FH Bielefeld
University of
Applied Sciences

AMMO
Angewandte Mathematische Modellierung & Optimierung

AMMO – Berichte aus Forschung und Technologietransfer

Formalismen für gefärbte Petri-Netze und Verfahren zur effizienten Bestimmung von aktiven Modus-Mengen

Julian Silberberg
Timo Lask
Bernhard Bachmann

Heft Nr. 8
Juli 2016

Veröffentlichungsreihe (Onlinepublikation):
AMMO – Berichte aus Forschung und Technologietransfer

ISSN
2198-4824

Erscheinungsort
http://www.fh-bielefeld.de/ammo/veroeffentlichungen/ammo-berichte-aus-forschung-und-technologietransfer

Herausgeber
Sprecher FSP AMMO, Fachhochschule Bielefeld

Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
FSP Angewandte Mathematische Modellierung und Optimierung
Interaktion 1
33619 Bielefeld

Vorwort

Das CPN-Team[1] des Forschungsschwerpunktes *Angewandte Mathematische Modellierung und Optimierung* (AMMO) hat 2014 als Band 2 der Forschungsreihe des Fachbereichs Ingenieurwissenschaften und Mathematik der Fachhochschule Bielefeld [ISSN 2196-6192] Lösungsansätze für allgemeine Konfliktsituationen bei Feuerprozessen in Petri-Netz-Modellen veröffentlicht [BKK+14]. Daran anknüpfend sollen in diesem Werk die gefärbten Petri-Netze weiterführend behandelt werden.

In diesem Werk findet einerseits eine Verallgemeinerung der gefärbten Petri-Netze statt. Dadurch kann im Zuge der Modellbildung viel Variabilität gewonnen werden. Andererseits jedoch ist es bei gefärbten Petri-Netzen notwendig für die erfolgreiche Abbildung dynamischer Geschehnisse, dass alle vorhandenen Variablen durch konstante Ausdrücke ersetzt werden, um einen Feuerprozess durchzuführen. Die vollständige Ersetzung aller Variablen wird *Modus* genannt. Für die Berechnung eines Modus ist speziell in größeren gefärbten Petri-Netzen ein vergleichsweise hoher Aufwand notwendig, um ein Ergebnis zu erhalten. Des Weiteren kann in der Regel zwischen mehreren *Modi* gewählt werden. Außerdem ist die Berücksichtigung vorgegebener Restriktionen ein weiterer Aspekt, wodurch die Komplexität der Berechnung weiter gesteigert wird. Diese Aspekte sollen in diesem Werk näher untersucht werden. Grundsätzlich soll effizient ein Modus ermittelt werden, welcher einen Feuerprozess ermöglicht. Auch die effiziente Bestimmung mehrerer feuerbarer Modi wäre wünschenswert.

Aufgrund der engen Anlehnung an [BKK+14] wird es für einen Leser dieses Heftes sicherlich eine Hilfe sein, sich das Buch zwecks Nachschlagewerk herunterzuladen: `http://www.fh-bielefeld.de/fb3/presse/veroeffentlichungen`.

Des Weiteren sind die Inhalte des 2. Kapitels eine Zusammenfassung der Inhalte aus [Sil15], für eine ausführliche Erläuterung sei auf diese verwiesen.

[1] CPN steht für **C**oloured **P**etri **N**et.

Autoren

Julian Silberberg (B. Sc.)
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
Masterstudent im Studiengang Optimierung und Simulation
julian.silberberg@fh-bielefeld.de

Fachgebiete: Petri-Netze, Numerische Mathematik

Absolvierte sein Bachelor-Studium in der *Angewandten Mathematik* an der Fachhochschule Bielefeld. Spezialisierte sich im Laufe des Studiums und einer WHK-Tätigkeit auf das Gebiet der Petri-Netze. Studiert momentan im Master-Studiengang *Optimierung und Simulation* an der Fachhochschule Bielefeld.

Timo Lask (M. Sc.), geb. Kleine-Döpke[1]
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
timo.kleine-doepke@fh-bielefeld.de

Fachgebiete: Petri-Netze, Personalplanung, Echtzeitsysteme

Absolvierte sein Master-Studium im Fach *Optimierung und Simulation* und sein Bachelor-Studium in der *Angewandten Mathematik* an der Fachhochschule Bielefeld. Arbeitete seit der Mitte seines Bachelor-Studiums als SHK im FSP-AMMO-Projekt „Coloured Petri Nets“ (CPN) von Prof. Dr. Hermann-Josef Kruse und hat sich in den letzten Jahren unter anderem in dieser Fachrichtung spezialisiert. Seit Ende 2014 an der Fachhochschule Bielefeld als wissenschaftlicher Mitarbeiter in Forschungsprojekten beschäftigt und seit Januar 2015 ein festes AMMO-Mitglied.

[1]Die Beteiligung an diesem Heft wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 02PQ2313 und 03FH005SX5 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Prof. Dr. phil. Bernhard Bachmann
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
bernhard.bachmann@fh-bielefeld.de

Fachgebiete: Numerische Mathematik, Optimierung, Symbolische und numerische Behandlung großer hybrider differential-algebraischer Gleichungen

Seit 1999 als Professor für Mathematik und ihre technischen Anwendungen an der Fachhochschule Bielefeld tätig und lehrt dort im Bachelor-Studiengang *Angewandte Mathematik* und im Master-Studiengang *Optimierung & Simulation* des Fachbereichs *Ingenieurwissenschaften & Mathematik*. Gründungsmitglied des Forschungsschwerpunktes *Angewandte Mathematische Modellierung & Optimierung* (FSP AMMO) der FH Bielefeld. Gründungsmitglied der *Modelica Association*. Gründungs- und Vorstandsmitglied des *Open Source Modelica Consortium* (OSMC). Nationale und internationale F&E-Projekte im Bereich der Modellierung, Simulation und Optimierung hybrider dynamischer Systeme.

Inhaltsverzeichnis

1 Einführung in gefärbte Petri-Netze

Es werden zunächst die wichtigsten Grundbegriffe aus dem Gebiet der Petri-Netze in enger Anlehnung an den Formalismus aus [BKK+14] eingeführt.

1.1 Gefärbte Petri-Netze ohne Variablen

Zunächst wird eine schematische Herleitung der gefärbten Petri-Netze (CPN) aus den Petri-Netzen (PN) an einem einfachen Beispiel demonstriert.[1]

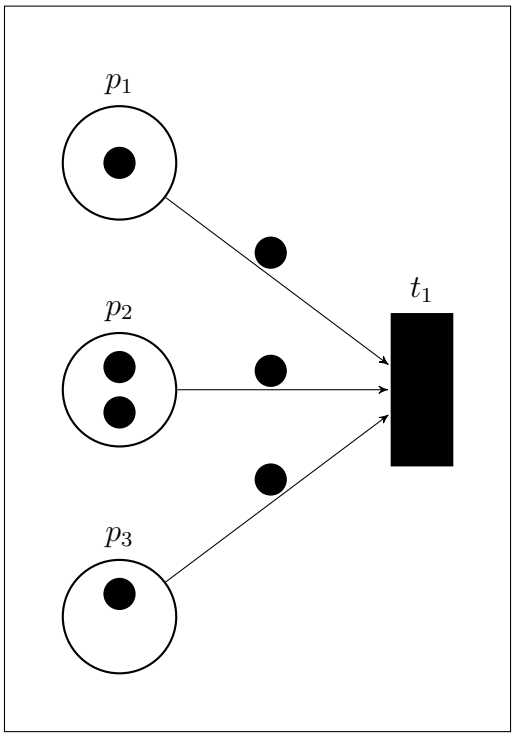

Abbildung 1.1: „Normales“ Petri-Netz

Betrachtet man das Petri-Netz aus Abb. 1.1, so sind alle Token im Netz „gleich“ und an sich nicht unterscheidbar; eine Unterscheidung ist nur möglich anhand der Plätze bzw. Pfeile, in bzw.

[1] Für eine ausführlichere Herleitung sei hier auf [Ger04] verwiesen.

an denen sie sich befinden. In der Anwendung könnten mögliche Interpretationen beispielsweise sein: p_1 enthält Schrauben, p_2 Holzbretter und p_3 Muttern, was durch eine extra Beschriftung klar gemacht werden müsste. Daher stellt sich die Frage: Wenn die Token unterschiedliche Objekte darstellen, warum unterscheiden sich die einzelnen Token nicht von andersartigen Token? Man stelle sich vor, dass nicht nur drei Komponenten in einem Kleinteilelager vorhanden sind, sondern dass man 500 oder mehr verschiedenartige Komponenten für unterschiedliche Produktionen zu bevorraten hat, wozu bei einer Modellierung von entsprechenden Produktionszusammenhängen ein relativ großes und unübersichtliches PN entstehen würde. Genau diese Probleme beheben CPN. Um aus einem PN ein CPN zu erhalten, werden mehrere gleiche Teile des Netzes zu einer vereinfachten Struktur durch „Faltung“ zusammengefasst, z.B. die drei Plätze aus Abb. 1.1. Damit aber die Unterscheidbarkeit erhalten bleibt, werden die Token „gefärbt“. Dafür wird jedem Platz eine andere Farbe zugewiesen, z.B.

$$\begin{aligned} p_1 &\rightarrow \text{ROT} \\ p_2 &\rightarrow \text{BLAU} \\ p_3 &\rightarrow \text{GRÜN} \end{aligned}$$

Nun wird jedes Token, das in einem Platz als Zustand oder an den Pfeilen als Gewichtung steht, in dieser Farbe gefärbt. Danach werden alle Plätze zu einem „gefärbten“ Platz zusammengefasst; dementsprechend werden auch alle Pfeilgewichte aggregiert (vgl. Abb. 1.2).

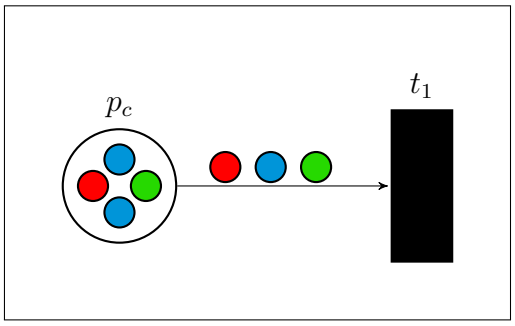

Abbildung 1.2: Gefärbtes Petri-Netz nach Faltung

In dieser Art kann man sich zunächst einmal den originären Zusammenhang zwischen PN und CPN bildlich vorstellen. Formal lassen sich CPN in unterschiedlichen Komplexitätsstufen definieren. Im Folgenden werden die Token nicht nur mit einem einzigen Attribut (nämlich mit dem Attribut „Farbe“) versehen, sondern es werden mehrere verschiedene Attribute definiert und den einzelnen Token auf den Plätzen bzw. Pfeilen zugewiesen (vgl. [KD13]). Hierbei wird der Begriff der *Multimenge* verwendet (Grundlagen hierzu siehe Anhang).

Als erstes soll ein *Netz* definiert werden.

Definition 1.1:
Ein ***Netz*** N ist ein Tupel $N = (P, T, F, B)$ mit einer endlichen Menge $P = \{p_1, \ldots, p_m\}$ und einer endlichen Menge $T = \{t_1, \ldots, t_n\}$, wobei $P \cap T = \emptyset$; zudem gelte $F \subseteq P \times T$, $B \subseteq T \times P$. Die Elemente aus P werden ***Plätze*** (places) und die Elemente aus T werden ***Transitionen*** (transitions) genannt. Die Elemente aus $F \cup B$ heißen ***Pfeile*** oder ***Bögen*** (arcs) oder ***gerichtete Kanten*** von N.

Offensichtlich stellt $G = (V, E)$ mit $V = P \cup T$ und $E = F \cup B$ einen 2-gefärbten Digraph dar.

Aufbauend auf dem Netz wird nun ein *Netz mit Klassen* definiert.

Definition 1.2:
Ein ***Netz mit Klassen*** ist ein Tupel $N = (P, T, F, B, A, K, k)$, wobei

- Das Tupel (P, T, F, B) ein Netz gemäß Def. 1.1 bildet;
- $A = \{A_1, \ldots, A_y\}$ eine endliche, nichtleere Menge von sog. ***Attributen*** ist, wobei die Attribute A_i ihrerseits endliche, nichtleere Mengen sind, die auf kanonische Weise geordnet sind und als Tupel dargestellt werden können: $A_i \equiv (a_1^i, \ldots, a_{|A_i|}^i)$, $i = 1, \ldots, y$, wobei die Elemente der Attribute auch ***Ausprägungen*** des jeweiligen Attributes genannt werden; dabei heißt
  $\mathbb{A} := \bigcup_{A_i \in A} A_i$ die ***Vereinigungsmenge aller Attribute***;
- $K = \{K_1, \ldots, K_x\}$ eine endliche, nichtleere Menge von sog. ***Klassen*** K_j ist, wobei K_j ein kartesisches Produkt aus Attributen einer Teilmenge von A ist, d.h.
  $K_j = A_{j_1} \times \ldots \times A_{j_s}$ mit $\{j_1, \ldots, j_s\} \subseteq \{1, \ldots, y\}$ für $j = 1, \ldots, x$; dabei heißt
  $\mathbb{K} := \bigcup_{K_j \in K} K_j$ die ***Vereinigungsmenge aller Klassen***;
- $k : P \to K$ eine Abbildung ist (***Klassifizierungsfunktion***), die jedem Platz $p \in P$ eine Klasse aus der Klassenmenge K zuweist.

Beispiel:
In diesem Beispiel wird ein Netz mit Klassen $N = (P, T, F, B, A, K, k)$ konstruiert:

1. Das Netz $N = (P, T, F, B)$ mit:

$$P = \{p_1, p_2, p_3, p_4, p_5\}$$
$$T = \{t_1, t_2\}$$
$$F = \{(p_1, t_1), (p_2, t_1), (p_3, t_2)\}$$
$$B = \{(t_1, p_3), (t_2, p_4), (t_2, p_5)\}$$

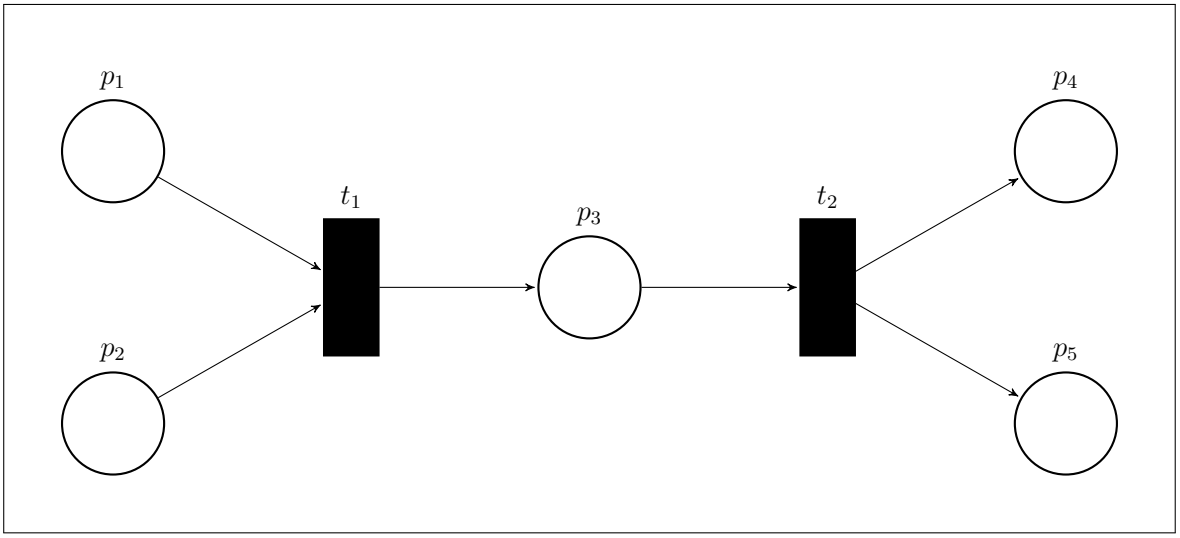

Abbildung 1.3: Beispiel eines Netzes

2. Die Attribute:

   - Es gibt zwei Attribute in $A = \{\text{COL,NUM}\}$ mit:

     $\text{COL} = \{\text{ROT,BLAU,GRÜN}\}$
     $\text{NUM} = \{1, 2, 3\}$

   - Demnach ergibt sich als Vereinigungsmenge aller Attribute:

     $\mathbb{A} = \{\text{ROT}, \text{BLAU}, \text{GRÜN}, 1, 2, 3\}$.

3. Die Klassen:

   - Aus den Attributen lassen sich auf kanonische Weise die folgenden drei Klassen erzeugen:
     K= $\{$FARBE,ZAHL,BUNTEZAHL$\}$ mit:

     $\text{FARBE} = \{(\text{ROT}),(\text{BLAU}),(\text{GRÜN})\} \equiv \text{COL} = \{\text{ROT,BLAU,GRÜN}\}$
     $\text{ZAHL} = \{(1), (2), (3)\} \equiv \text{NUM} = \{1, 2, 3\}$

$$\text{BUNTEZAHL} = \text{COL} \times \text{NUM} = \left\{ \begin{array}{c} (\text{ROT}, 1), (\text{ROT}, 2), (\text{ROT}, 3), \\ (\text{BLAU}, 1), (\text{BLAU}, 2), (\text{BLAU}, 3), \\ (\text{GRÜN}, 1), (\text{GRÜN}, 2), (\text{GRÜN}, 3) \end{array} \right\}$$

- Demnach ergibt sich als Vereinigungsmenge aller Klassen:

$$\begin{aligned}\mathbb{K} = \; & \{(1), (2), (3),\\ & (\text{ROT}), (\text{BLAU}), (\text{GRÜN}),\\ & (\text{ROT}, 1), (\text{ROT}, 2), (\text{ROT}, 3),\\ & (\text{BLAU}, 1), (\text{BLAU}, 2), (\text{BLAU}, 3),\\ & (\text{GRÜN}, 1), (\text{GRÜN}, 2), (\text{GRÜN}, 3)\}\end{aligned}$$

4. Die Klassifizierungsfunktion der Plätze:

$$k(p_1) = \text{FARBE}$$
$$k(p_2) = \text{ZAHL}$$
$$k(p_3) = \text{BUNTEZAHL}$$
$$k(p_4) = \text{ZAHL}$$
$$k(p_5) = \text{FARBE}$$

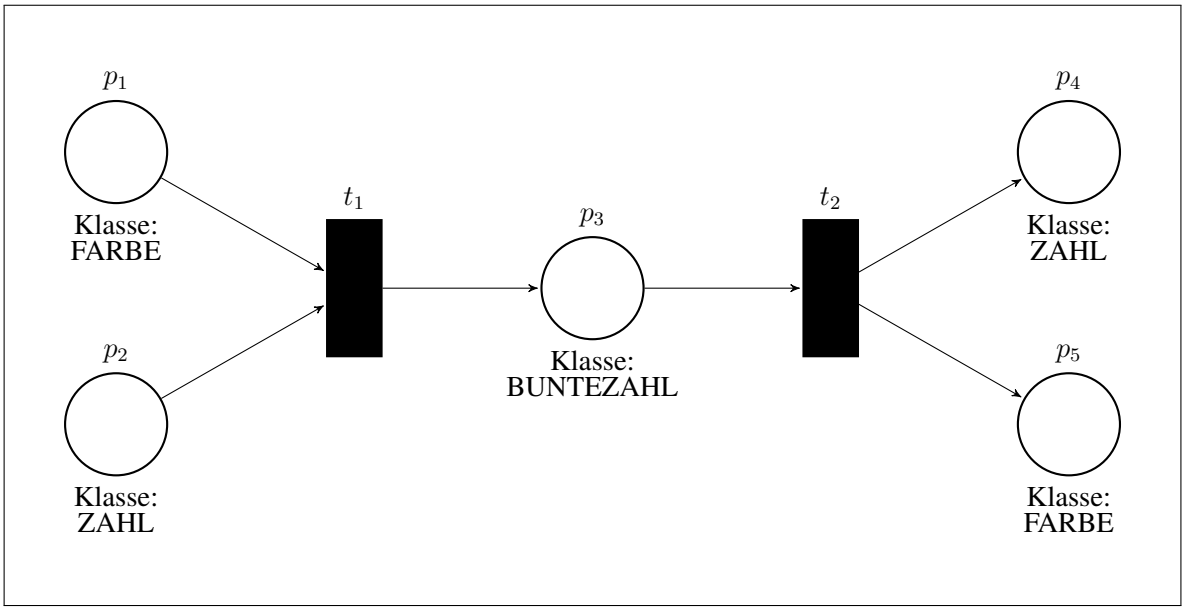

Abbildung 1.4: Beispiel eines Netzes mit Klassen

Um nun ein *mehrfach attribuiertes gefärbtes Petri-Netz* zu erhalten, muss das Netz mit Klassen noch um eine *Gewichtungsfunktion* ergänzt werden.

Definition 1.3:
Ein ***mehrfach attribuiertes gefärbtes Petri-Netz (MCPN[1])*** ist ein Tupel $N = (P, T, F, B, A, K, k, f)$, wobei

- das Tupel $N = (P, T, F, B, A, K, k)$ ein Netz mit Klassen gemäß Def. 1.2 bildet;
- $f : (F \cup B) \to \mathbb{M}(\mathbb{K})$ eine Abbildung ist (***Gewichtungsfunktion***), die jedem Pfeil $(p, t) \in F$ bzw. $(t, p) \in B$ eine der jeweiligen Klasse von p entsprechende Multimenge als ***Gewicht*** zuweist, d.h. $f(p, t)$ bzw. $f(t, p)$ ist eine Multimenge M in der Form $M = \{(a, h(a)) \mid a \in \mathbb{K}, h(a) \in \mathbb{N}_0\}$, wobei für die Stützmenge von M gilt:

$$supp(M) \subseteq k(p) \subseteq \mathbb{K}.^2$$

Bemerkung 1.4:
Aufgrund von großen Redundanzen ist es in einer grafischen Darstellung des MCPN empfehlenswert, wenn die Gewichte der einzelnen Pfeile nur aus der Stützmenge gebildet werden.

Beispiel:
In diesem Beispiel wird ein mehrfach attribuiertes gefärbtes Petri-Netz $N = (P, T, F, B, A, K, k, f)$, basierend auf dem Netz mit Klassen aus Abb. 1.4, konstruiert. Den Pfeilen aus $F \cup B$ werden folgende Multimengen als Gewichtung zugeordnet:

$$\begin{aligned}
f(p_1, t_1) &\equiv \{(\text{ROT}, 1), (\text{GRÜN}, 1)\} \\
f(p_2, t_1) &\equiv \{(1, 2)\} \\
f(t_1, p_3) &\equiv \{((\text{ROT}, 1), 1), ((\text{GRÜN}, 1), 1)\} \\
f(p_3, t_2) &\equiv \{((\text{ROT}, 1), 1), ((\text{BLAU}, 3), 1), ((\text{GRÜN}, 1), 1)\} \\
f(t_2, p_4) &\equiv \{(1, 2), (3, 1)\} \\
f(t_2, p_5) &\equiv \{(\text{ROT}, 1), (\text{BLAU}, 1), (\text{GRÜN}, 1)\}
\end{aligned}$$

Eigentlich hätte in diesem Beispiel jede Gewichtung die Form

$$\begin{aligned}
f(\ldots) = \; & \{(1, h(1)), (2, h(2)), (3, h(3)), \\
& (\text{ROT}, h(\text{ROT})), (\text{BLAU}, h(\text{BLAU})), (\text{GRÜN}, h(\text{GRÜN})), \\
& ((\text{ROT}, 1), h(\text{ROT}, 1)), ((\text{ROT}, 2), h(\text{ROT}, 2)), ((\text{ROT}, 3), h(\text{ROT}, 3)), \\
& ((\text{BLAU}, 1), h(\text{BLAU}, 1)), ((\text{BLAU}, 2), h(\text{BLAU}, 2)), ((\text{BLAU}, 3), h(\text{BLAU}, 3)), \\
& ((\text{GRÜN}, 1), h(\text{GRÜN}, 1)), ((\text{GRÜN}, 2), h(\text{GRÜN}, 2)), ((\text{GRÜN}, 3), h(\text{GRÜN}, 3))\}
\end{aligned}$$

[1] Die Abkürzung MCPN stammt vom engl. Begriff multi-attributed coloured Petri net.
[2] Man beachte, dass $supp(M) \neq \emptyset$ sein muss.

haben müssen, aber wie man sieht, ist diese Form sehr umständlich und unübersichtlich, daher wird hier die reduzierte Entsprechungsform gemäß Bemerkung 1.4 verwendet.

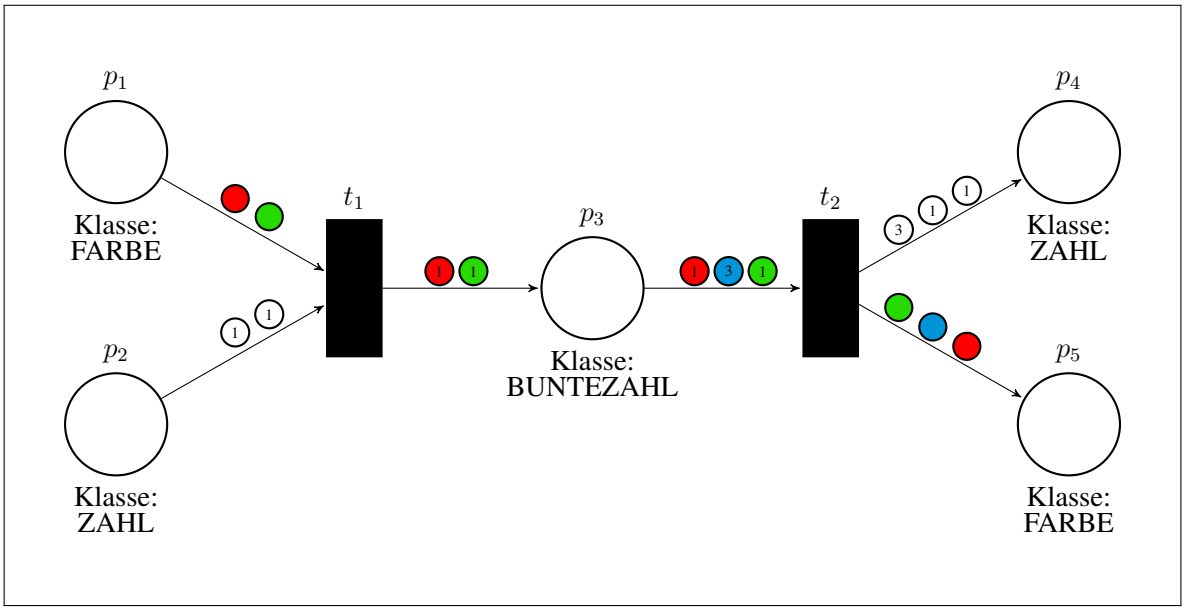

Abbildung 1.5: Beispiel eines gewichteten MCPN

Definition 1.5:
Es sei ein mehrfach attribuiertes gefärbtes Petri-Netz $N = (P, T, F, B, A, K, k, f)$ gegeben.

a) Jedem Platz $p \in P$ wird durch $\bullet p = \{t \in T \mid (t, p) \in B\}$ ein sog. ***Vorbereich*** (*set of inputs*) und durch $p\bullet = \{t \in T \mid (p, t) \in F\}$ ein sog. ***Nachbereich*** (*set of outputs*) zugewiesen.

b) Jeder Transition $t \in T$ wird durch $\bullet t = \{p \in P \mid (p, t) \in F\}$ ein sog. ***Vorbereich*** (*set of inputs*) und durch $t\bullet = \{p \in P \mid (t, p) \in B\}$ ein sog. ***Nachbereich*** (*set of outputs*) zugewiesen.

c) Die Elemente aus einem Vor- bzw. Nachbereich werden auch ***Vor-*** bzw. ***Nachbedingungen*** genannt.

Definition 1.6:
Es sei ein mehrfach attribuiertes gefärbtes Petri-Netz $N = (P, T, F, B, A, K, k, f)$ gegeben. Eine Abbildung $\mathbf{z} : P \to \mathbb{M}(\mathbb{K})$, die jedem Platz $p \in P$ eine der Klasse $k(p)$ entsprechende Multimenge als Gewicht zuweist, heißt ***Zustand*** oder ***Markierung*** von N, wobei $\mathbf{z}$ eindeutig als Vektor $\mathbf{z} = (z_1, \ldots, z_m)$ mit $z_i = \mathbf{z}(p_i)$ darstellbar ist, d.h. $\mathbf{z}(p_i)$ ist eine Multimenge M_i in der Form $M_i = \{(a, h_i(a)) \mid a \in \mathbb{K}, h_i(a) \in \mathbb{N}_0\}$, wobei für die Stützmenge gilt: $supp(M_i) \subseteq k(p_i)$.

