Fachbereich 3 - Informatik

Bachelorarbeit

Werkzeugbasierte Modellierung von Spielen mit Petrinetzen

Name:	Björn Scheetz
Matrikelnummer:	4000156
Datum:	10. Oktober 2016

1. Gutachterin:	Dr. Sabine Kuske
Zweitgutachten: Prof.	Dr. Hans-Jörg Kreowski

Erklärung

Ich versichere, die Bachelorarbeit ohne fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Bremen, den 10. Oktober 2016
Björn Scheetz

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1.

Einleitung

1.1. Motivation und Fragestellung

In den letzten Jahren ist das Verlangen nach Onlinespielen stetig gewachsen. Dadurch ist es wichtig zu wissen, wie komplex das Spiel an sich ist und wie viel Rechenleistung der Spielentwickler braucht, um das Programm ohne Probleme auszuführen. Die Komplexität der Spiele kann an der Anzahl der Zustände in einem Zustandsraum gemessen werden. Mit der Einführung der Petri-Netze im Jahr 1962 (vgl. [Pet62]) kann dieser Zustandsraum angezeigt werden. Darüber hinaus werden Petri-Netze für die Modellierung von Prozessen genutzt, aber im Bereich Spiele selten verwendet, obwohl der Erreichbarkeitsgraph, den wir im Verlauf noch erklären werden, alle möglichen Stellungen in Zuständen aufzeigt. Außerdem dienen Petrinetze als intuitive Darstellung und somit kommt man schnell auf die Idee, Spielfelder in Petri-Netze durch Stellen darzustellen. In den gefärbten Petrinetzen (vgl. [Jen92]) können mehrere vergleichbare Netzelemente zusammengefasst werden, die die Modellierung für Spiele vereinfacht. Zudem können gefärbte Petrinetze in äquivalente Stellen/Transition-Systeme umgewandelt werden, für die es gute formale Analysemethoden gibt (vgl. [Jen95]). Indem wir Stellen/Transition-Systeme und gefärbte Petrinetze mit Programmen erstellen, können interaktiv Simulationen durchführen. Ebenfalls bieten diese Programme Analysetools an, die wir im Verlauf verwenden werden, um Eigenschaften zu zeigen.

In dieser Arbeit beschäftige ich mich mit der Frage: Sind Petrinetze für das Modellieren von Brettspielen geeignet? Welche Eigenschaften können in solch einem Brettspiel mithilfe eines Petrinetzes gezeigt werden?

1.2. Zielsetzung

Das Ziel dieser Arbeit ist es, die Gebrauchstauglichkeit der Petrinetze bei Brettspielen anhand der Modellierung von TIC-TAC-TOE zu testen. Dabei spielen die Eigenschaften des Spiels eine große Rolle. Zusätzlich werden verschiedene Arten von Petrinetzen benutzt, um zu zeigen, welche Art sich am Besten eignet und welche Art selbst bei einem leichten Spiel wie TIC-TAC-TOE an seine Grenzen stößt.

1.3. Aufbau der Arbeit

Diese Arbeit setzt sich aus fünf Teilen zusammen. Zu Beginn werden einige Grundlagen im Bereich Stellen/Transitions-Systeme behandelt, damit wir mit diesen Petrinetzen arbeiten können. Anschließend wird das Spiel TIC-TAC-TOE in Tina modelliert und auf dem Weg einige Eigenschaften aufgezeigt. Tina ist ein Programm, welches Stellen/Transitions-Systeme erstellen und analysieren kann. In Kapitel 4 werden dann die gefärbten Petrinetze, die im Vergleich zu Stellen/Transitions-Systeme mehr Komponenten besitzen, behandelt. Diese werden anschließend im fünften Kapitel benutzt um TIC-TAC-TOE zu modellieren. Die Modellierung wird mit Unterstützung des Programms CPN-Tools erstellt, das gefärbte Petrinetze ebenfalls analysieren kann und in dem Buch [JK09] genutzt wird. Zusätzlich vergleichen wir in diesem Kapitel die Petrinetze miteinander. Im letzten Kapitel wird ein Fazit aus der Modellierung gezogen.

Kapitel 2.

Stellen/Transitions-Systeme

Petri-Netze wurden von Carl Adam Petri im Jahre 1960 aus endlichen Automaten entwickelt. Sie sind Modelle für jegliche Art von Prozessen oder Systemen. Stellen/Transitions-Systeme werden in der theoretischen Informatik genutzt, um Nebenläufigkeit darzustellen. Ebenso werden sie in der Praxis, speziell für die Verifikation von Hardwaresystemen genutzt (vgl. [JK09] S.3-4).

2.1. Netz

Grundsätzlich bestehen Petri-Netze aus Stellen, Transitionen und einer Flussrelation, wobei eine Stelle nie gleichzeitig eine Transition sein kann. Die Flussrelation verbindet Stellen mit Transitionen, genauso wie Transitionen mit Stellen aber niemals Stellen mit Stellen oder Transitionen mit Transitionen. Daraus ergibt sich nach [Rei86] und [Bau96] folgende Definition:

Ein Tripel $N = (S, T, F)$ heißt Netz, falls gilt:

- S ist eine endliche Menge von Stellen
- T ist eine endliche Menge von Transitionen
- S und T sind disjunkte Mengen, das heißt $S \cap T = \emptyset$
- $F \subseteq (S \times T) \cup (T \times S)$ ist eine zweistellige Relation, die Flussrelation von N.

Graphisch werden Stellen als Kreise und Transitionen als Rechtecke dargestellt. Die Flussrelation wird mit gerichteten Kanten zwischen Rechteck und Kreis gezeichnet.

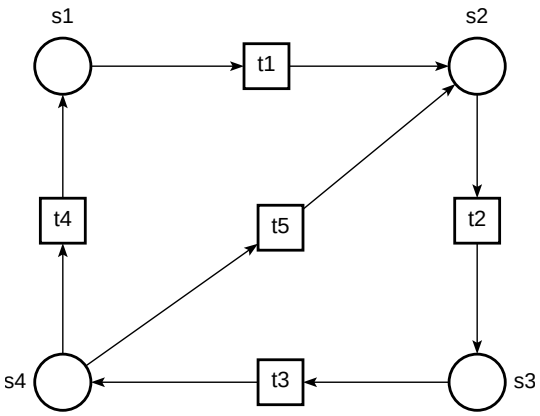


Abbildung 2.1.: Netz

Die Abbildung 2.1 zeigt ein Netz mit der Stellenmenge

- $S = \{s_1, ..., s_4\}$,
  der Transitionsmenge
- $T = \{t_1, ..., t_5\}$
  und der Flussrelation
- $F = \{(s_1, t_1), (t_1, t_2), (s_2, t_2), (t_2, s_3), (s_3, t_3), (t_3, s_4), (s_4, t_4), (s_4, t_5), (t_4, s_1), (t_5, s_2)\}$.

Damit eine Transition schalten kann, müssen wir erst den Vorbereich und den Nachbereich definieren. ${}^\bullet x$ ist der Vorbeireich von x. Der Vorbereich ist die Menge aller Eingangsknoten bzw. Inputknoten und wird definiert durch

$$ {}^\bullet x := \{y \mid (y, x) \in F\} $$

Der Nachbereich dagegen ist die Menge aller Ausgangs -bzw. Outputknoten und wird definiert durch

$$ x^\bullet := \{y \mid (x, y) \in F\} $$

Somit ist der Vorbereich von t_4 in Abbildung 2.1 derselbe wie von t_5, das heißt
${}^\bullet t_4 = {}^\bullet t_5 = \{s_4\}$

2.2. Stellen/Transitions-Systeme

Ein Stellen/Transition-System besteht nach [Rei86] und [Bau96] aus einem 5-Tupel $STS = (S, T, F, W, M_0)$. Wobei gilt:

- (S, T, F) ist ein Netz, wobei
- $W\colon F \to \mathbb{N}$ (Kantengewichte der Kanten)
- $M_0\colon S \to \mathbb{N}_0$ (Anfangsmarkierung)

Im klassischen Stellen/Transition-System werden Kapazitäten verwendet, die aber in dieser Arbeit keine Rolle spielen. Markierungen werden durch Marken in Stellen dargestellt. Die Marken werden durch kleine schwarze Punkte in den Stellen beschrieben. Falls die Anzahl der Marken größer als eins wird, wird aus Platzgründen in den Stellen, die Anzahl der Marken als ganze Zahl geschrieben. Kantengewichte $\neq 1$ werden an die Kanten geschrieben. Eine unbeschriebene Kante hat das Gewicht eins. Die formalen Begriffe werden wir nun an einem Beispiel veranschaulichen.

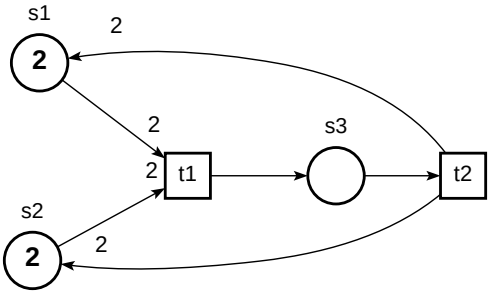


Abbildung 2.2.: Stellen/Transitions-System

Die Abbildung 2.2 zeigt ein Stellen/Transition-System mit der Stellenmenge

- $S = \{s_1, s_2, s_3\}$,
  der Transitionsmenge
- $T = \{t_1, t_2\}$
  und der Flussrelation
- $F = \{(s_1, t_1), (s_2, t_1), (t_1, s_3), (s_3, t_2), (t_2, s_1), (t_2, s_2)\}$
  Dazu kommen jetzt noch die Kantengewichte bzw. die Gewichtsabbildung:
- $W = \{((s_1, t_1), 2), ((s_2, t_1), 2), ((t_1, s_3), 1), ((s_3, t_2), 1), ((t_2, s_1), 2), ((t_2, s_2), 2)\}$
  Und ebenso die initiale Markierung:
- $M_0 = \{(s_1, 2), (s_2, 2), (s_3, 0)\}$

2.3. Markierungen

Eine Abbildung $M : S \to \mathbb{N}$ heißt *Markierung*.

2.4. Schaltungen

Nach ([Bau96] S.81) ist eine Transition $t \in T$ aktiviert unter M, geschrieben $M[t >$, wenn

- $\forall s \in {}^\bullet t : M(s) \geq W(s, t)$

Das heißt, wenn jede Stelle im Vorbeireich mindestens so viele Marken besitzt wie das Gewicht der Kanten zu der Transition, so kann die Transition geschaltet bzw. *gefeuert* werden. Dadurch entnehmen wir Marken aus den Eingangstellen und füllen Marken in die Ausgangsstellen gemäß den Kantengewichten:

$$M'(s) = \begin{cases} M(s) - W(s,t), & \text{falls } s \in {}^\bullet t \backslash t^\bullet, \\ M(s) + W(t,s), & \text{falls } s \in t^\bullet \backslash {}^\bullet t, \\ M(s) - W(t,s) + W(t,s), & \text{falls } s \in t^\bullet \cap {}^\bullet t, \\ M(s) & \text{sonst} \end{cases}$$

Wenn $t \in T$ von M nach M' schaltet, schreiben wir $M[t\rangle M'$, falls t aktiviert ist. M' ist die *unmittelbare Folgemarkierung von M unter Schalten von t* und wird auch als Mt bezeichnet. Falls wir uns in einer Markierung befinden, in der keine Transition aktiviert ist, nennen wir das Deadlock. Das Schalten einer Transition wollen wir an der Abbildung 2.2 zeigen.

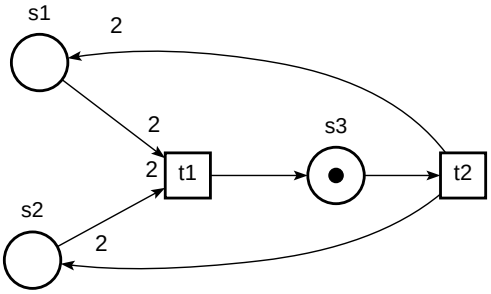


Abbildung 2.3.: Stellen/Transitions-System aus Abb. 2.2 nach dem Schalten von t_1

Im Stellen/Transitions-System von Abbildung 2.2 ist t_1 aktiviert, da ${}^\bullet t_1 = \{s_1, s_2\}$ mit

$$M_0\{s_i\} = 2 = W\{(s_i, t_1)\} \text{ für } i = 1, 2$$

ist die Bedingung, dass t_1 *aktiviert* ist, gegeben.

So erhalten wir durch das Schalten von t_1 die neue Markierung

$$M_0 t_1 = \{(s_1, 0), (s_2, 0), (s_3, 1)\}.$$

Das entsprechende Stellen/Transitions-System ist in Abbildung 2.3 zu sehen.

2.5. Schaltfolge

Um Erreichbarkeit zu definieren, müssen wir uns erst mit der Schaltfolge auseinander setzen. Nach [Bau96] S.84 lautet die Definition der Schaltfolge:

Es seien (S, T, F, W, M_0) ein Stellen/Transition-System und $w = t_1 ... t_n \in T^*$. Wir bezeichnen eine Markierung M'' *eine Folgemarkierung von* M_0 *unter* w und schreiben $M_0[w\rangle M''$, wenn:

- $w = \square \wedge M'' = M_0$, oder
- $\exists M' : M_0[t_1 ... t_{n-1}\rangle M' \wedge M'[t_n\rangle M''$.

Somit wäre w eine Schaltfolge. Im ersten Punkt wird keine Transition geschaltet und wir bleiben in M_0.
Im nächsten Punkt wird mindestens eine Transitionen geschaltet. Man schreibt $M_0[w\rangle$, falls w *aktiviert unter* M_0 ist.

Somit wäre die Markierung M'' zufälligerweise bei unserem Beispiel in Abbildung 2.2 durch das Schalten der Folge $w = t_1 t_2$ wieder:

$$M'' = \{(s_1, 2), (s_2, 2), (s_3, 0)\}$$

2.6. Erreichbare Markierungen

Erreichbare Markierungen sind nach [Bau96] S.84 definiert durch:

$$[M_0\rangle := \{M \mid M_0[w > M, w \in T^*\}$$

$[M_0\rangle$ wird als *Erreichbarkeitsmenge* des Systems bezeichnet und enthält alle möglichen Markierungen von M_0, die durch eine Schaltfolge erreicht werden können.

2.7. Erreichbarkeitsgraph

In einem sogenanten Erreichbarkeitsgraphen werden alle Zustände erfasst, die von einem Petri-Netz erreicht werden können. Also genau die *Erreichbarkeitsmenge*, die wir im vorherigen Unterkapitel behandelt haben. Nach [PW03] S.65 ist ein Erreichbarkeitsgraph wie folgt definiert:

Sei $N = (S, T, F)$ ein Netz und M_0 eine initiale Markierung von N. Der Erreichbarkeitsgraph $G(S, T, F, M_0)$ ist ein Graph mit Knoten und Kanten (V, E), wobei die Knoten $V = [M_0\rangle$.

Die Kantenmenge ist nach definiert durch:

$$Kanten\, E = \{(M', t, M'') \in [M_0\rangle \times T \times [M_0\rangle \mid M'[t\rangle M''\}$$

Im folgenden Beispiel stellen wir weiteres Stellen/Transitions-System vor, was in Abbildung 2.4 zu sehen ist.

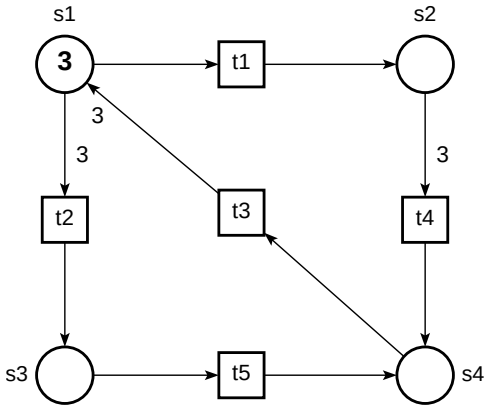


Abbildung 2.4.: Beschränktes Stellen/Transitions-System

Der daraus folgende Erreichbarkeitsgraph wurde mit einer Breitensuche erstellt und ist in Abbildung 2.5 zu sehen.

