formeln in $\mathcal{F}$ direkt aus der Wichtungsmatrix von $\mathcal{N}_T$ abgelesen werden können.

Wichtig ist, daß die Übertragung der Wichtungen in die Formeln nicht sequentialisiert wird, da dadurch die Anzahl der Wichtungen $O(|\mathcal{E}|^2)$ sein könnte. Ziel muß es sein, daß für jede Effekt-Proposition zumindest ein Prozessor existiert, der die Transformation Wichtungsmatrix $\mathcal{N}_T \rightarrow$ Formel $\mathcal{F} \rightarrow$ Wichtungsmatrix Hopfieldnetz durchführt.

Ein wesentlicher Vorteil einer Planfindung mit einem Hopfieldnetz ist, daß als Initialsituation schon Teillösungen angegeben werden können. Dies bedeutet im Sinne der Energieminimierung eine Vorgabe eines Startpunktes in der Energielandschaft. Da das normale Hopfieldnetz nur ein Abstieg zum nächsten Energieminimum durchführt, können solche Hinweise durchaus sinnvoll sein.

Andererseits wurde in [Roj93] bemerkt, daß bei einer ungünstig geformten Energielandschaft mit vielen lokalen Energieminima sich das Hopfieldnetz sehr häufig nur in lokale Minima bewegt und das Optimum nicht findet.

Abschließend ist noch zu bemerken, daß die Anwendung eines Hopfieldnetzes zur Berechnung der Modelle nicht der einzige Weg ist. In [HHSW94] wurden ebenfalls Algorithmen für die Lösung von Erfüllbarkeitsproblemen der Aussagenlogik behandelt. Teilweise lassen sich auch die dort angegebenen Algorithmen sehr gut parallelisieren. Die Anwendung Genetischer Algorithmen wurde in [Gro95] unteRucht.

Einen völlig anderer Weg kann beschritten werden, wenn eine rekurrente Architektur zum Einsatz kommt, die den eigentlichen Zustandsraum und die Transitionen in einem solchen lernt. Das zielgerichtete Vorgehen aus einem gegebenen Zustand in einen Zielzustand, kann über die Generalisierung von Zuständen erfolgen. Es müßten sich also Standardwege im Zustandsraum herausbilden, die hin und wieder durch Sonderfälle unterbrochen wären.

Ein ähnlich gelagertes Problem ist die Zeitreihenanalyse. Es wäre bedenkenswert, die dort gefundenen Lösungen auf das hier betrachtete Planungsproblem anzuwenden.

Literaturverzeichnis

[AK93] Khayri A. M. Ali and Roland Karlsson. Scheduling speculativ work in MUSE and performance results. Internal report, SICS, 1993.

[BG] Chitta Baral and Michael Gelfond. Representing concurrent actions in extended logic programming. Morgan Kaufmann. In Proceedings of the International Conference on Artificial Intelligence (IJCAI) 1993, S.866-871.

[BHS93] Wolfgang Bibel, Steffen Hölldobler, and T. Schaub. *Wissensrepräsentation und Inferenz*. Vieweg-Verlag, 1993.

[Bor94] Sven Erik Bornscheuer. Gleichzeitige Aktionen in einem ELP-basierten Planungsmodell, 1994. Diplomarbeit, TH-Darmstadt.

[Brä93] Thomas Bräunl. *Parallele Programmierung : Eine Einführung*. Vieweg, Braunschweig/Wiesbaden, 1993.

[BT94] Sven Erik Bornscheuer and Michael Thielscher. Representing concurrent actions and solving conflicts. 1994.

[GL92] Michael Gelfond and Vladimir Lifschitz. Representing actions in extended logic programming. In *Proceedings of IJCSLP'92*, pages 559–573. 1992.

[Gö95] Günther Görz (Hrsg.). *Einführung in die künstliche Intelligenz*. Addison-Wesley, 1995.

[Gre93] Steve Gregory. Experiments with speculative parallelism in Parlog. In Dale Miller, editor, *Logic Programming – Proceedings of the 1993 International Symposium*, pages 370–387. MIT Press, 1993.

[Gro95] Axel Großmann. Genetische Algorithmen und SAT-Probleme – Untersuchungen über den Einsatz evolutionärer Verfahren zur Lösung des Erfüllbarkeitsproblems für die Aussagenlogik, 1995. Großer Beleg: TU-Dresden.

[HHSW94] Steffen Hölldobler, Holger Hoos, Antje Strohmaier, and Andreas Weiss. The GSAT/SA Family – relating greedy satisfiability testing to simulated annealing. *Symbolic Computation*, 1994.

[Kal94] Yvonne Kalinke. Ein massiv paralleles Berechnungsmodell für normale logische Programme, 1994. Diplomarbeit, TU-Dresden.

[Kar92] Roland Karlsson. *A High-Performance OR-parallel Prolog System*. PhD thesis, The Royal Institute of Technology (KTH), Department of Telecommunication and Computerscience, 1992.

[KL93] G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects. Technical report, University of Texas at Austin, 1993.

[Lif94] Vladimir Lifschitz. A language for describing actions. Technical report, Department of Computer Science and Department of Philosophy, University of Texas at Austin, 1994.

[Llo87] John Wylie Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin Heidelberg New York, 1987.

[LMOS93] Edwing Lusk, Shyam Mudambi, Ross Overbeek, and Peter Szeredi. Applications of the aurora parallel prolog sytsem to computational molecular biology. In Dale Miller, editor, *Logic Programming – Proceedings of the 1993 International Symposium*, pages 353–369. MIT Press, 1993.

[NKK94] Detlef Nauck, Frank Klawoon, and Rudolf Kruse. *Neuronale Netze und Fuzzy-Systeme*. Vieweg-Verlag, Braunschweig/Wiesbaden, 1994.

[Pin90] Gadi Pinkas. The equivalence of energy minimization and propositional calculus satisfiability. Technical report, Center for Intelligent Computing Systems – Department of Computer Science, Washington University, 1990.

[Pin91] Gadi Pinkas. Symmetric neural networks and logic satisfiability. *Neural Computation*, (3), 1991.

[Roj93] R. Rojas. *Theorie der neuronalen Netze – Eine systematische Einführung*. Springer-Verlag, 1993.

[Sch92] Uwe Schöning. *Logik für Informatiker*. BI-Wissenschaftsverlag – Mannheim, Leipzig, Wien, Zürich, 1992.

[Sin93] Raed Yousef Sindaha. Branch-level Scheduling in Aurora: The Dharma Scheduler. In Dale Miller, editor, *Logic Programming – Proceedings of the 1993 International Symposium*, pages 403–419. MIT Press, 1993.