Bemerkung 1.7:
Aufgrund von großen Redundanzen ist es in einer grafischen Darstellung des mehrfach attribuierten gefärbten Petri-Netz empfehlenswert, wenn die Markierungen der einzelnen Plätze nur aus der Stützmenge gebildet werden (vgl. Bemerkung 1.4).

Beispiel:
Ein Beispiel einer Markierung des obigen MCPN ist:

$$\begin{aligned}
\mathbf{z}(p_1) &= z_1 \equiv \{(\text{ROT}, 3), (\text{BLAU}, 2), (\text{GRÜN}, 2)\} \\
\mathbf{z}(p_2) &= z_2 \equiv \{(1, 4), (2, 1), (3, 1)\} \\
\mathbf{z}(p_3) &= z_3 \equiv \{((\text{BLAU}, 3), 1)\} \\
\mathbf{z}(p_4) &= z_4 \equiv \{(1, 1), (2, 1)\} \\
\mathbf{z}(p_5) &= z_5 \equiv \{(\text{ROT}, 2), (\text{GRÜN}, 1)\}
\end{aligned}$$

oder als Vektor:

$$\mathbf{z} \equiv \begin{pmatrix} \{(\text{ROT}, 3), (\text{BLAU}, 2), (\text{GRÜN}, 2)\} \\ \{(1, 4), (2, 1), (3, 1)\} \\ \{((\text{BLAU}, 3), 1)\} \\ \{(1, 1), (2, 1)\} \\ \{(\text{ROT}, 2), (\text{GRÜN}, 1)\} \end{pmatrix}$$

Eigentlich hätte in diesem Beispiel jede Markierung in der Form

$$\begin{aligned}
\mathbf{z}(p) = \; & \{(1, h(1)), (2, h(2)), (3, h(3)), \\
& (\text{ROT}, h(\text{ROT})), (\text{BLAU}, h(\text{BLAU})), (\text{GRÜN}, h(\text{GRÜN})), \\
& ((\text{ROT}, 1), h(\text{ROT}, 1)), ((\text{ROT}, 2), h(\text{ROT}, 2)), ((\text{ROT}, 3), h(\text{ROT}, 3)), \\
& ((\text{BLAU}, 1), h(\text{BLAU}, 1)), ((\text{BLAU}, 2), h(\text{BLAU}, 2)), ((\text{BLAU}, 3), h(\text{BLAU}, 3)), \\
& ((\text{GRÜN}, 1), h(\text{GRÜN}, 1)), ((\text{GRÜN}, 2), h(\text{GRÜN}, 2)), ((\text{GRÜN}, 3), h(\text{GRÜN}, 3))\}
\end{aligned}$$

dargestellt werden müssen, aber wie man sieht, ist diese Form sehr umständlich und unübersichtlich, daher wird hier die reduzierte Entsprechungsform gemäß Bemerkung 1.7 verwendet.

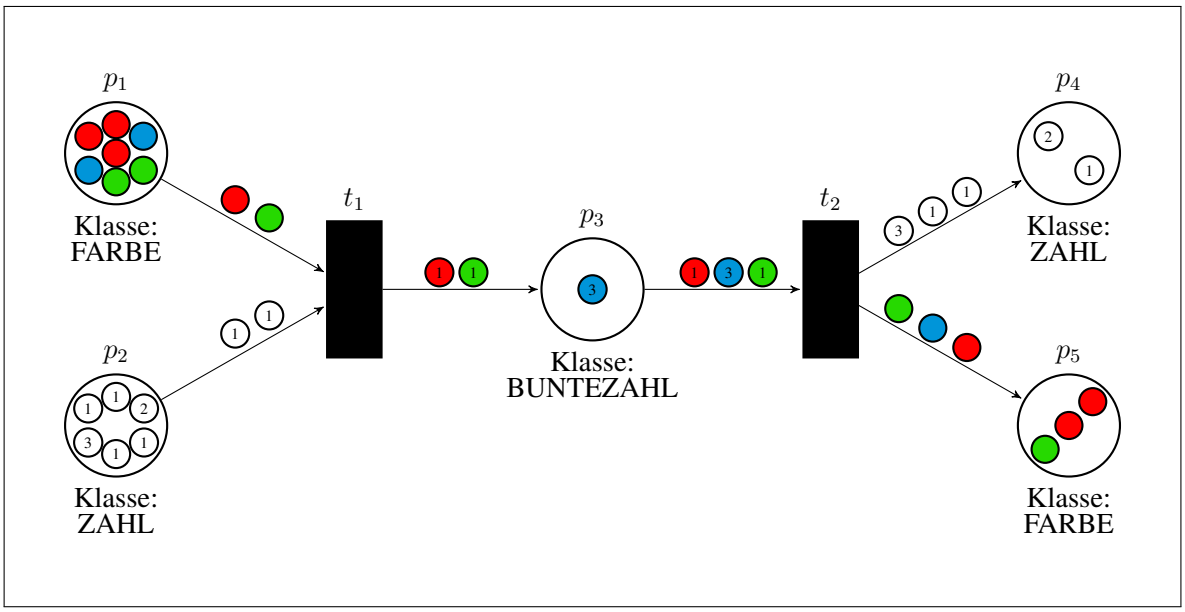

Abbildung 1.6: Beispiel einer Markierung eines MCPN

Im Folgenden sollen die Prozesse der Aktivierung und Feuerung von Transitionen in MCPN vorgestellten werden.

Definition 1.8:
Es sei ein mehrfach attribuiertes gefärbtes Petri-Netz $N = (P, T, F, B, A, K, k, f)$ und eine Markierung **z** von N gegeben.

a) Eine Transition $t \in T$ von N heißt ***aktiviert*** oder ***seriell feuerbar*** im Zustand **z**, wenn für alle $p \in \bullet t$ gilt: $f(p,t) \subseteq \mathbf{z}(p)$.

b) Eine im Zustand **z** aktivierte Transition t wird kurz auch **z-*feuerbar*** genannt.

c) Das ***Feuern*** einer z-feuerbaren Transition $t \in T$ von N ist der Übergang vom Zustand $\mathbf{z} = (z_1, \ldots, z_m)$ in den Zustand $\mathbf{z}' = (z'_1, \ldots, z'_m)$, wobei gilt:

$$z'_i = \left\{ \begin{array}{ll} (z_i \setminus f(p_i,t)) \uplus f(t,p_i) & \text{falls } p_i \in \bullet t \wedge p_i \in t\bullet \\ z_i \setminus f(p_i,t) & \text{falls } p_i \in \bullet t \wedge p_i \notin t\bullet \\ z_i \uplus f(t,p_i) & \text{falls } p_i \notin \bullet t \wedge p_i \in t\bullet \\ z_i & \text{falls } p_i \notin \bullet t \wedge p_i \notin t\bullet \end{array} \right\} \text{ für } i = 1, \ldots, m.$$

Beispiel:

Überprüft man die Transitionen des obigen MCPN-Beispiels, dann ist:

- t_1 aktiviert bzw. seriell feuerbar:

$$f(p_1, t_1) \subseteq \mathbf{z}(p_1)$$
$$f(p_2, t_1) \subseteq \mathbf{z}(p_2)$$

- t_2 nicht aktiviert, weil $f(p_3, t_2)$ keine Teilmenge von $\mathbf{z}(p_3)$ ist.

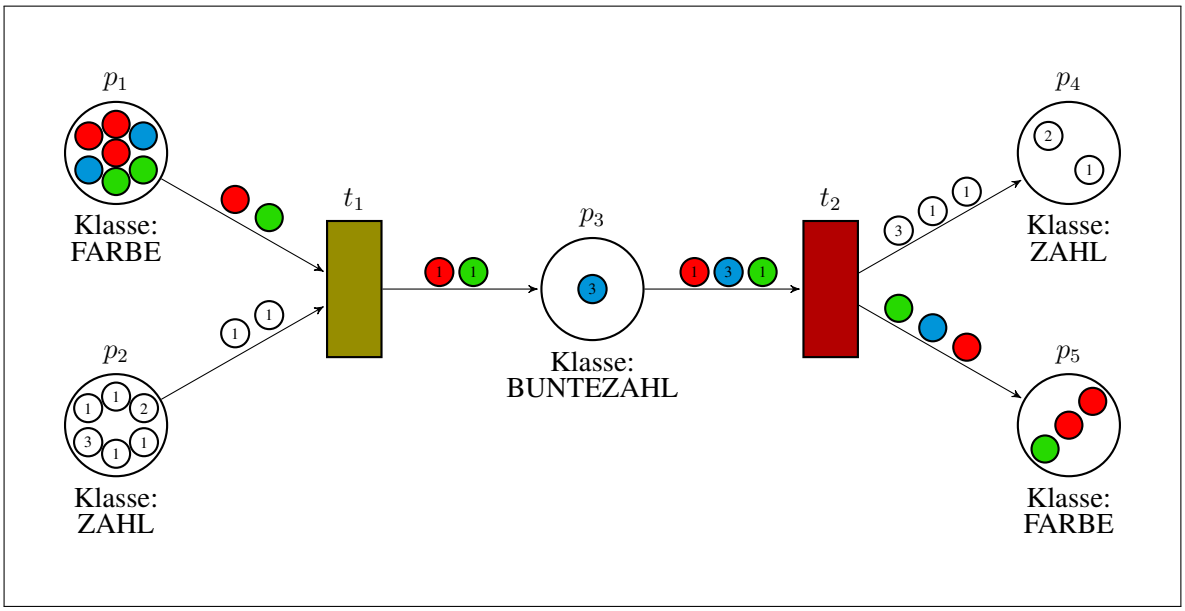

Abbildung 1.7: Feuerbarkeit des MCPN-Beispiels

Wird nun t_1 gefeuert, wird aus Zustand

$$\mathbf{z} \equiv \begin{pmatrix} \{(\text{ROT}, 3), (\text{BLAU}, 2), (\text{GRÜN}, 2)\} \\ \{(1, 4), (2, 1), (3, 1)\} \\ \{((\text{BLAU}, 3), 1)\} \\ \{(1, 1), (2, 1)\} \\ \{(\text{ROT}, 2), (\text{GRÜN}, 1)\} \end{pmatrix}$$

in den Zustand

$$\mathbf{z}' \equiv \begin{pmatrix} \{(\text{ROT},3),(\text{BLAU},2),(\text{GRÜN},2)\} \setminus \{(\text{ROT},1),(\text{GRÜN},1)\} \\ \{(1,4),(2,1),(3,1)\} \setminus \{(1,2)\} \\ \{((\text{BLAU},3),1)\} \uplus \{((\text{ROT},1),1),((\text{GRÜN},1),1)\} \\ \{(1,1),(2,1)\} \\ \{(\text{ROT},2),(\text{GRÜN},1)\} \end{pmatrix}$$

$$= \begin{pmatrix} \{(\text{ROT},2),(\text{BLAU},2),(\text{GRÜN},1)\} \\ \{(1,2),(2,1),(3,1)\} \\ \{((\text{ROT},1),1),((\text{BLAU},3),1),((\text{GRÜN},1),1)\} \\ \{(1,1),(2,1)\} \\ \{(\text{ROT},2),(\text{GRÜN},1)\} \end{pmatrix}$$

übergegangen.

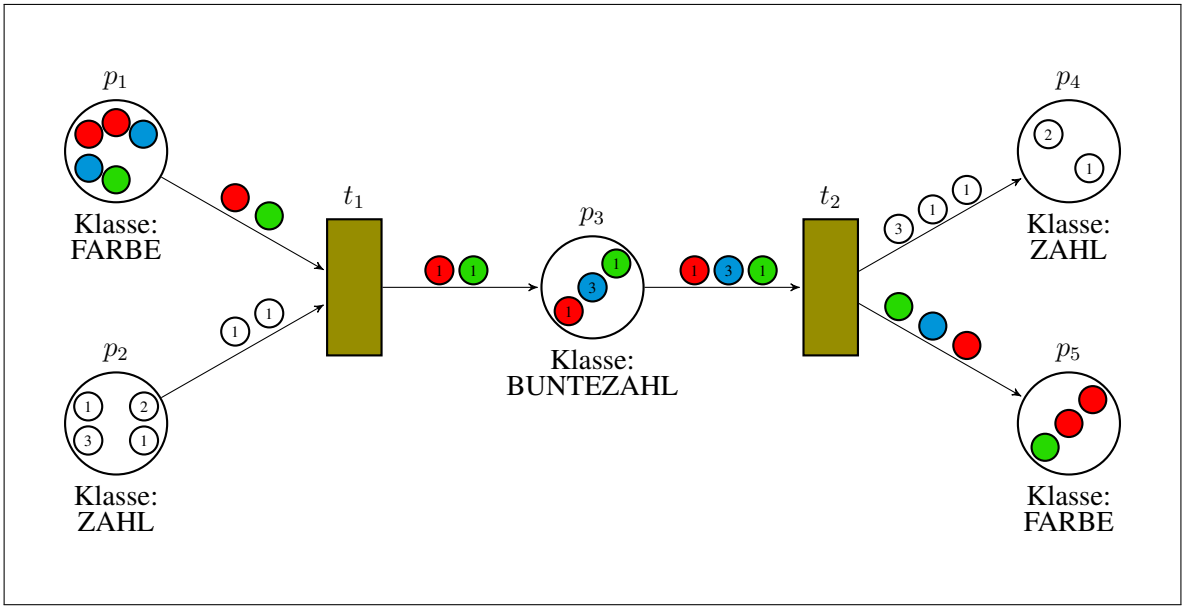

Abbildung 1.8: Zustandsübergang am MCPN-Beispiel

In diesem Zustand wären sowohl t_1 als auch t_2 aktiviert.

In diesem Abschnitt wurden bis jetzt nur CPN mit einer *konstanten* Pfeilgewichtung behandelt, was vor allem aus didaktischen Gründen geschieht. Die allgemeinere und in der Praxis häufiger benutzte Form von CPN mit *variabler* Pfeilgewichtung wird im folgenden Abschnitt beschrieben.

1.2 Gefärbte Petri-Netze mit Variablen

In der Praxis werden zumeist CPN mit variablen Pfeilgewichtungen verwendet (vgl. [JK09], [Ger04]). Die dazu eingeführten Variablen in den CPN lassen sich auf ähnliche Weise durch die Faltung herleiten wie die Farben der Tokens. Dies soll am folgenden Beispiel demonstriert werden.

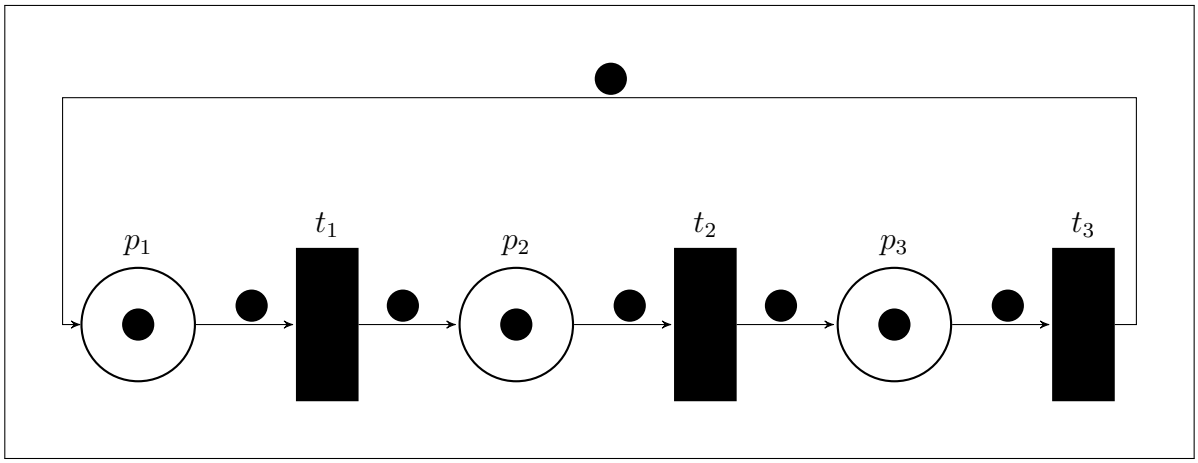

Abbildung 1.9: Nicht gefaltetes Petri-Netz

Das Petri-Netz in Abb. 1.9 stellt in seiner Graphenstruktur offensichtlich einen Zyklus dar. Durch „Konzentration“ der Plätze und der Transitionen entsteht eine vereinfachte Petri-Schlinge, indem die Plätze p_1, p_2 und p_3 zu einem „konzentrierten“ Platz p_c (vgl. Abb. 1.10) und die Transitionen t_1, t_2 und t_3 zu einer „konzentrierten“ Transition t_c „zusammengefaltet“ werden (siehe Abb. 1.11). Um bei diesem Faltvorgang die wesentlichen Informationen über die Struktur des ursprünglichen (nicht gefalteten) Petri-Netzes zu erhalten, werden die Token aus den Plätzen p_1, p_2 und p_3 und an den zugehörigen Pfeilen von p_i nach t_i $(i = 1, 2, 3)$ nach dem Schema

$$\begin{aligned} p_1 &\rightarrow \text{ROT} \\ p_2 &\rightarrow \text{BLAU} \\ p_3 &\rightarrow \text{GRÜN} \end{aligned}$$

gefärbt; man erhält das MCPN aus Abb. 1.10.

Um dabei die Reihenfolge im Zyklus des ursprünglichen Petri-Netzes erkenntlich zu erhalten, werden die Token an den Pfeilen (t, p) mit der jeweiligen Farbe des Platzes aus dem Nachbereich $t\bullet$ gefärbt. Damit lässt sich serielles Feuern im MCPN (Abb. 1.10) auch umkehrbar als serielles Feuern im Petri-Netz (Abb. 1.9) nachvollziehen. Beispielsweise wird beim Feuern von t_2 im MCPN dem Platz p_c ein blaues Token entzogen, dafür ein grünes Token hinzugefügt. Die neue Markierung des gefärbten Platzes p_c (1 rotes und 2 grüne Token) entspricht im nicht gefärbten

Petri-Netz dem analogen Zustand, dass p_1 ein Token, p_2 kein Token und p_3 zwei Token hat (vgl. Abb. 1.10).

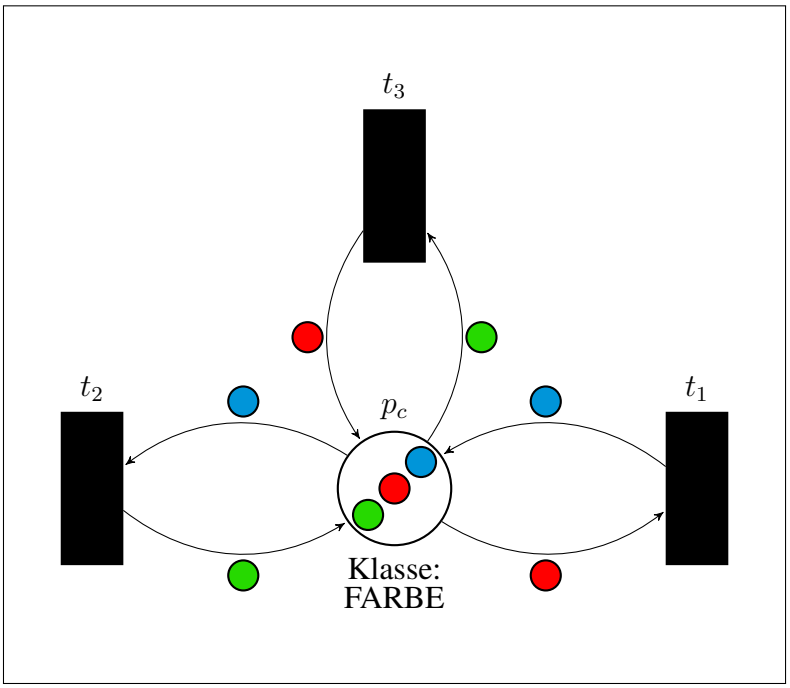

Abbildung 1.10: Zu einem MCPN gefaltetes PN

Im nächsten Faltungsschritt werden nun auch die Transitionen zusammengefasst. Es entsteht das gefaltete CPN mit Variablen in Abb. 1.11. Dabei ist zu beachten, dass die Pfeilgewichtungen der ehemals einzelnen Transitionen verschieden voneinander sind, was dazu führt, dass bei der Faltung der Transitionen zwangsläufig Variablen als Pfeilgewichtungen zu verwenden sind.

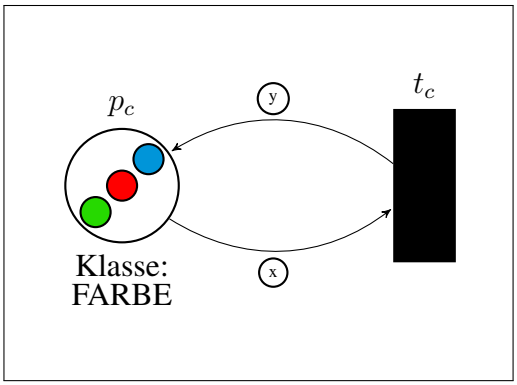

Abbildung 1.11: Gefaltetes CPN mit Variablen

Alle Variablen auf den mit einer Transition positiv oder negativ inzidenten Pfeilen bilden die *Variablenmenge* $Var(t)$ *der Transition* t. Im obigen Beispiel ist dies

$$Var(t_c) = \{x, y\}.$$

Nun werden diesen Variablen in passender Weise Ausprägungen aus der Farbenmenge zugeordnet. Eine solche Zuordnung nennt man *Modus* (vgl. [Rei10]). Sie ordnet jeder Variablen auf den Pfeilen einer Transition eine Ausprägung zu. Wie man aber erkennt, dürften im obigen Beispiel nur bestimmte Modi zugelassen werden, um die Äquivalenz zum ursprünglichen Petri-Netz zu erhalten. Diese grundsätzlich möglichen Modi einer Transition werden in der *Modus-Menge* $m(t)$ zusammengefasst. Die Modus-Menge für das obige Beispiel ist:

$$m(t_c) = \{\beta_1, \beta_2, \beta_3\} \text{ mit } \beta_i : \{x, y\} \to \{\text{ROT}, \text{BLAU}, \text{GRÜN}\} \text{ für } i = 1, 2, 3,$$

wobei konkret gilt:

$$\begin{aligned} \beta_1(x) &= \text{ROT}, & \beta_1(y) &= \text{BLAU}, \\ \beta_2(x) &= \text{BLAU}, & \beta_2(y) &= \text{GRÜN}, \\ \beta_3(x) &= \text{GRÜN}, & \beta_3(y) &= \text{ROT}. \end{aligned}$$

Dabei erweist sich die folgende Schreibweise in Anlehnung an [Rei10, S. 28] für die einzelnen Modi als praktikabel:

$$\begin{aligned} \beta_1 &: \{x = \text{ROT}, y = \text{BLAU}\}, \\ \beta_2 &: \{x = \text{BLAU}, y = \text{GRÜN}\}, \\ \beta_3 &: \{x = \text{GRÜN}, y = \text{ROT}\}. \end{aligned}$$

Soll nun die Transition t_c gefeuert werden, müsste zuvor ein Modus $\beta_i \in m(t_c)$ ausgewählt werden. Als Beispiel soll nun die Transition t_c im Modus β_1 gefeuert werden.

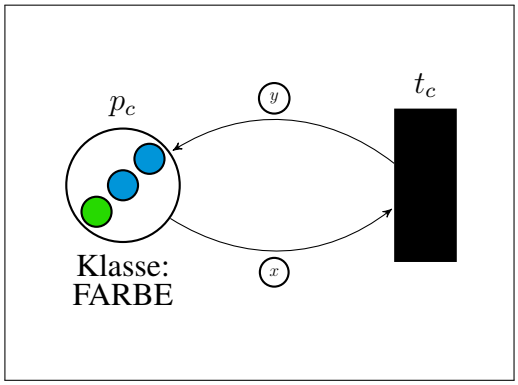

Abbildung 1.12: Zustand des CPN nach dem Feuern von Transition t_c im Modus β_1

Es ist einfach nachzuvollziehen, dass diese konkrete Moduswahl dem Feuern der Transition t_1 im ursprünglichen Petri-Netz entspricht (vgl. Abb. 1.9).

An dieser Stelle entsteht nun die Problematik, ob eine Transition überhaupt in einem bestimmten Modus feuerbar ist oder ob überhaupt ein feuerbarer Modus[1] existiert. Wie man im oberen Beispiel erkennt, ist es nicht möglich, in diesem Zustand die Transition t_c ein weiteres Mal im Modus β_1 zu feuern.

Nun sollen die Variablen für ein gefärbtes Petri-Netz definiert werden. Diese Variablen sind in einem gefärbten Petri-Netz von zentraler Bedeutung, da mit ihnen variable Kantengewichtungen möglich sind; zudem erleichtern sie die Verständlichkeit des Modells bei der Verwendung einer Formalsprache.

Definition 1.9:
Es seien ein Netz mit Klassen $N = (P, T, F, B, A, K, k)$ und eine endliche nichtleere Menge $\mathbb{V} = \{v_1, \ldots, v_\nu\}$ von sog. ***Variablen*** v_i gegeben.

a) Eine Abbildung $Attr : \mathbb{V} \to A$, die jeder Variablen $v_i \in \mathbb{V}$ ein Attribut aus der Attributenmenge A zuweist, bezeichnet man als ***Attributierungsfunktion***.

b) Mittels einer Abbildung $Var : T \to \wp(\mathbb{V})$, die jeder Transition $t \in T$ eine Menge von Variablen $Var(t) \subseteq \mathbb{V}$ zuordnet, bezeichnet man $Var(t)$ als ***Variablenmenge*** von t.[2]

Beispiel:
Gegeben sei das Netz mit Klassen aus Abb. 1.13 mit der Attributenmenge $A = \{\text{COL,NUM}\}$ und die Menge von Variablen $\mathbb{V} = \{x, y, n\}$.

- Den Variablen werden durch die Attributierungsfunktion folgende Attribute zugewiesen:

  $Attr(x) = \text{COL}$
  $Attr(y) = \text{COL}$
  $Attr(n) = \text{NUM}.$

- Den Transitionen t_1 und t_2 werden die folgenden Variablen zugewiesen:

  $Var(t_1) = \{x, n\}$
  $Var(t_2) = \{x, y, n\}.$

[1] Die Ausdrucksweise, dass ein Modus feuerbar ist, mag zunächst ungewöhnlich erscheinen, da ja eigentlich eine Transition in einem Modus feuert. Wenn man aber bedenkt, dass ein Modus einer Transition im ursprünglichen PN einer bestimmten Transition entspricht, ist diese Ausdrucksweise durchaus eine angebrachte Sprechweise.
[2] Der Ausdruck $\wp$ steht dabei für Potenzmenge.