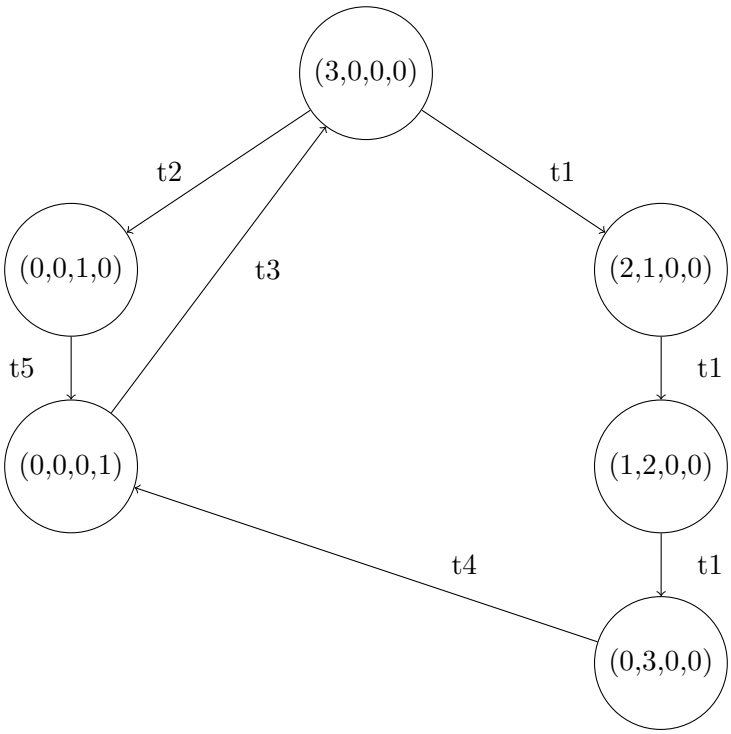


Abbildung 2.5.: Erreichbarkeitsgraph des Stellen/Transitions-System aus Abbildung 2.4

Über den Erreichbarkeitsgraph können einige Eigenschaften eines Stellen/Transitions-Systems gezeigt werden, die wir im Verlauf erklären werden.

2.8. Beschränkte Stellen/Transition-Systeme

Nach [Bau96] ist ein Stellen/Transitions-System beschränkt, wenn:

Seien $STS_B = (S, T, F, W, M_0)$ ein Stellen/Transitions-System und $B\colon S \to \mathbb{N}_0 \cup \{\infty\}$ eine Abbildung, die jeder Stelle eine 'kritische Markenzahl' zuordnet. N heißt *B-sicher* bzw. *B-beschränkt*, wenn:

$$\forall M \in [M_0\rangle, s \in S : M(s) \leq B(s).$$

Wenn $B = 1$, $B = 2$ spricht man von *1-sicher* ,*2-sicher* usw. STS_B heißt beschränkt, wenn es eine natürliche Zahl b gibt, für die STS_B *b-sicher* ist.

Darüber hinaus ist ein Stellen/Transitions-System genau dann beschränkt, wenn seine *Erreichbarkeitsmenge* endlich ist. Somit also auch der Erreichbarkeitsgraph.

2.9. Inhibitorkanten

Ein inhibitorisches Stellen/Transitions-System $STS_I = (S, T, F, W, M_0, inh)$ besteht aus einem Stellen/Transitionsystem (S, T, F, W, M_0) und einer Abbildung $inh : T \to 2^S$, die sogenannte inhibitorische Kanten von Stellen auf Transitionen definiert. Eine Transition $t \in T$ ist feuerbar in einer Markierung, genau dann, wenn für alle $s \in {}^\bullet t$ $M(s) \geq W(s,t)$ und für alle $s' \in inh(t)$ $M(s') = 0$ gilt. Siehe [PW03] S.166.

Das heißt, eine Transition $t \in T$ ist nur *aktiviert*, wenn mindestens genügend Marken in jeder Vorstelle sind und wenn die Markenanzahl in der Stelle 0 beträgt, falls es von

der Stelle eine Inhibitorkante zur Transition gibt. Bei Inhibitorkanten wird anstatt eines Pfeiles ein kleiner Kreis gezeichnet. In den folgenden Abbildungen 2.6 und 2.7 sehen wir uns ein Beispiel an.

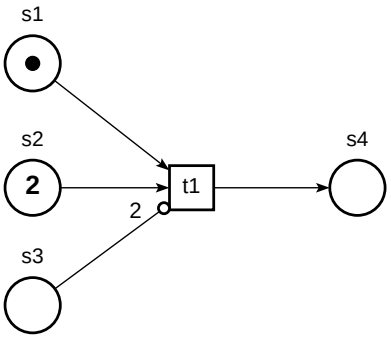


Abbildung 2.6.: Inhibitorisches Stellen/Transitions-System - t_1 ist aktiviert

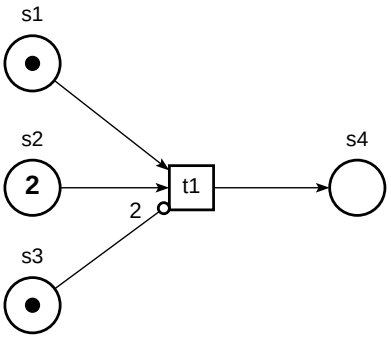


Abbildung 2.7.: Inhibitorisches Stellen/Transitions-System - t_1 ist nicht aktiviert

2.10. Prioritäten

Ein Stellen/Transitions-System mit Prioritäten $STS_P = (S, T, F, W, M_0, >)$ besteht aus einem Stellen/Transitions-System (S, P, F, W, M_0) und einer irreflexiven, transitiven Relation $> \subseteq T \times T$, die Prioritäten für Transitionen angibt. Eine Transiton $t \in T$ darf nur dann schalten, wenn $\nexists\, t' \in T : (M[t\rangle) \wedge (t' > t)$ gilt. Siehe [PW03] S.166.

Das heißt, wenn zwei Transition *aktiviert* sind und $t_1 > t_2$ ist, also t_1 eine höhere Priorität hat, muss zuerst t_1 geschaltet werden. In graphischer Form wird im Programm Tina ein gelber Pfeil von der höher priorisierten Transition zur niedrigeren Transition gezogen, was in der Abbildung 2.8 zu sehen ist.

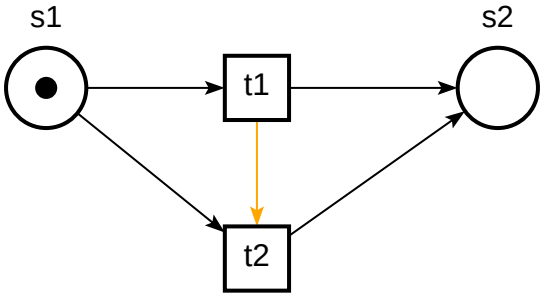


Abbildung 2.8.: Priorität

Kapitel 3.

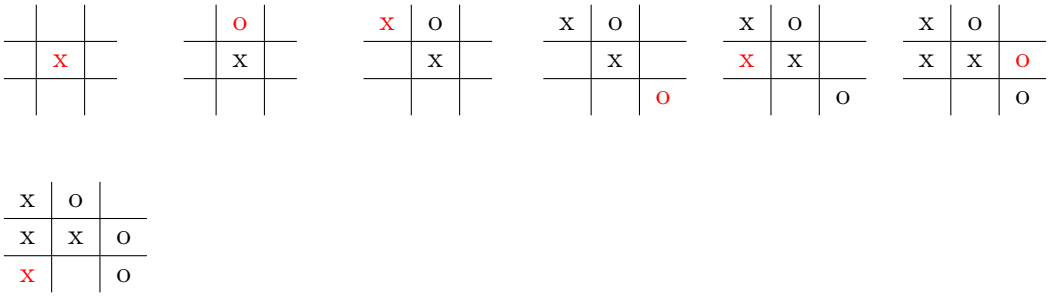
TIC-TAC-TOE in Tina

3.1. TIC-TAC-TOE

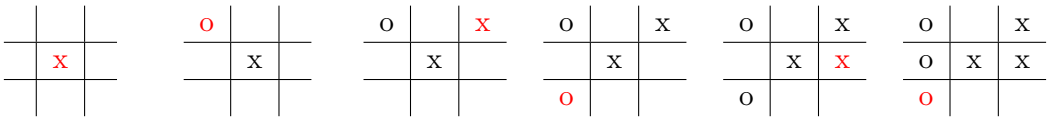
TIC-TAC-TOE ist ein Strategiespiel, das auf zwei Personen ausgelegt ist. Es wird auf einem 3x3 Spielfeld ausgetragen, wobei die Spieler abwechselnd ihr Zeichen in ein freies Feld eintragen. Die Zeichen bestehen meistens aus Kreuz und Kreis, wobei sich ein Spieler für ein Zeichen entscheiden muss. Das Ziel ist, drei gleiche Zeichen in einer Spalte, Reihe oder Diagonale zu bekommen. Falls die Spieler keine Fehler machen, wird dieses Ziel nach 9 Zügen nicht erreicht und das Spiel endet unentschieden.

Zur Anschauung werden wir drei Beispielpartien angucken, wobei einmal Spieler 1 bzw. Spieler 2 gewinnt und eine Partie, bei der das Spiel unentschieden ausgeht. Spieler 1 entscheidet sich für das Zeichen x und Spieler 2 entscheidet sich für das Zeichen o.

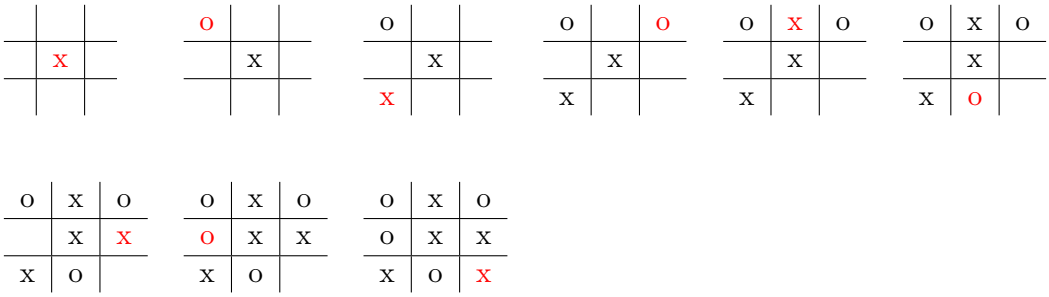
Spieler 1 gewinnt:



Spieler 2 gewinnt:



Unentschieden:



Mit der Modellierung durch Petri-Netze lassen sich Eigenschaften zeigen, wie jede Partie nach neun Zügen beendet sein muss, also immer terminiert, oder wie viele verschiedene Spielsituationen es geben kann.

3.2. Tina

Tina (Time petri Net Analyzer) ist ein Programm, welches Petri-Netze erstellen, bearbeiten und analysieren kann. Zusätzlich lassen sich Inhibitorkanten und Prioriäten einbauen, die wir in den vorherigen Abbildungen, die wir mit Tina erstellt haben, schon erkennen konnten. Darüber hinaus können wir Erreichbarkeitsgraphen, Überdeckungsgraphen (vgl. [PW03] S.66), Sturkturanalyse und viele weitere Analysen, die wir im Verlauf erklären, mit Tina modellieren.

3.3. TIC-TAC-TOE in Tina

Im folgenden Spiel werden wir die Modellierungen in kleinere Teile zerlegen, um die Übersicht des Petri-Netzes nicht zu verlieren. Zuerst modellieren wir das TIC-TAC-TOE-Spiel ohne die Gewinnbedingung und wir betrachten nur die ersten zwei Felder des 3x3 TIC-TAC-TOEs zur Übersicht.

Dafür nehmen wir das folgende Stellen/Transition-System in Abbildung 3.1
$STS_0 = (S, T, F, 1, M_0)$.

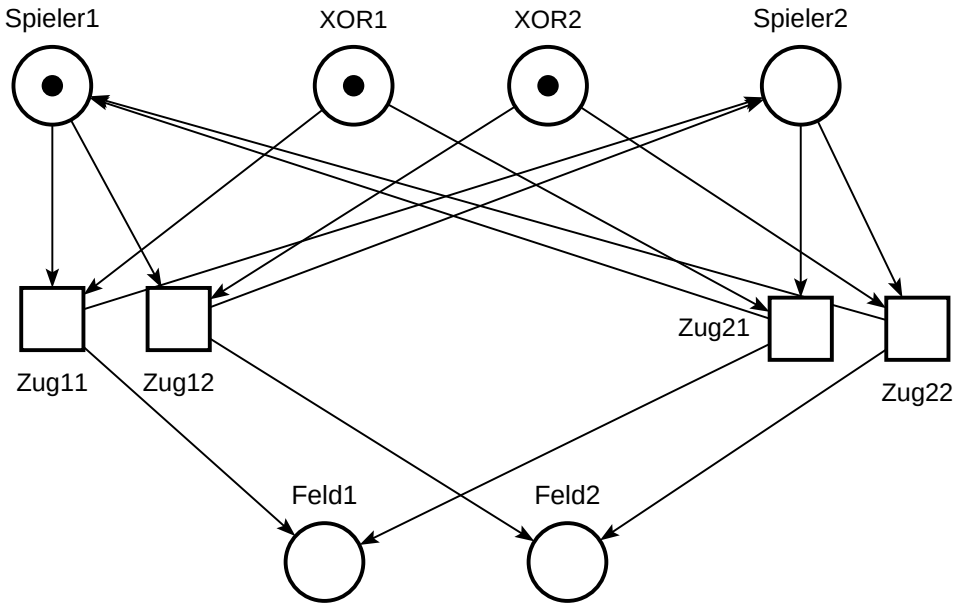


Abbildung 3.1.: 2 Felder TIC-TAC-TOE

- Die Stellen $Spieler_1, Spieler_2$ sind dafür zuständig, welcher Spieler am Zug ist.
- $Feld_1$ und $Feld_2$ sind die ersten zwei Felder, oben links und oben Mitte, des 3x3 TIC-TAC-TOE Feldes.
- Die Transitionen Zug_{11}, Zug_{12} machen die Kreuze für Spieler 1 auf den 2 Feldern des TIC-TAC-TOE Feldes.
- Zug_{21}, Zug_{22} füllen ebenfalls die Stellen $Feld_1, Feld_2$ mit Kreisen für Spieler 2.
- XOR_1 ist Vorstelle von Zug_{11} und Zug_{21}, damit nur einmal auf das Feld geschrieben werden kann. Ebenso gilt dies für XOR_2, die Vorstelle von Zug_{12} und Zug_{22} ist. Dies führt zu einem wechselseitigen Ausschluss (vgl. [Bau96] S.230).

Initial enthält $Spieler_1$ eine Marke, da Spieler 1 anfangen darf. Nachdem er beispielsweise Zug_{11} schaltet, wird aus $Spieler_1$ und XOR_1 eine Marke entfernt und das Feld $Feld_1$ mit einer Marke gefüllt. Ebenso wird eine Marke in $Spieler_2$ eingefügt, damit nun Spieler 2

am Zug ist. XOR_1 ist dauerhaft leer, damit niemand mehr Marken in $Feld_1$ schreiben kann.

Zur Übersicht definieren wir uns weitere Mengen, die Spieler, Felder, XOR und Zuege heißen, wobei:

- $Spieler = \{Sp_i \mid i \in [2]\}$
- $Felder = \{F_i \mid j \in [9]\}$
- $XOR = \{X_i \mid j \in [9]\}$
- $Zuege = \{Z_{ij} \mid i \in [2], j \in [9]\}$

Dabei bezeichnet $[n]$ mit $n \in \mathbb{N}$ die Menge $\{1, ..., n\}$.

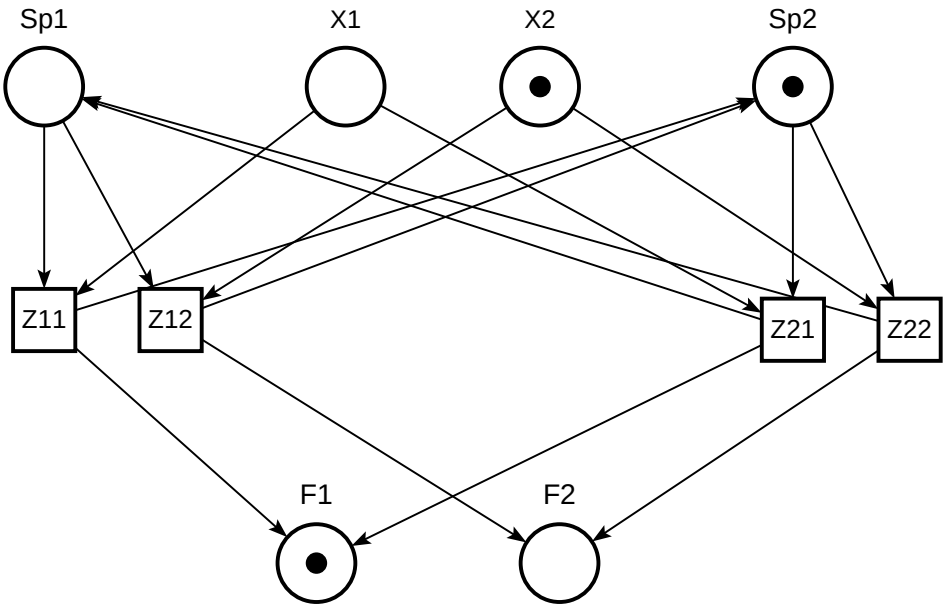


Abbildung 3.2.: 2 Felder TIC-TAC-TOE - Z_{11} geschaltet

In Abbildung 3.2 erkennt man gut, dass nur noch Z_{22} für Spieler 2 aktiviert ist, da XOR_1 keine Marke mehr enthält. Im folgenden Szenario wird ein Stellen/Transition-System vorgestellt, welches alle 9 TIC-TAC-TOE Felder besitzt.

