Grundpraktikum Programmierung Wintersemester 2023/24
Programmdokumentation

Inhaltsverzeichnis

Einleitung

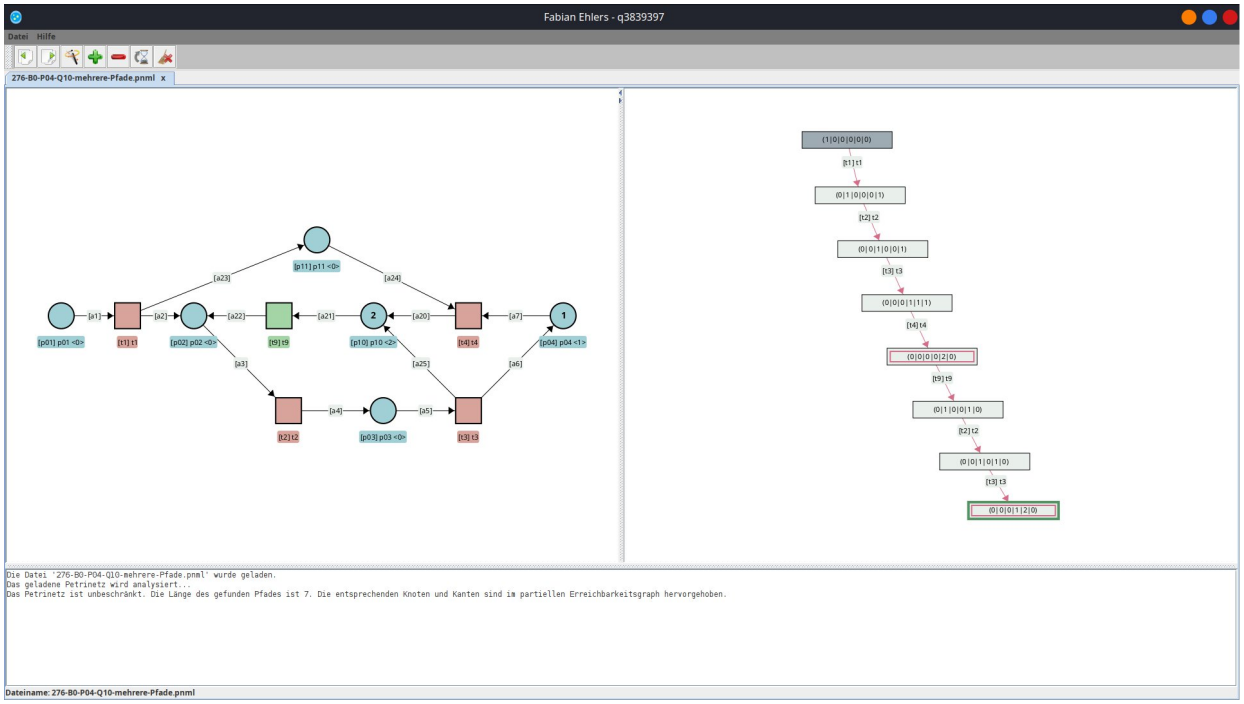

Abbildung 1: Hauptfenster mit Petri-Netz und zugehörigem Erreichbarkeitsgraph nach Beschränktheitsanalyse

Zur Einführung in das Programm und dessen Bedienung folgt eine kurze Erläuterung der GUI. Das Hauptfenster der Anwendung ist in *Abbildung 1* zu sehen.

Die Menüleiste befindet sich direkt unter der Titelleiste des Fensters. In dieser finden sich die Einträge *Datei* und *Hilfe*. In diesen finden sich alle, in der Aufgabenstellung geforderten, Einträge und weiter der Eintrag *Öffnen in neuem Tab* um explizit einen neuen Tab zu öffnen.

Die Toolbar befindet sich direkt unter der Menüleiste und enthält alle, in der Aufgabenstellung geforderten, Funktionalitäten. Die Buttons erklären sich folgendermaßen:

-  vorherige Datei aus dem aktuellen Verzeichnis öffnen.
-  nächste Datei aus dem aktuellen Verzeichnis öffnen.
-  Analyse des geöffneten Petri-Netzes.
-  Marke-Plus.
-  Marke-Minus.
-  Petri-Netz zurücksetzen.
-  Lösche-Erreichbarkeitsgraph.