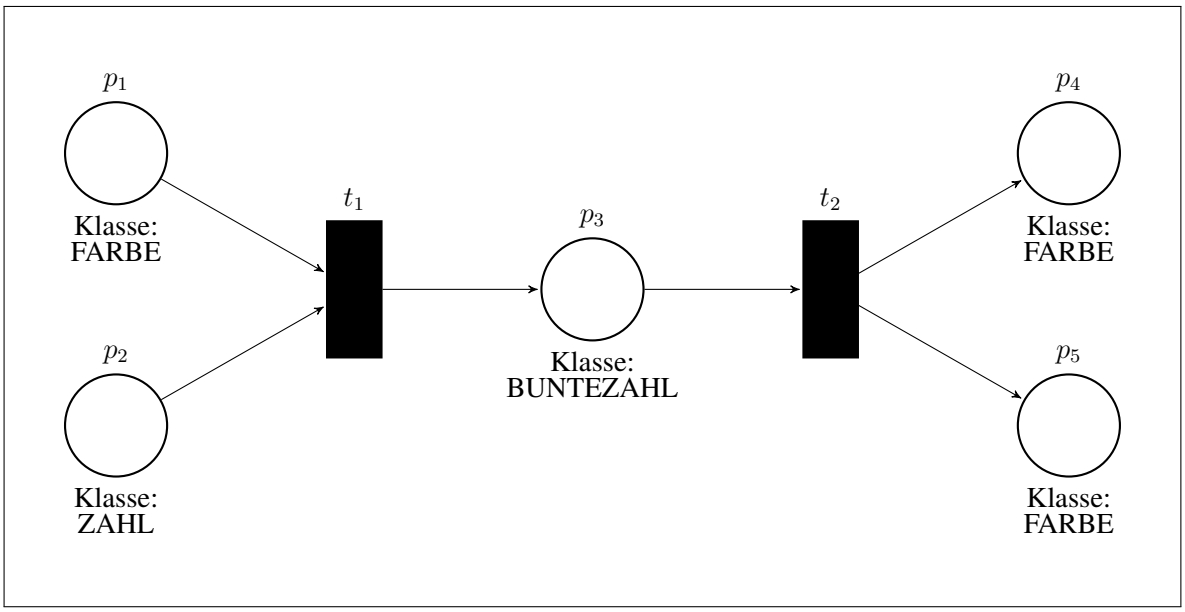

Abbildung 1.13: Netz mit Klassen

Nun soll den Transitionen eines Netzes mit Klassen und Variablen eine Menge von Modi mitgegeben werden, um nur bestimmte Zuordnungen zu erlauben, ähnlich wie in dem einleitenden Beispiel dieses Kapitels.

Definition 1.10:
Es sei ein Netz mit Klassen $N = (P, T, F, B, A, K, k)$ und eine Menge von Variablen $\mathbb{V} = \{v_1, \ldots, v_r\}$ gegeben.

a) Eine Abbildung $\beta : \mathbb{V} \to \mathbb{A}$, die jeder Variablen $v \in \mathbb{V}$ eine Ausprägung $a \in \mathbb{A}$ zuordnet, so dass $a \in Attr(v)$ gilt, bezeichnet man als ***Modus***.

b) Eine Menge $\mathbb{B} = \{\beta_1, \ldots, \beta_b\}$, welche eine endliche Menge von Modi ist, bezeichnet man als ***Modus-Menge von N***; dabei wird die ausgezeichnete Modus-Menge $\mathbb{B}_{max}$, für die gilt:

$$|\mathbb{B}_{max}| = \prod_{v \in \mathbb{V}} |Attr(v)|,$$

als ***Maximal-Modus-Menge*** bezeichnet.[1]

c) Mittels einer Abbildung $m : T \to \wp(\mathbb{B})$, die jeder Transition $t \in T$ eine Teilmenge der Modus-Menge $\mathbb{B}$ zuweist, bezeichnet man $m(t)$ als ***Modus-Menge einer Transition*** t.

[1] Mit $|Attr(v)|$ sei die Mächtigkeit der Menge $Attr(v)$ bezeichnet.

Beispiel:
Ein möglicher Modus für das obere Beispiel wäre β_1 mit:

$$\beta_1 : \{x = \text{ROT}, y = \text{ROT}, n = 1\}.$$

Für das obere Beispiel könnte die Gesamtheit der folgenden 27 Modi als Modus-Menge gewählt werden:

Tabelle 1.1: Modus-Menge mit Mächtigkeit 27

β	x	y	n	β	x	y	n	β	x	y	n
1	ROT	ROT	1	10	BLAU	ROT	1	19	GRÜN	ROT	1
2	ROT	ROT	2	11	BLAU	ROT	2	20	GRÜN	ROT	2
3	ROT	ROT	3	12	BLAU	ROT	3	21	GRÜN	ROT	3
4	ROT	BLAU	1	13	BLAU	BLAU	1	22	GRÜN	BLAU	1
5	ROT	BLAU	2	14	BLAU	BLAU	2	23	GRÜN	BLAU	2
6	ROT	BLAU	3	15	BLAU	BLAU	3	24	GRÜN	BLAU	3
7	ROT	GRÜN	1	16	BLAU	GRÜN	1	25	GRÜN	GRÜN	1
8	ROT	GRÜN	2	17	BLAU	GRÜN	2	26	GRÜN	GRÜN	2
9	ROT	GRÜN	3	18	BLAU	GRÜN	3	27	GRÜN	GRÜN	3

Diese Modus-Menge bildet zugleich auch die Maximal-Modus-Menge $\mathbb{B}_{max}$, da es nicht möglich ist, eine größere Modus-Menge für das Beispiel zu bilden.

Aus dieser Modus-Menge ist es beispielsweise möglich, folgende Modus-Mengen für die Transitionen t_1 und t_2 zu bilden:

$$m(t_1) = \{\beta_1, \beta_2, \beta_3, \beta_{13}, \beta_{14}, \beta_{15}, \beta_{25}, \beta_{26}, \beta_{27}\}$$

bzw.

$$m(t_2) = \{\beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_{12}, \beta_{16}, \beta_{17}, \beta_{18}, \beta_{19}, \beta_{20}, \beta_{21}, \beta_{22}, \beta_{23}, \beta_{24}\}.$$

Bemerkung 1.11:
Es sei an dieser Stelle noch angemerkt, dass man mit der Modus-Menge $m(t)$ einer Transition t bestimmte Bedingungen oder Abhängigkeiten modellieren kann. Zum Beispiel ist mit der Modus-Menge $m(t_2)$ die Bedingung erzeugt worden, dass die Variablen x und y unterschiedliche Werte annehmen müssen, demzufolge gilt:

$$x \neq y.$$

Solche Bedingungen werden unter anderem auch als *Wächter* (vgl. [JK09, S.85]) bezeichnet. Wächter sind eine praktikable Darstellungsform, wenn „viele“ Modi einer Transition zugewiesen werden, da so auch die Information über die Modi mit in die „grafische Darstellungsform“ übernommen wird, welche unter der jeweiligen Transition steht. Wird einer Transition die Maximal-Modus-Menge $\mathbb{B}_{max}$ bzw. eine Modus-Menge, wo alle Kombinationsmöglichkeiten für die Variablen der Transition $Var(t)$ enthalten sind, zugewiesen, wird der Wächter weggelassen. Die Darstellungsform mit Wächter sei in Abb. 1.14 exemplarisch gezeigt:

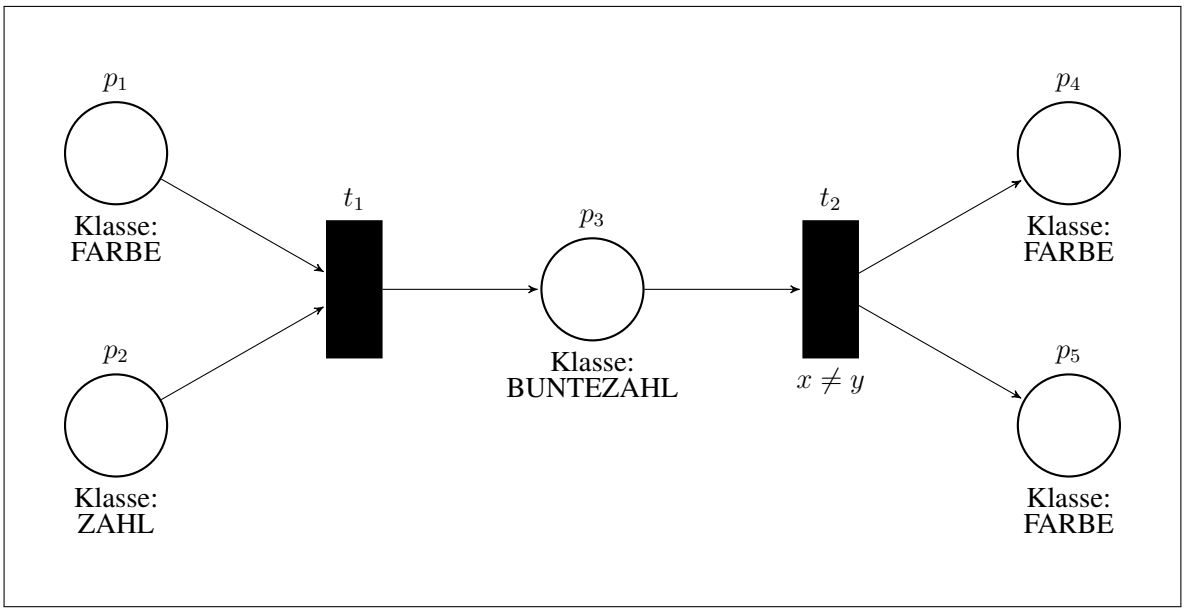

Abbildung 1.14: Darstellung von Wächter

Um übersichtliche und nachvollziehbare Modelle trotz komplexer Problemstellung zu erhalten, werden an den Pfeilen in einem CPN nicht nur Multimengen aus Konstanten und Variablen verwendet, sondern ein jeweilig passender „Ausdruck“ (vgl. expressions in [JK09, S.56f. und S.84.]), welcher eine *Multimengen-Erzeugungsfunktion* (MME-Funktion) darstellt. Dafür wird eine *Menge von MME-Funktionen* definiert, welche alle MME-Funktionen eines CPN enthält. Diese MME-Funktionen werden in der einschlägigen Literatur (vgl. [Ger04], [JK09], [Rei10]) recht unmathematisch in einer gewissermaßen sprachlichen Algebra (vgl. [Ger04]) oder in einer Pseudoprogrammiersprache (vgl. [JK09]) definiert, wie das folgende Beispiel in Abb. 1.15 zeigt.

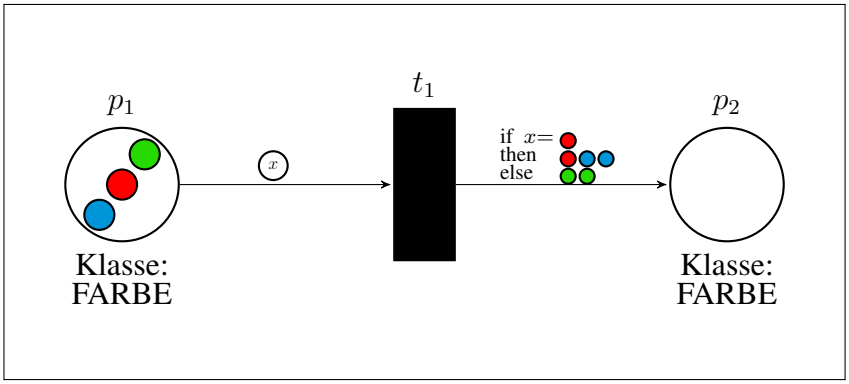

Abbildung 1.15: CPN mit Variablen (Pseudoprogrammiersprache)

Diese MME-Funktionen stellen im mathematischen Sinne Abbildungen dar. Anders als zu vermuten sind diese Abbildungen nicht von den Variablen im Petri-Netz abhängig (vgl. Abb. 1.16),

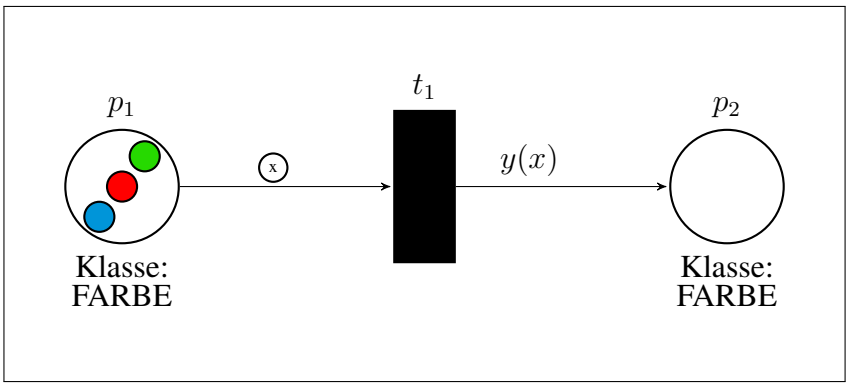

Abbildung 1.16: CPN mit Variablen (Funktion von x)

sondern von dem Modus β (vgl. Abb. 1.17), in welchem die jeweilige Transition gefeuert wird.

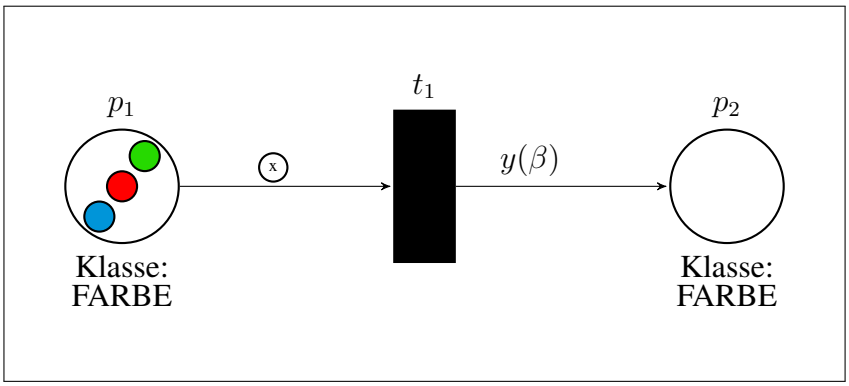

Abbildung 1.17: CPN mit Variablen (Funktion von β)

Bei dem obigen Beispiel würde für t_1 mit der Modus-Menge

$$m(t_1) = \{\beta_1, \beta_2, \beta_3\}$$

und den Modi

$$\begin{aligned}\beta_1 &: \{x = \text{ROT}\},\\ \beta_2 &: \{x = \text{BLAU}\},\\ \beta_3 &: \{x = \text{GRÜN}\}.\end{aligned}$$

der Ausdruck

$$\begin{aligned}&\text{if } x = \{(\text{ROT}, 1)\}\\ &\text{then } \{(\text{ROT}, 1), (\text{BLAU}, 2)\}\\ &\text{else } \{(\text{GRÜN}, 2)\}\end{aligned}$$

der folgenden Abbildung

$$y(\beta) = \begin{cases}\{(\text{ROT}, 1), (\text{BLAU}, 2)\} & \text{falls } \beta = \beta_1\\ \{(\text{GRÜN}, 2)\} & \text{falls } \beta \in \{\beta_2, \beta_3\}\end{cases}$$

entsprechen.

Definition 1.12:
Es sei ein Netz mit Klassen $N = (P, T, F, B, A, K, k)$, eine Variablen-Menge $\mathbb{V} = \{v_1, \ldots, v_r\}$ und eine Modus-Menge $\mathbb{B} = \{\beta_1, \ldots, \beta_b\}$ von N gegeben.

a) Eine Abbildung $Expr : \mathbb{B} \rightarrow \mathbb{M}(\mathbb{K})$, die jedem Modus β von $\mathbb{B}$ eine Multimenge $M \in \mathbb{M}(\mathbb{K})$ zuordnet, bezeichnet man als ***Multimengen-Erzeugungsfunktion*** (MME-Funktion).

b) Eine Menge $\mathbb{E} = \{Expr_1, \ldots, Expr_e\}$, welche eine endliche Menge von MME-Funktionen ist, bezeichnet man als ***Menge von Multimengen-Erzeugungsfunktionen von N***.

MME-Funktionen sind Abbildungen, die in Abhängigkeit von einem Modus eine Multimenge wiedergeben. Wie am Anfang dieses Abschnittes schon beschrieben wurde, können solche MME-Funktionen die unterschiedlichsten Darstellungsformen haben. Im folgenden Beispiel werden einige Beispiele von MME-Funktionen gezeigt und ihre Interpretation erläutert.

Beispiel:
In den folgenden MME-Funktionen wird die Modus-Menge aus Tabelle 1.1 zu Grunde gelegt.

Wie in der Einleitung gezeigt, werden oft auch MME-Funktionen verwendet, welche „if-Abfragen“ darstellen. Ein weiteres Beispiel hierzu sei eine MME-Funktion, die, wenn $n = 1$ ist, den Wert der Variablen x und sonst den Wert von y weitergibt (vgl. Abb. 1.18):

$$Expr_1(\beta_i) = \begin{cases} \{(\text{ROT}, 1)\} & \text{für } \beta_i \text{ mit } i = 1, 4, 7 \\ \{(\text{BLAU}, 1)\} & \text{für } \beta_i \text{ mit } i = 10, 13, 16 \\ \{(\text{GRÜN}, 1)\} & \text{für } \beta_i \text{ mit } i = 19, 22, 25 \\ \{(\text{ROT}, 1)\} & \text{für } \beta_i \text{ mit } i = 2, 3, 11, 12, 20, 21 \\ \{(\text{BLAU}, 1)\} & \text{für } \beta_i \text{ mit } i = 5, 6, 14, 15, 23, 24 \\ \{(\text{GRÜN}, 1)\} & \text{für } \beta_i \text{ mit } i = 8, 9, 17, 18, 26, 27 \end{cases}$$

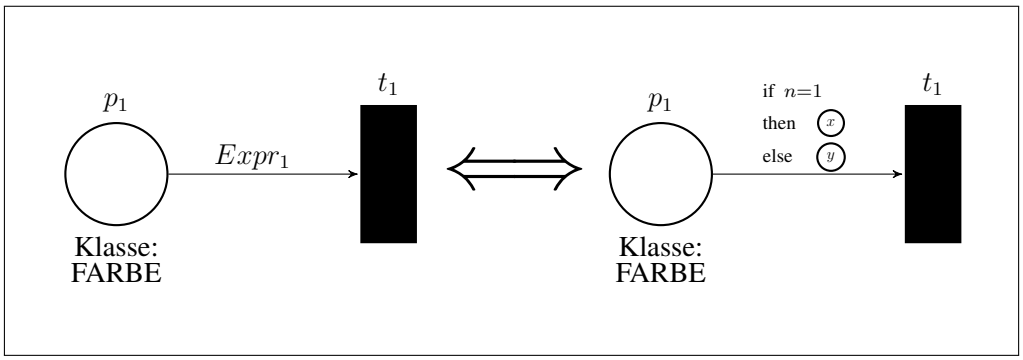

Abbildung 1.18: Darstellung von Exp_1 in einem Petri-Netz

Zwei weitere mögliche MME-Funktionen wären:

$$Expr_2(\beta_i) = \begin{cases} \{(\text{ROT}, 1)\} & \text{für } \beta_i \text{ mit } i = 1, \ldots, 9 \\ \{(\text{BLAU}, 1)\} & \text{für } \beta_i \text{ mit } i = 10, \ldots, 18 \\ \{(\text{GRÜN}, 1)\} & \text{für } \beta_i \text{ mit } i = 19, \ldots, 27 \end{cases}$$

und

$$Expr_3(\beta_i) = \begin{cases} \{(1, 1)\} & \text{für } \beta_i \text{ mit } i = 1, 4, 7, 10, 13, 16, 19, 22, 25 \\ \{(2, 1)\} & \text{für } \beta_i \text{ mit } i = 2, 5, 8, 11, 14, 17, 20, 23, 26 \\ \{(3, 1)\} & \text{für } \beta_i \text{ mit } i = 3, 6, 9, 12, 15, 18, 21, 24, 27 \end{cases}$$

Die MME-Funktionen $Expr_2$ und $Expr_3$ geben jeweils den durch den Modus zugeordneten Wert der Variable x bzw. n als Multimenge mit der Häufigkeit 1 wieder. Daher geben diese beiden MME-Funktionen die Variablen x bzw. n wieder (vgl. Abb. 1.19 und Abb. 1.20).

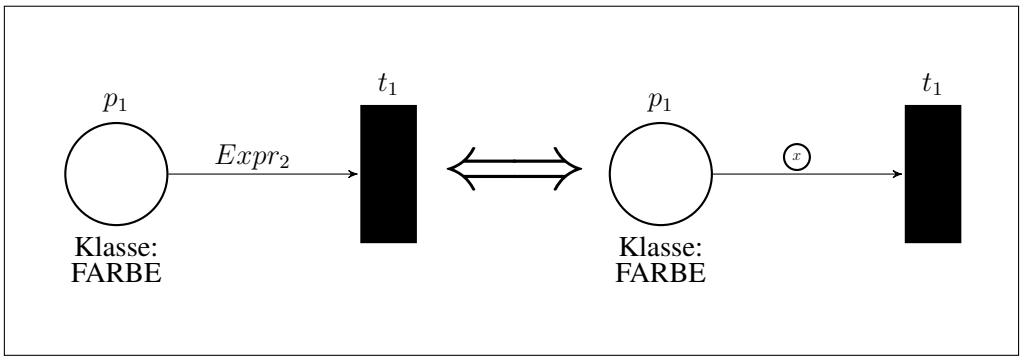

Abbildung 1.19: Darstellung von Exp_2 in einem Petri-Netz

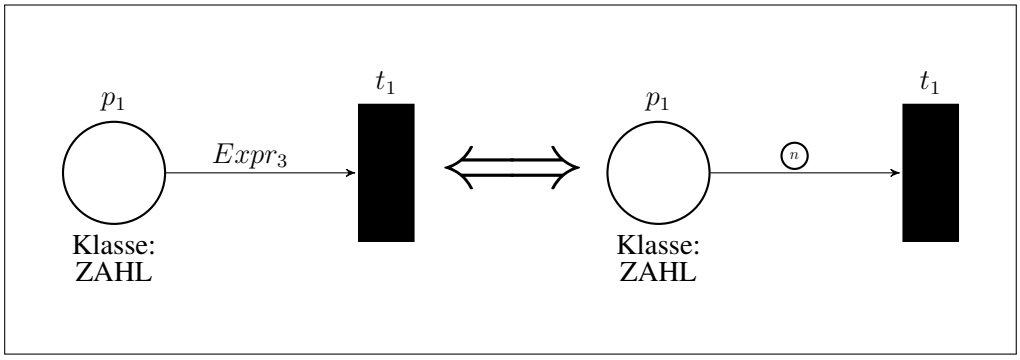

Abbildung 1.20: Darstellung von Exp_3 in einem Petri-Netz

Eine weitere Möglichkeit, die mit MME-Funktionen realisiert werden kann, ist, dass einzelne Variablen miteinander kombiniert werden können, um so z.B. zusammengesetzte Klassen wie BUNTEZAHL zu erhalten. Dies soll an der MME-Funktion $Expr_4$ exemplarisch gezeigt werden:

$$Expr_4(\beta_i) = \begin{cases} \{((\text{ROT}, 1), 1)\} & \text{für } \beta_i \text{ mit } i = 1, 4, 7 \\ \{((\text{ROT}, 2), 1)\} & \text{für } \beta_i \text{ mit } i = 2, 5, 8 \\ \{((\text{ROT}, 3), 1)\} & \text{für } \beta_i \text{ mit } i = 3, 6, 9 \\ \{((\text{BLAU}, 1), 1)\} & \text{für } \beta_i \text{ mit } i = 10, 13, 16 \\ \{((\text{BLAU}, 2), 1)\} & \text{für } \beta_i \text{ mit } i = 11, 14, 17 \\ \{((\text{BLAU}, 3), 1)\} & \text{für } \beta_i \text{ mit } i = 12, 15, 18 \\ \{((\text{GRÜN}, 1), 1)\} & \text{für } \beta_i \text{ mit } i = 19, 22, 25 \\ \{((\text{GRÜN}, 2), 1)\} & \text{für } \beta_i \text{ mit } i = 20, 23, 26 \\ \{((\text{GRÜN}, 3), 1)\} & \text{für } \beta_i \text{ mit } i = 21, 24, 27 \end{cases}$$

Diese MME-Funktion stellt ein Token mit Klasse BUNTEZAHL dar, welches die „Farbe“ von der Variablen x und die „Zahl“ von der Variablen n enthält (vgl. Abb. 1.21).

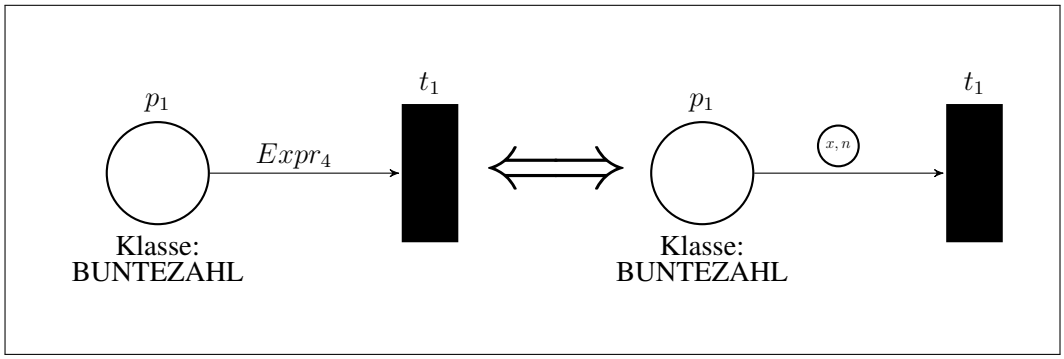

Abbildung 1.21: Darstellung von Exp_4 in einem Petri-Netz

Natürlich können mit MME-Funktionen auch Konstanten (vgl. Abb. 1.22) dargestellt werden, wie bei MME-Funktion $Expr_5$ gezeigt:

$$Expr_5(\beta_i) = \{(\text{ROT}, 1)\} \text{ für } \beta_i \text{ mit } i = 1, \ldots, 27.$$

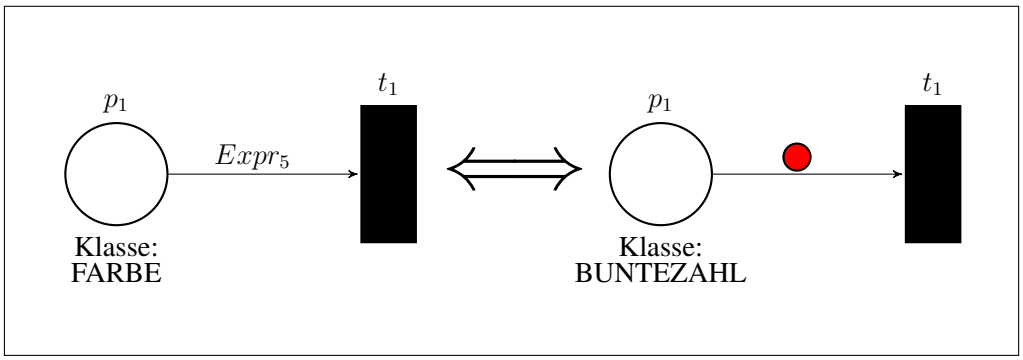

Abbildung 1.22: Darstellung von Exp_5 in einem Petri-Netz

Nun soll ein gefärbtes Petri-Netz definiert werden, welches MME-Funktionen als Pfeilgewichtungen verwendet.

Definition 1.13:
Ein ***gefärbtes Petri-Netz mit Variablen*** ist ein Tupel $N = (P, T, F, B, A, K, k, \mathbb{V}, Attr, Var, \mathbb{B}, m, \mathbb{E}, f)$, wobei

- das Tupel (P, T, F, B, A, K, k) ein Netz mit Klassen gemäß Def. 1.2 bildet;
- $\mathbb{V} = \{v_1, \ldots, v_r\}$ eine Menge von Variablen gemäß Def. 1.9 ist;
- $Attr : \mathbb{V} \to A$ eine Attributierungsfunktion der Variablen gemäß Def. 1.9 ist;
- $Var : T \to \wp(\mathbb{V})$ die Variablenmenge zu den Transitionen gemäß Def. 1.9 bildet;
- $\mathbb{B} = \{\beta_1, \ldots, \beta_b\}$ eine Modus-Menge von N gemäß Def. 1.10 ist;
- $m : T \to \wp(\mathbb{B})$ die Modus-Mengen zu den Transitionen gemäß Def. 1.10 bildet;
- $\mathbb{E} = \{Expr_1, \ldots, Expr_e\}$ eine MME-Funktionsmenge von N gemäß Def. 1.12 ist;
- $f : (F \cup B) \to \mathbb{E}$ eine Abbildung ist (***Gewichtungsfunktion***), die jedem Pfeil $(p, t) \in F$ bzw. $(t, p) \in B$ eine MME-Funktion $Expr \in \mathbb{E}$ zuweist, wobei jede Stützmenge $supp(M)$ der Multimenge $M \in$ Bild$(Expr)$ eine Teilmenge der Klasse des inzidenten Platzes bildet: $supp(M) \subseteq k(p)$.