Wir nehmen wieder ein Stellen/Transition-System $STS_1 = (S, T, F, 1, M_0)$.

- $S = Spieler \cup Felder \cup XOR$
- $T = Zuege$
- $F = \{\{(Sp_i, Z_{ij}) \mid i \in [2], j \in [9]\} \cup \{(Z_{ij}, F_j) \mid i \in [2], j \in [9]\} \cup$
  $\{(X_j, Z_{ij}) \mid i \in [2], j \in [9]\} \cup \{(Z_{2j}, Sp_1) \mid j \in [9]\} \cup \{(Z_{1j}, Sp_2) \mid j \in [9]\}\}$
- $M_0 = \{(Sp_1, 1), (Sp_2, 0) \cup \{(X_j, 1) \mid j \in [9]\} \cup \{(F_j, 0) \mid j \in [9]\}\}$

In der folgenden Abbildung 3.3 wird deutlich, dass das Programm sogar ohne Gewinnbedingung jetzt schon unübersichtlich wird. Doch dieses Petri-Netz arbeitet genauso wie das Stellen/Transition-System in Abbildung 3.2. Jede Transition hat zwei eingehende und zwei ausgehende Kanten. Der Vorbereich besteht aus der Spielerstelle Sp_1 oder Sp_2 und besitzt eine gemeinsame Vorstelle $\{X_1, ..., X_9\}$ damit nur einmal auf dem Feld interagiert werden kann. Der Nachbereich besteht aus den Feldern $\{F_1, F_2\}$ und dem jeweiligen Spieler, der den nächsten Zug macht, Sp_1 oder Sp_2.

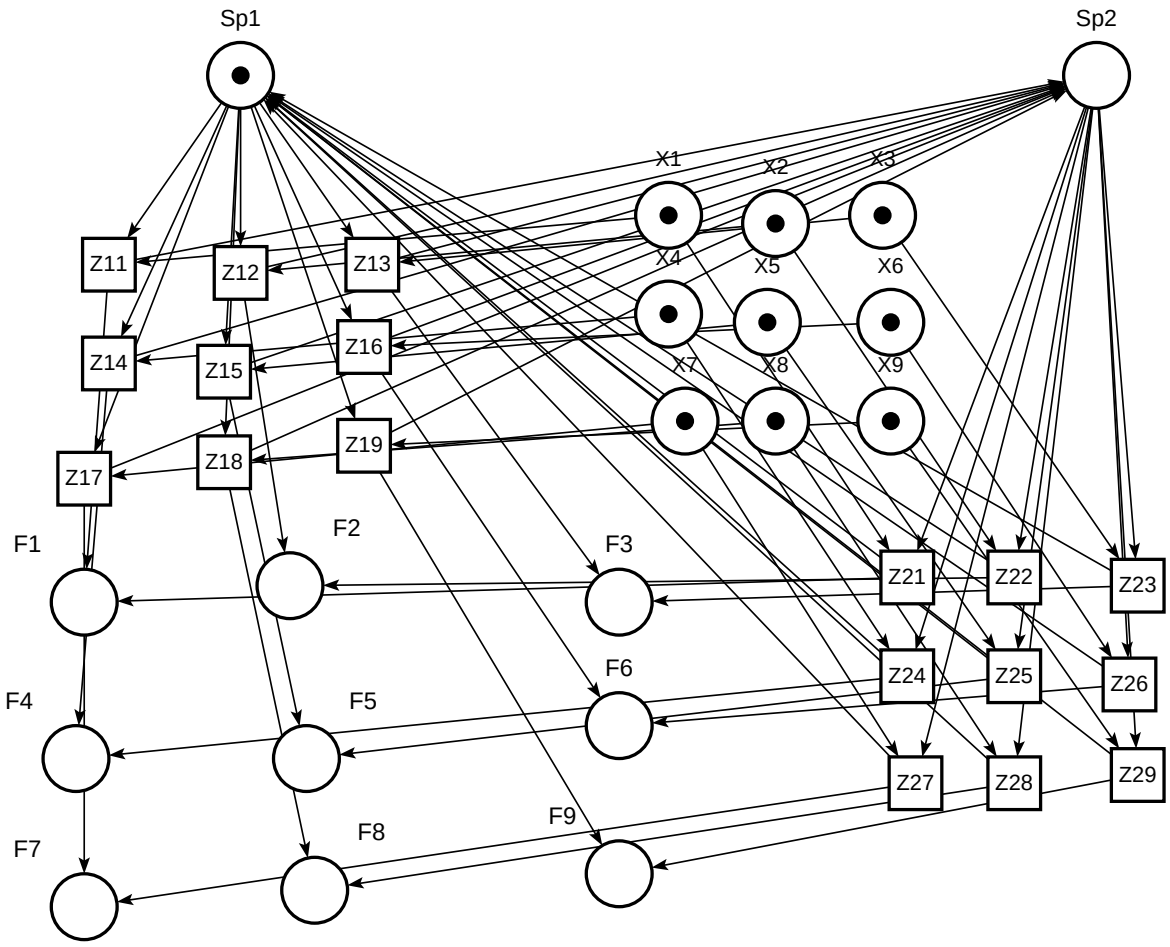


Abbildung 3.3.: 9 Felder TIC-TAC-TOE

Aus dieser Konstellation lassen sich interessante Eigenschaften vermuten. Über Tina lässt sich der Erreichbarkeitsgraph erzeugen. Dazu klickt der User über den Reiter *Tools* auf *reachability analysis*. Danach öffnet sich ein Fenster, worauf sich einige Aktionen ausführen lassen. Neben dem Erreichbarkeitsgraph *(marking graph)* lassen sich auch Überdeckungsgraphen *(coverability graph)* und viele weitere Graphen erzeugen, die sich über verschiedene Algorithmen aufbauen lassen. Der erzeugte Erreichbarkeitsgraph ist in der Abbildung 3.4 zu sehen. Dabei beschränken wir uns auf dem Bereich *count*.

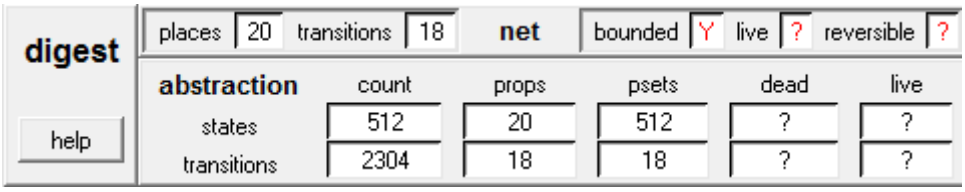


Abbildung 3.4.: Erreichbarkeitsmenge des Stellen/Transitions-System aus Abbildung 3.3

Nachdem wir den Erreichbarkeitsgraph erstellt haben, werden 512 mögliche Zustände ausgegeben. Wegen der Größe wird der Erreichbarkeitsgraph in graphischer Form auf der CD zu finden sein, die sich im Anhang befinden. Daraus lässt sich vermuten, dass sich die Menge an Zuständen aus $Spieler^{Felder}$ bildet, also $2^9 = 512$ Zustände. Um diese These weiter zu überprüfen, entfernen wir eine Stelle aus der Menge $s3, ..., s11$. Dadurch hat das TIC-TAC-TOE Spiel nur noch acht Felder. Der daraus folgende Erreichbarkeitsgraph hat 256 Stellen, somit ist die These für einen weitere natürliche Zahl bestätigt, da $2^8 = 256$.

Um das Stellen/Transition-System übersichtlicher zu gestalten, benutzen wir nun Inhibitorkanten. Dadurch kann die Stellenmenge $\{X_1, ..., X_9\}$ eingespart werden, da wir Inhibitorkanten von den TIC-TAC-TOE Feldern $\{F_1, ..., F_9\}$ zu den dazugehörigen Transitionen $\{Z_{ij} \mid i \in [2], j \in [9]\}$ ziehen. Der Effekt ist der gleiche, da nur geschaltet werden darf, wenn in den Stellen $\{F_1, ..., F_9\}$ keine Marke vorhanden ist.

Wir nehmen ein Stellen/Transition-System $STS_2 = (S, T, F, 1, M_0, inh)$, was in Abbildung 3.5 zu sehen ist.

- $S = Spieler \cup Felder$
- $T = Zuege$
- $F = \{\{(Sp_i, Z_{ij}) \mid i \in [2], j \in [9]\} \cup \{(Z_{ij}, F_j) \mid i \in [2], j \in [9]\} \cup \{(Z_{(2j}, Sp_1) \mid j \in [9]\} \cup \{(Z_{1j}, Sp_2) \mid j \in [9]\}\}$
- $M_0 = \{(Sp_1, 1), (Sp_2, 0) \cup \{(F_i, 0) \mid i \in [9]\}\}$
- $inh = \{(F_j, Z_{ij}) \mid i \in [2], j \in [9]\}$

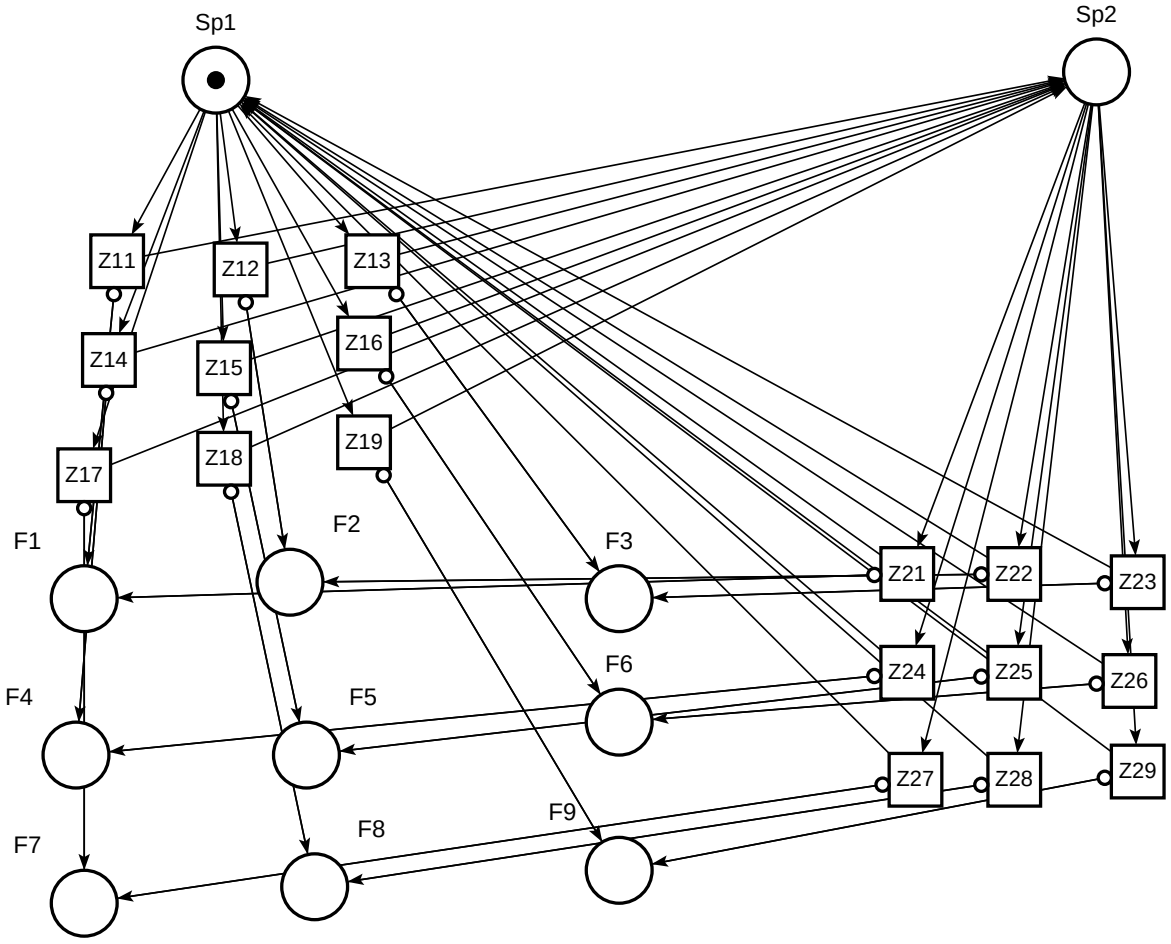


Abbildung 3.5.: Stellen/Transition-System mit Inhibitorkanten

Der davon erzeugte Erreichbarkeitsgraph enthält ebenfalls 512 Zustände. Doch bestimmte Analysetools von *Tina* scheinen Probleme mit Inhibitorkanten zu haben. Der Überdeckungsgraph scheint 1023 Zustände gegenüber dem Erreichbarkeitsgraph zu haben, obwohl die *Erreichbarkeitsmenge* endlich ist und nach [PW03] S.84 ist der Überdeckungsgraph beschränkter Netze $Cov(N) = G(S, T, F, M_0)$. Das heißt, die Anzahl der Zustände im Überdeckungsgraphen müssten gleich der Anzahl der Zustände im Erreichbarkeitsgraphen sein.

Marken werden auch anonyme Marken genannt, da sie keine Informationen mitliefern (vgl. [Bau96] S.206). Da wir in TIC-TAC-TOE zwischen Kreuzen und Kreisen unterscheiden müssen, können wir nicht einfach die acht Gewinnmöglichkeiten verbinden und den Sieger ausgeben. Damit das Petri-Netz vorerst übersichtlich bleibt, zeigen wir die Umsetzung für eine Gewinnsituation.

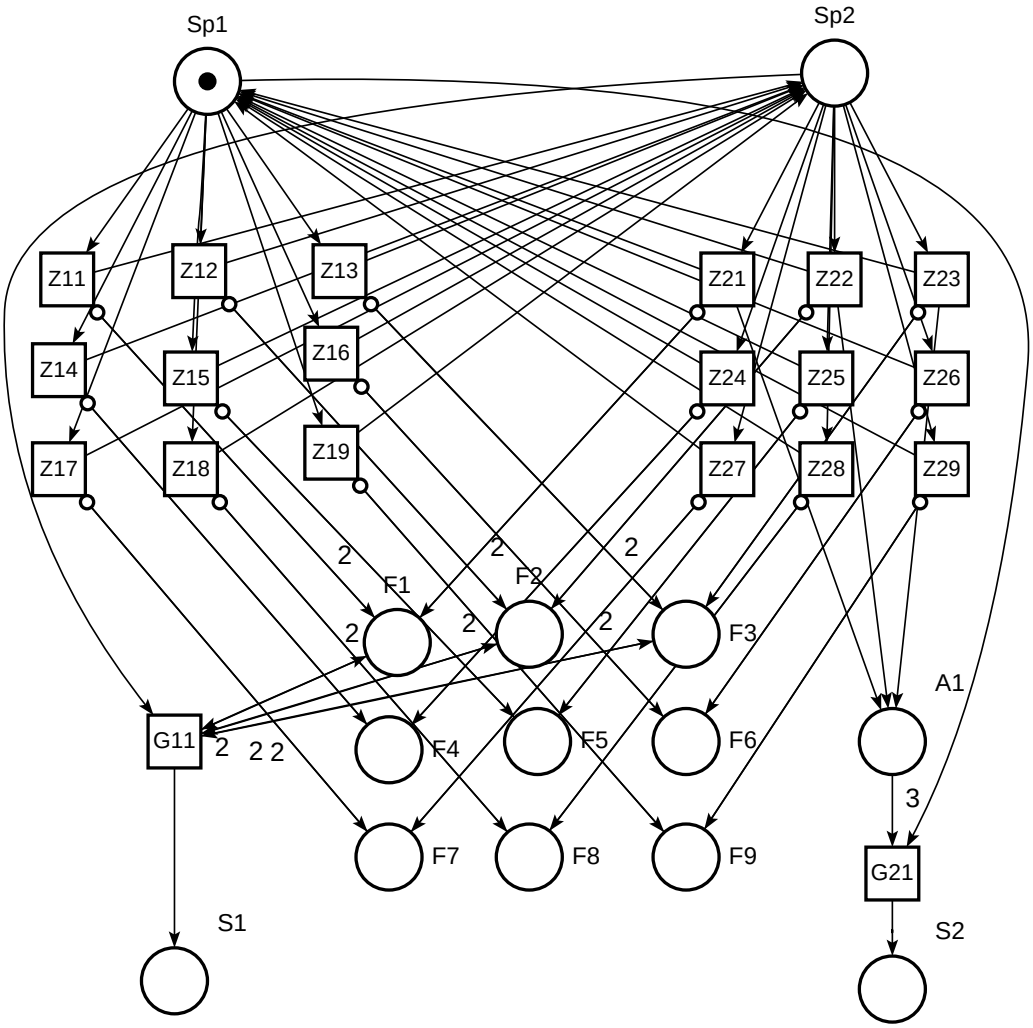


Abbildung 3.6.: Gewinnsituation für eine Reihe

In der Abbildung 3.6 können wir erkennen, dass Spieler 1 immer zwei Marken in die Stellen $\{F_1, .., F_3\}$ legt. Spieler 2 addiert immer noch eine Marke in die Stellen $\{F_1, ..., F_9\}$. Somit können wir graphisch unterscheiden, was $''Kreuze''$ bzw. $''Kreise''$ sind. In der Abbildung ist G_{11} *aktiviert*, wenn die Stellen F_1, F_2, F_3 jeweils zwei Marken besitzen. Nachdem G_{11} gefeuert wurde, wird die Marke aus Sp_2 entfernt und das Spiel endet in einem Deadlock. Zudem füllen wir die Stellen F_1, F_2, F_3 wieder mit zwei Marken, damit einerseits im Feld erkannt wird, welcher Spieler gewonnen hat und damit wir Zustände im Erreichbarkeistgraphen sparen.