Unter der Toolbar befindet sich die Tableiste. In dieser werden alle geöffneten Tabs angezeigt. In dem Programm ist es möglich mehrmals die gleiche Datei zu öffnen. Dies ermöglicht den schnellen Vergleich verschiedener Wege im gleichen Netz. Da die Veränderungen in diesem Programm nicht gespeichert werden, kann es auch durch mehrfaches Öffnen der gleichen Datei keine Inkonsistenzen in den Dateien geben.

Alle im Programm verwendeten Icons sind vom von Oxygen Team unter LGPL-Lizenz veröffentlicht und zur Verfügung gestellt von *https://www.iconarchive.com/show/oxygen-icons-by-oxygen-icons.org.html.*

Graphendesign

Petri-Netz-Graph

Die Repräsentation des Petri-Netzes folgt in den Formen der gängigen Petri-Netz-Darstellung mit Stellen als Kreise, Transitionen als Quadrate und Arcs als Pfeile. Da die Transitionen verschiedene Zustände haben können, werden aktivierte Transitionen grün ausgefüllt dargestellt und nicht-aktivierte Transitionen rot dargestellt. Wird im Petri-Netz ein Knoten angeklickt, erhält er eine doppelte Umrandung, von der es jedoch zu jedem Zeitpunkt nur eine im ganzen Graphen gibt. Bei Stellen kann diese Hervorhebung durch einen weiteren Klick auf diese Stelle entfernt werden. Für Transitionen ist dies nicht möglich, da es sich hierbei hauptsächlich um die Bedingung handelt um einer Stelle Marken hinzuzufügen oder abzuziehen. Das Zurücksetzen des Petri-Netzes entfernt die Hervorhebung in jedem Fall. Das Löschen von Petri-Netz und Erreichbarkeitsgraph entfernt die Hervorhebung, wenn sie auf einer Transition sitzt, nicht jedoch wenn sie auf einer Stelle sitzt, da mit diesem Vorgehen schnell mehrere Marken zu einer Stelle hinzugefügt werden können ohne jedes Mal die Stelle neu anzuklicken.

Erreichbarkeitsgraph

Der Wurzelknoten des Graphen ist immer Dunkelgrau und alle weiteren Knoten Hellgrau hinterlegt. Den grünen Rahmen hält immer der Knoten, der die aktuelle Markierung des Petri-Netzes repräsentiert, er existiert somit auch zu jedem Zeitpunkt einmalig. Werden Transitionen im Petri-Netz geschaltet, so wird immer die Kante zwischen den beiden aufeinanderfolgenden Knoten grün hervorgehoben. Wird manuell eine bestimmte Markierung im Erreichbarkeitsgraphen angeklickt, wird die grüne Hervorhebung der Kante entfernt, da diese nur Schaltvorgänge im Petri-Netz abbildet. Wird eine Beschränktheitsanalyse durchgeführt und das Ergebnis ist unbeschränkt, so werden die Knoten m und m‘ mit einem inneren roten Rahmen hervorgehoben. Weiter werden dann auf dem direkten Pfad vom Wurzelknoten über Knoten m bis zu m‘ alle Kanten rot hervorgehoben.

Bedienungsanleitung

Multiple-Document-Interface (MDI)

Die Datenstruktur zur Realisierung des MDI besteht daraus, dass für jeden Tab eine eigene Einheit nach MCV-Muster existiert. Damit besteht ein Tab aus einem Controller, der Operationen verwaltet, auslöst und durchführt, einem Datenmodell für die in diesem Tab geöffnete Datei und einer View, welche die Graphen darstellt.

Alle Dateien, die in dem Programm geöffnet werden, werden damit in einem eigenen Tab geöffnet. Damit nicht jedes Mal, wenn eine Datei geöffnet wird, auch als Standardverhalten ein neuer Tab geöffnet wird, gibt es im Menüeintrag *Datei* die Einträge *Öffnen in neuem Tab* und *Öffnen*. Mit Ersterem wird eine neu geöffnete Datei in einem neuen Tab geöffnet. Dieser wird an das Ende der Tableiste angeheftet. Mit Letzterem wird die neu geöffnete Datei in dem aktuellen Tab geöffnet und damit die vorherige Datei ersetzt.