Beispiel:
Im folgenden Beispiel sollen die Komponenten aus den vorherigen Beispielen (Netz mit Klasse, Variablen, Modi, MME-Funktionen) in ein gefärbtes Petri-Netz mit Variablen zusammengeführt werden. Dazu werden den Pfeilen die folgenden MME-Funktionen zugeordnet:

$$
\begin{aligned}
f(p_1, t_1) &= Expr_2 \\
f(p_2, t_1) &= Expr_3 \\
f(t_1, p_3) &= Expr_4 \\
f(p_3, t_2) &= Expr_4 \\
f(t_2, p_4) &= Expr_5 \\
f(t_2, p_5) &= Expr_1
\end{aligned}
$$

Man erhält das CPN mit Variablen in Abb. 1.23.

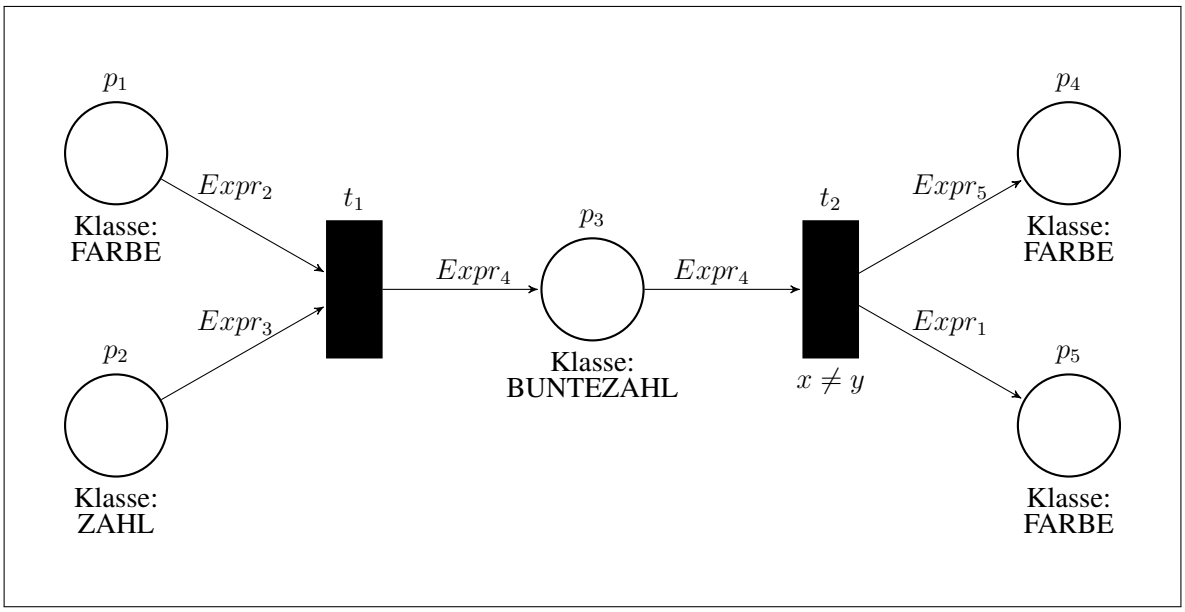

Abbildung 1.23: CPN mit Variablen

In einer „Formalsprache“ könnte das CPN die folgende Darstellungsform in Abb. 1.24 haben.

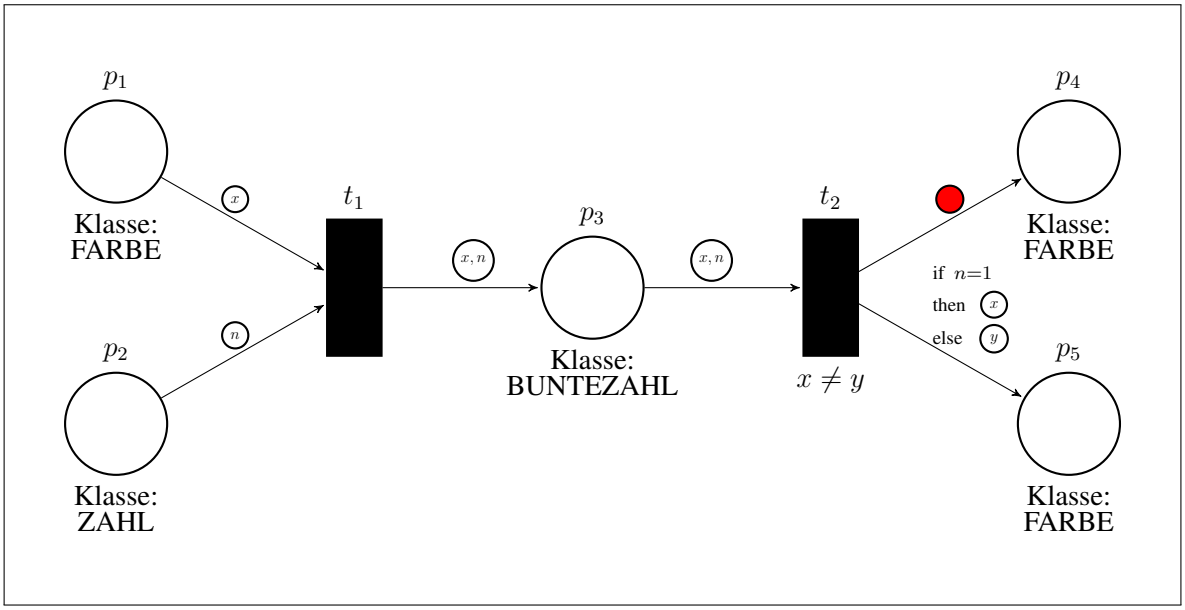

Abbildung 1.24: CPN mit Variablen (Formalsprache)

Definition 1.14:
Es sei ein ***gefärbtes Petri-Netz mit Variablen*** $N = (P, T, F, B, A, K, k, \mathbb{V}, Attr, Var, \mathbb{B}, m, \mathbb{E}, f)$ gegeben, zudem ein Modus β und eine Transition $t \in T$. Dann bezeichnet man mit

$$f_\beta(p,t) = [f(p,t)](\beta) = Expr(\beta)$$

bzw.

$$f_\beta(t,p) = [f(t,p)](\beta) = Expr(\beta)$$

ein ***Pfeilgewicht im Modus β bzgl. Transition t***.

Beispiel:
Für das Beispiel aus Abb. 1.23 ist das Pfeilgewicht

$$f(p_1, t_1) = Expr_2$$

im Modus

$$\beta_1 : \{x = \text{ROT}, y = \text{ROT}, n = 1\}$$

gegeben als

$$f_{\beta_1}(p_1, t_1) = f(p_1, t_1)(\beta_1) = Expr_2(\beta_1) = \{(\text{ROT}, 1)\}.$$

Wenn nur für eine Transition ein Modus gewählt wird, ergibt sich auf kanonische Weise ein durch β und t ***induziertes MCPN***, kurz (β,t)-MCPN (vgl. [BKK+14, S.117 f.]).

Beispiel:
Für das Beispiel aus Abb. 1.23 sei der Modus

$$\beta_1 : \{x = \text{ROT}, y = \text{ROT}, n = 1\}$$

für Transition t_1 gewählt, daher erhält man das folgende (β_1,t_1)-MCPN:

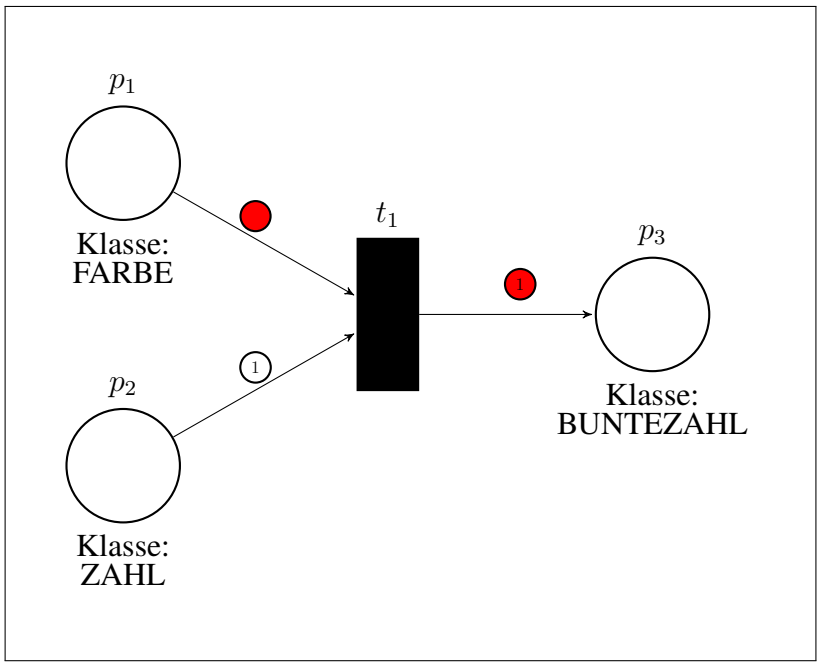

Abbildung 1.25: Durch Modus-Wahl entstandenes (β_1,t_1)-MCPN aus CPN mit Variablen in Abb. 1.23 bzw. Abb. 1.24

Bemerkung 1.15:

Man beachte, dass man ein MCPN erhält, wenn für jede Transition genau ein Modus gewählt wird und dadurch alle zugehörigen Pfeile in diesem Modus sind. Daher lassen sich viele Grundbegriffe auf die CPN mit Variablen leicht übertragen bzw. können äquivalent benutzt werden, unter anderem die Begriffe *Vor-/Nachbereich* (vgl. Def. 1.5) und *Zustand* (vgl. Def. 1.6). Des Weiteren lassen sich, wie aus dem obigen Beispiel ersichtlich, nicht nur die Begriffe *aktivierbar, feuerbar* (vgl. [BKK$^+$14, Def. 6.7]) und *nebenläufig feuerbar* (vgl. [BKK$^+$14, Def. 6.8]) eins zu eins übertragen, sondern auch der *allgemeine Konflikt* (vgl. [BKK$^+$14, Def. 5.2]) und die dazu gehörigen Konflikt-Lösungsstrategien (vgl. [BKK$^+$14, Abschn. 5]), wodurch das *Feuerszenario* 4 aus [BKK$^+$14, S.138] auch auf CPN mit Variablen anwendbar ist.

2 Berechnung der aktiven Modus-Menge durch die Pattern-Matching-Methode

Generell werden jeder Transition t bestimmte Modi aus der Modus-Menge $\mathbb{B}$ zugewiesen. Die Mächtigkeit der verwendeten Klassen ist dabei u.a. maßgeblich für die maximale Mächtigkeit der Menge $\mathbb{B}$ und somit auch für die zugewiesenen Modus-Mengen $m(t)$. Andererseits ist die Quantität der Variablen in hohem Maße mitentscheidend.

Im Sinne von Feuerprozess-Szenarien (vgl. [BKK$^+$14, Kap. 3 und Kap. 7]) und daran anknüpfend bei der Modellierung von Petri-Netzen ist es entscheidend zu wissen, ob eine Transition t in einem gegenwärtigen Zustand aktiv ist und daher mindestens ein feuerbarer Modus existiert. In Bezug auf CPN mit Variablen müssen die Modi aus der Modus-Menge $m(t)$ ausfindig gemacht werden, die dafür sorgen, dass die Transition t aktiv wird. Mit anderen Worten sollen die Modi ausgeschlossen werden, welche zu einer nicht aktivierbaren Transition t führen. Daher soll eine Modus-Menge bestimmt werden, welche ausschließlich alle aktiven Modi angibt. Diese Menge soll *aktive Modus-Menge* genannt werden.

Definition 2.1:
Das Tupel $N = (P, T, F, B, A, K, k, \mathbb{V}, Attr, Var, \mathbb{B}, m, \mathbb{E}, f)$ sei ein gefärbtes Petri-Netz mit Variablen und Modus-Menge und $\mathbf{z}$ eine Markierung von N. Ein Modus $\beta \in m(t)$ ist ein ***aktiver Modus*** für eine Transition $t \in T$ von N und Zustand $\mathbf{z}$, wenn für alle $p \in {\bullet}t$ gilt:

$$f_\beta(p, t) \subseteq \mathbf{z}(p).$$

Des Weiteren sei mit der Menge

$$Am_{\mathbf{z}}(t) := \{\beta \in m(t) | \, \forall \, p \in {\bullet}t : f_\beta(p, t) \subseteq \mathbf{z}(p)\}$$

die ***aktive Modus-Menge einer Transition im Zustand z*** bezeichnet, die jeder Transition $t \in T$ genau die Teilmenge der Modus-Menge $m(t)$ zuweist, welche lediglich die im Zustand **z** aktiven Modi beinhaltet.

Ist die Markierung **z** fest vorgegeben bzw. als bekannt vorauszusetzen, wird vereinfacht auch $Am(t)$ statt $Am_{\mathbf{z}}(t)$ geschrieben.

Für die Bestimmung der aktiven Modus-Menge kann im schlimmsten Fall eine vollständige Suche erforderlich sein. Dazu wird das Beispiel mit der Klasse ZAHL $:= \{1, \ldots, 100\}$ und der Modus-Menge $m(t_1) = \mathbb{B}_{max}$ betrachtet.

Allgemein kann für die Berechnung der aktiven Modus-Menge $Am(t_1)$ die Methode der *vollständigen Enumeration*, also Überprüfung aller theoretisch möglichen Modi, verwendet werden. Bei diesem Beispiel (Abb. 2.1) müssten ca. $100^{10} = 100.000.000.000.000.000.000$ (100 Trillionen) Kombinationen geprüft werden.

Damit die aktive Modus-Menge $Am(t_1)$ in akzeptabler Rechenzeit berechnet werden kann, muss eine angemessene Lösungsstrategie ermittelt werden. Anhand der sog. *Pattern-Matching-Methode* nach [JK09], [KC04] ist ein Vorgehen möglich, wonach mit annehmbarem Rechenaufwand die aktive Modus-Menge bestimmt werden kann. Hierfür müssen zunächst einmal die Strukturen der variablen Pfeilgewichtungen betrachtet werden, da bestimmte Strukturen die Anwendung eines Verfahrens beeinflussen. Generell gilt, je „einfacher“ die Struktur der Pfeilgewichte ist, desto effizienter kann ein Verfahren zur Bestimmung der aktiven Modus-Menge verwendet werden.

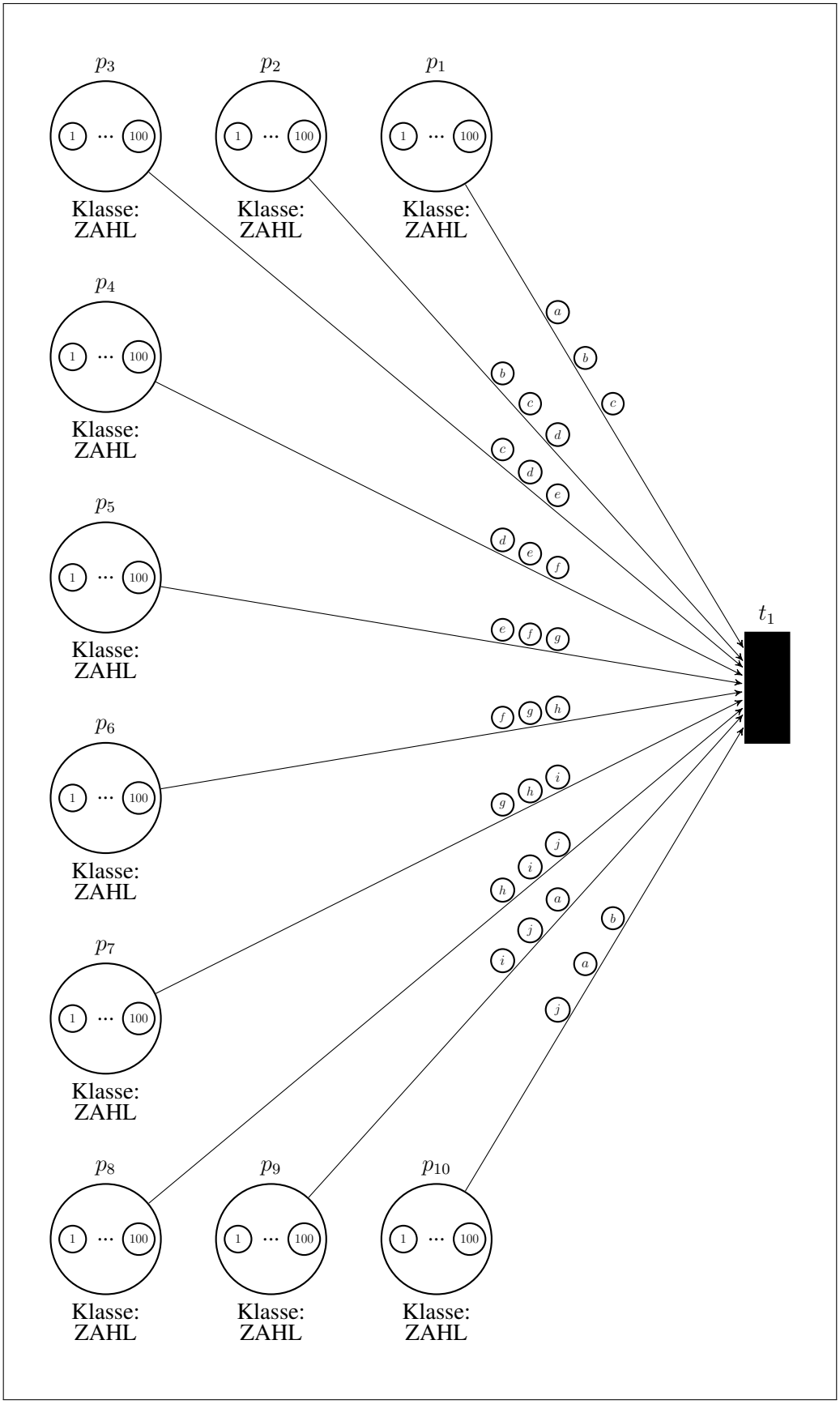

Abbildung 2.1: Problematik der vollständigen Suche bei der Modus-Wahl

2.1 Strukturen der variablen Pfeilgewichtungen

Allgemein lässt sich die Struktur von variablen Pfeilgewichtungen in folgende Kategorien einteilen:

- **Konstante Pfeilgewichtung**, die unabhängig von dem Modus eine konstante Multimenge als Pfeilgewicht verwendet (vgl. Abb. 1.22),
- **Einfache Pfeilgewichtung**, die eine Multimenge aus Variablen als Pfeilgewicht verwendet (vgl. Abb. 1.19 und Abb. 1.20),
- **Tupel Pfeilgewichtung**, die eine Multimenge aus Variablen-Tupeln als Pfeilgewicht verwendet (vgl. Abb. 1.21),
- **Operator Pfeilgewichtung**, die durch eine Algebra auf den Variablen eine Multimenge als Pfeilgewicht erzeugt (vgl. Abb. 1.18).

Bei dem grundsätzlichen Vorgehen der Pattern-Matching-Methode wird in diesem Kapitel o.B.d.A. davon ausgegangen, dass ausschließlich einfache Pfeilgewichtungen vorhanden sind.

Bemerkung 2.2:
Wenn nur einfache Pfeilgewichtungen vorhanden sind, kann als Gewichtungsfunktion folgende Abbildung verwendet werden:

> $fe : (F \cup B) \to \mathbb{M}(\mathbb{V})$, die jedem Pfeil $(p,t) \in F$ bzw. $(t,p) \in B$ eine der jeweiligen Klasse von p entsprechende Multimenge als Gewicht zuweist, indem $fe(p,t)$ bzw. $fe(t,p)$ eine Multimenge $M = \{(a, h(a)) \mid a \in \mathbb{V}, h(a) \in \mathbb{N}_0\}$ ist, wobei $\forall\, a \in supp(M)$ gilt:
>
> $$k(a) = k(p),$$
>
> die man als **einfache Gewichtungsfunktion** bezeichnet.

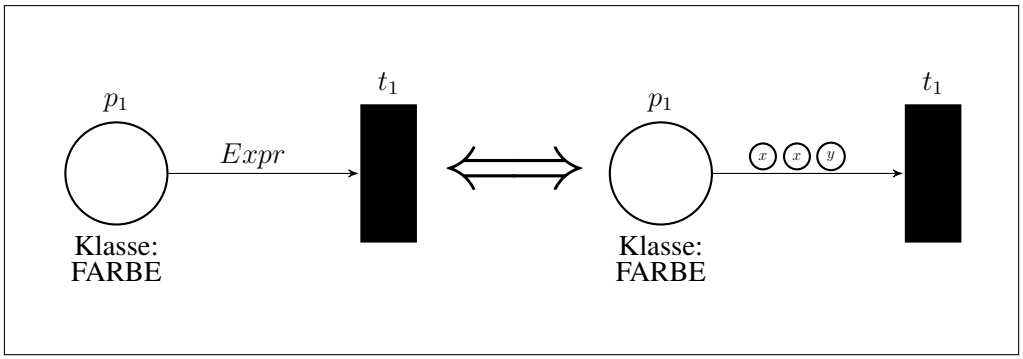

Abbildung 2.2: Darstellung eines CPN mit einfacher Pfeilgewichtung

Beispiel 2.3:
Gegeben sei das CPN aus Abb. 2.2 mit folgender Modus-Menge

Tabelle 2.1: Modus-Menge mit Mächtigkeit 9

β	x	y
1	ROT	ROT
2	ROT	BLAU
3	ROT	GRÜN
4	BLAU	ROT
5	BLAU	BLAU
6	BLAU	GRÜN
7	GRÜN	ROT
8	GRÜN	BLAU
9	GRÜN	GRÜN

und der MME-Funktion

$$Expr(\beta_i) = \begin{cases} \{(\text{ROT}, 3)\} & \text{falls } \beta = \beta_1 \\ \{(\text{ROT}, 2), (\text{BLAU}, 1)\} & \text{falls } \beta = \beta_2 \\ \{(\text{ROT}, 2), (\text{GRÜN}, 1)\} & \text{falls } \beta = \beta_3 \\ \{(\text{ROT}, 1), (\text{BLAU}, 2)\} & \text{falls } \beta = \beta_4 \\ \{(\text{BLAU}, 3)\} & \text{falls } \beta = \beta_5 \\ \{(\text{BLAU}, 2), (\text{GRÜN}, 1)\} & \text{falls } \beta = \beta_6 \\ \{(\text{ROT}, 1), (\text{GRÜN}, 2)\} & \text{falls } \beta = \beta_7 \\ \{(\text{BLAU}, 1), (\text{GRÜN}, 2)\} & \text{falls } \beta = \beta_8 \\ \{(\text{GRÜN}, 3)\} & \text{falls } \beta = \beta_9 \end{cases}$$

die man auch als folgende einfache Gewichtungsfunktion

$$fe(p_1, t_1) = \{(x, 2), (y, 1)\}$$

darstellen kann.

2.2 Die Pattern-Matching-Methode

Bei dem grundsätzlichen Problem in CPN mit Variablen geht es darum, festzustellen, welche Ausprägungen den Variablen zugeordnet werden, um schlussendlich die aktive Modus-Menge $Am(t)$ angeben zu können. Mit der Verwendung der Pattern-Matching-Methode kann ein Vorgehen vorgeschlagen werden, um ausschließlich aktive Modi einer Transition t zu ermitteln.

Die Pattern-Matching-Methode zeichnet sich durch die Verwendung von *Mustern* (*Pattern*) aus. Ein Muster beinhaltet Variablen der Pfeilgewichtungen des gefärbten Petri-Netzes. Da nur die Variablen im Vorbereich einer Transition t relevant für die Untersuchung der Feuerbarkeit sind, werden die Variablen im Nachbereich im Laufe des Verfahrens zunächst außer Acht gelassen.

Um Zuordnungen für die Variablen des Vorbereichs, die nun als Muster vorliegen, zu erhalten, wird ein *Abgleich* (*Matching*) durchgeführt. Konkret werden dabei die vorher festgelegten Muster mit Ausprägungen abgeglichen. Dabei sind die unterschiedlichen Ausprägungen auf den dazugehörigen Plätzen im Vorbereich einer Transition t gemeint. Der Abgleich eines Musters erzeugt sogenannte *partielle* Modi. Diese speziellen Modi zeichnen sich dadurch aus, dass in der Regel nur einem bestimmten Anteil aller Variablen von t, $Var(t)$, eine Ausprägung zugeordnet werden konnte. Des Weiteren ist deren Mächtigkeit von der Quantität verschiedener Token in einem Platz abhängig.

Für gewöhnlich enthält ein gefärbtes Petri-Netz mehrere Muster, sodass entsprechend mehrere partielle Modi entstehen. Das Ziel ist es, aus der unübersichtlichen Anzahl partieller Modi eine *Zusammenführung* (*Merge*) zu erzielen, damit am Ende die aktive Modus-Menge $Am(t)$ angegeben werden kann.

Das folgende Beispiel soll die einleitenden Erläuterungen des Musterabgleiches (Pattern-Matching) unterstützen und den Vorgang der Zusammenführung an einem CPN mit einfachen Pfeilgewichtungen verdeutlichen.

Beispiel 2.4:

Gegeben sei das gefärbte Petri-Netz mit Variablen aus Abb. 2.3,

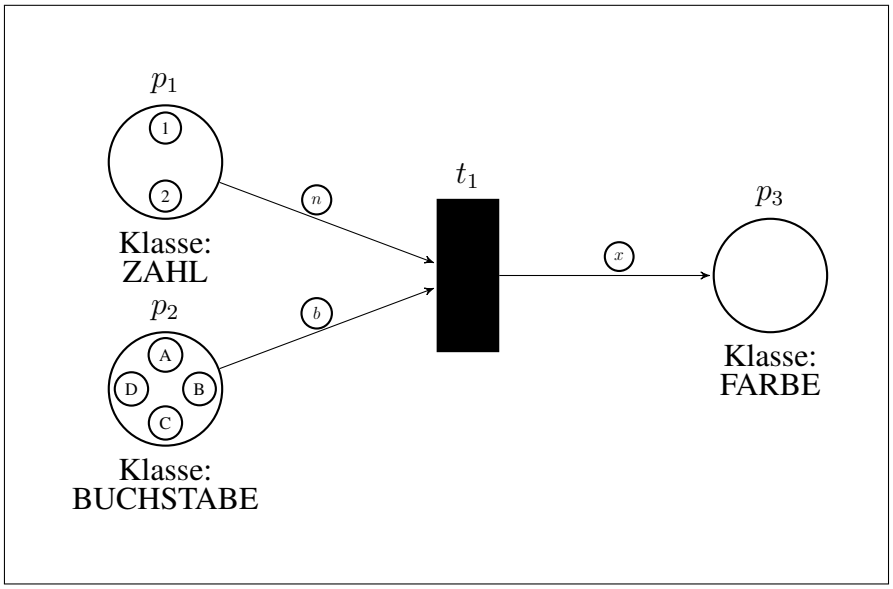

Abbildung 2.3: Beispiel eines gefärbten Petri-Netzes mit Variablen

mit den folgenden:

- Klassen $K = \{$BUCHSTABE,ZAHL,FARBE$\}$ mit:

  BUCHSTABE $= \{\text{A}, \ldots, \text{Z}\}$
  ZAHL $= \{1, 2, 3\}$
  FARBE $= \{$SCHWARZ,ROT,BLAU,GELB$\}$

- Variablen $\mathbb{V} = \{b, n, x\}$

- Klassifizierungen der Plätze und Attributierungen der Variablen:

  Plätze

  $k(p_1) =$ ZAHL
  $k(p_2) =$ BUCHSTABE
  $k(p_3) =$ FARBE

  Variablen

  $k(n) =$ ZAHL
  $k(b) =$ BUCHSTABE
  $k(x) =$ FARBE

- einfachen Gewichtungsfunktionen:

$$fe(p_1, t_1) = \{(n, 1)\}$$
$$fe(p_2, t_1) = \{(b, 1)\}$$
$$fe(t_1, p_3) = \{(x, 1)\}$$

Für das vorliegende Beispiel ist ein Modus β folgendermaßen aufgebaut:

$$\beta : \{b, n, x\},$$

wobei die explizite Angabe der 32 aktiven Modi nun durch den Muster-Abgleich exemplarisch vorgestellt wird.

Definition 2.5:
Gegeben sei eine einfache Pfeilgewichtung $fe(p, t)$, dann werden die Elemente $Pat \in supp(fe(p, t))$ als Muster bezeichnet.