Leider ist das anders herum nicht der Fall, da wir nur nach einer Marke abfragen würden.

Darum wird beim Schalten einer Transition bei Spieler 2 eine Marke in einer extra Stelle A_1 gespeichert. Falls es Spieler 2 schafft, Z_{21}, Z_{22} und Z_{23} zu schalten, hat Spieler 2 gewonnen und in der Stelle A_1 werden sich drei Marken befinden. Somit ist auch G_{21} *aktiviert* und entfernt die Marke aus der Spielerstelle Sp_1. Somit würde das Petri-Netz in einem Deadlock landen. Spieler 1 hätte gewonnen, wenn der Spieler eine Marke in S_1 hätte. Spieler 2 hätte gewonnen, wenn eine Marke in S_2 ist. Mithilfe einer Schaltfolge $w \in T^*$ zeigen wir in einem Anwendungsfall die Möglichkeiten an diesem reduzierten Beispiel.

Nach der Schaltfolge $w_1 = Z_{11}Z_{24}Z_{12}Z_{25}Z_{13}G_{11}$ hat Spieler 1 gewonnen.

Nach der Schaltfolge $w_2 = Z_{17}t_{21}Z_{18}Z_{22}t_{14}Z_{23}G_{21}$ hat Spieler 2 gewonnen.

Nach der Schaltfolge $w_3 = Z_{15}Z_{21}Z_{14}t_{26}Z_{t17}Z_{23}Z_{12}Z_{28}Z_{19}$ hat keiner gewonnen.

Leider gibt es ein großes Problem bei dieser Implementierung. Da es noch andere Möglichkeiten gibt, außer G_{11} bzw. G_{21} zu feuern, könnten die Spieler einfach weiter spielen. Hierbei werden uns die Prioritäten helfen, damit erzwungen wird, die gewinnbringende Transition zu feuern.

Die gewinnbringende Transition muss eine höhere Priorität haben als die gegnerischen Transition zur Setzung eines Symboles.

Dazu nehmen wir wieder ein Stellen/Transitions-System $STS_2 = (S, T, F, W, M_0, inh, >)$, was in Abbildung 3.7 zu sehen ist.

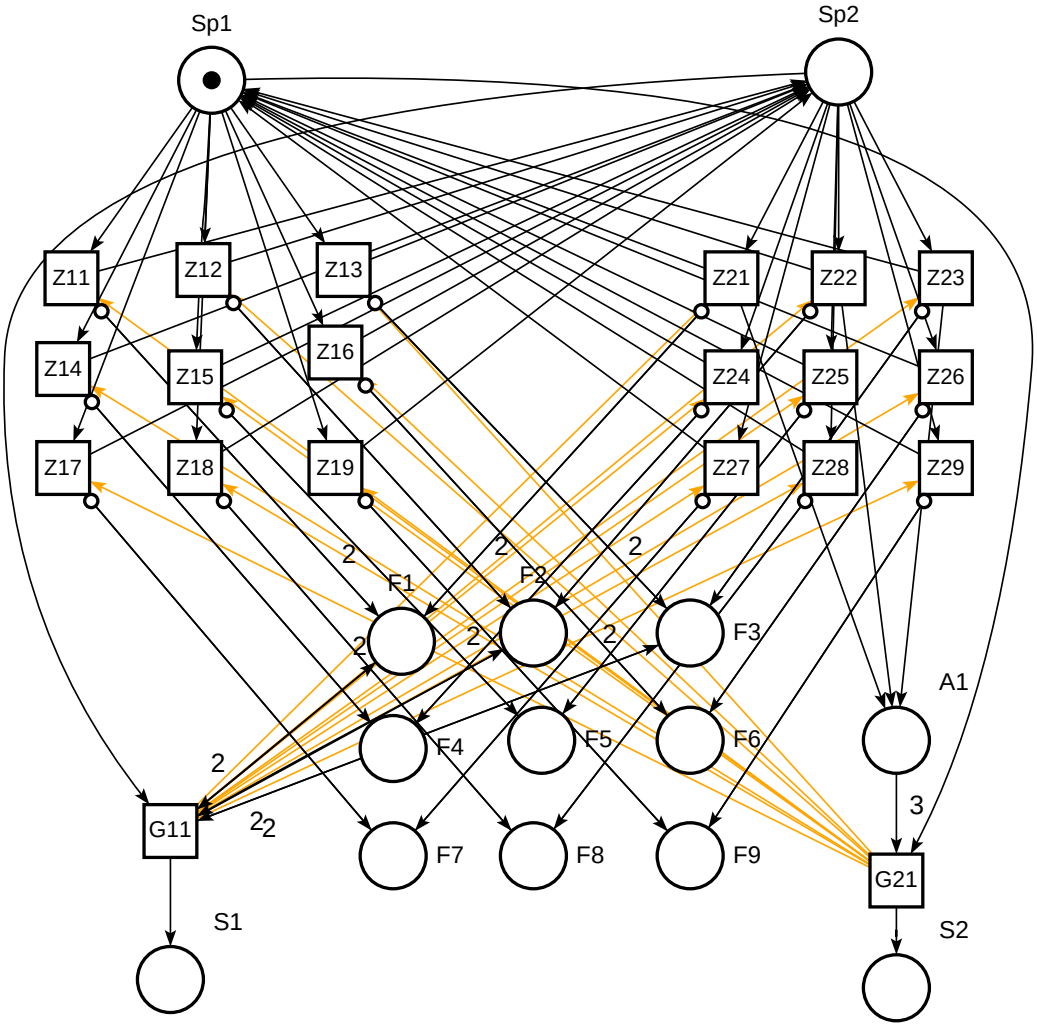


Abbildung 3.7.: Gewinnsituation für eine Reihe mit Prioritäten

Somit kann sichergestellt werden, dass das Spiel beendet wird, wenn ein Spieler gewonnen hat.

Bevor wir das Spiel mit voller Funktonalität zeigen, definieren wir weitere Mengen. Die heißen Gewinn, Auslagerung und Sieg, wobei:

- $Gewinn = \{G_{ij} \mid i \in [2], j \in [8]\}$
- $Auslagerung = \{A_j \mid j \in [8]\}$
- $Sieg = \{S_i \mid i \in [2]\}$

Damit wir alle acht Gewinnmöglichkeiten für Spieler 1 speichern können, brauchen wir sieben weitere Transitionen $\{G_{12}, ..., G_{18}\}$. Für Spieler 2 benötigen wir ebenfalls sieben weitere Stellen $\{A_2, .., A_9\}$ plus sieben weitere Transition $\{G_{22}, ..., G_{28}\}$. Darüber hinaus müssen alle gewinnbringenden Transitionen $\{G_{ij} \mid i \in [2], j \in [8]\}$ eine höhere Priorität besitzen, als die Transitionen, die für das Setzen der gegnerischen Symbole zuständig sind.

Die folgende Abbildung 3.8 ist sehr unübersichtlich und der Leser erkennt gut, dass das Stellen/Transitionsystem trotz extra Prioritäten und Inhibitorkanten selbst bei TIC-TAC-TOE schnell wachsen kann. Ohne Prioritäten würde die Speicherung der gewinnbringenden Transitionen sehr viel komplexer ausfallen, da wir nach einem Sieg in einem Deadlock landen müssen. Die Umsetzung von TIC-TAC-TOE lässt sich in gefärbten Petri-Netzen einfacher umsetzen, doch dazu kommen wir noch.

Das vollständige TIC-TAC-TOE Spiel besteht aus $STS_3 = (S, T, F, W, M_0, inh, >)$.

Wegen der Komplexität des Stellen/Transitions-Systems STS_3 wird auf eine Formalisierung des Petri-Netzes verzichtet.

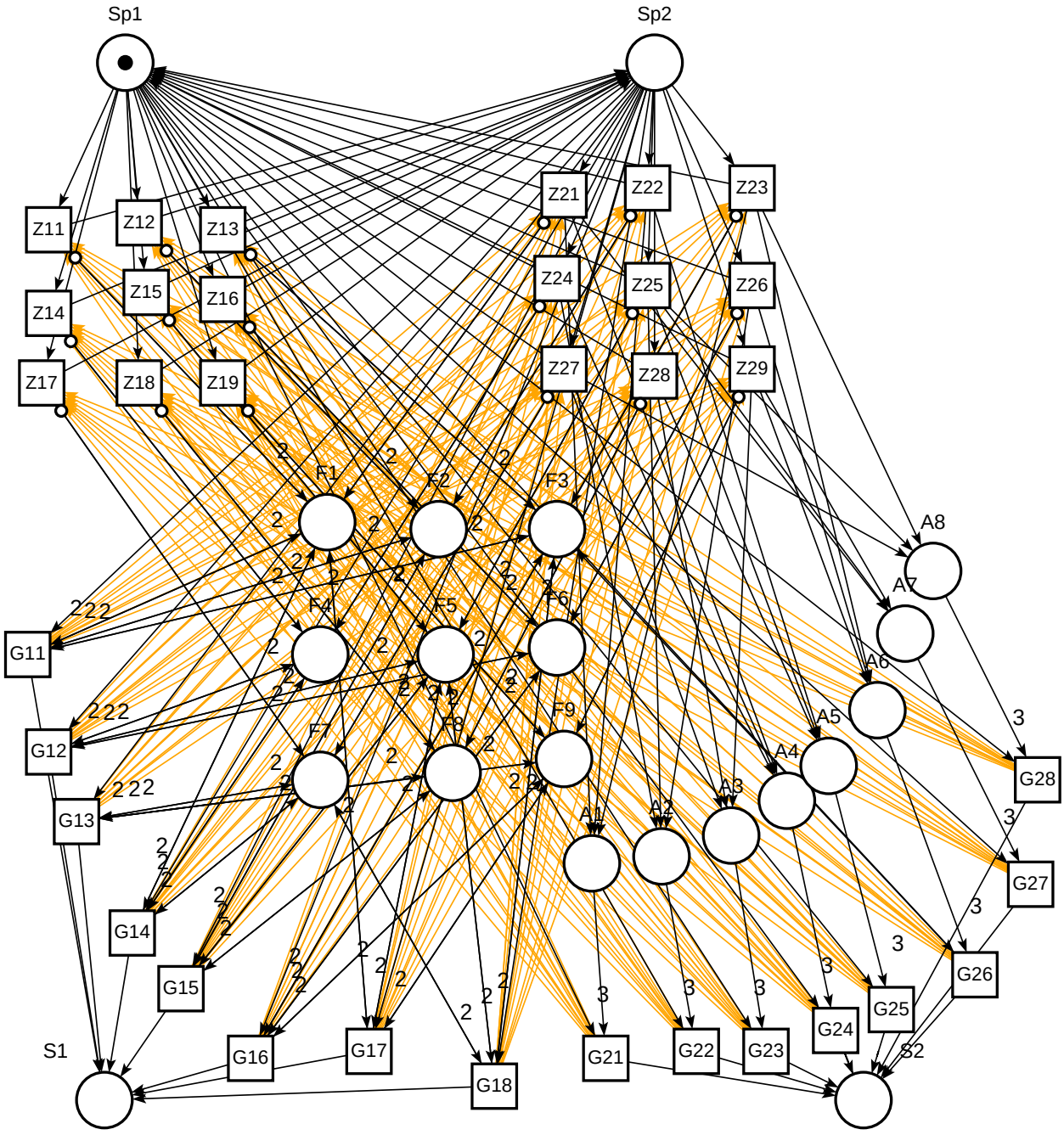


Abbildung 3.8.: TIC-TAC-TOE - vollständig

Kapitel 4.

Gefärbte Petrinetze

Gefärbte Petrinetze sind in der praktischen Anwendung sehr beliebt, da sie einen kompakten Aufbau haben und parametrisierte Modelle ermöglichen (vgl. [JK09] S.4) . Die Entwicklung der gefärbten Petrinetze steht eng im Einklang mit der Sprache CPN ML (Coloured Petri Nets - Modeling Language). Die Sprache ist für Definitionen einiger Mengen, wie Typen, Variablen und Ausdrücke zuständig, die wir im Verlauf erklären werden. Die folgenden Definitionen wurden aus [JK09] S.79 - 91 entnommen.

4.1. Nicht-hierarchisches gefärbtes Petrinetz

Grundsätzlich besteht ein gefärbtes Petri-Netz aus einem 9-Tupel
CPN $= (S, T, F, \Sigma, V, C, G, E, I)$, wobei:

- (S,T,F) ein Netz ist
- Σ eine nicht endliche Menge von Typen ist.
- V eine endliche Menge von Typvariablen mit $Type[v] \in \Sigma, \forall\, v \in V$ ist.
- $C\colon S \to \Sigma$ ist eine Typfunktion ist, wo jede Stelle ein Typ zugewiesen wird.
- $G\colon T \to EXPR_V$ ist eine Wächterfunktion ist, die jeder Transition t einen Wächter zuweist, mit $Type[G(t)] = BOOL$.
- $E\colon F \to EXPR_V$ ist eine Kantenausdrucksfunktion ist, die jeder Kante f einen Kantenausdruck zuweist, mit $Type[E(f)] = C(s)_{MS}$, wobei s die Stelle ist, die mit f verbunden ist.
- $I\colon S \to EXPR_{MS}$ ist eine Initialisierungsfunktion ist, wobei jeder Stelle s ein Initialisierungsausdruck zugewiesen wird, mit $Type[I(s)] = C(s)_{MS}$.

Die Mengen S, T und F sind im Kapitel 2.1 beschrieben. Die anderen sechs Komponenten werden in den folgenden Kapiteln mithilfe von Beispielen genauer erklärt.

4.2. Multimengen

Wir starten bei der Formalisierung mit den Multimengen. Sie werden in späteren Definitionen wie Markierungen und Schalten verwendet.

Sei $B = \{b_1, b_2, b_3\}$ eine nicht leere Menge. Dann ist eine Multimenge über B eine Funktion $m\colon B \to \mathbb{N}$. Eine Multimenge kann somit auch als Summe geschrieben werden:

$$\sum_{b \in B} = m(b)'b = m(b_1)'b_1 +\!\!+\ m(b_2)'b_2 +\!\!+\ m(b_3)'b_3 +\!\!+\ ...$$

Die Menge aller Multimengen über B wird mit B_{MS} und die leere Menge Multimenge mit $\emptyset_{MS}$ bezeichnet.

Seien m_1, m_2 und m Multimengen und $n \in \mathbb{N}$, dann sind Addition, Skalarmultiplikation, Vergleich, Länge und Subtraktion wie folgt definiert:

- $Addition = \forall b \ \in B : (m_1 +\!\!+ \ m_2)(b) = m_1(b) + m_2(b)$.
- $Skalarmultiplikation = \forall b \in B : (n * *m)(b) = n * m(b)$.
- $Vergleich = m_1 \ll= m_2 \Leftrightarrow \forall b \in B \colon m_1(b) \leq m_2(b)$.
- $Länge = |m| = \sum_{b \in B} m(b)$.
- $Subtraktion = \forall b \in B : (m_2 - -m_1)(s) = m_2(b) - m_1(b)$. Dabei wird vorausgesetzt, dass $m_1 \ll= m_2$ gilt.

Innerhalb der CPN ML würde eine Deklaration etwa so aussehen:

$$multi = 3'(''Anfang'') +\!\!+ \ 2'(''Ende'');$$

Das heißt, wir haben insgesamt fünf Elemente. Drei Elemente mit dem Inhalt *Anfang* und zwei Elemente mit Inhalt *Ende*.