Bei diesem Vorgehen gibt es eine Ausnahme, welche die Stapelverarbeitung mehrerer Dateien betrifft. In Tabs, welche einzelne Dateien anzeigen, sind verschiedene Interaktionen möglich. In Tabs, welche eine Stapelverarbeitung anzeigen, sind diese Interaktionen nicht möglich. Wird im Menü der Eintrag *Analyse mehrerer Dateien* ausgewählt, so wird der aktuelle angezeigte Tab nur in dem Fall ersetzt, dass dieser bereits das Ergebnis einer Stapelverarbeitung anzeigt. Sonst wird ein neuer Tab an die Leiste angeheftet. Umgekehrt gilt das gleiche, also wird bereits eine Stapelverarbeitung angezeigt und der Menüeintrag *Öffnen* gewählt, so wird ein neuer Tab angeheftet und nicht der aktuelle ersetzt.

Hervorhebung von Knoten im Petri-Netz Graph

Sobald ein Knoten im Petri-Netz Graph angeklickt wurde, wird er hervorgehoben. Bei weiteren Klicks wandert die Hervorhebung zum jeweiligen Knoten.

Handelt es sich um eine nicht-aktivierte Transition, geschieht nichts weiter, handelt es sich um eine aktivierte Transition, wird auch direkt geschaltet. Wird hier der Reset- oder Lösche-EG-Button gedrückt, wird die Hervorhebung entfernt.

Handelt es sich um eine Stelle, geschieht ebenfalls nichts weiter, jedoch ist diese Hervorhebung eine Bedingung um die Anzahl der Marken für eine Stelle zu verändern. Damit die Hervorhebung nicht immer wieder gesetzt werden muss, wenn mehrmals die Markenanzahl der gleichen Stelle verändert werden soll, wird diese Hervorhebung bei dem Löschen des Erreichbarkeitsgraphen nicht entfernt. So kann mehrmals hintereinander der Marke-Plus- oder Marke-Minus-Button geklickt werden. Wird der Reset-Button geklickt, wird die Hervorhebung hingegen entfernt, da diese Aktion nur manuell ausgelöst werden kann und nicht mit der Veränderung der Markenanzahl von Stellen in Verbindung steht. Die Hervorhebung einer Stelle kann auch durch einen erneuten Klick auf den Knoten entfernt werden.

Programmstruktur

Das Programm ist in drei Hauptteile gemäß des Model-View-Controller Musters aufgebaut. Dabei kommt ein erweiterter Ansatz des Musters zum Einsatz. Hierbei handelt es sich um Muster mit einem höheren Abstraktionsgrad zwischen Model und View. Während Aktualisierungen des Models im klassischen Ansatz direkt an die View kommuniziert werden, werden alle Aktionen in diesem erweiterten Ansatz über den Controller geführt, ohne dass die Bereiche Model und View Kenntnis voneinander haben. Der Algorithmus wird im Bereich *Algorithmus* dieser Dokumentation separat erläutert.

Model

Im Model können zwei Unterbereiche unterschieden werden. Hier gibt es die Klassen, welche die Elemente und das Modell eines Petri-Netzes repräsentieren und die Klassen, welche die Elemente und das Modell eines Erreichbarkeitsgraphen repräsentieren. Diese Aufteilung hat jedoch keinen Einfluss darauf, dass ein Petri-Netz im Rahmen dieses Projekts immer einen zugehörigen Erreichbarkeitsgraphen hat, dessen Knoten die Zustände bzw. Markierungen des Petri-Netzes abbilden. Diese Verbindung zwischen den beiden Modellen wird von dem Controller verwaltet, nicht von den Datenmodellen selbst.

<u>Petri-Netz:</u>

- Arc: Repräsentiert eine Kante und ihre Eigenschaften eines Petri-Netzes.
- Place: Repräsentiert eine Stelle und ihre Eigenschaften eines Petri-Netzes.
- Transition: Repräsentiert eine Transition und ihre Eigenschaften eines Petri-Netzes.
- PetrinetModel: Repräsentiert ein komplettes Petri-Netz und verwaltet seine Elemente. Wird eine Datei in dem Programm geöffnet, wird ein neues Petri-Netz erzeugt und die in der Datei beschriebenen Elemente (Arc, Place, Transition) werden in dieses Petri-Netz eingefügt. Im Petri-Netz werden die Operationen durchgeführt, in denen Eigenschaften des gesamten Netzes nötig sind. Das sind die Prüfung der Korrektheit des Netzes nach dem Erzeugen, die Prüfung ob eine Transition aktiviert ist, das Schalten einer Transition und das Setzen des Petri-Netzes auf einen bestimmten Zustand. Wobei einige Operationen hier nur verwaltet und an die entsprechenden Elemente weitergeleitet werden um die eigentliche Operation, durchzuführen.