Beispiel 2.6:
Nachfolgende Muster sind im Vorbereich des CPN vorhanden:

für die Pfeilgewichtung $fe(p_1, t_1)$ das Muster $Pat_1 = \{n\}$
für die Pfeilgewichtung $fe(p_2, t_1)$ das Muster $Pat_2 = \{b\}$

Definition 2.7:
Eine Abbildung $\rho : \mathbb{V} \to \mathbb{K}^*$, die jeder Variablen $v \in \mathbb{V}$ eine Ausprägung $a \in \mathbb{K}^*$ zuordnet, bezeichnet man als **partiellen Modus**, wobei $\mathbb{K}^* = \mathbb{K} \cup \{?\}$.

Bemerkung 2.8:
Es ist zu beachten, dass der Ausdruck *?* keineswegs aus einer Klassen-Menge *Sonderzeichen* stammt, sondern einzig und allein angibt, dass beim Abgleich keine Ausprägung für die dazugehörige Variable $v \in \mathbb{V}$ zugeordnet werden konnte.

Definition 2.9:
Das Tupel $N = (P, T, F, B, A, K, k, \mathbb{V}, Attr, Var, \mathbb{B}, m, fe)$ sei ein gefärbtes Petri-Netz mit Variablen, Modus-Menge, einfacher Gewichtungsfunktion und **z** eine Markierung von N. Zudem sei ein partieller Modus ρ und eine Transition $t \in T$ gegeben. Dann bezeichnet man mit

$$fe_\rho(p,t) = \{(\rho(a), h(a)) \mid (a, h(a)) \in fe(p,t) \wedge \rho(a) \neq ?\}$$

bzw.

$$fe_\rho(t,p) = \{(\rho(a), h(a)) \mid (a, h(a)) \in fe(t,p) \wedge \rho(a) \neq ?\}$$

ein ***Pfeilgewicht im partiellen Modus ρ bzgl. Transition t***.

Definition 2.10:
Es sei ein partieller Modus ρ und eine Transition $t \in T$ gegeben. Der partielle Modus ρ heißt **zulässig**, wenn für alle $p \in {}^\bullet t$ gilt: $fe_\rho(p,t) \subseteq z(p)$.

Bemerkung 2.11:
Falls $\rho(v) \neq ?$ für alle $v \in \mathbb{V}$ gilt, liegt ein aktiver Modus vor.

Beispiel 2.12:
Man beginne z.B. mit Muster Pat_1. Beim Vergleich zwischen Pat_1 und dem Platz p_1 aus dem Vorbereich entstehen folgende partielle Modi:

$$\rho_1 : \{b = ?, n = 1, x = ?\},$$
$$\rho_2 : \{b = ?, n = 2, x = ?\}.$$

Des Weiteren entstehen folgende partielle Modi beim Abgleich des Musters Pat_2 mit den Token des Platzes p_2:

$$\rho_3 : \{b = \mathrm{A}, n = ?, x = ?\},$$
$$\rho_4 : \{b = \mathrm{B}, n = ?, x = ?\},$$
$$\rho_5 : \{b = \mathrm{C}, n = ?, x = ?\},$$
$$\rho_6 : \{b = \mathrm{D}, n = ?, x = ?\}.$$

Anschließend erfolgt eine Zusammenführung der partiellen Modi.

Das Zusammenführen zweier partieller Modi ist unterteilt in zwei Schritte, *Verträglichkeit* und

Vereinigung. Zwei partielle Modi ρ_1, ρ_2 sind verträglich unter der Voraussetzung, dass sie in den Ausprägungen der Variablen, falls zugeordnet, übereinstimmen.

Definition 2.13:
Zwei partielle Modi ρ_1 und ρ_2 heißen **verträglich**, wenn

- $\forall\, v \in \mathbb{V} : (\rho_1(\mathrm{v}) \neq ? \wedge \rho_2(\mathrm{v}) \neq ?) \Rightarrow \rho_1(\mathrm{v}) = \rho_2(\mathrm{v})$

Die Vereinigung von zwei verträglichen partiellen Modi ρ_1, ρ_2 in den partiellen Modus $\hat{\rho}$ geschieht nach folgendem Schema:

$$\hat{\rho}(v) = \begin{cases} \rho_1(v) & \textit{falls } \rho_1(v) \neq ? \wedge \rho_2(v) = ? \\ \rho_2(v) & \textit{falls } \rho_2(v) \neq ? \wedge \rho_1(v) = ? \\ \rho_1(v) & \textit{falls } \rho_1(v) = \rho_2(v) \end{cases}$$

Zusammenfassend lässt sich das Zusammenführen (Merge) von zwei partiellen Modi ρ_1 und ρ_2 folgendermaßen formal aufschreiben:

Da partielle Modi eine Abbildung von $\mathbb{V} \rightarrow \mathbb{K}^*$ sind, können sie auch als eine Relation von $\mathbb{V} \times \mathbb{K}^*$ aufgefasst werden, sodass die Menge aller möglichen partiellen Modi eines CPN eine Teilmenge der Potenzmenge $\wp(\mathbb{V} \times \mathbb{K}^*)$ ist. Das Zusammenführen (Merge) soll aus zwei partiellen Modi einen partiellen Modus erzeugen, mit:

$$\text{Merge} : \wp(\mathbb{V} \times \mathbb{K}^*) \times \wp(\mathbb{V} \times \mathbb{K}^*) \rightarrow \wp(\mathbb{V} \times \mathbb{K}^*),$$

wobei $\text{Merge}(\rho_1, \rho_2)$ sich nach den obigen Prinzipien für $\hat{\rho}$ bildet.

An dieser Stelle sei noch angemerkt, dass im Zuge der Zusammenführung von zwei partiellen Modi-Mengen jede Kombinationsmöglichkeit hinsichtlich der obengenannten Prinzipien durchgeführt werden muss. Zudem kann die Anzahl der partiellen Modi, die als Resultat entstehen, größer bzw. kleiner sein als vor dem Zusammenführen. Abhängig ist dies von den Ausprägungen, die den Variablen bereits zugewiesen wurden.

Beispiel 2.14:
Für die Zusammenführung werden zunächst die partiellen Modi der Muster Pat_1 und Pat_2 betrachtet und miteinander verglichen. Folgende partielle Modi ergeben sich nach der Zusammenführung:

$$\hat{\rho}_1 : \{b = \text{A}, n = 1, x = ?\},$$
$$\hat{\rho}_2 : \{b = \text{A}, n = 2, x = ?\},$$
$$\hat{\rho}_3 : \{b = \text{B}, n = 1, x = ?\},$$
$$\hat{\rho}_4 : \{b = \text{B}, n = 2, x = ?\},$$
$$\hat{\rho}_5 : \{b = \text{C}, n = 1, x = ?\},$$
$$\hat{\rho}_6 : \{b = \text{C}, n = 2, x = ?\},$$
$$\hat{\rho}_7 : \{b = \text{D}, n = 1, x = ?\},$$
$$\hat{\rho}_8 : \{b = \text{D}, n = 2, x = ?\}.$$

Für die Variable n sind beim zweiten Abgleich keine Ausprägungen zugewiesen worden. Selbiges gilt beim ersten Abgleich für die Variable b. Die partiellen Modi können daher vollständig zusammengeführt werden. Sollte z.B. die Variable n des partiellen Modus ρ_6 die Ausprägung 3 aufweisen, könnte dieser nicht zusammengeführt werden. Dies liegt daran, dass kein partieller Modus mit identischer Zuweisung beim ersten Abgleich entstanden ist.

Beispielsweise sind die beiden partiellen Modi

$$\rho_1 : \{b = \text{A}, n = 3, x = ?\},$$
$$\rho_2 : \{b = ?, n = 2, x = ?\},$$

nicht verträglich und können nicht zusammengeführt werden.

Abschließend werden Ausprägungen für die Variable x aus dem Nachbereich festgelegt, d.h. es werden alle möglichen Ausprägungen aus der Klasse FARBE gewählt.

Die vollständige aktive Modus-Menge der Transition t_1 ist in Tabelle 2.2 zu finden.

Tabelle 2.2: Aktive Modus-Menge mit der Mächtigkeit 32

β	b	n	x
1	A	1	SCHWARZ
2	A	1	ROT
3	A	1	BLAU
4	A	1	GELB
5	A	2	SCHWARZ
6	A	2	ROT
7	A	2	BLAU
8	A	2	GELB
9	B	1	SCHWARZ
10	B	1	ROT
11	B	1	BLAU
12	B	1	GELB
13	B	2	SCHWARZ
14	B	2	ROT
15	B	2	BLAU
16	B	2	GELB
17	C	1	SCHWARZ
18	C	1	ROT
19	C	1	BLAU
20	C	1	GELB
21	C	2	SCHWARZ
22	C	2	ROT
23	C	2	BLAU
24	C	2	GELB
25	D	1	SCHWARZ
26	D	1	ROT
27	D	1	BLAU
28	D	1	GELB
29	D	2	SCHWARZ
30	D	2	ROT
31	D	2	BLAU
32	D	2	GELB

Die aktive Modus-Menge $Am(t_1)$ beinhaltet also, wie vorher schon vorweggenommen, 32 verschiedene Modi. Es wird deutlich, dass bereits ein übersichtliches CPN mit einfachen Variablen viele aktive Modi aufweisen kann.

Anders als im vorherigen Beispiel (siehe Abb. 2.3) müssen bei größeren CPN mit einfachen Pfeilgewichtungen mehrere partielle Modi erstellt und zusammengeführt werden.

Die folgende Abb. 2.4 zeigt ein komplexeres Netz.

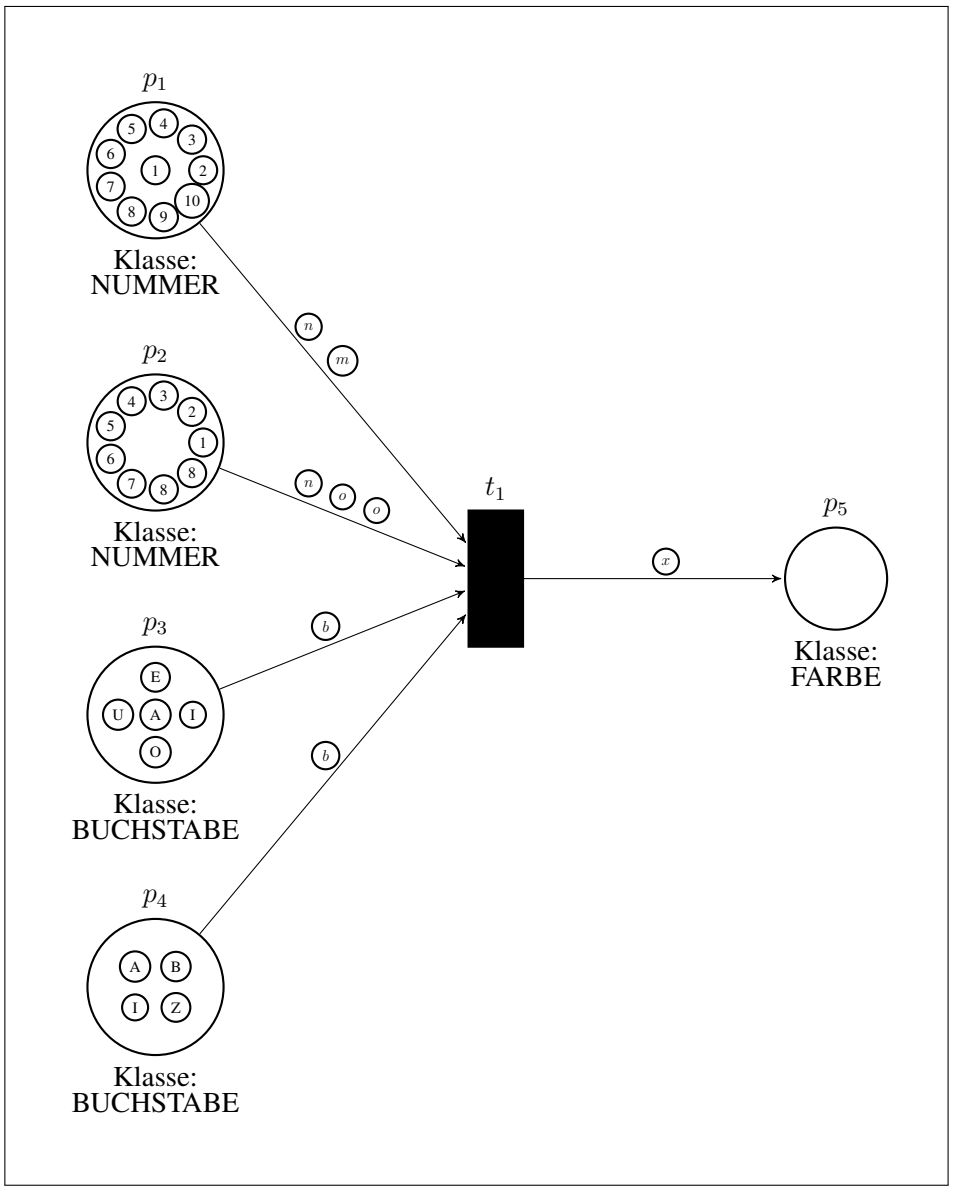

Abbildung 2.4: Beispiel eines größeren Netzes

Zur zusätzlichen Verdeutlichung des höheren Aufwandes werden einige Inhalte aus dem CPN klassifiziert.

1. Die Klassen aus $K = \{$BUCHSTABE,NUMMER,FARBE$\}$ mit:

   BUCHSTABE $= \{\text{A}, \ldots, \text{Z}\}$
   NUMMER $= \{1, ..., 10\}$
   FARBE $= \{$SCHWARZ,ROT,BLAU,GELB$\}$

2. Die Muster:

   $Pat_1 = \{n\}$ und $Pat_2 = \{m\}$ für die Pfeilgewichtung $fe(p_1, t_1)$,
   $Pat_3 = \{n\}$ und $Pat_4 = \{o\}$ für die Pfeilgewichtung $fe(p_2, t_1)$,
   $Pat_5 = \{b\}$ für die Pfeilgewichtung $fe(p_3, t_1)$,
   $Pat_6 = \{b\}$ für die Pfeilgewichtung $fe(p_4, t_1)$.

Durch das Auftreten mehrerer Muster kann festgestellt werden, dass viel mehr partielle Modi (in diesem Fall 38) erzeugt und im Anschluss zusammengeführt werden müssen.

Eine algorithmische Beschreibung der Patern-Matching Methode für CPN mit einfachen Pfeilgewichtungen ist im Anhang zu finden.

2.3 Erweiterungsmöglichkeit der Pattern-Matching-Methode

Im Folgenden sollen einige effiziente Erweiterungsmöglichkeiten der Pattern-Matching-Methode vorgestellt werden, mit denen auch Wächter und beliebige Strukturen von Pfeilgewichtungen behandelt werden können.

2.3.1 Wächter

Es ist möglich, das Verfahren so zu modifizieren, dass der sogenannte Wächter (Guard) mit berücksichtigt wird. Wächter sind boolesche Ausdrücke an den Transitionen, welche zusätzliche Bedingungen aufstellen und somit die Modus-Menge $m(t)$ einschränken (vgl. Bem. 1.11).

Beispielsweise könnte die Transition im vorherigen Netz (vgl. Abb. 2.4) exemplarisch den Wächter 'n>1' aufweisen. Damit würde z.B. ein Modus ungültig werden, dessen Variable n den Wert 1 aufweist.

Formal gesehen ist ein Wächter in einer konjunktiven Form gegeben:

$$G(t) = \wedge_{i=1}^{m} G_i$$

Dabei ist jede Form G_i ein eigenständiger Ausdruck, der mit den gängigen logischen Operatoren wie z.B. '$n \neq$1' oder 'n<1' versehen werden kann.

Jeder Ausdruck G_i wird als separates Muster aufgefasst.

Folgendes Beispiel zeigt das Netz aus Abb. 2.4 inklusive Wächter, welcher angelehnt an den obigen Formalismus zwei separate Muster enthält.

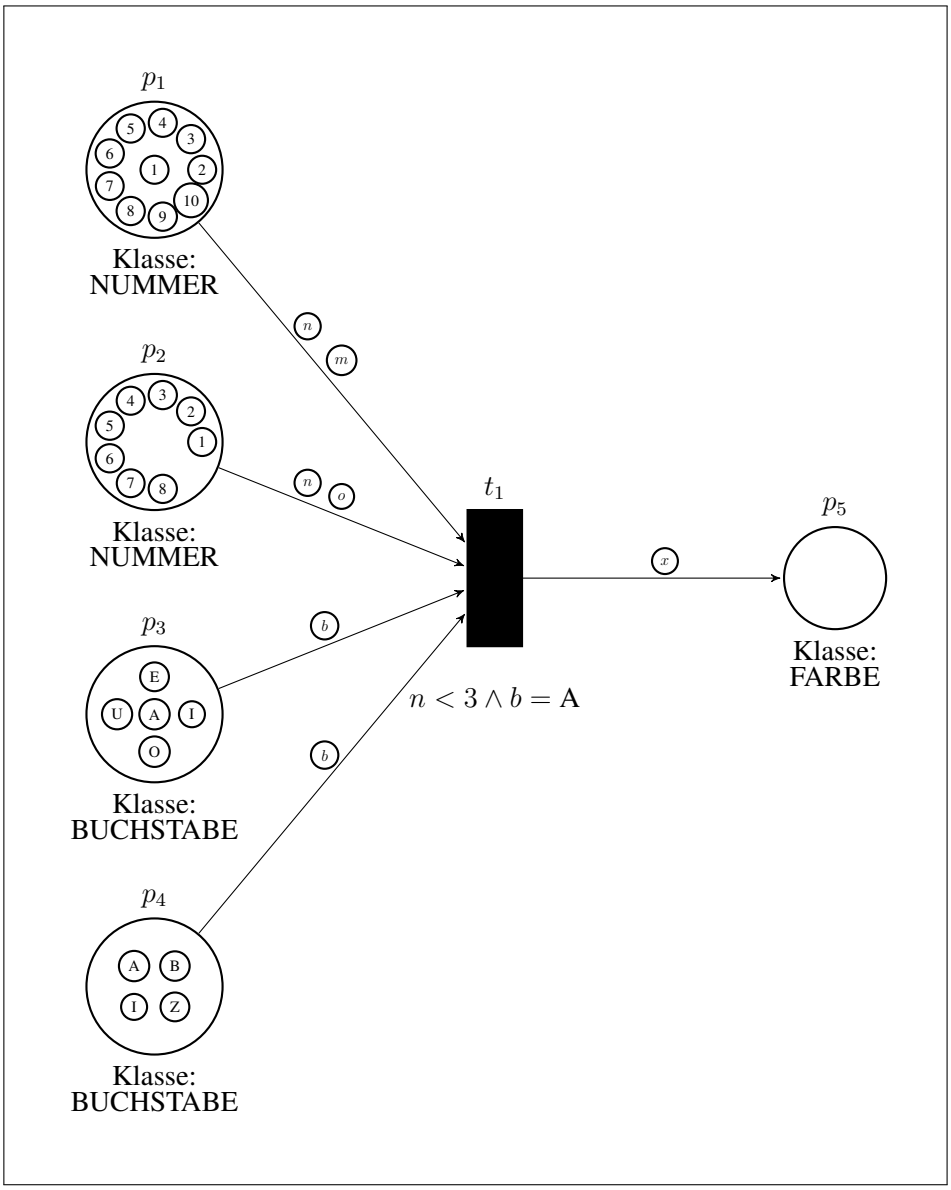

Abbildung 2.5: Beispiel eines größeren CPN inklusive Wächter

Aufgrund der Restriktionen des Wächters und der Token im Platz p_1 und p_2 kann die Variable n lediglich auf 1 oder 2 gesetzt werden.

Im obigen Beispiel (Abb. 2.5) würde durch die Restriktion des Wächters die Variable b auf 'A' gesetzt werden. Somit ergibt sich eine Reduzierung des Rechenaufwandes für die Plätze p_3 und p_4, um im Gesamtergebnis die aktive Modus-Menge $Am(t)$ anzugeben.

Weiter sind aufgrund der aktuellen Markierung des Platzes p_1 zehn verschiedene Ausprägungen vorhanden und somit hinsichtlich zweier Variablen auf der dazugehörigen Pfeilgewichtung 100 verschiedene Möglichkeiten auswählbar. Jedoch kann durch die Restriktion des Wächters die Variable n nur zwei verschiedene Attribute annehmen. Aus diesem Grund sind nur noch 20 Möglichkeiten realisierbar, was ausschließlich für den Platz p_1 eine Reduzierung von 80 Möglichkeiten bzw. 80 % darstellt. Es wird daher deutlich, dass durch einen Wächter eine hohe Einschränkung auf die Modus-Menge ausgeübt werden und somit die Angabe der aktiven Modus-Menge $Am(t)$ schneller erfolgen kann.

2.3.2 Ansatz über die optimierte Muster-Menge

Im Folgenden wird eine Möglichkeit nach [LH10] aufgezeigt, um die Angabe der aktiven Modus-Menge $Am(t)$ effizienter durchzuführen. Wie bei dem bisherigen Pattern-Matching werden anhand einer lexikografischen Sortierung der Variablen partielle Modi aufgestellt und anschließend zusammengeführt. Es kann durchaus nützlich sein, dass das Aufstellen der partiellen Modi einer Reihenfolge unterliegt, die das Ziel verfolgt, die Rechenzeit so kurz wie möglich zu halten. Vielmehr soll der Prozess der Zusammenführung möglichst überschaubar aufgerufen werden. Durch den folgenden Testfall lässt sich dies sehr gut verdeutlichen.

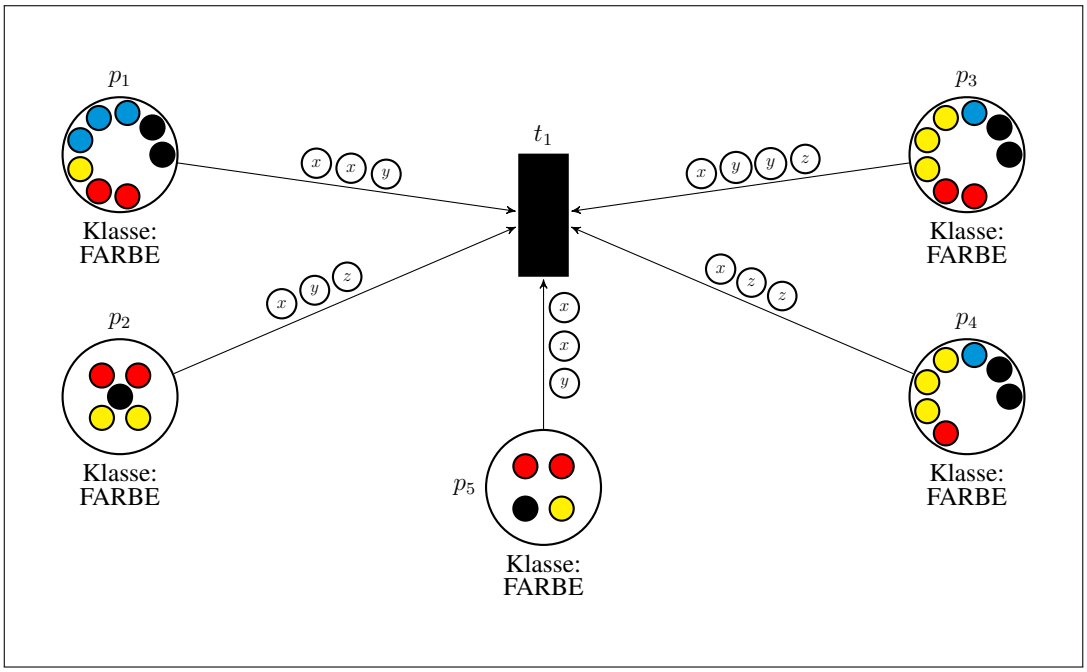

Abbildung 2.6: Effizienzproblematik der Pattern-Matching-Methode

Man betrachte den Platz p_5 mit entsprechender Kantenverbindung. Durch die gegebene Markierung wird festgestellt, dass die Variable x ausschließlich die Ausprägung ROT annehmen kann.

Infolge dieser Tatsache könnten alle im Netz vorkommenen Variablen x vernachlässigt werden. Weiterhin ist zu erkennen, dass die Variable y die Zuweisung ROT nicht erhalten kann, weswegen die restlichen Schritte angepasst werden können.

Auf dieser Grundlage kann das Netz weiter systematisch durchlaufen werden, sodass ein möglichst geringer Aufwand entsteht. Man kann die Zuweisungsmöglichkeiten der Variablen x und y mit dem Platz p_4 weiter konkretisieren, damit eine noch effektivere Vorgehensweise möglich wird. Anschaulich gesagt können lediglich die Ausprägungen GELB und SCHWARZ für die Variable z festgelegt werden. Durch diese Erkenntnis können wiederum weitere unzulässige partielle Modi vermieden werden.

Es wird angestrebt, viele unzulässige partielle Modi bereits im Vorfeld zu vermeiden, da somit auch der vergleichbare „teure“ Prozess der Zusammenführung reduziert werden kann. Es ist entscheidend, welche partiellen Modi an erster Stelle gebildet werden und welche Feststellungen dadurch entnommen werden können. Dadurch wird ein logischer, effektiver Verlauf für das Bestimmen der aktiven Modus-Menge $Am(t)$ ermöglicht.

Diese Überlegungen über eine Optimierungsmöglichkeit führen zu der Vorgehensweise nach [LH10], um die Berechnung der aktiven Modus-Menge $Am(t)$ noch effizienter vorzunehmen.

Im Folgenden sollen die Hauptaspekte dieser Lösungsstrategie näher gebracht werden. Wie im Abschnitt 2.2 beschrieben wurde, werden alle auftretenden Variablen aus dem Vorbereich einer Transition t als Muster aufgefasst und im Anschluss abgeglichen. In der vorliegenden Herangehensweise werden zu Beginn alle Muster in einer *Muster-Menge* zusammengefasst und nach bestimmten Kriterien umstrukturiert. Die Hauptmodifikation ist im Vergleich zu der gewöhnlichen Pattern-Matching-Methode eine Neuordnung der Muster. Ansonsten wird derselbe Mechanismus verwendet, um partielle Modi zu erzeugen und zusammenzuführen. Um die Strategie in vollem Umfang zeigen zu können, wird von einem Netz mit Wächter-Funktion und umfangreicheren Strukturen der Pfeilgewichtungen ausgegangen. Im Detail werden als erstes alle Muster auf den Kantenverbindungen des Vorbereichs in der Menge $AS(t)$ zusammengefasst. Die Muster des Wächters finden sich in der Menge $GS(t)$ wieder. Daraus ergibt sich die gesamte Muster-Menge $PS(t) = AS(t) \cup GS(t)$.

Gegenüber der üblichen Vorgehensweise aus Abschnitt 2.2 ist für diesen Ablauf jedes Muster $Pat \in PS(t)$ durch ein Quintupel $Pat = (M_T, E, X, z(p), \mu)$ definiert, wobei

- M_T den Typ des Musters festlegt (Konstant,Variable,Tupel,Wächter etc.),
- E die formalsprachliche Darstellung des Musters widerspiegelt,
- X die Menge der Variablen im Muster angibt,
- $z(p)$ die Markierung des Platzes angibt, mit dem das Muster durch den Pfeil verbunden ist,
- μ dem Multiplikator des Musters entspricht.

Für das Muster eines Wächters werden lediglich die ersten drei Komponenten M_T,E und X belegt sein. Bei einem konstanten Muster wird X immer $\varnothing$ gesetzt.

Für die Angabe einer exemplarischen Muster-Menge $PS(t)$ wird das Netz der Abb. 2.5 modifiziert. Zum einen werden die Pfeilgewichtungen der Kanten (p_2, t_1) und (p_4, t_1) verändert. Zum anderen wird der Platz p_2 um zwei Token ergänzt.

Die Muster-Menge $PS(t)$ enthält acht Elemente, wobei die jeweiligen Elemente beschrieben sind durch:

- Pat_1 = (Variable,$n, \{n\}$,$\{1^1, 2^1, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1, 9^1, 10^1\}$,1),
- Pat_2 = (Variable,$m, \{m\}$,$\{1^1, 2^1, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1, 9^1, 10^1\}$,1),
- Pat_3 = (Konstant,$9, \varnothing$,$\{1^1, 2^2, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1, 9^1\}$,1),
- Pat_4 = (Variable,$n, \{n\}$,$\{1^1, 2^2, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1, 9^1\}$,2),
- Pat_5 = (Variable,$o, \{o\}$,$\{1^1, 2^2, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1, 9^1\}$,1),
- Pat_6 = (Variable,$b, \{b\}$,$\{\mathrm{A}^1, \mathrm{E}^1, \mathrm{I}^1, \mathrm{O}^1, \mathrm{U}^1\}$,1),
- Pat_7 = (Tupel,$(b, n), \{b, n\}$,$\{(\mathrm{A}, 1)^1, (\mathrm{A}, 8)^1, (\mathrm{I}, 3)^1, (\mathrm{I}, 7)^1, (\mathrm{O}, 3)^1, (\mathrm{U}, 2)^1\}$,1),
- Pat_8 = (Wächter,b=A,$\{b\}$).