4.3. Typen

Gefärbte Petri-Netze unterstützen einige Typen, wie UNIT, BOOL, STRING, INT, REAL und TIME. Trotzdem werden wir uns im folgenden Beispiel auf STRING und INT beschränken. Grundsätzlich ist Σ eine endliche nicht leere Menge von Typen. Zuerst definieren wir drei Typen, mithilfe des Befehls *colset* von CPN ML. Die Typen in der Abbildung 4.1 sind wie folgt definiert:

- colset ZAHLEN = int;
- colset DATEN = string;
- colset ZAHLENxDATEN = product ZAHLEN * DATEN;

Produkte lassen sich mithilfe des Befehls *product* erzeugen und können sich aus zwei oder mehreren Komponenten zusammen setzen. Nun besitzen wir drei Elemente in der Menge Σ.

$$\Sigma = \{\text{ZAHLEN, DATEN, ZAHLENxDATEN}\}$$

Darüber hinaus gibt es Variablen, die ebenfalls in CPM ML deklariert werden können. Dazu bezeichnen $EXPR$ die Menge der Ausdrücke und $Type[e]$ den Typ des Ausdrucks e, wobei $e \in EXPR$. Außerdem bezeichnen wir die Menge der freien Variablen in einem Ausdruck e als $Var[e]$ und den Typ der Variable v als $Type[v]$. Freie Variablen können als Kantenausdruck, Typen und Wächterfunktionen verwendet werden. Die Variablen in der Abbildung 4.1 sind wie folgt definiert:

$$Var[e] = \begin{cases} \{n\} & \text{if } e = n \\ \{x\} & \text{if } e = x \\ \{n, x\} & \text{if } e = (n, x) \end{cases}$$

4.4. Typvariablen

Variablen müssen einem Typ zugeordnet werden. V ist eine endliche Menge von Typvariablen mit $Type[v] \in \Sigma$ für $\forall\, v \in V$. Die Typvariablen in der Abbildung 4.1 sind wie folgt definiert:

$$V = \{\text{n:ZAHLEN, x:DATEN}\}$$

4.5. Typfunktion

Die Typfunktion $C\colon S \to \Sigma$ ordnet jeder Stelle s einen Typen zu. Die Typfunktion in der Abbildung 4.1 ist wie folgt definiert:

$$C(s) = \begin{cases} \text{ZAHLEN} & \text{if } s = s_1 \\ \text{DATEN} & \text{if } s = s_2 \\ \text{ZAHLENxDATEN} & \text{if } s = s_3 \end{cases}$$

4.6. Wächterfunktion

Die Wächterfunktion $G\colon T \to EXPR_V$ ordnet jeder Transition t einen Wächter $G(t)$ zu, mit $Type[G(t)] = BOOL$. Die Wächterfunktion in der Abbildung 4.1 ist wie folgt definiert:

$$G(t_1) = [n = 1]$$

Falls es keinen expliziten Wächter gibt, wird true zurückgeben.

4.7. Kantenausdrucksfunktion

Die Kantenausdrucksfunktion $E\colon F \to EXPR_V$ weist jeder Kante f eine Kanteausdrucksfunktion zu. Wie bei der Wächterfunktion benötigen wir freie Variablen von $E(f)$, die eine Teilmenge von V sind. Das heißt, dass $E(f) \in EXPR_V$. Für eine Kante $(s,t) \in F$, von der es eine Verbindung von Stelle $s \in S$ zur Transition $t \in T$ gibt, ist es erforderlich, dass der Typ der Kantenausdrucksfunktion ein Multimengentyp von Typ $C(s)$ in der Stelle ist. Das heißt $Type[E(s,t)] = C(s)_{MS}$. Genauso gilt dies anders herum, wenn es eine Kante $(t,s) \in A$ gibt, dann muss sie ebenso $Type[E(t,s)] = C(s)_{MS}$.
Die Kantenausdrucksfunktion in der Abbildung 4.1 ist wie folgt definiert:

$$E(f) = \begin{cases} 1'n & \text{if } f = (s_1, t_1) \\ 1'x & \text{if } f = (s_2, t_1) \\ 1'(n,x) & \text{if } f = (t_1, s_3) \end{cases}$$

4.8. Initialisierungsfunktion

Die Initialisierungsfunktion $I\colon S \to EXPR_{\emptyset}$ ordet jedem $s \in S$ einen initialen Ausdruck $I(s)$ zu. Dieser wertet zu einer Multimenge über $C(s)$ aus, das heißt $Type[I(s)] = C(s)_{MS}$. Es ist zu beachten, dass $I(s)$ keine Variablen enthält. Die Initialisierungsfunktion in der Abbildung 4.1 ist wie folgt definiert:

$$I(s) = \begin{cases} 2'1 & \text{if } s = s_1 \\ 1'(''Hello'') +\!\!+ 1'(''World'') & \text{if } s = s_2 \\ \emptyset_{MS} & otherwise \end{cases}$$

Die entsprechenden Ausdrücke stehen in Abbildung 4.1 neben den Stellen, sind aber nicht grün unterlegt.

4.9. Markierungen

Eine Markierung ist eine Abbildung M, die jeder Stelle s eine Multimenge aus $C(s)_{MS}$ zuordnet. Die initiale Markierung eines gefärbten Petrinetzes ist definiert durch $M_0(s) = I(s)\langle\rangle$ für jede Stelle s. Das heißt, $M_0(s)$ besteht aus der durch $I(s)$ spezifizierten Multimenge.

Da wir nun alle Komponenten eines CPN beschrieben haben, wenden wir uns nun einem graphischen Beispiel zu.

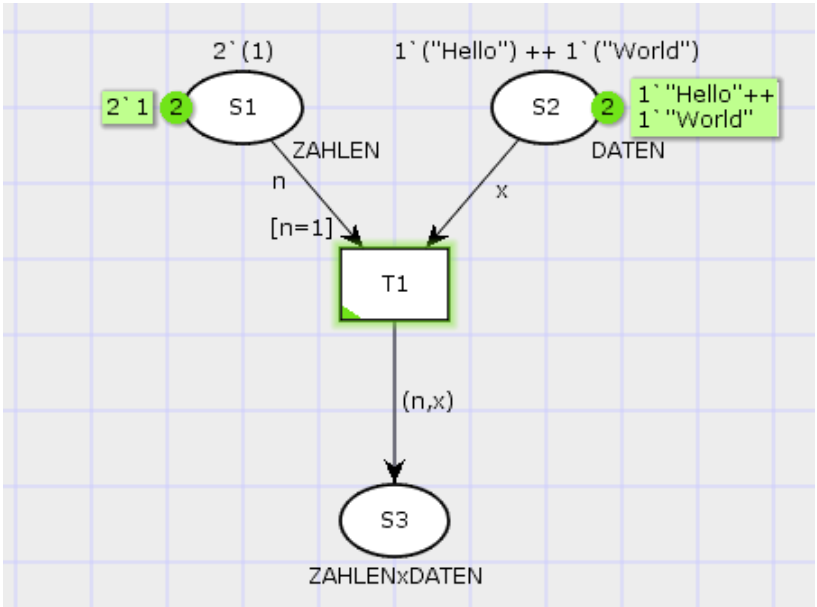


Abbildung 4.1.: Gefärbtes Petri-Netz

In der Abbildung 4.1 sind die Typen $\Sigma = \{ZAHLEN,\ DATEN,\ ZAHLEN \times DATEN\}$ jeweils einer Stelle zugeteilt. Nebenbei erkennt man gut, dass die Typvariablen an die Kanten geschrieben werden. Der Wächter besitzt wie erwartet die extra Bedingung für t_1, $n = 1$. Zudem erkennt der Leser die initiale Markierung M_0 in dem Text über den Stellen s_1 und s_2, die sich toolbedingt nicht ändern. In der Stelle s_1 sind zwei Marken mit dem Inhalt *1* und in der Stelle s_2 befinden sich ebenfalls zwei Marken, einmal $''Hello''$ und einmal $''World''$. In den grünen Felder sind die Markierungen, die sich beim Schalten verändern. Die Zahl am Rand einer Stelle gibt die Anzahl der Marken wieder.

4.10. Schalten

Bevor wir mit dem Thema Schalten anfangen können, müssen wir noch die *Variablen einer Transition* $Var(t)$ definieren, wobei $Var(t) \subseteq V$. Hierzu zählen alle freien Variablen, die in ihrem Wächter vorhanden sind und alle Variablen, die in den Ausdrücken derjenigen Kanten stehen und mit der Transition verbunden sind. In unserem Beispiel wären die Variablen der Transition t_1 folgende:

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

Eine *Belegung einer Transition* t ist eine Funktion b, die von jeder Variable $v \in Var(t)$ in einen Wert $b(v) \in Type[v]$ abbildet. Die Menge aller Belegungen von t wird als $B(t)$ definiert. Darüber hinaus können wir nun das *Belegungselement* definieren. Ein Belegungselement ist ein Paar (t, b), wobei $t \in T$ und $b \in B(t)$. Die Menge aller Belegungselemente $BE(t)$ ist definiert durch $BE(t) = \{(t, b) \mid b \in B(t)\}$. BE bezeichnet die Menge aller Belegungselemente. Nun haben wir die erforderlichen Mittel um Schalten zu definieren.

Ein Belegungselement $(t, b) \in BE(t)$ ist *aktiviert* in einer Markierung M, wenn folgende Eigenschaften erfüllt sind.

- $G(t)\langle b\rangle$.
- $\forall s \in S\colon E(s,t)\langle b\rangle \ll= M(s)$.

Falls M durch Schalten von t zu M' kommt, geschrieben $M[t\rangle M'$. Dann würde die M' wie folgt aussehen:

$$M'(s) = \begin{cases} M(s) - -E(s,t)\langle b\rangle, & \text{falls } s \in {}^\bullet t \backslash t^\bullet, \\ M(s) +\!+ E(s,t)\langle b\rangle, & \text{falls } s \in t^\bullet \backslash {}^\bullet t, \\ M(s) - -E(s,t)\langle b\rangle +\!+ E(s,t)\langle b\rangle, & \text{falls } s \in t^\bullet \cap {}^\bullet t, \\ M(s) & \text{sonst} \end{cases}$$

Die Auswertung vom Belegungselement (t, b) wird als $G(t)\langle b\rangle$ und das Ergebnis der Kantenausdrucksfunkton $E(f)$ für eine Kante f wird als $E(f)\langle b\rangle$ geschrieben. In der ersten Bedingung muss die Wächterfunktion $G(t)\langle b\rangle = true$ ergeben und in der zweiten Bedingung wird das Ergebnis $E(s,t)\langle b\rangle$ von einem Ausdruck $E(s,t)$, welcher an der Kante (s,t) steht, mindestens so viele Marken in der Stelle s verlangen, wie in der Kantenausdrucksfunktion beschrieben. Nachdem Schalten werden die erforderlichen Marken, die in der Kantenausdrucksfunktion (s,t) stehen, aus der Stelle s entfernt und in die entsprechenden Stellen, die in der Kantenausdrucksfunktion (t,s) aufgeführt sind, mit Marken gefüllt.

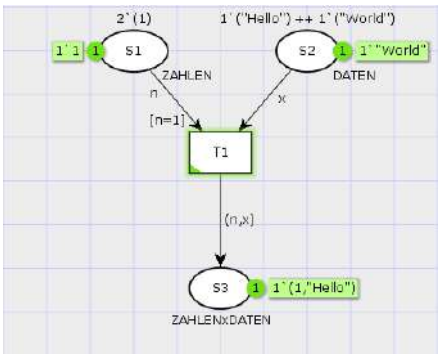


Abbildung 4.2.: Gefärbtes Petri-Netz - t_1 geschaltet - Variante 1

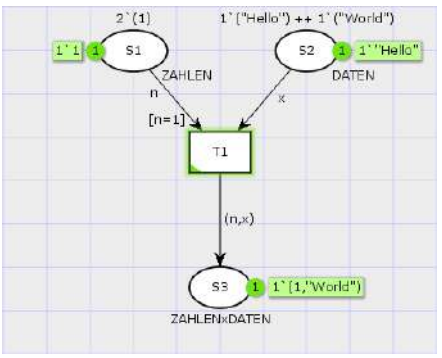


Abbildung 4.3.: Gefärbtes Petri-Netz - t_1 geschaltet - Variante 2

In dem gefärbten Petrinetz aus Abbildung 4.1 ist die Transition t_1 aktiviert, da sie einmal $G(t)\langle b\rangle$ erfüllt. Das heißt, die Variable $n = 1$ und es wird genau ein Element x aus DATEN und n aus ZAHLEN verlangt. Das eine Transition in CPN-Tools aktiviert ist, erkennt man an der grünen Umrandung an der Transition. Beim Schalten von t_1 gibt es jetzt zwei Möglichkeiten, da die Marken zufällig aus der Stelle entfernt werden. In der Stelle s_1 wird sicherlich eine 1 entfernt, aber in Stelle s_2 gibt es die Möglichkeit das einmal "$Hello$" bzw. "$World$" entfernt wird, was in den Abbildung 4.2 und 4.3 gut zu erkennen ist.

Kapitel 5.

TIC-TAC-TOE in CPN-Tools

Nachdem wir die gefärbten Petrinetze vorgestellt haben, werden wir nun TIC-TAC-TOE in einem gefärbten Petrinetze erstellen. Dabei verwenden wir das Programm CPN-Tools, das Kurt Jensen in seinem Buch [JK09] zur Modellierung genutzt hat. In der Modellierung von Stellen/Transition-Systemen hat sich herausgestellt, das TIC-TAC-TOE schnell unübersichtlich wurde. Nun verwenden wir gefärbte Petrinetze, die mehr Komponenten zu bieten haben, als beim einem Stellen/Transitions-System und überprüfen nun, ob sich gefärbte Petrinetze besser eignen.

5.1. CPN-Tools

CPN-Tools ist ein Werkzeug, um gefärbte Petrinetze zu erstellen, simulieren und analysieren. Das Tool verfügt über eine inkrementelle Syntaxübereinstimmung und Codegenerierung, die während der Erstellung stattfindet. Es gibt die Möglichkeit Erreichbarkeitsgraphen zu erzeugen und zu analysieren. Außerdem lassen sich Eigenschaften prüfen, ob ein gefärbtes Petrinetz beschränkt oder lebendig ist (vgl. [PW03] S.85 - 87).
Beim Schalten einer Transition t wird durch einen Nichtdeterminismus zufällig eine Marke entfernt. Somit können beim Schalten t verschiedene Markierungen entstehen.

5.2. TIC-TAC-TOE in CPN-Tools

Wie zuvor zerlegen wir die Modellierung des Spiels in kleinere Teile, damit wir die Übersicht des Petri-Netzes nicht verlieren. Dabei betrachten wir die ersten zwei Felder des TIC-TAC-TOE Spiels, was in Abbildung 5.1 zu sehen ist.
Dafür nehmen wir uns ein gefärbtes Petri-Netz ohne Wächterfunktion aber mit Inhibitorkanten $CPN_0 = (S, T, F, \Sigma, V, C, E, I, inh)$.

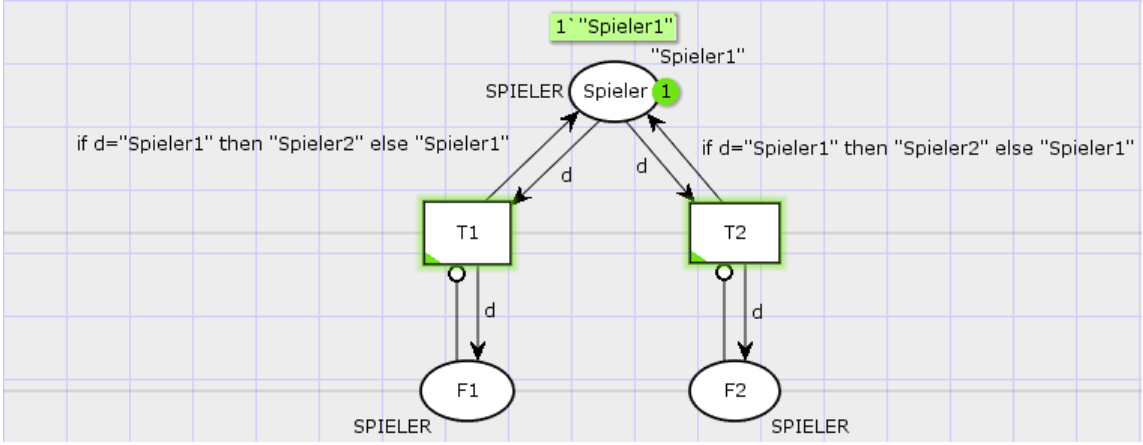


Abbildung 5.1.: TIC-TAC-TOE 2 Felder in CPN-Tools

Bei der Modellierung in gefärbten Petri-Netzen gibt es schon einen großen Unterschied zur Modellierung in Stellen/Transitions-Systeme, da wir nicht mit anonymen Marken arbeiten.

Somit brauchen wir keine Stelle für den zweiten Spieler. Durch die Kantenausdrucksfunktion an den Kanten $(t_1, Spieler)$ und $(t_2, Spieler)$ wird sichergestellt, dass die Spieler abwechselnd am Zug sind.

Das bedeutet, falls $''Spieler1''$ beispielsweise durch das Schalten von t_1 in F_1 geschrieben wird, wird in die Stelle Spieler $''Spieler2''$ geschrieben. Dies ist in Abbildung 5.2 zu sehen. Genauso gilt dies anders herum, falls $''Spieler2''$ in t_1 geschrieben wird, endet die if-Klausel in dem *else*-Zweig und es wird $''Spieler1''$ in Spieler geschrieben.