<u>Erreichbarkeitsgraph:</u>

- ReachabilityNode: Repräsentiert einen Knoten eines Erreichbarkeitsgraphen. In diesem wird eine Markierung eines Petri-Netzes gespeichert.
- ReachabilityEdge: Repräsentiert eine Kante eines Erreichbarkeitsgraphen. In dieser wird gespeichert durch das Schalten, welcher Transition der Übergang von einer Markierung zu einer Folgemarkierung erfolgt ist.
- ReachabilityGraphModel: Repräsentiert einen Erreichbarkeitsgraphen und verwaltetet seine Elemente. Es enthält immer mindestens einen Knoten, den Wurzelknoten. Dieser wird bereits bei der Erzeugung des Modells eingefügt. Für jede Markierung die ein zugehöriges Petri-Netz jemals erreicht hat, wird in diesem Modell ein neuer Knoten eingefügt. Diese Verwaltung wird jedoch von dem Controller übernommen, da es keine direkte Verbindung zwischen den Datenmodellen gibt. Das Modell kann auf seinen Ursprungszustand zurückgesetzt werden. Dabei werden alle Elemente gelöscht und der Wurzelknoten erneut eingefügt.

View

In diesem Bereich befinden sich die Elemente des Programms, welche die grafische Repräsentation der Graphen, eines Tabs zur Anzeige im Hauptfenster und das Hauptfenster selbst bilden.

Graphen:

Eine wichtige Feststellung ist, dass die Graphen eine Graphstream-Repräsentation der Datenmodelle sind. Das heißt, sie dienen lediglich zur grafischen Darstellung dieser. Auf ihnen werden keine Operationen durchgeführt, die die Datenmodelle verändern. Weiter werden die hier enthaltenen Klassen von einem Controller verwaltet, da dieser dafür zuständig ist, dass die Graphen immer den aktuellen Zustand eines Datenmodells darstellen. Dieser Umfang kann nicht von diesen Klassen geleistet werden, da sie ihre zugrunde liegenden Datenmodelle im Sinne der Kapselung nicht kennen sollen.

Weiter verwaltet die Klasse die Hervorhebungen in der Anzeige der Graphen. Diese können in den Graphen selbst verwaltet werden, da hierfür keine Operationen auf dem Datenmodell nötig sind und der Controller des jeweiligen Tabs jederzeit die Hervorhebungen prüfen kann, da er die Graphen verwaltet. Aktualisierungen eines Graphen müssen daher immer von seinem Controller ausgelöst werden um die Anzeige konsistent zu den Datenmodellen zu halten.

Das Graph-Design wird von einer CSS-Datei für jeden Graph festgelegt.

- PetrinetGraph: Die Klasse repräsentiert die Graph-Darstellung eines Petri-Netzes. Bei der Erzeugung des Graphen wird das Datenmodell eines Petri-Netzes übergeben und alle Elemente aus diesem in den Graph übernommen.
- ReachabilityGraph: Die Klasse repräsentiert die Graph-Darstellung eines Erreichbarkeitsgraphen. Bei der Erzeugung des Graphen wird der Wurzelknoten übergeben und in den Graph übernommen. Jeder Knoten stellt als Label die Markierung des zugehörigen Petri-Netzes, die er repräsentiert, dar. Jede Kante stellt als Label den Namen der Transition dar, die geschaltet wurde um die Markierung, auf die die Kante zeigt, zu erreichen.

Fenster:

- TabView: Die Klasse repräsentiert einen Tab der im Hauptfenster angezeigt wird. Es existieren, analog zu der Klasse TabController, zwei Ausführungen von Tabs. Ein Tab zur Anzeige einer Stapelanalyse stellt lediglich ein Textfeld dar, in dem die Ausgabe des Ergebnisses erfolgt. Ein Statuslabel am unteren Rand zeigt Informationen über die Anzahl und das Verzeichnis der geprüften Dateien. Ein Tab zur Anzeige zweier Graphen stellt die Graphen, ein Textfeld zur Anzeige von Informationen über erfolgte Operationen und ein Statuslabel zur Anzeige des Dateinamens dar.
- PetrinetMainFrame: Stellt das Hauptfenster der Anwendung dar. Er enthält die Menüs, die Toolbar und das Panel zur Anzeige der Tabs inklusive Tableiste.