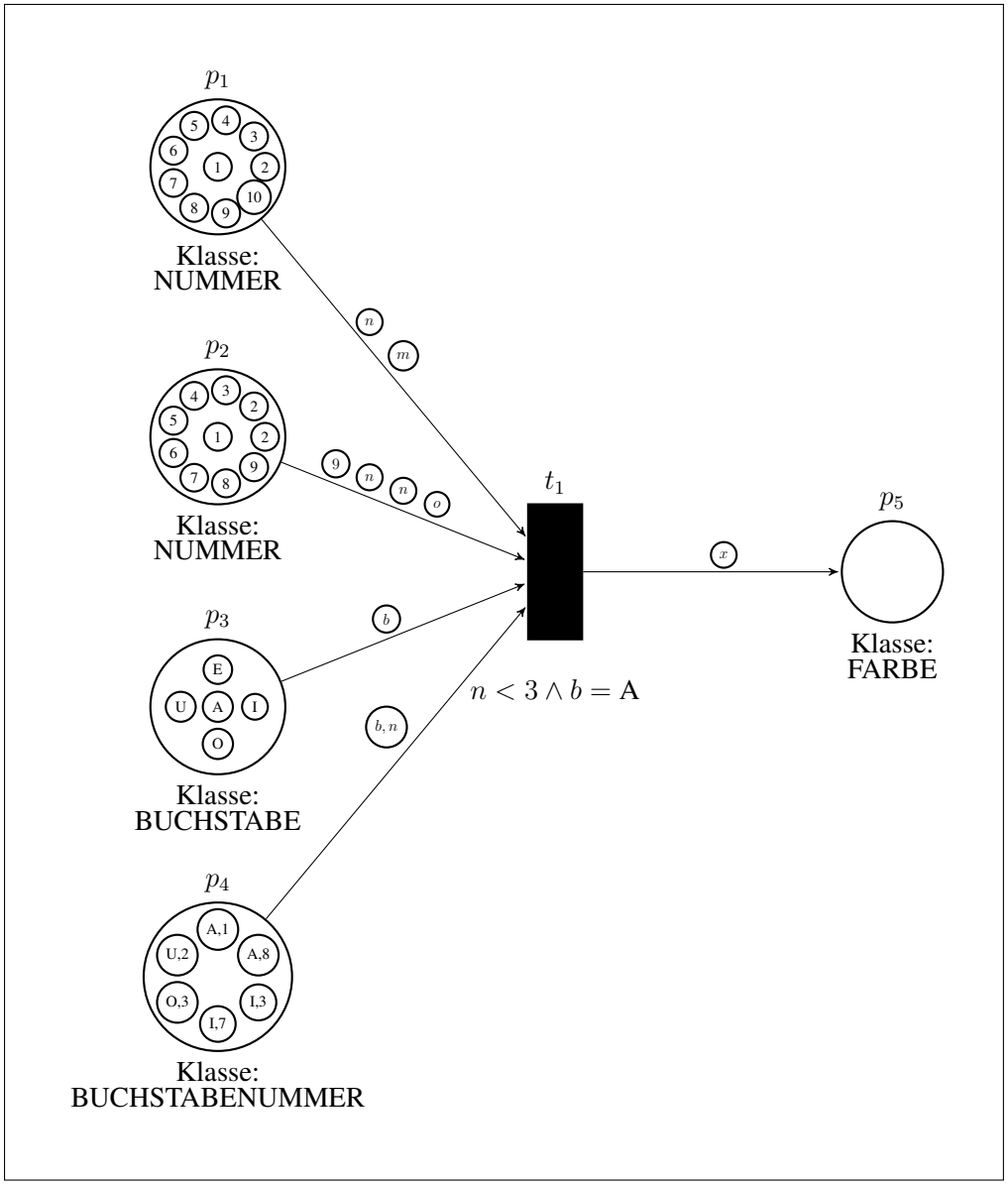

Abbildung 2.7: Abgeändertes Netz aus Abb. 2.5

Bei genauerer Betrachtung fällt auf, dass in dem obigen CPN neun andersartige Muster vorhanden sind. Jedoch befindet sich in der Muster-Menge $PS(t)$ ein Muster weniger als erwartet. Dies liegt an der Struktur des nicht berücksichtigten Musters '$n < 3$'. Analog zum Aufbau der Muster-Menge wurde verdeutlicht, dass jedem Muster ein bestimmter Typ zugewiesen wird, welchen er verkörpert. Allerdings ist es nicht möglich, dem Muster entsprechend genannter Syntax eindeutig zuzuordnen.

Aus diesem Grund wird im Zuge des Lösungsansatzes eine dritte Muster-Menge aufgestellt. Diese beinhaltet all die Muster, die aufgrund ihrer Struktur nicht zugewiesen werden können.

Jedes auf diese Weise festgestellte „Nicht-Muster“ wird in der Menge $NS(t)$ zusammengetragen. Des Weiteren ist jedes „Nicht-Muster“ in der Menge $NS(t)$ ein Tripel S = (E,X,$z(p)$), wobei

- E die formalsprachliche Darstellung ist,
- X die Menge der Variablen angibt,
- $z(p)$ die Markierung des Platzes widerspiegelt, mit dem der Pfeil verbunden ist.

Für alle „Nicht-Muster“ $S \in NS(t)$, die aus einem Wächter stammen, wird $z(p)$ immer $\varnothing$ gesetzt.

Die erste Besonderheit ist, dass bei der Bestimmung partieller Modi die Menge $NS(t)$ zum Tragen kommt. Dabei werden die ermittelten Zuweisungen der partiellen Modi mit allen entsprechenden „Nicht-Muster“ der Menge $NS(t)$ getestet.

In dem obigen Netz aus Abb. 2.7 kann bspw. für die Variable n und den Platz p_1 folgender partieller Modus entstehen:

$$\rho_1 : \{n = 5, m = ?, o = ?, b = ?\}.$$

Es kann festgestellt werden, dass dies ein unzulässiger partieller Modus ist. Mithilfe des Testvorgangs des „Nicht-Musters“ '$n < 3$' wird ermittelt, dass eine ungültige Zuweisung vorliegt, sodass ρ_1 verworfen wird. Analog zu der Definition für zulässige partielle Modi (siehe Def. 2.10) können durch diese Herangehensweise nicht zulässige partielle Modi festgestellt werden, bevor eine eventuelle zeit- und kostenintensive Berechnung des Befehls „Zusammenführen“ stattfindet.

Die weitere besondere Abwandlung des Algorithmus ist die bereits erwähnte Umstrukturierung der Muster-Menge $PS(t)$. Die Muster-Menge $PS(t)$ wird in die modifizierte und optimierte Muster-Menge $OPS(t)$ überführt, wobei die folgenden Eigenschaften vorhanden sein müssen:

1. $Var(OPS(t)) = Var(PS(t))$, wobei $Var(OPS(t))$ bzw. $Var(PS(t))$ die Menge der Variablen der Muster-Menge wiedergibt,
2. $OPS(t) \subseteq PS(t)$,
3. $\forall\ Pat_i \in GS(t) : Pat_i \in OPS(t)$.

Das genaue Vorgehen, um die Muster-Menge $OPS(t)$ zu erhalten, wird im folgenden Abschnitt für die Abb. 2.7 beschrieben.

Diese Vorberechnung ist von großer Bedeutung, da dadurch im Vorfeld eine Vielzahl von ungültigen Zuordnungen verworfen werden kann. Somit erhöht sich auch die Effizienz des Algorithmus. Drei wesentliche Schritte werden dafür durchlaufen: 1. Test auf Vielfachheit, 2. Zusammenfügen identischer Muster, 3. Muster sortieren.

1. Testen auf Vielfachheit

In diesem Teil können Token aus der Berechnung entfernt werden, die nicht ausreichend für die jeweiligen Muster vorhanden sind. Dies wird durchgeführt, indem getestet wird, ob die Häufigkeit eines jeden Tokens für den entsprechenden Multiplikator des Musters ausreichend ist. Dabei wird zwischen einem konstanten und anderen möglichen Mustern unterschieden. Als Eingabe fungiert die Menge PS(t).

Anhand der zuvor erstellten Muster wird für jedes Muster einzeln überprüft, ob die Anzahl der Token ausreichend ist.

- Pat_1 = (Variable,n, $\{n\}$,$\{1^1, 2^1, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1, 9^1, 10^1\}$,1),
- Pat_2 = (Variable,m, $\{m\}$,$\{1^1, 2^1, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1, 9^1, 10^1\}$,1),
- Pat_4 = (Variable,n, $\{n\}$,$\{2^2\}$,2),
- Pat_5 = (Variable,o, $\{o\}$,$\{1^1, 2^2, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1\}$,1),
- Pat_6 = (Variable,b, $\{b\}$, $\{\mathrm{A}^1, \mathrm{E}^1, \mathrm{I}^1, \mathrm{O}^1, \mathrm{U}^1\}$,1),
- Pat_7 = (Tupel,(b, n), $\{b, n\}$,$\{(\mathrm{A}, 1)^1, (\mathrm{A}, 8)^1, (\mathrm{I}, 3)^1, (\mathrm{I}, 7)^1, (\mathrm{O}, 3)^1, (\mathrm{U}, 2)^1\}$,1),
- Pat_8 = (Wächter,b=A,$\{b\}$).

Ergebnis: Die Anzahl der Token des Musters Pat_4 wurden stark reduziert. Zudem wurde das konstante Muster Pat_3 entfernt, da das notwendige Token vorhanden ist und nicht weiter beachtet werden muss. Auch wurde die Ausprägung '9' des Musters Pat_5 entfernt, damit es nicht belegt werden kann. Sollte im Allgemeinen für ein konstantes Muster das entsprechende Token fehlen oder für andere Muster kein zulässiges Token vorhanden sein, wird die untersuchte Transition t auf inaktiv gesetzt und der Algorithmus beendet.

2. Zusammenfügen identischer Muster

Für gewöhnlich kommt es vor, dass für eine Transition dieselben Muster mehrfach vorhanden sind. In diesem Fall kann ein Zusammenfügen identischer Muster eine Vielzahl von ungültigen Zuordnungen im Vorfeld aussortieren.

Dabei wird schemenhaft wie folgt vorgegangen:

- Bestimme zwei Muster Pat_i und Pat_j, welche dieselbe Menge an Variablen angeben,
- γ_i bzw. γ_j geben die Auflistung der Token auf dem dazugehörigen Platz des jeweiligen Musters an,
- bestimme die Schnittmenge $\gamma_k = \gamma_i \cap \gamma_j$,
- wenn $\gamma_k \neq \emptyset$, erstelle ein neues Muster Pat_k mit dem übereinstimmenden Schnitt γ_k und entferne Pat_i, Pat_j aus der Muster-Menge,
- wenn $\gamma_k = \emptyset$, setze die Transition t auf inaktiv und beende den Algorithmus.

Angewendet auf das gewählte Beispiel ergibt sich folgende Abwandlung:

- Pat_2 = (Variable,m, $\{m\}$,$\{1^1, 2^1, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1, 9^1, 10^1\}$,1),
- Pat_{14} = (Variable,n, $\{n\}$,$\{2^1\}$,1),
- Pat_5 = (Variable,o, $\{o\}$,$\{1^1, 2^2, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1\}$,1),
- Pat_6 = (Variable,b, $\{b\}$,$\{\mathrm{A}^1, \mathrm{E}^1, \mathrm{I}^1, \mathrm{O}^1, \mathrm{U}^1\}$,1),
- Pat_7 = (Tupel,(b, n), $\{b, n\}$,$\{(\mathrm{A}, 1)^1, (\mathrm{A}, 8)^1, (\mathrm{I}, 3)^1, (\mathrm{I}, 7)^1, (\mathrm{O}, 3)^1, (\mathrm{U}, 2)^1\}$,1),
- Pat_8 = (Wächter,b=A,$\{b\}$).

Ergebnis: Die Muster Pat_1 und Pat_4 wurden in dem Muster Pat_{14} zusammengefasst. Es wurde ermittelt, dass deren Schnittmenge lediglich durch das Token 2 charakterisiert ist. Trotz der unterschiedlichen Multiplikatoren und Häufigkeiten werden diese in dem zusammengefassten Muster gleichgestellt. Es muss berücksichtigt werden, dass diese Gleichstellung im Zuge von Feuerprozessen anhand ursprünglicher Ausgangslage der Pfeilgewichtungen wieder rückgängig gemacht werden muss. Jedoch ist dies für die vorliegende Vorüberprüfung nicht relevant.

3. Muster sortieren

Im letzten Schritt werden die Muster sortiert. Dies geschieht nach dem Prinzip *„Wer am wenigsten unterschiedliche Ausprägungen aufweist, wird zuerst bearbeitet“*. Damit sind die Muster gemeint, welche die geringste Häufigkeit an Ausprägungen in der Markierung des verbundenen Platzes aufweisen. Muster-Wächter haben dabei eine übergeordnete Rolle. Diese werden an die ersten Stellen gesetzt. Auf diese Weise können zu Anfang Zuweisungen ermittelt werden, wodurch nachfolgende unzulässige partielle Modi erst gar nicht entstehen können.
Dadurch ergibt sich folgende Neuordnung:

- Pat_8 = (Wächter,b=A,$\{b\}$),
- Pat_{14} = (Variable,n, $\{n\}$,$\{2^1\}$,1),
- Pat_6 = (Variable,b, $\{b\}$,$\{\mathrm{A}^1, \mathrm{E}^1, \mathrm{I}^1, \mathrm{O}^1, \mathrm{U}^1\}$,1),
- Pat_7 = (Tupel,(b, n), $\{b, n\}$,$\{(\mathrm{A}, 1)^1, (\mathrm{A}, 8)^1, (\mathrm{I}, 3)^1, (\mathrm{I}, 7)^1, (\mathrm{O}, 3)^1, (\mathrm{U}, 2)^1\}$,1),
- Pat_5 = (Variable,o, $\{o\}$,$\{1^1, 2^2, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1\}$,1),
- Pat_2 = (Variable,m, $\{m\}$,$\{1^1, 2^1, 3^1, 4^1, 5^1, 6^1, 7^1, 8^1, 9^1, 10^1\}$,1).

Schlussendlich wurde die optimierte Muster-Menge $OPS(t)$ bestimmt, wodurch eine elegante Grundlage für die effiziente Berechnung der aktiven Modus-Menge $Am(t)$ gelegt wurde. Beispielhaft kann beim Abgleich zwischen den vorgestellten Voruntersuchungen und der Abb. 2.6 festgestellt werden, dass eine Verbesserung des Rechenaufwands erwirkt werden kann. Des Weiteren kann in [LHY12] neben weiteren Strategien für die Vorüberprüfung ein Beispiel gefunden werden, in dem die vorgestellten Optimierungsmethoden Verwendung finden.

3 Effiziente Berechnung eines aktiven Modus

Grundsätzlich ist der Aufwand für das Aufstellen der aktiven Modus-Menge $Am(t)$ von vielen Faktoren abhängig. Zum einen ist die Anzahl der Variablen und Plätze im Vorbereich der Transition t entscheidend. Zum anderen ist die Struktur der einzelnen Variablen auf den Pfeilgewichtungen von Bedeutung. Insbesondere für den Fall, dass Petri-Netz-Modelle zur Simulation eingesetzt werden, ist das Bestimmen von ausschließlich einem aktiven Modus zielführender. Dadurch kann der Zustandsübergang vollzogen, die Simulation fortgesetzt und die Laufzeit verbessert werden. Aus diesem Grund wird im Folgenden eine Vorgehensweise für die Ermittlung eines aktiven Modus in einem gefärbten Petri-Netz mit einfachen Pfeilgewichtungen beschrieben.

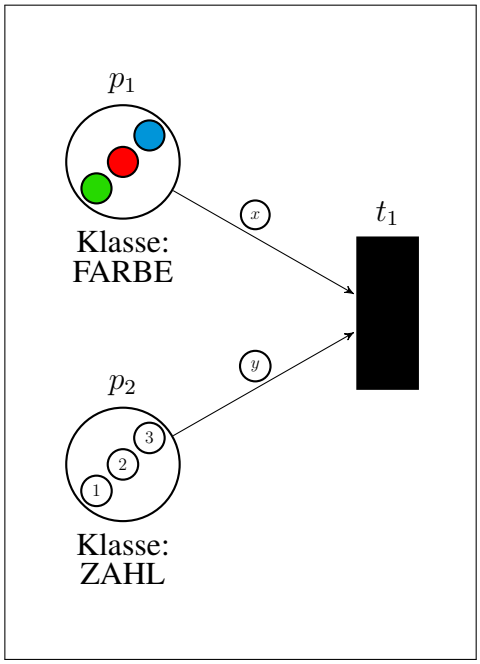

Abbildung 3.1: Kleines Netz mit einfacher Pfeilgewichtung

3.1 Grundgedanke

Um das grundsätzliche Vorgehen besser zu veranschaulichen, wird im Folgenden das Netz aus Abb. 3.1 betrachtet.

Im Allgemeinen sind eine Vielzahl von Kombinationsmöglichkeiten in einem Netz möglich, um für die auftretenden Variablen einen aktiven Modus zu bestimmen. Analog zu der Pattern-Matching Methode können die Ausprägungen in den Plätzen des Vorbereichs berücksichtigt und mit den Variablen auf den Kanten abgeglichen werden. Dieses Vorgehen lässt sich grundsätzlich, bspw. für das Netz aus Abb. 3.1, in der in Abb. 3.2 zu findenden Baumstruktur darstellen.

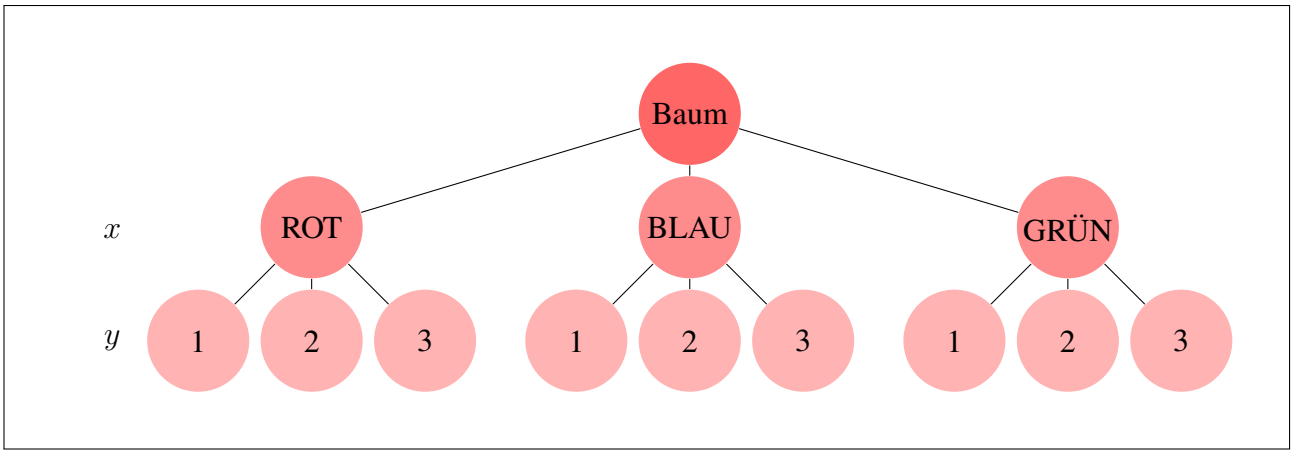

Abbildung 3.2: Vollständiger Baum aller Kombinationsmöglichkeiten des Netzes aus Abb. 3.1

Im Sinne der Pattern-Matching Methode werden in der Baumstruktur diejenigen Pfade gesucht, die für jede Variable eine zulässige Zuweisung ermöglichen. Wird exemplarisch ab dem Wurzelknoten kontinuierlich der linke Pfad gewählt, kann der Modus $\beta = \{x = \text{ROT}, y = 1\}$ angegeben werden. Für das triviale Beispiel aus Abb. 3.1 ist zu sehen, dass die aktive Modus-Menge neun Modi enthält. Diese Vorgehensweise bezeichnet man als Tiefensuche [CLRS13]. Die Tiefensuche charakterisiert sich dadurch, dass zunächst ein Pfad vollständig in die Tiefe beschritten wird, bevor die Suche in abzweigende Pfade fortgesetzt wird. Anhand des Aufbaus aus Abb. 3.2 ist zu sehen, dass für zwei vorkommende Variablen eine Tiefensuche bis in eine Tiefe von 2 erforderlich ist. Auf diese Weise lassen sich genau neun aktive Modi ermitteln.

Für das Ziel der Ermittlung ausschließlich eines aktiven Modus wird im Sinne einer Baumstruktur keine komplette Tiefensuche durchgeführt, sondern genau ein kompletter Pfad in die Tiefe des Baums gesucht. Da bereits anschauliche gefärbte Petri-Netze mit einfachen Pfeilgewichtungen viele aktive Modi aufweisen können und die dazugehörigen Baumstrukturen schnell einen sehr hohen Speicherbedarf benötigen, wird im Folgenden von der Baumstruktur abgewichen und die Realisierung mittels doppelt verketteter Listen bewerkstelligt [GD14, S.520-528], [CLRS13, S.237-241]. Dadurch wird wie in der Baumstruktur eine dynamische Speicherverwaltung ermöglicht. Des Weiteren wird die Anzahl der vorkommenden Listenknoten auf einen kompletten Pfad, also einen Modus, reduziert, sodass die Anzahl vorkommender Variablen mit den Listenknoten identisch ist.

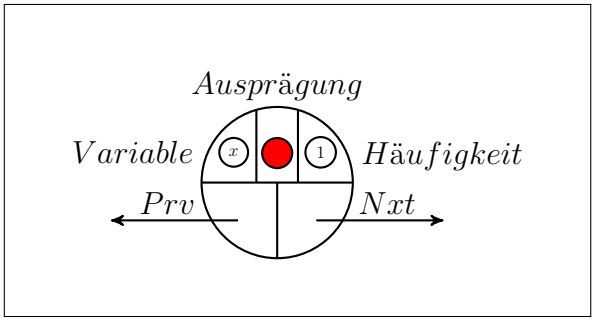

Abbildung 3.3: Beispiel eines Knotens der doppelt verketteten Listenstruktur

Ein Knoten der Listenstruktur (siehe Abb. 3.3) setzt sich aus fünf Bestandteilen zusammen. Neben dem Zeiger *Prv* auf den vorherigen und dem Zeiger *Nxt* auf den nachfolgenden Knoten, enthält ein Knoten eine Variable, die zugewiesene Ausprägung und die Ausprägungshäufigkeit. Beispielsweise ist im obigen Knoten der Variablen x die Ausprägung ROT zugewiesen worden.[1]

Neben der Struktur der einzelnen Knoten, ist deren maximale Anzahl in der Listenstruktur ein wesentlicher Bestandteil des Algorithmus. Diese entspricht der Mächtigkeit der Variablenmenge $Var(t)$, sodass für jede Variable des Netzes genau ein Knoten vorgesehen ist. Aufgrund des dynamischen Grundcharakters ist es allerdings möglich, dass im Laufe des Algorithmus die Knotenanzahl variieren kann, bevor schlussendlich ein aktiver Modus angegeben wird.

Es ist zu betonen, dass das Verfahren die zugewiesenen Klassen der vorkommenden Variablen nach und nach durcharbeitet. Diese Vorgehensweise wird gewählt, da es in den vorkommenden Netzen einerseits keine Abhängigkeit zwischen Variablen unterschiedlicher Klassen geben kann. Andererseits ist auf diese Weise die Abgrenzung der einzelnen Variablen deutlich zu erkennen.
Zu Beginn des Verfahrens wird eine erste Klasse, eine beliebige Variable v und ein Token $(a, h(a))$ aus dem korrespondierenden Platz des Vorbereichs der Transition t gewählt. Sollte dadurch ein zulässiger partieller Modus erzeugt werden, wird das Tupel bzw. die Kombination $(v, (a, h(a)))$ in der Liste gespeichert, die Ausprägung $(a, h(a))$ aus dem Zustand entfernt und mit der nächsten Variablen fortgefahren. Ansonsten wird der nicht zulässige partielle Modus in einer Tabuliste vermerkt. Es ist anzumerken, dass im Laufe des Verfahrens für jede Variable eine eigene Tabuliste erstellt wird.

Unter der Voraussetzung, dass auf diese Weise kein partieller Modus gefunden werden kann, wird ein Rückschritt durchgeführt. Es wird anhand des *Prev* Zeigers zum vorherigen Knoten zurück-

[1] Es ist anzumerken, dass, bei Vernachlässigung der Ausprägungshäufigkeit, die Zuweisung einer Variablen in einem Listenknoten ein partieller Modus in der Form $\rho(x) = ROT$ ist. Die Notwendigkeit der Ausprägungshäufigkeit wird in der Folge beschrieben.

gekehrt und die dortige Zuweisung in die Tabuliste gespeichert und eine alternative Zuweisung ermittelt. Falls der Listenanfang bereits erreicht ist und für eine Variable keine Zuweisung gefunden werden kann, ist die Transition t nicht aktivierbar. Nachdem für jede Variable ein zulässiger partieller Modus ermittelt ist, kann die Listenstruktur durchlaufen werden und ein aktiver Modus angegeben werden.

Da sich die beschriebene Bestimmung eines aktiven Modus nur auf den Vorbereich einer Transition bezieht, müssen für die vollständige Angabe eines Modus unter Umständen die Variablen im Nachbereich betrachtet werden und, z.B. mittels eines Zufallsgenerators, jeweils eine Zuweisung erhalten.

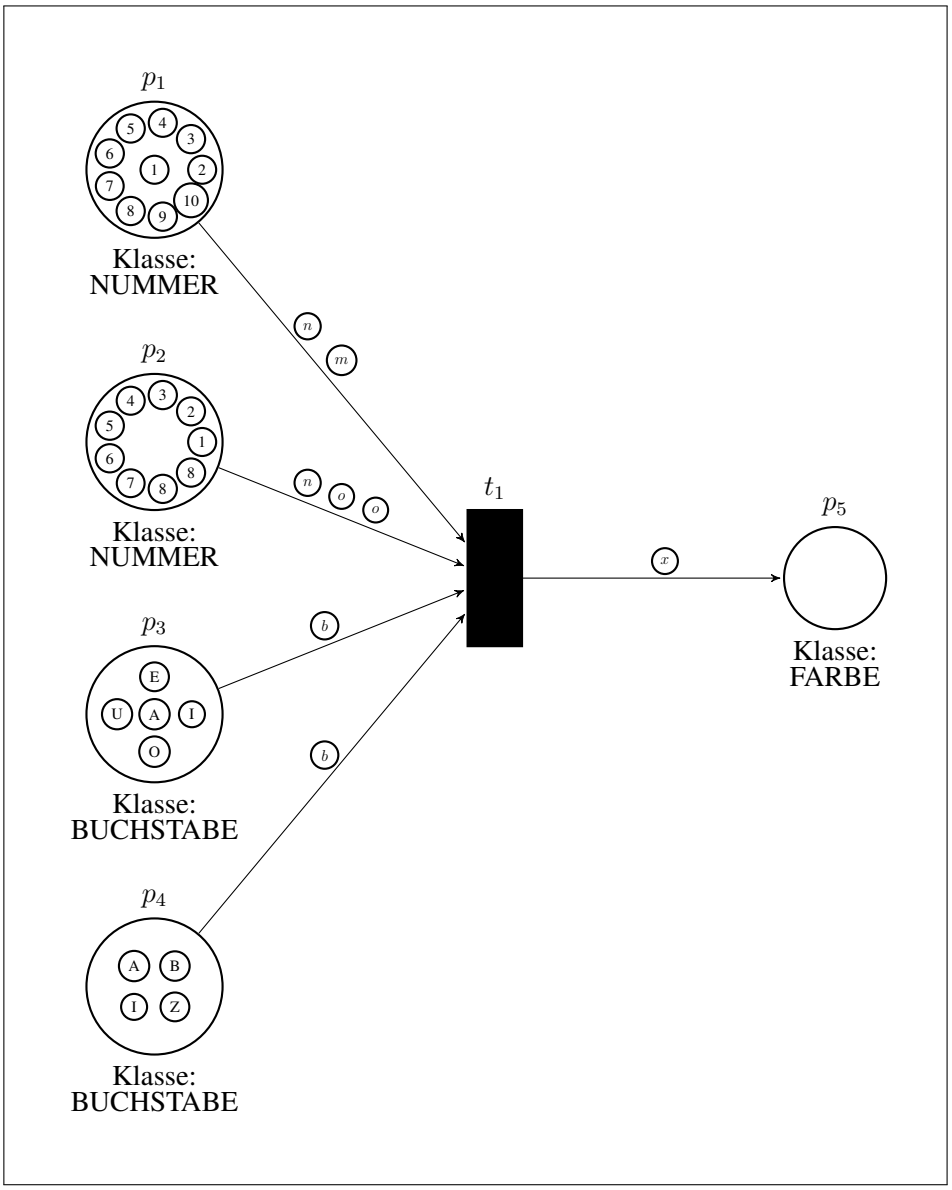

Abbildung 3.4: Beispiel des gefärbten Petri-Netzes aus Abb. 2.4

3.2 Vorstellen des Verfahrens

Anhand des Beispiels aus Abb. 3.4 wird die Vorgehensweise des Algorithmus verdeutlicht und in den einzelnen Schritten ausführlich erläutert.