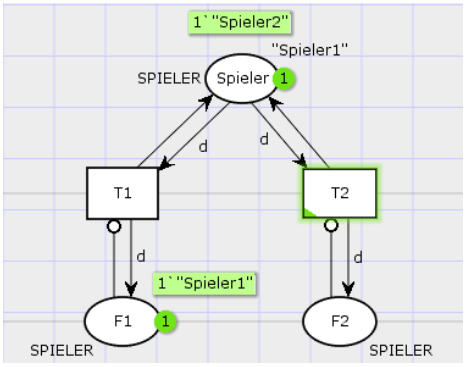


Abbildung 5.2.: TIC-TAC-TOE 2 Felder - t_1 geschaltet

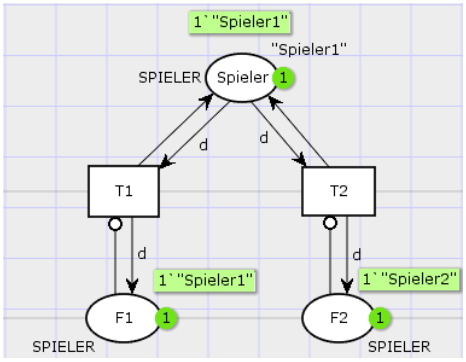


Abbildung 5.3.: TIC-TAC-TOE 2 Felder - t_1t_2 geschaltet

Auf der Abbildung 5.2 und 5.3 wurde die if-Klausel aus Platzgründen weggelassen. Nachdem t_1 gefeuert wurde, liegt in der Stelle $Spieler$ eine Marke mit $''Spieler2''$. Nun kann durch Schalten von t_2 eine Marke Spieler 2 in das Feld F_2 geschrieben werden. Somit können die Symbole viel einfacher unterschieden werden, als in einem Stellen/Transitions-System.

Bevor wir zu neun TIC-TAC-TOE Feldern kommen, definieren wir weitere Mengen, wie bei den Stellen/Transitions-Systemen. Die heißen $Zuege$ und $Felder$, wobei:

- $Zuege = \{Z_j \mid j \in [9]\}$
- $Felder = \{F_j \mid j \in [9]\}$

Nachdem wir das TIC-TAC-TOE Spiel für zwei Felder modelliert haben, entwickeln wir nun ein gefärbtes Petrinetz mit neun Feldern. Die folgende Abbildung 5.4 entspricht der gleichen Funktionalität des Stellen/Transitions-System in der Abbildung 3.5. Die Felder $\{F_1, ..., F_9\}$ entsprechen den TIC-TAC-TOE Feldern. Sie wurden graphisch in einer Reihe angeordnet, damit die Übersicht erhalten bleibt. Außerdem wurde die Kantenausdrucksfunktion nur einmal in der Abbildung 5.4 verwendet. Trotzdem steht die if-Klausel für alle Kanten $\{(Z_j, Spieler) \mid j \in [9]\}$.

Hierfür nehmen wir wieder ein gefärbtes Petrinetz $CPN_1 = (S, T, F, \Sigma, V, C, E, M_0, inh)$.

- $S = \{Spieler\} \cup Felder$
- $T = Zuege$
- $F = \{(Spieler, Z_j) \mid j \in [9]\} \cup \{(Z_j, Spieler) \mid j \in [9]\} \cup \{(Z_j, F_j) \mid j \in [9]\}$
- $\Sigma = \{SPIELER\}$
- $V = \{d : SPIELER\}$
- $C(s) = SPIELER$, mit $s \in S$

$$\bullet\ E(f) = \begin{cases} 1'd & \text{if } f \in \{(Spieler, Z_j), (Z_j, F_j) \mid j \in [9]\} \\ if\ d =''\ Spieler1''\ then & \text{if } f \in \{(Z_j, Spieler) \mid j \in [9])\} \\ ''Spieler2''\ else\ ''Spieler1'' & \end{cases}$$

$$\bullet\ I(s) = \begin{cases} 1'(''Spieler1'') & \text{if } s = Spieler \\ \emptyset_{MS} & otherwise \end{cases}$$

- $inh = \{(F_j, Z_j) \mid j \in [9]\}$

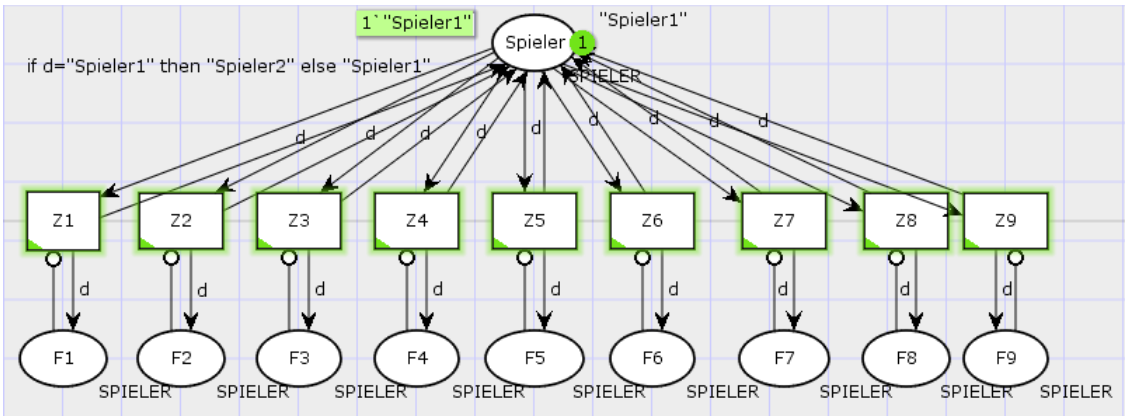


Abbildung 5.4.: TIC-TAC-TOE 9 Felder in CPN-Tools

Das gefärbte Petrinetz in Abbildung 5.4 benötigt keine extra Züge für Spieler 2, somit sparen wir neun Transition. Dadurch halbiert sich die Anzahl an Kanten von 72 Kanten in Abbildung 3.5 auf 36.

Für die Gewinnbedingung benötigen wir ebenfalls nur noch die Hälfte an Transitionen, da die Information, welcher Spieler gewonnen hat, in der Markierung steht. Dadurch entfallen zusätzlich die Stellen Auslagerung, die wir in der Abbildung 3.6 benötigen.

Die Abbildung 5.5 zeigt die gleiche Funktionalität wie das Stellen/Transitions-System in Abbildung 3.6. G_1 übernimmt in die Aufgaben von G_{11}, G_{21} und A_1. Die Kante $(Spieler, G_1)$ besitzt die Kantenausdrucksfunkiton $if\ d = ''Spieler1''\ then\ ''Spieler2''\ else\ ''Spieler1''$, da beim Schalten der Gewinnsituation der Gegner am Zuge ist. Somit würde G_1 nicht aktiviert sein, wenn beispielsweise in der Stelle F_1, F_2 und F_3 $''Spieler1''$ liegen würde, da trotzdem in Spieler $''Spieler2''$ steht. Deshalb müssen $''Spieler1''$ und $''Spieler2''$ getauscht werden. Mit dieser Kante wird wieder sichergestellt, dass das gefärbte Petrinetz nach dem Schalten von G_1 in einem Deadlock landet. Mithilfe einer Schaltfolge $w \in T^*$ zeigen wir die drei Anwendungsfälle.

Nach der Schaltfolge $w_4 = Z_1 Z_4 Z_2 Z_5 Z_3 G_1$ hat Spieler 1 gewonnen.

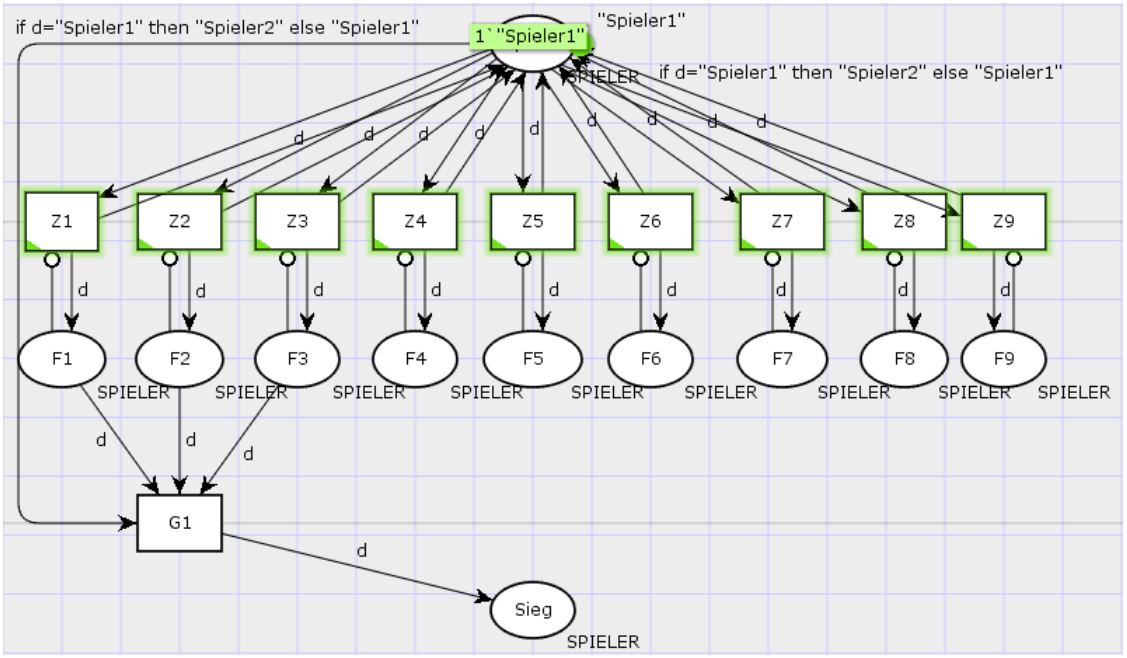


Abbildung 5.5.: Umsetzung einer Gewinnmöglichkeit

Nach der Schaltfolge $w_5 = Z_7Z_1Z_8Z_2Z_4Z_3G_1$ hat Spieler 2 gewonnen.

Nach der Schaltfolge $w_6 = Z_5Z_1Z_4Z_6Z_7Z_3Z_2Z_9$ hat keiner gewonnen.

In diesen Anwendungsbeispielen erkennt man nicht mehr an der letzten geschalteten Transition welcher Spieler gewonnen hat, da G_1 unabhängig aktiviert sein kann, egal welcher Spieler am Zug ist. Spieler 1 hat gewonnen, wenn in der Stelle *Sieg* durch Schalten von $G_1, ..., G_8$ $''Spieler1''$ steht, bzw. Spieler 2 hat gewonnen, wenn in der Stelle *Sieg* $''Spieler2''$ steht. Unentschieden geht das Spiel aus, wenn nach neun Zügen keine Marke in Sieg vorhanden ist. Bei diesem Anwendungsfall endet das Spiel ebenfalls in einem Deadlock, da alle Stellen $\{F_1, ..., F_9\}$ nicht leer sind und die Inhibitorkanten $\{(F_j, Z_j) \mid j \in [9]\}$ verhindert, dass eine Transition $\{Z_1, ..., Z_9\}$ schalten kann.

Da wir wieder das Problem haben, dass G_1 aktiviert ist, aber nicht geschaltet werden muss, verwenden wir Prioritäten in CPN-Tools. In CPN-Tools gibt es standardgemäß drei Prioritäten P_{HIGH}, P_{NORMAL} und P_{LOW}. Doch es gibt die Möglichkeit weitere Prioritäten in CPN-Tools zu deklarieren. Eine Priorität wird in CPN-Tools einer Transition zugewiesen. Falls eine Transition P_{HIGH} zugewiesen wird, besitzt die Transition eine höhere Priorität als alle anderen Transition, die standardgemäß P_{NORMAL} besitzen. Wären $\{t_0, ..., t_{99}\} = T$ und t_3 würde P_{HIGH} zugewiesen bekommen, dann hätte t_3 eine höhere Priorität als alle anderen 99 Transitionen in T, also $t_3 > T \setminus t_3$.

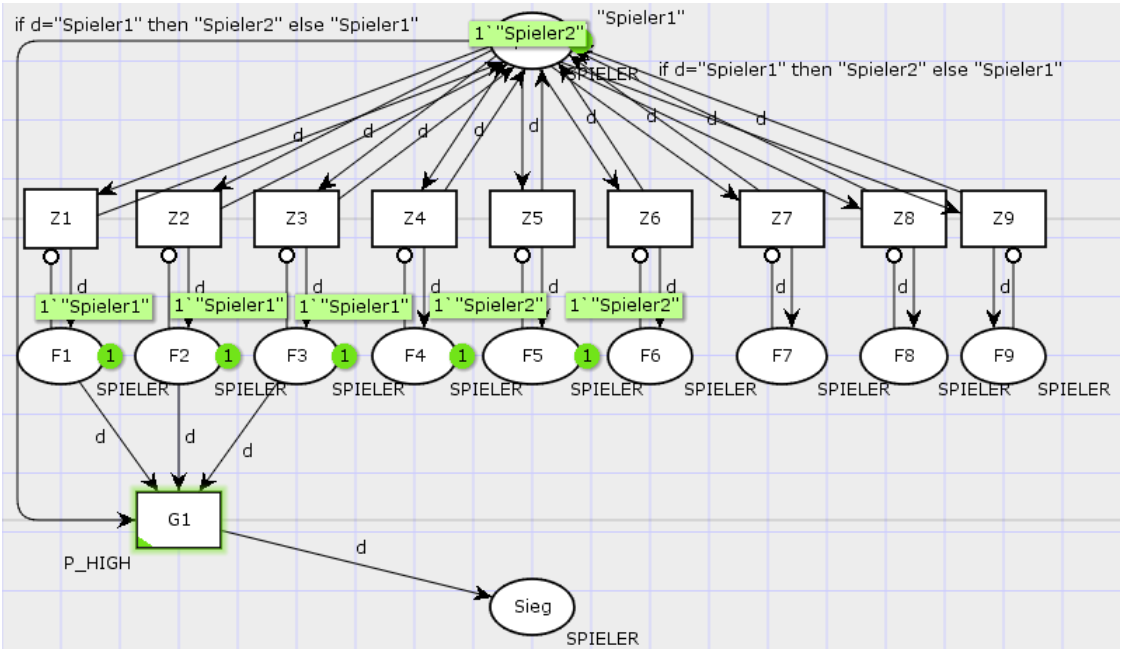


Abbildung 5.6.: Umsetzung einer Gewinnmöglichkeit mit Prioritäten

Die Markierung in Abbildung 5.6 entsteht nach der Schaltfolge $w_7 = Z_1Z_4Z_2Z_5Z_3$. Das ist die Schaltfolge w_4, die Spieler 1 gewinnen lässt, außer das G_1 geschaltet wurde. In dem gefärbten Petrinetz ist nur G_1 aktiviert, was der Leser an der grünen Umrandung erkennen kann. Das liegt daran, dass G_1 eine hohe Priorität zugewiesen wurde, welche man links neben der Transition G_1 in der Abbildung 5.6 erkennt. Damit wird erzwungen, dass G_1 als einzige Transition aktiviert ist. Nach dem Schalten von G_1 wird eine Marke $''Spieler1''$ in die Stelle Sieg gelegt und Spieler 1 hat gewonnen.

Diese Gewinnbedingung muss auf alle anderen sieben Gewinnmöglichkeiten übertragen werden. Die sieben anderen Gewinntransition $\{G_2, ..., G_8\}$ müssen ebenfalls eine höhere Priorität besitzen als die Züge. In diesem Fall weisen wir in CPN-Tools alle Gewinnmöglichkeiten P_{HIGH} zu.