Control

In diesem Bereich befinden sich die Controller des Programms und weitere Klassen die eine Steuerungsfunktion oder einen Dienst im Programmverlauf übernehmen.

Dienste:

- Petrinets_3839397_Ehlers_Fabian: Der Programmeinstieg um das Programm zu starten. Hier wird das Hauptfenster der Anwendung erzeugt.
- TabListener: Der Listener verarbeitet Klicks, die in der Anzeige, also in den Graphen auftreten. Die Anzeige jedes Graphen bekommt einen eigenen Listener, der die Klicks an seine Anzeige weiterleitet, von wo sie an den Controller des jeweiligen Tabs weitergeleitet werden.
- SimplePetrinetParser: Die Klasse erweitert die Klasse PNMLWopedParser des Projekts ProPra-WS23-Basis. Der Parser liest PNML-Dateien ein und veranlasst die Verarbeitung der eingelesenen Elemente in dem Datenmodell des Petri-Netzes. Nachdem eine Datei eingelesen wurde, wird von ihm die Überprüfung des aus der Datei erzeugten Datenmodells auf Korrektheit im Rahmen dieses Programms überprüft.

Controller:

- FrameController: Der Controller, der das Hauptfenster der Anwendung verwaltet. Er verwaltet die Interaktion mit den Menüs, der Toolbar und den Tabs. Hier werden alle Interaktionen, die über eines dieser Bedienungselemente gesteuert werden, gesteuert und die entsprechenden Aktionen werden ausgelöst. Da die Durchführung dieser Aktionen in Verbindung mit einem Petri-Netz oder der Stapelverarbeitung von selbigen in Verbindung stehen, wird ihre Durchführung an den Controller des Tabs, für den die Operation angefordert wurde, weitergereicht und von diesem ausgeführt.
- TabController: Der Controller ist das zentrale Element zur Verwaltung eines Tabs und steuert alle Operationen auf den Datenmodellen und der Anzeige. Er kennt damit alle Elemente eines Tabs. Dies sind die Datenmodelle des Petri-Netzes und des Erreichbarkeitsgraphen, die Graphen, welche jeweils diese Datenmodelle repräsentieren, die Anzeige, in der die beiden Graphen dargestellt werden und das Hauptfenster, in dem die Tabs dargestellt werden. Der TabController ist auch dafür verantwortlich die Konsistenz zwischen den Datenmodellen und ihrer Anzeige sicherzustellen.

  Es existieren in diesem Programm zwei verschiedene Arten von TabControllern, da sich die Verwaltung eines Petri-Netzes und seines Erreichbarkeitsgraphen sehr von der Verwaltung einer Stapelverarbeitung mehrerer Dateien unterscheidet. Für die Stapelverarbeitung sind keine Operationen auf einem bestimmten Petri-Netz möglich, dafür müssen hier mehrere Petri-Netze analysiert und die Ergebnisse ausgegeben werden. Der benötigte TabController wird mit der Auswahl des Menüeintrages automatisch erzeugt.

Algorithmus

Überblick

Der Algorithmus zur Durchführung einer Beschränktheitsanalyse eines Petri-Netzes ist in dem Programm in einem eigenständigen Bereich angeordnet. Damit arbeitet er in einem separaten Bereich und kann nicht auf Informationen zugreifen, die mit der Durchführung der Analyse keine Schnittmenge haben. Zur Durchführung wird ein eigenes Analyse-Objekt erstellt und dieses bekommt die Datenmodelle eines Petri-Netzes und eines Erreichbarkeitsgraphen übergeben. Diese beiden Modelle befinden sich zum Zeitpunkt der Übergabe in ihrem jeweiligen Anfangszustand, wie er direkt nach der Erzeugung der Modells ist. Somit herrscht zum Start der Analyse ein definierter Zustand.

Der Algorithmus führt eine rekursive Tiefensuche auf dem Ausgangszustand des Petri-Netzes aus. Der Algorithmus hat dabei direkten Zugriff auf die ihm übergebenen Datenmodelle und verändert diese in seinem Verlauf. Weiter erzeugt jede Rekursionsebene eigene Hilfsstrukturen zur Speicherung des bisher durchlaufenen Pfades, also der besuchten Knoten und der dazwischen liegenden Kanten des Datenmodells des Erreichbarkeitsgraphen. Das Analyse-Objekt enthält auch eine eigene Struktur um die Ergebnisse der Analyse abzurufen. Das sind im Wesentlichen das Ergebnis der Beschränktheitsanalyse, der direkte Pfad vom Wurzelknoten des Erreichbarkeitsgraphen über den Knoten m bis zu dem Knoten m‘ und gesondert die Knoten m und m‘. Diese Daten können nach der Analyse vom Controller des Tabs verarbeitet werden um z. B. den Pfad und die Knoten im Graphen der Anzeige hervorzuheben, das Ergebnis der Analyse zu kommunizieren oder die Ausgabe der Stapelverarbeitung darzustellen.