Beispiel 3.1:
Man beginne bspw. mit der Klasse NUMMER, der Variablen m, der Ausprägung 2 und erzeugt den partiellen Modus

$$\rho_1 : \{b = ?, m = 2, n = ?, o = ?\}.$$

Es kann festgestellt werden, dass der partielle Modus ρ_1 zulässig ist und in der Listenstruktur abgespeichert werden kann. Es wird mit der Variablen n fortgefahren und bspw. mit der Ausprägung 8 der partielle Modus

$$\rho_2 : \{b = ?, m = ?, n = 8, o = ?\}$$

erstellt. Auch hier liegt ein zulässiger partieller Modus vor und kann in der Liste gespeichert werden. Zum jetzigen Zeitpunkt sind zwei Knoten in der Liste zu finden (siehe Abb. 3.5).

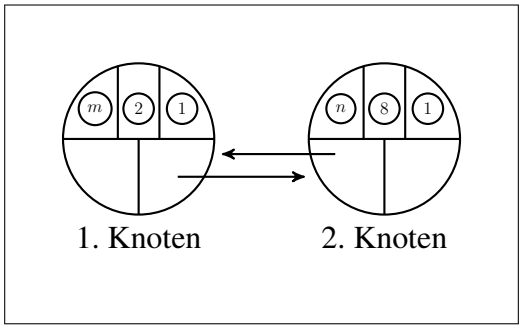

Abbildung 3.5: Listenstruktur, nachdem die Variablen m und n abgearbeitet sind

Wird mit der Variablen o fortgefahren und ein zulässiger partieller Modus gesucht, führt dies nicht zum Erfolg. Durch den partiellen Modus $\rho_2(n) = 8$ ist es nicht mehr möglich, eine zulässige Zuweisung für die Variable o zu ermitteln. Daher wird mittels des $Prev$ Zeigers ein Rückschritt durchgeführt, wodurch der partielle Modus ρ_2 bzw. der zweite Listenknoten entfernt und eine alternative Zuweisung gesucht wird. Des Weiteren wird die Ausprägung $(8, 1)$ im Zustand wieder hinzugefügt und die Tabuliste $T(n) = \{8\}$ angelegt.

Unter Berücksichtigung der Tabuliste wird erneut ein partieller Modus für die Variable n bestimmt, sodass bspw. die Ausprägung 5 gewählt wird. Mittels

$$\rho_2 : \{b = ?, m = ?, n = 5, o = ?\}$$

liegt nun ein zulässiger Modus vor und kann in der Liste abgespeichert werden. Aufgrund des partiellen Modus ρ_2 ist es nun möglich, für die Variable o eine gültige Zuweisung zu finden. Es lässt sich feststellen, dass der zulässige partielle Modus

$$\rho_3 : \{b = ?, m = ?, n = ?, o = 8\}$$

gebildet werden und in der Liste abgespeichert werden kann.

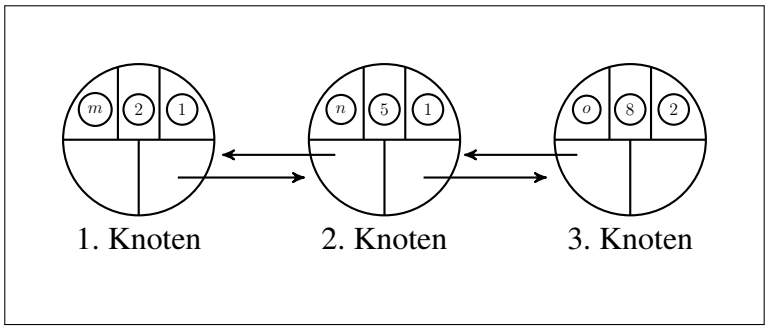

Abbildung 3.6: Listenstruktur, nachdem die Variablen m, n und o abgearbeitet sind

Mit der letzten Klasse BUCHSTABE ist noch für die Variable b ein zulässiger partieller Modus notwendig, der in die Listenstruktur (siehe Abb. 3.6) hinzugefügt werden muss, damit ein aktiver Modus angegeben werden kann. Für die Zuweisung kann festgestellt werden, dass, anhand der Ausprägungen im Platz p_3 und p_4, z.B. die Ausprägungsauswahl O oder U keine zulässigen partiellen Modi erzeugen, sodass diese Kombinationen in der Tabuliste $T(b) = \{\mathrm{O}, \mathrm{U}\}$ vermerkt werden. Es lässt sich z.B. durch die Ausprägung A für die Variable b ein zulässiger partielle Modus angeben:

$$\rho_4 : \{b = \mathrm{A}, m = ?, n = ?, o = ?\}.$$

Die Abb. 3.7 visualisiert die vollständig gefüllte Listenstruktur. Abschließend ist es möglich, die Listenstruktur und den aktiven Modus mit

$$\beta : \{b = \mathrm{A}, m = 2, n = 5, o = 8\}.$$

anzugeben.

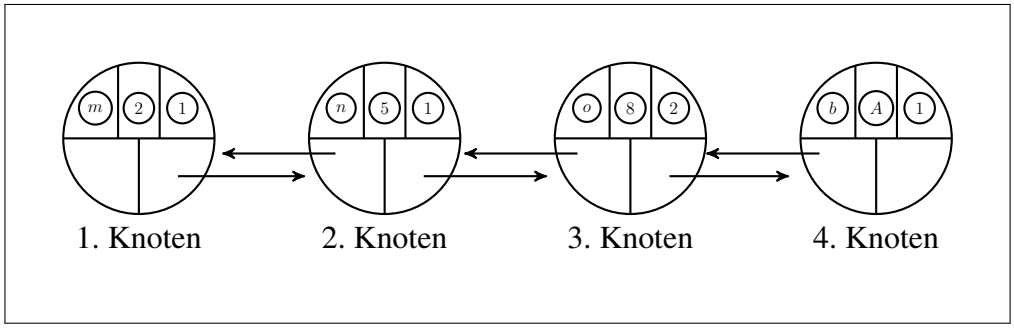

Abbildung 3.7: Listenstruktur nachdem alle Variablen abgearbeitet sind

Es ist zwar nicht die komplette aktive Modus-Menge $Am(t)$ bestimmt worden, allerdings ist im Vergleich ein wesentlich geringerer Aufwand für die Angabe eines aktiven Modus nötig als im Vergleich zur Pattern-Matching Methode. Dadurch kann im Zusammenhang der Simulation von Petri-Netz-Modellen die Laufzeit erheblich verbessert werden.

Soll hingegen die komplette aktive Modus-Menge $Am(t)$ bestimmt werden, kann dieses Verfahren dahingehend leicht modifiziert werden. Das Verfahren endet dann nicht bei dem ersten aktiven Modus, sondern läuft weiter, bis der Baum vollständig durchsucht ist bzw. aufgrund der Tabulisten keine weiteren Möglichkeiten bestehen und man somit alle aktiven Modi erhält.

Wächter können bei dieser Vorgehensweise mit berücksichtigt werden, indem sie bei der Erzeugung von Tabulisten mit beachtet werden. Des Weiteren kann man als Alternative zu den Tabulisten eine Liste verwenden, in der nur die möglichen Ausprägungen für eine Variable enthalten sind und aus dieser Liste die nicht zulässigen dann im Laufe des Verfahrens entfernen.

3.3 Algorithmus

Die Ermittlung eines aktiven Modus ist im folgenden Algorithmus zusammengefasst.[1]

Algorithmus 3.2: Berechnung eines aktiven Modus

Input:

fe $:= (fe(p_1, t), \ldots, fe(p_m, t))$ (Einfache Pfeilgewichtungen)

$V := Var(t)$ (Variablenmenge der Transition t)

Z $:= (\mathbf{z}(p_1), \ldots, \mathbf{z}(p_m))$ mit $p_i \in \bullet t$ (Zustand des Vorbereichs der Transition t)

[1] Es ist anzumerken, dass ebenfalls Vorüberprüfungen des Algorithmus B.1 durchgeführt werden. Wenn nicht weiter angegeben korrespondiert im Algorithmus die Variable v_x bzw. die Ausprägung a_y mit der Kombination an der Stelle l_s der Listenstruktur.

Output:

$L := \{(l_1, l_2, \ldots, l_{|V|}) \mid l_\lambda \in (v_x, (a_y, h(v_x))), \lambda = 1, \ldots, |V|\}$, wobei $v_x \in V$, $a_y \in \mathbb{A}$ und $h(v_x) \in \mathbb{N}$ (Lösung in Form einer doppelt verketteten Liste)

Verfahren:

Setze:

$TL(v) := \emptyset \mid \forall\, v \in V$ (Unzulässige Zuordnungen für jede Variable (Tabuliste)),

$s := 1$ (Zähler für die Liste),

$\hat{v} :=$ nicht gesetzt (Hilfsvariable),

1. Schritt: Falls $V = \emptyset$, terminiere.

Falls gilt: $\hat{v} \in V$, bestimme $v_x \in V \mid k(v_x) = k(\hat{v})$, sonst wähle $v_x \in V$ beliebig.

Ermittle:

$h(v_x) = \max(h(v) \mid (v, h(v)) \in fe(p, t))$

(Bei Mehrdeutigkeit wähle willkürlich).

$\hat{v} = v_x$.

2. Schritt: Wähle $(a_y, h(a_y)) \in z(p)$ mit

$h(a_y) = \min(h(a_k) \mid h(a_k) \geq h(v_x) \wedge a_k \notin TL(v_x))$

(Bei Mehrdeutigkeit wähle willkürlich).

Falls $\rho(v_x) := a_y$ zulässig, setze:

$l_s := (v_x, (a_y, h(v_x)))$,

$\forall\, p \in {}^\bullet t$ mit $(v_x, h(v_x)) \in fe(p, t)$ setze $\mathbf{z}(p) := \mathbf{z}(p) \setminus (a_y, h(v_x))$,

$V := V \setminus v_x$,

$s := s + 1$,

gehe zu Schritt 1,

sonst

falls $s > 1$ gilt, setze:

$s := s - 1$,

$\forall\, p \in {}^\bullet t$ mit $(v_x, h(v_x)) \in fe(p, t)$ setze $\mathbf{z}(p) := \mathbf{z}(p) \uplus (a_y, h(v_x))$,

$V := V \cup v_x$,

$TL(v_x) = TL(v_x) \cup a_y$,

$l_s := \emptyset$,

gehe zu Schritt 1,

sonst terminiere und setze die Transition t auf nicht aktivierbar.

4 Ausblick

Im ersten Kapitel dieser Abhandlung wurde eine konsequente Weiterführung und Verallgemeinerung des CPN-Formalismus aus [BKK+14] in Anlehnung an einschlägige Literatur (vgl. [JK09], [Rei10] und [Ger04]) vorgenommen, sodass auf diesen Formalismus weitere aufbauende Erweiterungen und Verallgemeinerungen, wie Petri-Netze mit Zeitkonzepten sowie kontinuierliche und hybride Petri-Netze (vgl. [Pro13]), in diesem Formalismus mit eingebettet werden können.

Des Weiteren wurden in dieser Abhandlung zwei mögliche Wege zur Bestimmung von aktiven Modi vorgestellt, einmal den Weg über die Pattern-Matching-Methode, um die gesamte aktive Modus-Menge zu ermitteln, und zum andern die Bestimmung eines aktiven Modus über eine Tiefensuche.

An sich ist es in der Regel effizienter, nur einen aktiven Modus zu berechnen. Allerdings soll für eine umfangreichere Modellbildung der zu feuernde Modus nach gewissen Auswahlkriterien bestimmt werden, wie in [BKK+14, Abschn. 6.2] beschrieben. Hier stellt sich nun die Frage, ist es effizienter, erst die gesamte aktive Modus-Menge zu ermitteln und ein Auswahlkriterium zu verwenden oder die Auswahlkriterien in die Tiefensuche mit einzubeziehen und so z. B. nach einen aktiven Modus mit einem maximalen Nutzen zu suchen. Dies wird Teil weiterführender Untersuchungen in diesem Gebiet sein.

Insgesamt soll der Petri-Netz Formalismus mit der Bezeichnung „Extended Hybrid Petri Net“ (kurz: xHPN) aus der Dissertation von Sabrina Proß (vgl. [Pro13]) auf einen gefärbten Formalismus verallgemeinert werden. Dementsprechend soll auch die von ihr entwickelte Petri-Netz-Bibliothek namens PNlib[1], welche mit der objektorientierten Modellierungssprache *Modelica* erstellt wurde, um die Möglichkeit der Modellierung mit gefärbten Petri-Netzen ergänzt werden. Hierdurch soll es ermöglicht werden, umfangreichere und komplexere Systeme effizient zu

[1] Die Bibliothek PNlib wurde auf der 9. Internationalen Modelica-Konferenz 2012 preisgekrönt (http://www.fh-bielefeld.de/presse/archiv/library-award-fuer-mathematiker-der-fh-bielefeld)

modellieren und simulieren, da durch die Faltung eines Petri-Netzes zu einem gefärbten Petri-Netz eine drastische Reduktion von Plätzen und Transitionen stattfindet. Auf diese Weise ist es möglich, kleinere und übersichtlichere Modelle zu erhalten, die mit einer geringeren Laufzeit simuliert werden können.

Symbolverzeichnis

A	Menge von Attributen, Seite 3
A_i	Attribut, Seite 3
$\mathbb{A}$	Vereinigungsmenge aller Attribute, Seite 3
a	Ausprägung eines Attributs, Seite 3
$Am(t)$	Aktive Modus-Menge einer Transition, Seite 28
$Attr$	Attributierungsfunktion, Seite 15
B	Pfeilmenge von Transitionen nach Plätzen, Seite 3
$\mathbb{B}$	Modus-Menge eines CPN, Seite 16
$\mathbb{B}_{max}$	Maximal-Modus-Menge eines CPN, Seite 16
β bzw. β_i	Modus, Seite 16
$\mathbb{E}$	Menge von Multimengen-Erzeugungsfunktionen, Seite 20
$Expr$	Multimengen-Erzeugungsfunktion, Seite 20
F	Pfeilmenge von Plätzen nach Transitionen, Seite 3
f	Gewichtungsfunktion, Seite 6
f_β	Pfeilgewicht in einem Modus, Seite 26
fe	Einfache Pfeilgewichtung, Seite 31
fe_ρ	Einfache Pfeilgewichtung in einem partiellen Modus, Seite 36

h	Häufigkeitsfunktion einer Multimenge, Seite 65
K	Menge von Klassen, Seite 3
K_j	Klasse, Seite 3
$\mathbb{K}$	Vereinigungsmenge aller Klassen, Seite 3
k	Klassifizierungsfunktion, Seite 3
$\mathbb{M}(A)$	Menge aller Multimengen über eine Menge, Seite 70
$m(t)$	Modus-Menge einer Transition, Seite 16
N	Netz bzw. Petri-Netz, Seite 3
P	Menge von Plätzen, Seite 3
p bzw. p_i	Platz, Seite 3
$\bullet p$	Vorbereich eines Platzes, Seite 7
$p\bullet$	Nachbereich eines Platzes, Seite 7
Pat	Muster, Seite 35
ρ bzw. ρ_i	partieller Modus, Seite 35
$supp(M)$	Stützmenge einer Multimenge, Seite 66
T	Menge von Transitionen, Seite 3
t bzw. t_j	Transition, Seite 3
$\bullet t$	Vorbereich einer Transition, Seite 7
$t\bullet$	Nachbereich einer Transition, Seite 7
$\mathbb{V}$	Menge von Variablen, Seite 15
v bzw. v_i	Variable, Seite 15
$Var(t)$	Variablenmenge einer Transition, Seite 15
$\mathbf{z}$	Zustand eines Petri-Netz, Seite 8
z_i bzw. $\mathbf{z}(p_i)$	Zustand eines Platzes, Seite 8

Literaturverzeichnis

[BKK+14] Bachmann, B., Kleine-Döpke, T., Kruse, H.-J., Ochel, L. und Pross, S. : Petri-Netz-Formalismen und Lösungsansätze für allgemeine Konfliktsituationen bei Feuerprozessen in Petri-Netz-Modellen. Forschungsreihe des Fachbereichs Ingenieurwissenschaften und Mathematik der Fachhochschule Bielefeld, Band 2, 2014.

[CLRS13] Cormen, T. H., Leiserson, C. E., Rivest, R. und Stein, C.: Algorithmen - Eine Einführung. Oldenburg, München, 4., überarb. und erw. Auflage, 2013.

[GD14] Goll, J. und Dausmann, M.: C als erste Programmiersprache: Mit den Konzepten von C11. Esslingen, 8. Auflage, 2014.

[Ger04] Gerber, S.: Petri-Netze 2. Institut für Informatik, Universität Leipzig, Sommersemester 2004.

[JK09] Jensen, K. und Kristensen, L. M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Berlin-Heidelberg, 2009.

[KC04] Kristensen, L. M. und Christensen, S.: Implementing Coloured Petri Nets Using a Functional Programming Language. In: High Order and Symbolic Computation, 216-224, 2004.

[KD13] Kleine-Döpke, T.: Konfliktlösungen für den nebenläufigen Feuerprozess bei gefärbten Petri-Netzen und ihre Anwendungen. Bachelorarbeit im Studiengang Angewandte Mathematik der Fachhochschule Bielefeld, 2013.

[LH10] Liu, f. und Heiner, M.: Computation of Enabled Transition Instances for Colored Petri Nets. Proceedings of the 17th German Workshop on Algorithmus and Tools for Petri Nets, 2010.

[LHY12] Liu, f., Heiner, M. und Yang, M.: An Efficient Method for unfolding Colored Petri Nets. In: Winter Simulation Conference, 2012.

[Mon01] MONRO, G.P.: The Concept of Multiset. In: Proceedings of the Workshop on Multiset Processing, 347-358, London 2001.

[Pro13] PROSS, S.: Hybrid Modeling and Optimization of Biological Processes. Forschungsreihe des Fachbereichs Ingenieurwissenschaften und Mathematik der Fachhochschule Bielefeld, Band 1, 2013.

[Rei10] REISIG, W.: Petrinetze. Wiesbaden, 2010.

[Sil15] SILBERBERG, J.: Effiziente Berechnung der aktiven Modus-Menge in gefärbten Petri-Netzen mit einfachen Variablen. Bachelorarbeit im Studiengang Angewandte Mathematik der Fachhochschule Bielefeld, 2015.

[Syr87] SYROPOULOS, A.: Mathematics of Multisets. Zeitschrift f. math. Logik und Grundlagen d. Math., 33:171–178, 1987.

Anhang: Multimengen

Eine Multimenge ist eine modifizierte Form des Mengenbegriffs aus der klassischen Mengenlehre nach Cantor, indem es bei einer Multimenge erlaubt ist, dass ein und dasselbe Element mehrmals enthalten ist. Dementsprechend müssen auch die Mengenoperationen auf Multimengen modifiziert werden (hierzu siehe [Mon01], [Syr87]).

Definition A.1:
Eine ***Multimenge*** M über einer klassischen Menge A wird als eine klassische Menge geordneter Paare $(a, h(a))$ aufgefasst, wobei gilt:

$$a \in A$$
$$h : A \to \mathbb{N}_0$$

Dabei gibt $h(a)$ die Häufigkeit des Elementes a in M an. Die klassische Menge A wird auch als die ***Grundmenge*** der Multimenge M und h als die ***Häufigkeitsfunktion*** von M bezeichnet.

Beispiel:

(a) Ein Beispiel einer Multimenge über der Menge $A = \{\text{ROT,BLAU,GRÜN}\}$ ist:

$$M = \{(\text{ROT}, 3), (\text{BLAU}, 2), (\text{GRÜN}, 1)\},$$

was auch interpretiert werden kann als

$$M = \{\text{ROT}, \text{ROT}, \text{ROT}, \text{BLAU}, \text{BLAU}, \text{GRÜN}\}$$

oder auch ungeordnet als

$$M = \{\text{ROT}, \text{BLAU}, \text{ROT}, \text{GRÜN}, \text{ROT}, \text{BLAU}\}.$$

(b) Ein weiteres Beispiel einer Multimenge über der Menge A ist:

$$M = \{(\text{ROT}, 2), (\text{BLAU}, 0), (\text{GRÜN}, 1)\},$$

was auch interpretiert werden kann als

$$M = \{\text{ROT}, \text{ROT}, \text{GRÜN}\}$$

oder auch ungeordnet als

$$M = \{\text{ROT}, \text{GRÜN}, \text{ROT}\} = \{\text{GRÜN}, \text{ROT}, \text{ROT}\}.$$

Definition A.2:
Es seien A eine klassische Menge und $M = \{(a, h(a)) \mid a \in A, h(a) \in \mathbb{N}_0\}$ eine Multimenge über A. Die klassische Teilmenge $supp(M) = \{a \in A \mid h(a) > 0\}$ von A wird ***stützende Menge***, ***Stützmenge*** oder ***reduzierte Grundmenge*** der Multimenge M genannt.

Im Extremfall kann die stützende Menge einer Multimenge die Grundmenge selber sein $(supp(M) = A)$, wenn es kein Element $a \in A$ mit $h(a) = 0$ gibt. Umgekehrt kann die stützende Menge aber im anderen Extremfall auch die leere Menge $\emptyset$ sein, wenn nämlich jedem Element $a \in A$ die Häufigkeit $h(a) = 0$ zugewiesen wird.

Beispiel:
Gegeben über der Menge $A = \{\text{ROT}, \text{BLAU}, \text{GRÜN}\}$ sei die Multimenge:

$$M = \{(\text{ROT}, 3), (\text{BLAU}, 2), (\text{GRÜN}, 0)\}.$$

Die Stützmenge von M ist

$$supp(M) = \{\text{ROT}, \text{BLAU}\}.$$

Definition A.3:
Zu einer Grundmenge A sei die Multimenge $M = \{(a, h(a)) \mid a \in A, h(a) \in \mathbb{N}_0\}$ über A gegeben. Dann ist die ***Mächtigkeit der Multimenge*** M definiert als:

$$|M| := \sum_{a \in A} h(a)$$

Beispiel:
Gegeben über der Menge $A = \{\text{ROT}, \text{BLAU}, \text{GRÜN}\}$ sei die Multimenge:

$$M = \{(\text{ROT}, 3), (\text{BLAU}, 2), (\text{GRÜN}, 0)\}.$$

Die Mächtigkeit der Multimenge von M ist

$$|M| = h(\text{ROT}) + h(\text{BLAU}) + h(\text{GRÜN}) = 3 + 2 + 0 = 5.$$

Definition A.4:
Zu einer Grundmenge A seien die Multimengen $M_1 = \{(a, h_1(a)) \mid a \in A, h_1(a) \in \mathbb{N}_0\}$ und $M_2 = \{(a, h_2(a)) \mid a \in A, h_2(a) \in \mathbb{N}_0\}$ über A gegeben.

- M_1 heißt ***Teilmenge*** von M_2 $(M_1 \subseteq M_2)$, wenn $\forall\, a \in A : h_1(a) \leq h_2(a)$.
- M_1 heißt ***echte Teilmenge*** von M_2 $(M_1 \subset M_2)$, wenn $M_1 \subseteq M_2$ und $\exists\, a \in A : h_1(a) < h_2(a)$.
- M_1 und M_2 heißen ***gleich*** $(M_1 = M_2)$, wenn $\forall\, a \in A : h_1(a) = h_2(a)$.

Beispiel:
Gegeben über der Menge $A = \{\text{ROT}, \text{BLAU}, \text{GRÜN}\}$ seien die zwei Multimengen:

$$M_1 = \{(\text{ROT}, 3), (\text{BLAU}, 2), (\text{GRÜN}, 1)\},$$
$$M_2 = \{(\text{ROT}, 2), (\text{BLAU}, 1), (\text{GRÜN}, 1)\}.$$

M_2 ist eine (echte) Teilmenge von M_1 $(M_2 \subseteq M_1)$:

$$h_2(\text{ROT}) = 2 \leq 3 = h_1(\text{ROT}),$$
$$h_2(\text{BLAU}) = 1 \leq 2 = h_1(\text{BLAU}),$$
$$h_2(\text{GRÜN}) = 1 \leq 1 = h_1(\text{GRÜN}).$$

Definition A.5:
Zu einer Grundmenge A seien die Multimengen $M_1 = \{(a, h_1(a)) \mid a \in A, h_1(a) \in \mathbb{N}_0\}$ und $M_2 = \{(a, h_2(a)) \mid a \in A, h_2(a) \in \mathbb{N}_0\}$ über A gegeben. Dann heißen die Multimengen

- $M_1 \uplus M_2 := \{(a, s(a)) \mid a \in A, s(a) = h_1(a) + h_2(a)\}$ die ***Vereinigungsmenge*** (oder ***Summe***) von M_1 und M_2,
- $M_1 \setminus M_2 := \{(a, d(a)) \mid a \in A, d(a) = max(h_1(a) - h_2(a); 0)\}$ die ***Differenzmenge*** (oder ***Differenz***) von M_1 und M_2.
- $M_1 \cap M_2 := \{(a, s(a)) \mid a \in A, s(a) = min(h_1(a), h_2(a))\}$ die ***Schnittmenge*** von M_1 und M_2,

Bemerkung:
Im Fall $M_2 \subseteq M_1$ ergibt sich die Differenz $M_1 \setminus M_2$ vereinfacht als

$$M_1 \setminus M_2 = \{(a, d(a)) \mid a \in A, d(a) = h_1(a) - h_2(a)\}.$$

Beispiel:
Gegeben über der Menge $A = \{\text{ROT}, \text{BLAU}, \text{GRÜN}\}$ seien die zwei Multimengen:

$$M_1 = \{(\text{ROT}, 2), (\text{BLAU}, 2), (\text{GRÜN}, 1)\},$$
$$M_2 = \{(\text{ROT}, 2), (\text{BLAU}, 1), (\text{GRÜN}, 2)\}.$$

(a) Die Vereinigungsmenge von M_1 und M_2 ist

$$\begin{aligned} M_1 \uplus M_2 &= \{(\text{ROT}, 2), (\text{BLAU}, 2), (\text{GRÜN}, 1)\} \uplus \{(\text{ROT}, 2), (\text{BLAU}, 1), (\text{GRÜN}, 2)\} \\ &= \{(\text{ROT}, 2+2), (\text{BLAU}, 2+1), (\text{GRÜN}, 1+2)\} \\ &= \{(\text{ROT}, 4), (\text{BLAU}, 3), (\text{GRÜN}, 3)\}. \end{aligned}$$

(b) Die Differenzmenge von M_1 und M_2 ist

$$\begin{aligned} M_1 \setminus M_2 &= \{(\text{ROT}, 2), (\text{BLAU}, 2), (\text{GRÜN}, 1)\} \setminus \{(\text{ROT}, 2), (\text{BLAU}, 1), (\text{GRÜN}, 2)\} \\ &= \{(\text{ROT}, \max(2-2; 0)), (\text{BLAU}, \max(2-1; 0)), (\text{GRÜN}, \max(1-2; 0))\} \\ &= \{(\text{ROT}, 0), (\text{BLAU}, 1), (\text{GRÜN}, 0)\}. \end{aligned}$$

(c) Die Schnittmenge von M_1 und M_2 ist

$$\begin{aligned} M_1 \cap M_2 &= \{(\text{ROT}, 2), (\text{BLAU}, 2), (\text{GRÜN}, 1)\} \cup \{(\text{ROT}, 2), (\text{BLAU}, 1), (\text{GRÜN}, 2)\} \\ &= \{(\text{ROT}, \min(2; 2)), (\text{BLAU}, \min(2; 1)), (\text{GRÜN}, \min(1; 2))\} \\ &= \{(\text{ROT}, 2), (\text{BLAU}, 1), (\text{GRÜN}, 1)\}. \end{aligned}$$

Definition A.6:
Zu zwei Grundmengen A und B seien die Multimengen $M_1 = \{(a, h_1(a)) \mid a \in A, h_1(a) \in \mathbb{N}_0\}$ über A und $M_2 = \{(a, h_2(a)) \mid a \in B, h_2(a) \in \mathbb{N}_0\}$ über B gegeben. Dann heißt die Multimenge

$$M_1 \uplus M_2 := \left\{ (a, s(a)) \mid a \in (A \cup B), s(a) = \left\{ \begin{array}{ll} h_1(a) + h_2(a) & \text{für } a \in A \wedge a \in B \\ h_1(a) & \text{für } a \in A \wedge a \notin B \\ h_2(a) & \text{für } a \notin A \wedge a \in B \end{array} \right\} \right\}$$

die ***Vereinigungsmenge*** (oder ***Summe***) von M_1 und M_2.