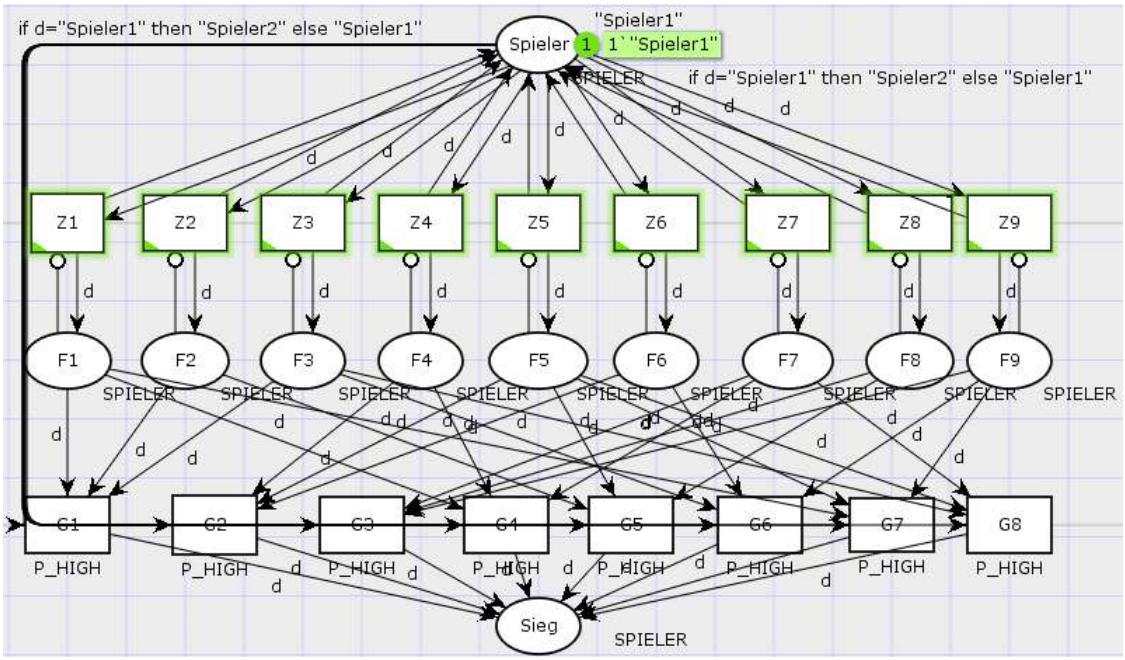


Abbildung 5.7.: TIC-TAC-TOE vollständig mit 8 Gewinntransitionen

In Abbildung 5.7 sind alle acht Gewinnsituationen eingebaut und das gefärbte Petrinetz besitzt alle Funktionalitäten des TIC-TAC-TOE-Spiels, also die Funktionalitäten des Stellen/Transitions-Systems in Abbildung 3.8. Trotzdem ist der Bereich zwischen den Feldern und den Gewinnsituationen sehr unübersichtlich und wir optimieren das gefärbte Netz mithilfe eines Wächters. Mit diesem Wächter können wir alle acht Transitionen in eine Transition überführen. Im ersten Schritt entfernen wir die Inhibitorkanten $\{(F_j, Z_j) \mid j \in [9]\}$ und führen unsere Menge $XOR = \{X_1, ..., X_9\}$ wieder ein, die auf der Abbildung 3.3 zu sehen ist. XOR wird mit einem leeren String $''\,''$ gefüllt und wir verbinden XOR mit den Zügen. Mithilfe einer neuen Typvariable $V = \{a : Spieler\}$ können wir sicher gehen, dass die Transitionen $\{Z_1, ..., Z_9\}$ trotzdem aktiviert sind, obwohl in Spieler $''Spieler1''$ steht. Danach legen wir in den Feldern 8 unterschiedliche Strings in die Felder. In unserem Fall entscheiden wir uns für die Strings $''a'',''b'',''c'',''d'',''e'',''f'',''g'',''h'',''i''$. Die initiale Markierung in den Stellen $\{F_1, ..., F_9\}$ ist uninterssant. Sie dienen nur dazu, dass das Spiel beendet werden kann, obwohl nicht alle neun Züge ausgeführt wurden.

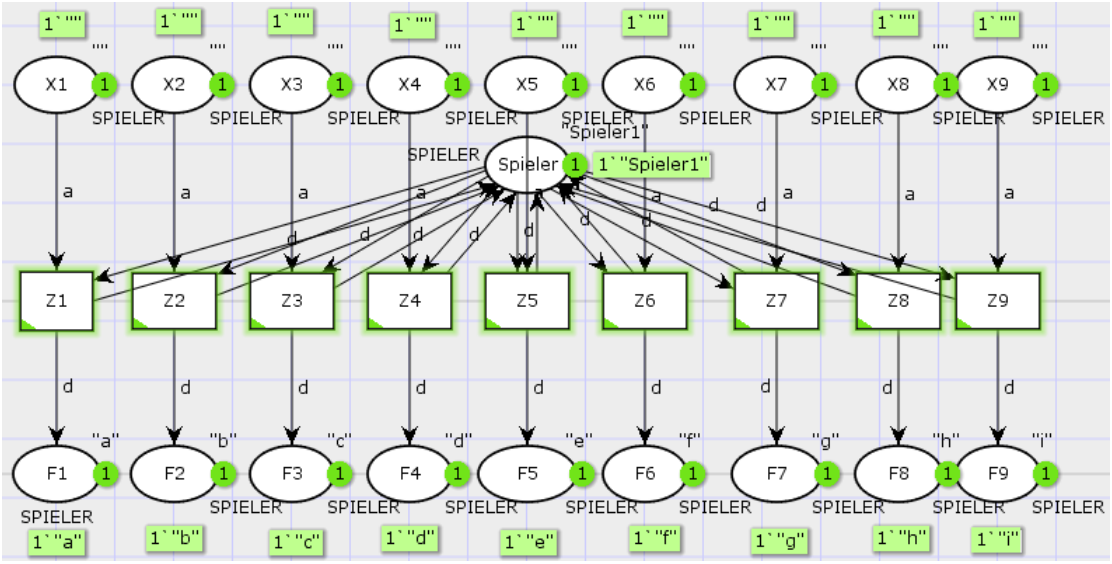


Abbildung 5.8.: TIC-TAC-TOE mit zwei Typvariablen

In Abbildung 5.8 erkennen wir, dass alle Transitionen $\{Z_1, .., Z_9\}$ aktiviert sind. Nun führen wir für jede Zelle in dem 3x3 Feld eine weitere Typvariable ein und vergleichen sie miteinander. Somit brauchen wir für die Zellen insgesamt neun Typvariablen, die alle dem Typ Spieler zugewiesen werden. Wir nennen sie $a, b, c, d, e, f, g, h, i$. Die Kanten $\{(F_j, Gewinn) \mid j \in [9]\}$ bekommen alle unterschiedliche Typvariablen. Zusätzlich brauchen wir noch eine weitere Typvariable $\{j : SPIELER\}$, die uns sagt, welcher Spieler am Ende gewonnen hat. Sie wird die Kantenausdrucksfunktion an der Kante $(Spieler, Gewinn)$, denn wir wissen, der Spieler der gewonnen hat, steht nicht in der Stelle $Spieler$. In Abbildung 5.9 sehen wir nun, dass in der Stelle $Sieg$ j negiert wird. Das heißt, wenn $''Spieler1''$ in der Stelle $Spieler$ steht, muss im nächsten Schritt $''Spieler2''$ in $Sieg$ stehen.

Die Wächterfunktion von Gewinn vergleicht die Einträge in den Stellen bzw. Zellen untereinander und überprüft, ob ein Spieler eine Gewinnbedingung erfüllt hat. Beispielsweise wird in der ersten Zeile der Wächterfunktion abgefragt, ob $a = b = c$ ist. Das heißt, ob die obere erste Reihe drei gleiche Symbole besitzt. Falls das stimmt, wissen wir, dass Spieler 1 oder Spieler 2 gewonnen hat. Derjenige Gewinner wird aus j ermittelt. Somit besitzen wir ein kompakte und übersichtliche Gewinntransition, die durch eine if-Klausel alle acht Gewinnmöglichkeiten auffängt.

Hierfür nehmen wir wieder ein gefärbtes Petrinetz mit Prioritäten
$CPN_2 = (S, T, F, \Sigma, V, C, G, E, I, >)$

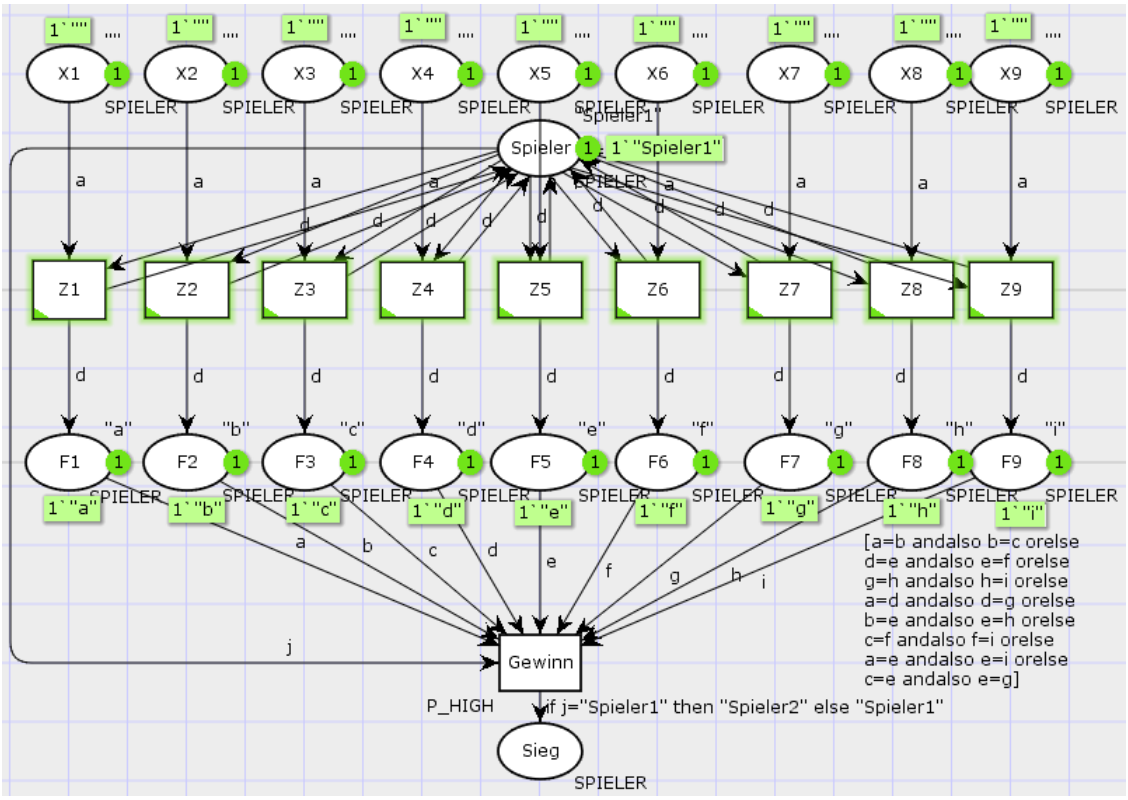


Abbildung 5.9.: TIC-TAC-TOE vollständig mit Wächterfunktion

Die Komponenten entsprechen der Abbildung 5.9.

- $S = XOR \cup \{Spieler\} \cup Felder \cup \{Sieg\}$
- $T = Zuege \cup \{Gewinn\}$
- $F = \{\{(X_j, Z_j) \mid j \in [9]\} \cup \{(\{Spieler\}, Z_j) \mid j \in [9]\} \cup \{(Z_j, \{Spieler\}) \mid j \in [9]\} \cup \{(Z_j, F_j) \mid j \in [9]\}, \{(F_j, Gewinn) \mid j \in [9]\} \cup (Gewinn, Sieg) \cup ((Spieler, Gewinn)\}$
- $\Sigma = \{SPIELER\}$
- $V = \{a : SPIELER, b : SPIELER, c : SPIELER, d : SPIELER, e : SPIELER, f : SPIELER, : g : SPIELER, h : SPIELER, i : SPIELER, j : SPIELER\}$
- $C(s) = SPIELER$, mit $s \in S$
- $G(Gewinn) = [a = b$ andalso $b = c$ orelse $d = e$ andalso $e = f$ orelse $g = h$ andalso $h = i$ orelse $a = d$ andalso $d = g$ orelse $b = e$ andalso $e = h$ orelse $c = f$ andalso $f = i$ orelse $a = e$ andalso $e = i$ orelse $c = e$ andalso $e = g]$

- $E(f) =$

$$\begin{cases} 1'a & \text{if } f \in \{(X_j, Z_j) \mid j \in [9], (F_1, Gewinn)\} \\ 1'b & \text{if } f = (F_2, Gewinn) \\ 1'c & \text{if } f = (F_3, Gewinn) \\ 1'd & \text{if } f \in \{(Spieler, Z_j) \mid j \in [9], (Z_j, F_j) \mid j \in [9], (F_4, Gewinn)\} \\ 1'e & \text{if } f = (F_5, Gewinn) \\ 1'f & \text{if } f = (F_6, Gewinn) \\ 1'g & \text{if } f = (F_7, Gewinn) \\ 1'h & \text{if } f = (F_8, Gewinn) \\ 1'i & \text{if } f = (F_9, Gewinn) \\ 1'j & \text{if } f = (Spieler, Gewinn) \\ \text{if d=”Spieler1” then ”Spieler2” else ”Spieler1”} & \text{if } f \in \{(Z_j, Spieler) \mid j \in [9]), (Gewinn, Sieg)\} \end{cases}$$

- $I(s) =$

$$\begin{cases} 1'(\text{''}\,\text{''}) & \text{if } s \in \{X_1, ..., X_9\} \\ 1'(\text{''}a\text{''}) & \text{if } s = F_1 \\ 1'(\text{''}b\text{''}) & \text{if } s = F_2 \\ 1'(\text{''}c\text{''}) & \text{if } s = F_3 \\ 1'(\text{''}d\text{''}) & \text{if } s = F_4 \\ 1'(\text{''}e\text{''}) & \text{if } s = F_5 \\ 1'(\text{''}f\text{''}) & \text{if } s = F_6 \\ 1'(\text{''}e\text{''}) & \text{if } s = F_7 \\ 1'(\text{''}g\text{''}) & \text{if } s = F_8 \\ 1'(\text{''}h\text{''}) & \text{if } s = F_9 \\ \emptyset_{MS} & \text{otherwise} \end{cases}$$

- $> = \{Gewinn > T \setminus \{Gewinn\}\}$

Nun wollen wir die Petrinetze miteinander vergleichen. Zuerst behandeln wir das Stellen/ Transition-System in der Abbildung 3.8. Wenn wir von diesem Stellen/Transition-System den Erreichbarkeitsgraph erstellen, gibt es 6420 Zustände. Das Programm Tina bietet nur eine kleine Anzahl von Analysetools wie die Erstellung eines Erreichbarkeitsgraphen oder Überdeckungsgraphen. Zusätzlich bietet Tina einen *stepper simulator* an, mit dem wir Transitionen interaktiv schalten können.

Wenn wir aber die gefärbten Petrinetze in den Abbildungen 5.7 und 5.9 angucken, lassen sich einige Eigenschaften zeigen. Das Programm CPN-Tools bietet nicht nur Erreichbarkeitsgraphen an, sondern es lassen sich auch Grenzen, Deadlocks, tote Transitionen und lebendige Transitionen (vgl. [PW03] S.86) mithilfe der *state space* Analyse erstellen.

In Tina und CPN-Tools kann der Erreichbarkeitsgraph nur erstellt werden, wenn die Petrinetze beschränkt sind. Falls die Petrinetze nicht beschränkt sind und somit der Erreichbarkeitsgraphen unendlich ist, hört Tina nicht auf zu rechnen und die Analyse muss abgebrochen werden. Bei CPN-Tools wird nach die Berechnung nach fünf Minuten automatisch abgebrochen und das Programm gibt eine Fehlermeldung zurück.

Graphische Erreichbarkeitsgraphen können bei CPN-Tools nur per Mausklick von Zustand zu Zustand ausgerollt werden. Das heißt, zu Anfang wird nur die initale Markierung angezeigt und wir können nur einzeln per Mausklick den Nachfolgezustand anzeigen lassen.

Wenn wir in den beiden gefärbten Petrinetzen in Abbildung 5.7 und 5.9 Kanten von den Gewinnmöglichkeiten zurück zu den Feldern legen, also beispielsweise bei Abbildung 5.9 die Flussrelation um $\{(Gewinn, F_j) \mid j \in [9]\}$ erweitern, so dass sich die Markierungen in den Feldern nicht ändert, wenn eine Gewinnmöglichkeit geschaltet wird, erhalten wir bei beiden Petrinetzen ebenfalls 6420 Zustände. Somit können wir vermuten, das die Petrinetze die gleiche Funktionalität haben. Für den nächsten Schritt müssten wir einen Korrektheitsbeweis einführen, um sicher zu stellen, dass die Petrinetze auch die gleiche Funktionalität haben, wie das TIC-TAC-TOE Spiel doch das würde den Rahmen dieser Arbeit sprengen.

Die Behauptung, dass das TIC-TAC-TOE Spiel 6420 Zustände besitzt, ist falsch, da wir extra Gewinntransitionen haben. Wenn wir nun aber das gefärbte Petrinetz um eine Unentschieden-Transition erweitern, so dass das Spiel nach neun Zügen unentschieden ausgeht, erhalten wir 16 Zustände mehr und besitzen 6436 Zustände. Nun wissen wir schon das es 16 mögliche Unentschieden Zustände gibt, was in Abbildung 5.10 zu sehen ist. Die Wächterfunktion von Unentschieden überprüft, ob in den Stellen $\{F_1, ..., F_9\}$ $''Spieler1''$ oder $''Spieler2''$ steht.

Nun können wir die Deadlocks von den Zuständen abziehen und erhalten die möglichen Zustände in TIC-TAC-TOE, also 6436 - 958 = 5478. In verschiedenen Quellen wie [HMO99] S.190 und [GSS14] S.627 wurden mathematisch die möglichen Zustände in einem TIC-TAC-TOE Spiel berechnet und erhalten ebenfalls 5478 Zustände. Das ist ein starker Indiz, dass die Funktionalität der Petrinetze den Spielregeln von TIC-TAC-TOE entsprechen.