Die Abbruchbedingung des Algorithmus ist erreicht, sobald zwei Knoten m und m‘ im Erreichbarkeitsgraphen gefunden wurden. Damit wird die Ausführung direkt beendet und alle höheren Rekursionsebenen in die zurückgekehrt wird, werden auch sofort beendet. Mit diesem Verhalten wird erreicht, dass im Fall der Unbeschränktheit die Datenmodelle und damit die Graphen nicht weiter verändert werden, also der Knoten m‘ im Erreichbarkeitsgraphen der letzte erreichte Knoten bleibt und das Petri-Netz auch im äquivalenten Zustand. Bei beschränkten Petri-Netzen ist die Abbruchbedingung erreicht, wenn der Erreichbarkeitsgraph vollständig aufgebaut ist. Das Petri-Netz wird in diesem Fall auf die Anfangsmarkierung gesetzt um einen konsistenten Zustand herzustellen.

Pseudocode

Zur genaueren Erläuterung folgt eine Darstellung des Algorithmus in Pseudocode. In der Methode analyse() ist der Algorithmus enthalten. Hilfsmethoden sind in dieser Darstellung nicht erläutert. Die Bezeichnung Knoten im Algorithmus meint immer Knoten des Erreichbarkeitsgraphen. Die Elemente des Petri-Netzes werden direkt als Stelle oder Transition bezeichnet.

```
 1  PetrinetAnalysis
 2
 3  START analyse(besuchteKnoten, erzeugteKanten, aktuelleMarkierungKnoten)
 4      erzeuge aktivierteTransitionenListe
 5      IF aktivierteTransitionenListe ist NICHT leer THEN
 6          // lokale Listen erzeugen, damit jede Rekursionsebene ihren Pfad kennt
 7          erzeuge leere knotenListe
 8          erzeuge leere kantenListe
 9          FOR jeden Knoten von besuchteKnoten
10              Knoten in knotenListe einfügen
11          END FOR
12          aktuelleMarkierungKnoten in knotenListe einfügen
13          FOR jede Kante von erzeugteKanten
14              Kante in kantenListe einfügen
15          END FOR
16          IF nicht erster Aufruf von analyse() THEN
17              letzteKante von EG-Modell in kantenListe einfügen
18          END IF
19
20          FOR jede Transition von aktivierteTransitionenListe
21              schalte die Transition in PN-Modell
22              speichere Folgemarkierung nextMarkierungKnoten
23              IF nextMarkierungKnoten ist NICHT in EG-Modell enthalten THEN
24                  nextMarkierungKnoten in EG_Modell
25                  erzeuge neueKante zu nextMarkierungKnoten
26                  neueKante in EG-Modell einfügen
27                  IF nextMarkierungKnoten markiert PN als unbeschränkt THEN
28                      speichere m und m'
29                      markiere PN als unbeschränkt
30                      speichere knotenListe
31                      speichere kantenListe
32                      RETURN beende die Ausführung dieser Rekursionsebene
33                  ELSE der Knoten markiert das PN nicht als unbeschränkt THEN
34                      START analyse(knotenListe, kantenListe, nextMarkierung)
35
36                      IF PN als unbeschränkt markiert THEN
37                          RETURN beende die Ausführung dieser Rekursionsebene
38                      END IF
39                  END ELSE
40              ELSE nextMarkierungKnoten ist in EG-Modell enthalten THEN
41                  finde nextMarkierungKnoten in EG-Modell
42                  erzeuge neueKante zu nextMarkingKnoten
43                  neueKante in EG-Modell einfügen
44              END ELSE
45              /*
46               * wenn alle nach dem Schalten in tieferen Rekursionsebenen
47               * möglichen Schaltvorgänge durchgeführt sind und PN unbeschränkt ist.
48               */
49              setze PN-Modell auf Markierung von aktuelleMarkierungKnoten
50          END FOR
51      END IF
52  END
```

Abbildung 2: Pseudocode für den Beschränktheitsalgorithmus