Beispiel:
Gegeben über der Menge $A = \{\text{ROT}, \text{BLAU}, \text{GRÜN}\}$ sei die Multimenge

$$M_1 = \{(\text{ROT}, 2), (\text{BLAU}, 2), (\text{GRÜN}, 1)\}$$

und über der Menge $B = \{\text{ROT}, \text{PINK}, \text{SCHWARZ}\}$ sei die Multimenge

$$M_2 = \{(\text{ROT}, 2), (\text{PINK}, 1), (\text{SCHWARZ}, 2)\}.$$

Die Vereinigungsmenge von M_1 und M_2 ist

$$\begin{aligned} M_1 \uplus M_2 &= \{(\text{ROT}, 2), (\text{BLAU}, 2), (\text{GRÜN}, 1)\} \uplus \{(\text{ROT}, 2), (\text{PINK}, 1), (\text{SCHWARZ}, 2)\} \\ &= \{(\text{ROT}, 2 + 2), (\text{BLAU}, 2), (\text{GRÜN}, 1), (\text{PINK}, 1), (\text{SCHWARZ}, 2)\} \\ &= \{(\text{ROT}, 4), (\text{BLAU}, 2), (\text{GRÜN}, 1), (\text{PINK}, 1), (\text{SCHWARZ}, 2)\}. \end{aligned}$$

Definition A.7:
Es seien N Multimengen $M_i = \{(a, h_i(a)) \mid a \in A, h_i(a) \in \mathbb{N}_0\}$ von $i = 1, \ldots, N$ über A gegeben. Dann heißt die Multimenge

$$\biguplus_{i=1,\ldots,N} M_i := M_1 \uplus M_2 \uplus M_3 \uplus \ldots \uplus M_{N-1} \uplus M_N$$

die ***N-äre Vereinigungsmenge*** (oder ***N-äre Summe***) aller M_i.

Beispiel:
Gegeben über der Menge $A = \{\text{ROT}, \text{BLAU}, \text{GRÜN}\}$ seien die vier Multimengen:

$$M_1 = \{(\text{ROT}, 3), (\text{BLAU}, 2), (\text{GRÜN}, 1)\},$$
$$M_2 = \{(\text{ROT}, 2), (\text{BLAU}, 1), (\text{GRÜN}, 1)\},$$
$$M_3 = \{(\text{ROT}, 1), (\text{BLAU}, 1), (\text{GRÜN}, 0)\},$$
$$M_4 = \{(\text{ROT}, 3), (\text{BLAU}, 2), (\text{GRÜN}, 1)\}.$$

Die N-äre Vereinigungsmenge aller M_i ist

$$\biguplus_{i=1,\ldots,4} M_i = M_1 \uplus M_2 \uplus M_3 \uplus M_4 = \{(\text{ROT}, 9), (\text{BLAU}, 6), (\text{GRÜN}, 3)\}.$$

Definition A.8:
Es sei A eine klassische Menge. Dann heißt die Menge

$$\mathbb{M}(A) = \{M \mid M = \{(a, h(a)) \mid a \in A, h(a) \in \mathbb{N}_0\}\}$$

die ***Menge aller Multimengen*** über der Menge A.

Abschließend werde noch eine alternative Schreibweise für Multimengen eingeführt, die insbesondere bei ausführlichen Mengenausdrücken für mehr Übersichtlichkeit sorgen kann. Diese Darstellungsform lehnt sich an die Schreibweise für algebraische Potenzausdrücke und diesbezügliche Potenzregeln an, etwa: $a^3 \cdot b^2 \cdot c \equiv a^3 \cdot b^2 \cdot c^1 \cdot d^0$.

Über einer endlichen Grundmenge $A = \{a_i \mid i = 1, \ldots, m\}$ sei $M = \{(a_i, h(a_i)) \mid i = 1, \ldots, m\}$ eine Multimenge gemäß Def. A.1. Diese Multimenge lässt sich folgendermaßen auch in **Potenzdarstellung** schreiben:

$$M = \prod_{i=1}^{m} a_i^{h_i} = a_1^{h_1} \cdot a_2^{h_2} \cdot \ldots \cdot a_m^{h_m} \text{ mit } h_i := h(a_i) \text{ für } i = 1, \ldots, m.$$

Hieraus ergibt sich auf kanonische Weise eine vereinfachte Schreibweise für die in Def. A.5 eingeführte Summe für zwei Multimengen $M_1 = \{(a_i, h(a_i)) \mid i = 1, \ldots, m\}$ und $M_2 = \{(a_i, g(a_i)) \mid i = 1, \ldots, m\}$ über der Grundmenge A, wobei h und g die Häufigkeitsfunktionen von M_1 bzw. M_2 seien sowie $h_i := h(a_i)$ und $g_i := g(a_i)$ für $i = 1, \ldots, m$ gelte:

$$M_1 \uplus M_2 := \prod_{i=1}^{m} a_i^{h_i} \cdot \prod_{i=1}^{m} a_i^{g_i} = \prod_{i=1}^{m} a_i^{h_i + g_i}.$$

Diese Schreibweise lässt sich ohne Weiteres auch auf die N-äre Summe übertragen.

Im Fall $M_2 \subset M_1$ lässt sich die Differenz $M_1 \setminus M_2$ ebenfalls vereinfacht schreiben als:

$$M_1 \setminus M_2 = \frac{\prod_{i=1}^{m} a_i^{h_i}}{\prod_{i=1}^{m} a_i^{g_i}} = \prod_{i=1}^{m} a_i^{h_i - g_i}.$$

Anhang: Algorithmen und Testfälle zum Pattern-Matching

B.1 Algorithmen

Die folgenden Algorithmen sind aus [Sil15, S. 29 f.] entnommen. In diesen wird zunächst eine Vorüberprüfung durchgeführt. Es wird für alle vorkommenden Kanten überprüft, ob aufgrund der aktuellen Markierung mindestens eine Pfeilgewichtung vorliegt, sodass keine Lösung existieren kann. Dafür wird zum einen die Häufigkeit, zum anderen das Maximum untersucht. Anschaulich gesagt, wird überprüft, ob einerseits die Häufigkeit der Ausprägungen eines Platzes ausreichend ist, damit alle Variablen eine notwendige Zuordnung erhalten können. Andererseits wird auf Grundlage des Maximums der Variablen einer jeden Pfeilgewichtung auf dem dafür korrespondierenden Platz abgeglichen, ob eine Zuweisung realisierbar ist.

Neben den beiden erwähnten Untersuchungen gibt es noch weitere Möglichkeiten, um durch geeignete Tests eine Vorüberprüfung zu durchlaufen. Zum Beispiel kann aufgrund der Maxima-Untersuchung eine unausweichliche Setzung einer Variable notwendig werden, da ggf. die derzeitige Markierung nur eine Möglichkeit bereitstellt, wohingegen aufgrund dieser Setzung solch eine Zuweisung in einem weiteren Platz nicht durchführbar ist. Deshalb liegt in diesem fiktiven Szenario keine Lösung vor. Allerdings verwendet der Algorithmus ausschließlich die erläuterten einfachen Zusammenhänge für die Vorüberprüfung. Die Testfälle 1 und 2 im folgenden Abschnitt verdeutlichen die oben genannten Vorgehensweisen.

Algorithmus B.1: Algorithmus zur Berechnung der aktiven Modus-Menge

Input:

fe $:= (fe(p_1, t), \ldots, fe(p_m, t))$ (Einfache Pfeilgewichtungen)

Z $:= (\mathbf{z}(p_1), \ldots, \mathbf{z}(p_m))$ mit $p_i \in \bullet t$ (Zustand des Vorbereichs der Transition t)

Output:

$Am(t)$

Vorüberprüfung:

Für $i = 1, \ldots, m$ ermittle:

> $|fe(p_i, t)|, |\mathbf{z}(p_i)|$. (Mächtigkeit)
>
> $h(v^*) = \max(h(v)|(v, h(v)) \in fe(p_i, t))$. (Maximum der Kante)
>
> $h(a^*) = \max(h(a)|(a, h(a)) \in \mathbf{z}(p_i))$. (Maximum des Zustandes)
>
> Falls gilt: $|\mathbf{z}(p_i)| < |fe(p_i, t)| \vee h(a^*) < h(v^*)$,
>
> Terminiere.

Hauptverfahren:

Setze: $C := \emptyset$ (Modus-Menge).

Für $j = 1, \ldots, m$ führe aus:

> Setze: $\hat{C} := \emptyset$. (partielle Modus-Menge),
>
> Für alle $(v, h(v)) \in fe(p_j, t)$, falls gilt $h(v) > 0$, führe aus:
>
> > Für alle $(a, h(a)) \in \mathbf{z}(p_j)$, falls gilt $h(a) \geq h(v)$, führe aus:
> >
> > > Setze: $\rho(v) := a$. (partieller Modus),
> > >
> > > setze: $\hat{C} := \hat{C} \cup \{\rho\}$.
> >
> > $C := Merge(C, \hat{C})$. (Externe Merge-Funktion, siehe Algo. B.2)

Nachüberprüfung:

Setze: $Am(t) := \emptyset$ (Aktive Modus-Menge).

Für alle Modi $\beta \in C$ führe aus:

> Setze: $\mathbf{fe}_\beta$.
>
> Falls Active($\mathbf{fe}_\beta, z$) erfüllt, (Externe Active-Funktion, siehe Algo. B.3),
>
> > setze: $Am(t) := Am(t) \cup \{\beta\}$.

Algorithmus B.2: Merge-Funktion

Input:

$\rho = (\rho_1, \ldots, \rho_x)$ (Partielle Modus-Menge 1)

$\tilde{\rho} = (\tilde{\rho}_1, \ldots, \tilde{\rho}_y)$ (Partielle Modus-Menge 2)

Output:

$\hat{\rho} = (\hat{\rho}_1, \ldots, \hat{\rho}_z)$ (Zusammengeführte partielle Modus-Menge)

Verfahren:

Setze: $a := 1$.

Für $i = 1, \ldots, x$ führe aus:

Für $j = 1, \ldots, y$ führe aus:

Falls $(\rho_i, \tilde{\rho}_j)$ verträglich,

setze für alle $v \in \mathbb{V}$,

$$\hat{\rho}_a(v) = \begin{cases} \rho_i(v) & \textit{falls } \rho_i(v) \neq ? \\ \tilde{\rho}_j(v) & \textit{sonst} \end{cases}$$

Setze $a := a + 1$.

Algorithmus B.3: Active-Funktion

Input:

$\mathbf{fe}_\beta := (fe_\beta(p_1, t), \ldots, fe_\beta(p_m, t))$ (Einfache Pfeilgewichtungen im Modus β)

$\mathbf{Z} := (\mathbf{z}(p_1), \ldots, \mathbf{z}(p_m))$ mit $p_i \in {\bullet}t$ (Zustand des Vorbereichs der Transition t)

Output:

Bool $result$

Verfahren:

Setze: $result := true$.

Falls für ein $p \in {\bullet}t$ gilt: $fe_\beta(p, t) \not\subseteq \mathbf{z}(p)$,

setze: $result := false$.

B.2 Testfälle

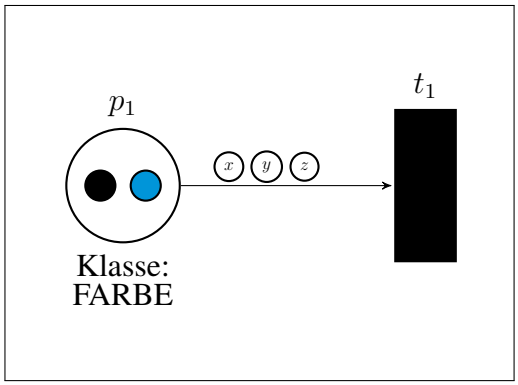

Abbildung B.1: Ein kleines CPN mit Variablen (Testfall 1)

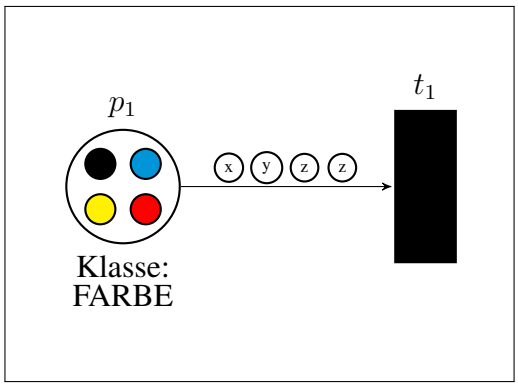

Abbildung B.2: Ähnliches CPN mit Variablen aus Abb.B.1 (Testfall 2)

Auffällig ist, dass in Abb. B.1 die Häufigkeit der Token im Platz p_1 für die vorliegenden Variablen nicht ausreicht. In Abb. B.2 kann ermittelt werden, dass aufgrund der Häufigkeiten eine Lösung möglich erscheint. Im Gegensatz dazu kann anhand des Maximums der Pfeilgewichtung festgestellt werden, dass die Variable z von keinem Attribut des Platzes p_1 angenommen werden kann. Aus diesem Grund ist auch in diesem Testfall keine Lösung möglich.

Mithilfe des nächsten Beispiels wird die Notwendigkeit der Nachüberprüfung näher gebracht (Abb. B.3).

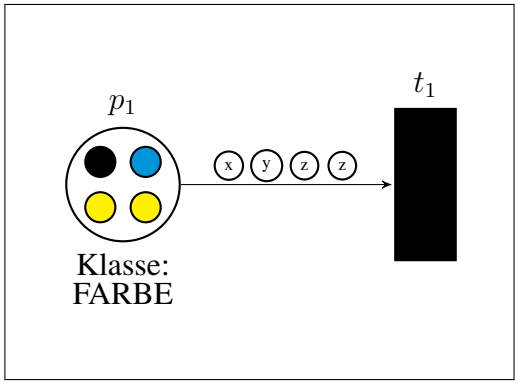

Abbildung B.3: Abgeändertes Netz aus Abb. B.2 (Testfall 3)

Es kann festgestellt werden, dass die Variable z ausschließlich das Attribut 'GELB' als Zuweisung erhalten kann. Infolgedessen können die Variablen x und y nur noch durch die beiden verbleibenden Ausprägungen 'BLAU' und 'SCHWARZ' belegt werden. Dadurch wird ersichtlich, dass die aktive Modus-Menge $Am(t_1)$ folgende zwei Modi beinhalten muss:

$$\beta_1 : \{x = \text{BLAU}, y = \text{SCHWARZ}, z = \text{GELB}\},$$
$$\beta_2 : \{x = \text{SCHWARZ}, y = \text{BLAU}, z = \text{GELB}\}.$$

Nachdem das Hauptverfahren des Algorithmus, also die im vorherigen Kapitel eingeführte Pattern-Matching-Methode, durchgeführt wurde, werden mehr Modi als erwartet angegeben. Sie werden mit β^* gekennzeichnet.

$$\beta_1^* : \{x = \text{GELB}, y = \text{GELB}, z = \text{GELB}\},$$
$$\beta_2^* : \{x = \text{GELB}, y = \text{BLAU}, z = \text{GELB}\},$$
$$\beta_3^* : \{x = \text{GELB}, y = \text{SCHWARZ}, z = \text{GELB}\},$$
$$\beta_4^* : \{x = \text{BLAU}, y = \text{GELB}, z = \text{GELB}\},$$
$$\beta_5^* : \{x = \text{BLAU}, y = \text{BLAU}, z = \text{GELB}\},$$
$$\beta_6^* : \{x = \text{BLAU}, y = \text{SCHWARZ}, z = \text{GELB}\},$$
$$\beta_7^* : \{x = \text{SCHWARZ}, y = \text{GELB}, z = \text{GELB}\},$$
$$\beta_8^* : \{x = \text{SCHWARZ}, y = \text{BLAU}, z = \text{GELB}\},$$
$$\beta_9^* : \{x = \text{SCHWARZ}, y = \text{SCHWARZ}, z = \text{GELB}\}.$$

Es wird im weiteren Verlauf erläutert, aus welchem Grund eine verfälschte Ausgabe vorliegt. Das Hauptverfahren des implementierten Lösungsansatzes erstellt mittels der lexikografischen Ordnung alle möglichen partiellen Modi für die vorliegenden Variablen. Es wird nicht beachtet, ob die Häufigkeit einer Ausprägung bereits überschritten wurde bzw. ob bereits ein unzulässiger partieller Modus vorliegt (vgl. Def 2.10). Daher kann es vorkommen, dass ein Modus gespeichert wird, welcher fälschlicherweise mehr Token einer Ausprägung voraussetzt. Aus diesem Grund müssen bestimmte Modi aussortiert werden.

Aufgrund dieses Sachverhalts wird eine Nachüberprüfung der ermittelten Modus-Menge durchgeführt. Es werden alle ungültigen Modi ausgefiltert, sodass im Ergebnis die vollständige aktive Modus-Menge $Am(t_1)$ angegeben wird. Dafür werden alle Pfeilgewichtungen analog zu Def. 2.9 stückweise mittels der vorläufig ermittelten Modi in konstante Ausdrücke überführt. Mittels des Algorithmus B.3 wird nach und nach jeder Modus überprüft, ob eine aktive Transition t herbeigeführt werden kann. Das im Anschluss erhaltene Resultat entspricht den im Vorfeld erwarteten Modi β_1 und β_2.

Abschließend soll im letzten Testfall ein größeres Petri-Netz gezeigt werden.

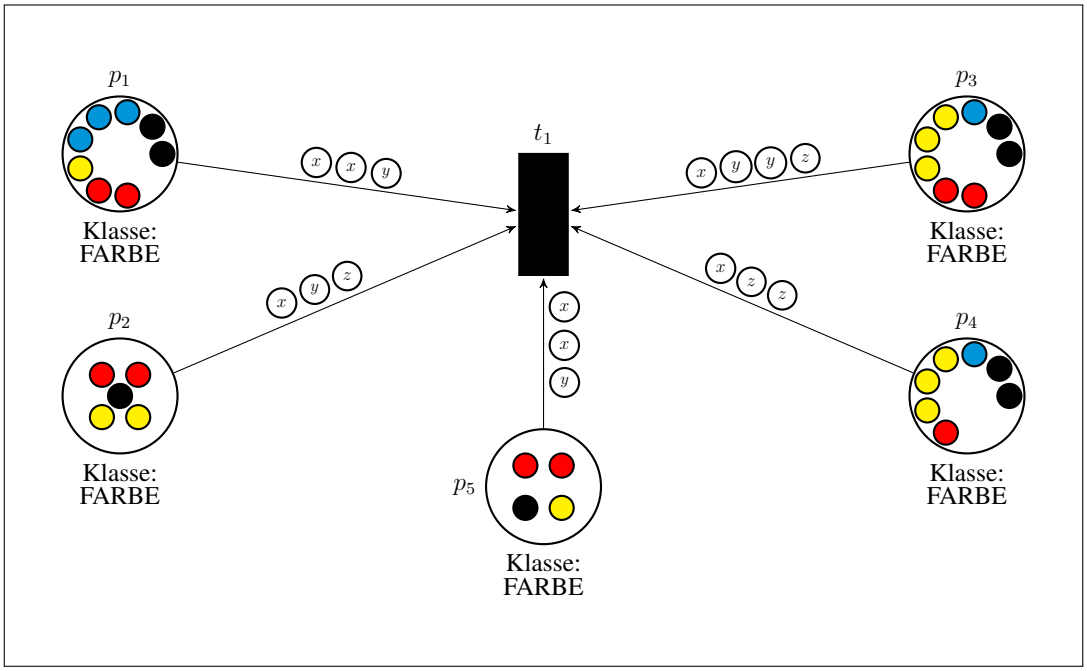

Abbildung B.4: Beispiel eines großen Netzes (Testfall 4)

Im Gegensatz zu den vorherigen Testfällen, ist es in diesem Beispiel nicht trivial, einen aktiven Modus anzugeben. Durch den programmierten Ansatz lässt es sich aber elegant bewältigen. Auf den Vorgang des Zusammenführens ist nochmals hingewiesen (siehe Algorithmus B.2).

Angesichts der Vorgehensweise ist zu sehen, dass die Prüfung der Verträglichkeit besonders beim vorliegenden Testfall Bedeutung findet. Im Laufe des Verfahrens ergibt sich bspw. anhand des Platzes p_1 und der dazugehörigen Kante, dass für die Variable y die Zuweisung 'BLAU' möglich ist. Allerdings ist dies beim Abgleich mit dem Platz p_5 und entsprechender Kante nicht machbar. Mithilfe des Tests auf Verträglichkeit wird so im Laufe des Algorithmus die Zuweisung 'BLAU' für die Variable y nicht mehr auftauchen.

Nachdem das komplette Verfahren abgearbeitet wurde, finden sich drei Modi in der aktiven Modus-Menge:

$$\beta_1 : \{x = \text{ROT}, y = \text{GELB}, z = \text{GELB}\},$$
$$\beta_2 : \{x = \text{ROT}, y = \text{GELB}, z = \text{SCHWARZ}\},$$
$$\beta_3 : \{x = \text{ROT}, y = \text{SCHWARZ}, z = \text{GELB}\}.$$

Letztlich soll auf mögliche Auswirkungen eingegangen werden, welche bei einer geringen Änderung des Zustandes entstehen können.

In der nachfolgenden Abbildung findet sich erneut das obige Beispiel aus Abb. B.4 wieder. Es wurde lediglich das Token 'SCHWARZ' aus dem Platz p_2 entfernt und durch ein Token des Attributes 'BLAU' ersetzt.

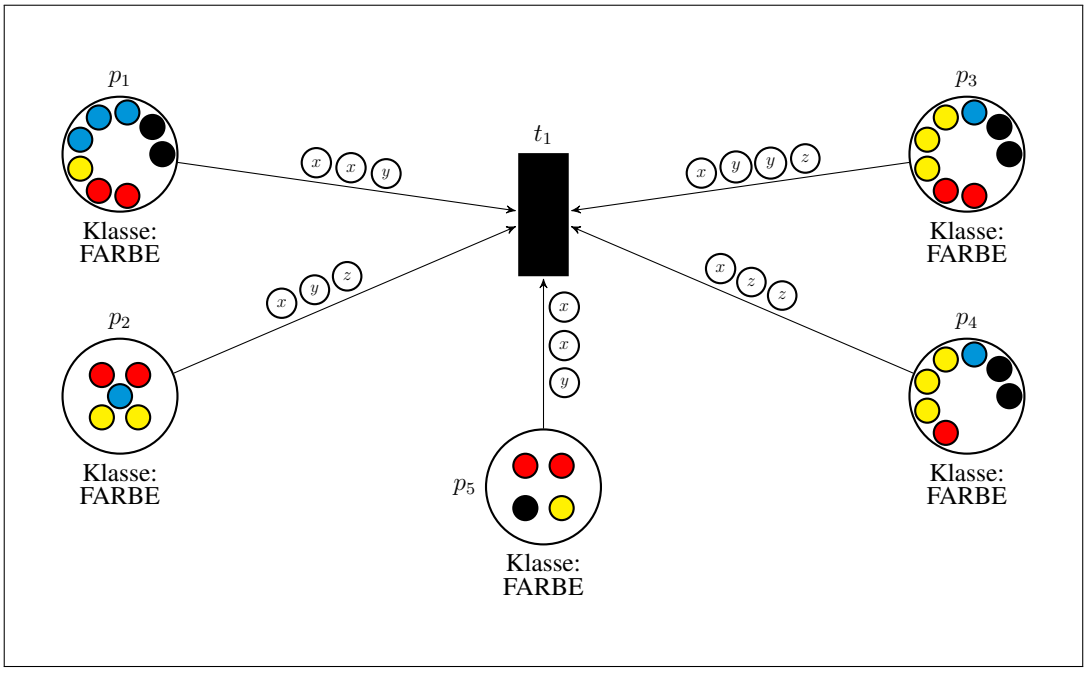

Abbildung B.5: Netz aus Abb. B.4 mit geringfügiger Änderung

Ausschließlich aufgrund der marginalen Änderung der Markierung ist die Mächtigkeit der aktualisierten aktiven Modus-Menge $Am(t)$ auf nur noch einen aktiven Modus stark reduziert worden.

$$\beta_1 : \{x = \text{ROT}, y = \text{GELB}, z = \text{GELB}\}.$$

Kontaktdaten

Autoren

Julian Silberberg (B. Sc.)
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
Interaktion 1
33619 Bielefeld
julian.silberberg@fh-bielefeld.de

Timo Lask (M. Sc.)
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
Interaktion 1
33619 Bielefeld
Telefon +49.521.106-7403
timo.lask@fh-bielefeld.de
Raum A 504

Prof. Dr. phil. Bernhard Bachmann
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
Interaktion 1
33619 Bielefeld
Telefon +49.521.106-7407
Telefax +49.521.106-7190
bernhard.bachmann@fh-bielefeld.de
Raum D 230

FSP AMMO

Sprecherin

Prof. Dr. rer. nat. Svetozara Petrova
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
FSP Angewandte Mathematische Modellierung und Optimierung
Interaktion 1
33619 Bielefeld
Telefon +49.521.106-7410
Telefax +49.521.106-7190
svetozara.petrova@fh-bielefeld.de
Raum D 226

Stellv. Sprecherin

Dr. rer. nat. Sabrina Proß
Fachhochschule Bielefeld
Fachbereich Ingenieurwissenschaften und Mathematik
FSP Angewandte Mathematische Modellierung und Optimierung
Schulstraße 10
33330 Gütersloh
Telefon +49.5241.21143-21 (70121)
sabrina.pross@fh-bielefeld.de
Raum 303

Veröffentlichungsreihe: AMMO – Berichte aus Forschung und Technologietransfer

Heft 7: T. Lye, H.-J. Kruse, T. Lask, *Heuristische Lösungsmethoden für eine Klasse von Ware-zum-Mensch-Kommissionierungsproblemen*. März 2016.

Heft 6: T. Kleine-Döpke, H.-J. Kruse, *Lösungsansätze für Konfliktsituationen bei Feuerprozessen in kapazitierten Petri-Netzen*. Juni 2015.

Heft 5: H.-J. Kruse, *Optimumgraphen*. Oktober 2014.

Heft 4: S. Proß, *Diskrete Modellierung und Optimierung praxisrelevanter Prozesse mit Petri-Netzen*. September 2014.

Heft 3: R. Ueckerdt, H.-W. Schmidt, M. Weber, E. Mindlina, *Entwicklung einer Dispatcherfunktion zur Überprüfung von Nominierungsmengen in der Betriebsführung von Erdgasspeichern*. Juli 2014.

Heft 2: R. Walden und V.-M. Roemer, *Methoden der quantitativen rechnergestützten CTG-Analyse*. April 2014.

Heft 1: AMMO-Team, *Informationen über den Forschungs- und Entwicklungsschwerpunkt Angewandte Mathematische Modellierung und Optimierung*. Dezember 2013.

ISSN 2198-4824
Herausgeber:
Vorstand von FSP AMMO
Fachhochschule Bielefeld