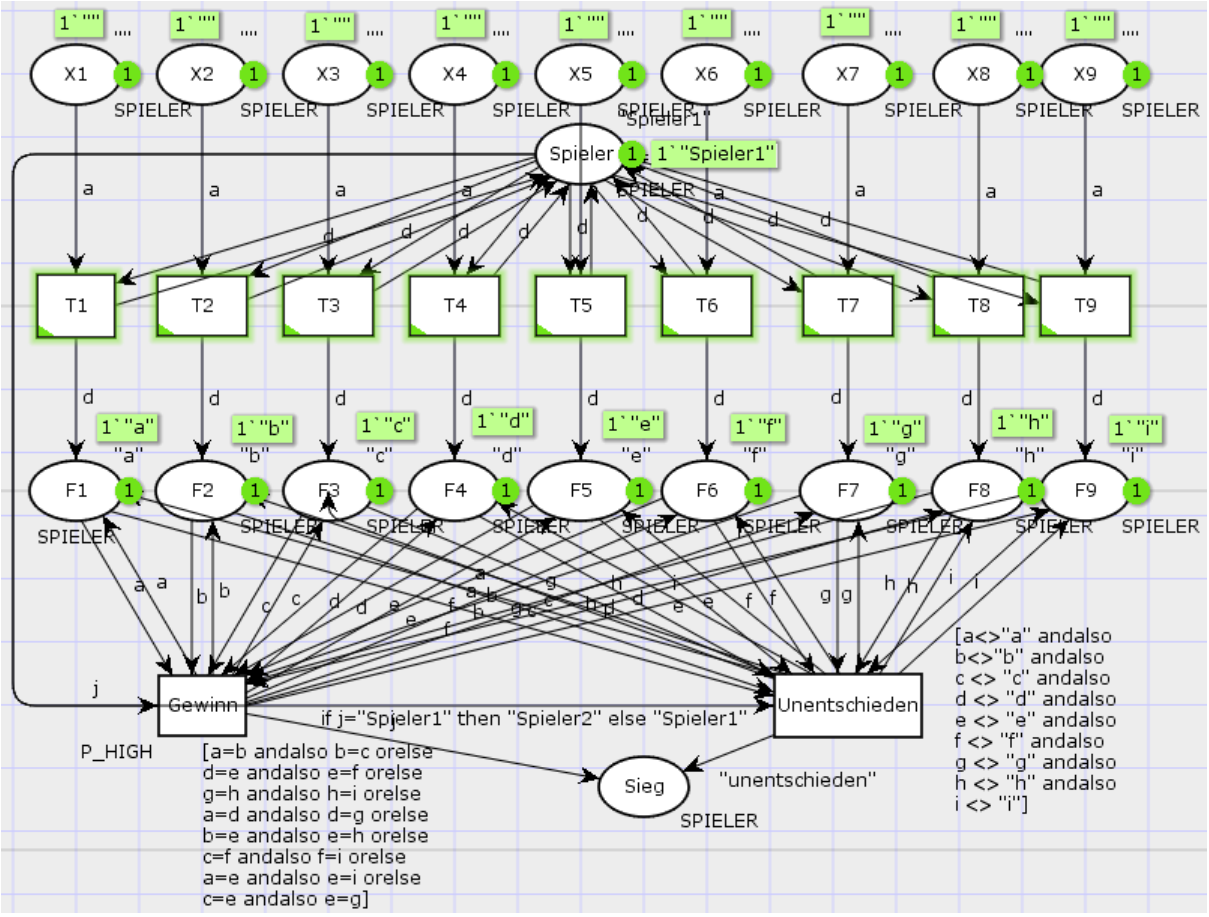


Abbildung 5.10.: Unentschieden-Transition

Neben den 16 Zuständen für unentschieden, interessieren wir uns für die möglichen Gewinnzustände des TIC-TAC-TOE-Spiels. Damit wir die möglichen Gewinnzustände für Spieler 1 heraus finden, ändern wie die Unentschieden-Transition aus der Abbildung 5.10 in eine zweite Gewinntransition $Gewinn_2$. Bei der zweiten Gewinntransition entfernen wir die Kante $(Gewinn_2, Sieg)$ und ersetzen sie durch $(Gewinn_2, Spieler)$. Nun landen wir beim Schalten von $Gewinn_2$ wieder in dem gleichen Zustand. Die Wächterfunktion von $Gewinn$, die in Abbildung 5.11 zu sehen ist, wird um eine Bedingung $j =″ Spieler2″$ erweitert. So wird sichergestellt, dass $Gewinn$ nur geschaltet werden kann, wenn Spieler 1 gewinnt. Ebenso wird bei $Gewinn_2$ die Wächterfunktion von $Gewinn$ übernommen und anstatt $j =″ Spieler2″$, wird die Bedingung $j =″ Spieler1″$ hinzugefügt. Nun wird sichergestellt, dass wenn Spieler 2 gewinnt immer in dem gleichen Zustand landet.

Nun können wir mithilfe der *state space* Analyse die Deadlocks herausfinden, denn die Deadlocks entsprechen genau den Zuständen in denen Spieler 1 gewonnen hat. Bei der Analyse kam heraus, dass 642 Zustände Deadlocks sind und somit können wir sagen, dass es genau 642 Stellungen gibt, in denen Spieler 1 gewinnen kann.

Wenn wir nun die Bedingungen $j =″ Spieler1″$ und $″j = Spieler2″$ tauschen, werden in $Gewinn$ genau die Zustände in einem Deadlock landen, in denen Spieler 2 gewinnt. Somit können wir wieder mit der *state space* Analyse herausfinden, in wie vielen Stellungen Spieler 2 gewonnen hat. Die Analyse ergab 332 Zustände für Spieler 2. Somit lässt sich vermuten, dass Spieler 1 bessere Chancen zu gewinnen hat, da es für Spieler 1 mehr mögliche Zustände für Siege gibt, als bei Spieler 2.

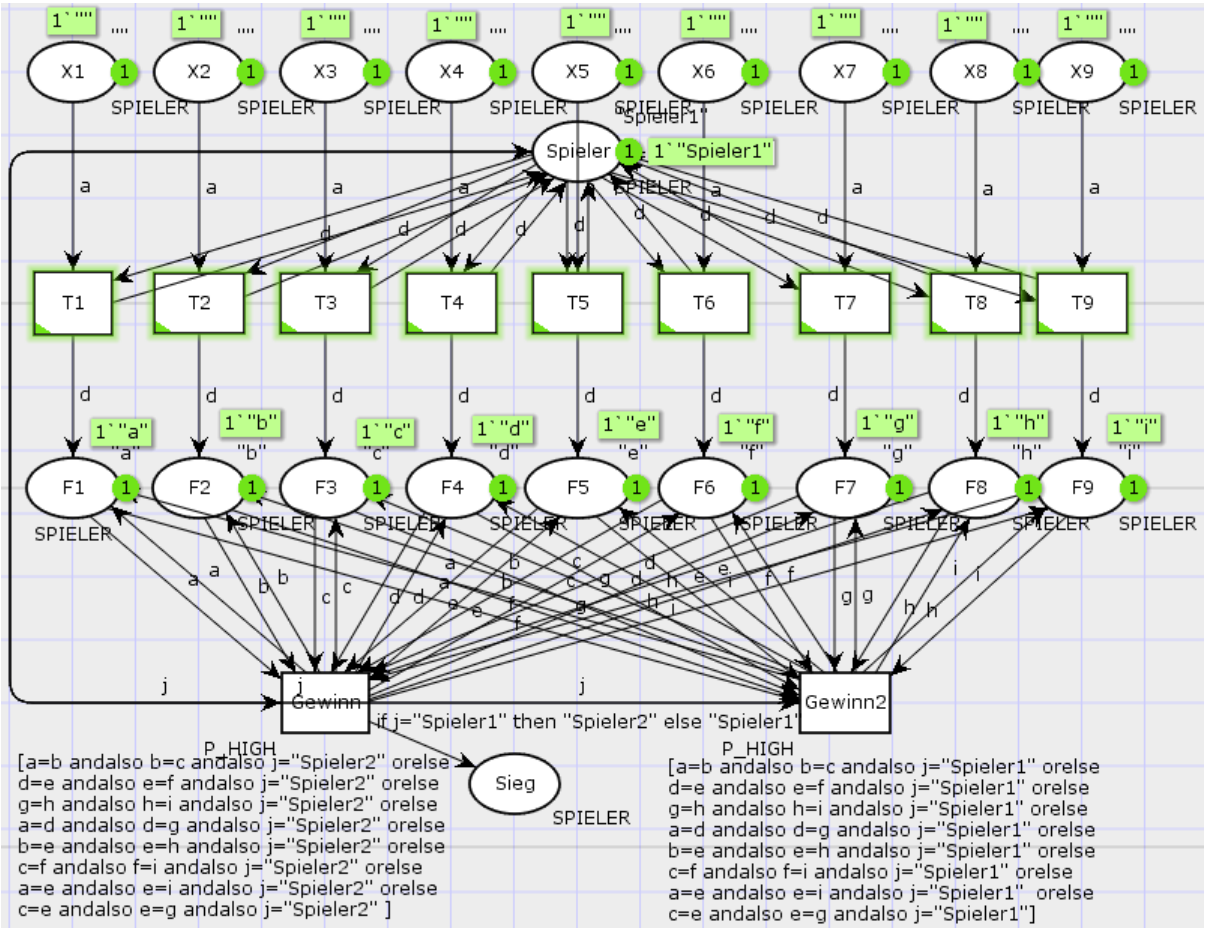


Abbildung 5.11.: Sieg-Transition

In den gefärbten Petrinetzen sind nach der *state space* Analyse keine toten Transitionen vorhanden. Die Petrinetze und die Auswertung der Petrinetze sind auf der CD zu finden.

Kapitel 6.

Fazit

In der Praxis hat sich gezeigt, dass selbst einfache Spiele wie TIC-TAC-TOE in einem Stellen/Transitions-System mit Prioritäten und Inhibitorkanten sehr unübersichtlich wird und die Anzahl der Kanten sehr hoch ist. Auf der einen Seite sind die anonymen Marken das Problem, die eine Unterscheidung von Symbolen wie beim Spiel TIC-TAC-TOE oder auch verschiedene Spielsteine nur sehr schwer realisieren lassen. Auf der anderen Seite vergrößert sich die Anzahl der Transitionen ohne Wächterfunktion sehr schnell, da jede einzelne Möglichkeit mit einer Transition abgedeckt werden muss.

Die gefärbten Petrinetze dagegen fangen diese Probleme auf. Mit Typen können Informationen in den Marken übertragen werden, die das ganze TIC-TAC-TOE-Spiel halbieren. Ebenso kann mithilfe verschiedener Typvariabeln jede einzelne Zelle verglichen werden und wir können mit einer Wächterfunktion die Gewinnbedingungen auf eine Transition reduzieren.

Außerdem besitzt CPN-Tools einige Analysetools, die zur Berechnung mathematischer Eigenschaften beitragen. Da die CPN ML auf einer inkrementellen Programmiersprache (Haskell) basiert, gibt es die Möglichkeit auch Funktionen zu schreiben, die dann als Kantenausdrucksfunktion wieder verwendet werden können. Dadurch ist eine Übertragung des gefärbten Petrinetzes in eine Programmiersprache kein großer Umstand.

In nächsten Schritt sollte ein Korrektheitsbeweis durchgeführt werden, der zeigt, dass das Petrinetz mit den Spielregeln von TIC-TAC-TOE übereinstimmt. Zudem könnte man versuchen das gefärbte Netz mithilfe von Listen weiter zu optimieren und somit Stellen wie $\{X_1, .., X_9\}$ und $\{F_1, .., F_9\}$ auf jeweils eine Stelle zu reduzieren. Zudem könnte man in CPN-Tools mehr Funktionalitäten einbauen, wie beispielsweise eine Anfrage auf dem Erreichbarkeitsgraphen, in welcher Markierung steht welcher Inhalt. So könnten wir die Abbildungen 5.10 und 5.11 sparen, indem wir fragen, in wie vielen Markierungen steht $''Spieler1''$ bzw. $''Spieler2''$ in der Stelle *Sieg*.

Desweiteren wäre eine Modellierung ohne Prioritäten in den gefärbten Petrinetzen angebracht, da nur so eine semantisch äquivalente Umwandlung in ein Stellen/Transition-System möglich ist. Mit dieser Umwandlung könnten wichtige Eigenschaften, wie das Erreichbarkeitsproblem, gelöst werden (vgl. [PW08]).

Hinzukommend ist die Komplexität von TIC-TAC-TOE mit 5748 Zuständen im Vergleich zu Schach verschwindend gering, da nach Schätzungen von 10^{43} bis 10^{120} möglichen Zuständen ausgegangen wird (vgl. [Sha50]). Das heißt, um genauere Vermutungen anzustellen, sollten komplexere Brettspiele ausprobiert werden.

Am Ende lässt sich vermuten, dass Stellen/Transitions-Systeme für Brettspiele nicht geeignet sind, da sie mit komplexen Systemen ohne viel Platzaufwand nicht zu erstellen sind. Gefärbte Petrinetze dagegen können ihren Platz als Model in Spielen finden, da sie logisch wie eine Programmiersprache aufgebaut sind und viele Systeme mithilfe weniger Stellen und Transitionen erstellt werden können. Für weitere Prognosen müssen mehr Spiele in Petri-Netze modelliert werden, damit wir eine aussagekräftige Bewertung abgeben können.

Literatur

[Sha50] Claude E. Shannon. *Programming a Computer for Playing Chess*. Bell Telephone Laboratories, Inc., New York, 1950.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. Technische Hochschule Darmstadt, 1962.

[Rei86] Wolfgang Reise. *Petrinetze - Eine Einführung*. 3-540-16622-X. Springer-Verlag Berlin, 1986.

[Jen92] Kurt Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use - Vol. 1*. 978-3-540-55597-8. Springer-Verlag Berlin, 1992.

[Jen95] Kurt Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use - Vol. 2*. 978-3-540-58276-2. Springer-Verlag Berlin, 1995.

[Bau96] Bernd Baumgarten. *Petri-Netze - Grundlagen und Anwendungen*. 978-3-827-40175-5. Spektrum Akademischer Verlag GmbH, 1996.

[HMO99] Jarkko Hietaniemi, John Macdonald und Jon Orwant. *Mastering Algorithms with Perl*. 1-56592-398-7. O'Reilly & Associates, Inc., 1999.

[PW03] Lutz Priese und Harro Wimmel. *Theoretische Informatik Petri-Netze*. 978-3-540-44289-9. Springer-Verlag Berlin Heidelberg New York, 2003.

[PW08] Lutz Priese und Harro Wimmel. *Petri-Netze*. 978-3-540-76970-5. Springer-Verlag Berlin Heidelberg, 2008.

[JK09] Kurt Jensen und Lars M. Kristensen. *Coloured Petri Nets - Modelling and Validation of Concurrent Systems*. 978-3-642-00283-0. Springer-Verlag Berlin Heidelberg, 2009.

[GSS14] Günther Görz, Josef Schneeberger und Ute Schmid. *Handbuch der Künstlichen Intelligenz*. 978-3-486-71307-7. Oldenbourg Wissenschaftsverlag GmbH, 2014.

Anhang A.

Inhalt der beigefügten CD

Folgender Inhalt befindet sich auf der beigefügten CD:

- Alle verwendeten Stellen/Transitionsysteme und die dazugehörigen Abbildungen in PDF-Format.
  - Die zwei Kapitel dieser Arbeit haben zwei Verzeichnisse mit Kapitelnamen
  - Alle Stellen/Transitions-Systeme haben eine die Abbildungsnummer wie in diesem Dokument. (z.B. Abbildung 2.1 wird zu *Abbildung_2_1.ndr* für Tina und *Abbildung_2_1.pdf* als Grafik)
- Alle verwendeten gefärbten Petrinetze und die dazugehörigen Abbildungen in PNG-Format.
  - Die zwei Kapitel dieser Arbeit haben ein weiteres Verzeichnis mit Kapitelnamen
  - Alle gefärbten Petrinetze haben eine die Abbildungsnummer wie in diesem Dokument. (z.B. Abbildung 4.1 wird zu *Abbildung_4_1.cpn* für CPN-Tools und *Abbildung_4_1.png* als Grafik)
  - Die gefärbten Petrinetze und Stellen/Transitions-Systeme im fünften Kapitel mit denen wir einen Erreichbarkeitsgraphen erstellt haben, enthalten zusätzlich die Analyseergebnisse. Die Analyse der gefärbten Petrinetze wird *Abbildung_X_Y_Zustand*.txt benannt. Die Analyse des Stellen/Transitons-System wird STS_6420.png benannt.
- Der vollständige Erreichbarkeitsgraph von dem Stellen/Transitions-System in Abbildung 3.3 in PDF und ADR.Format für Tina.
- Die Modellierung von dem Spiel Halli-Galli befindet sich im Ordner Halli_Galli