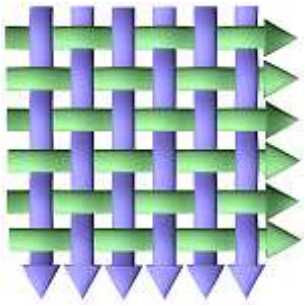
Sonderforschungsbereich 559
Modellierung großer Netze in der Logistik



Technical Report 03011

ISSN 1612-1376

Abbildung von ProC/B nach Petri-Netzen - Version 2

Teilprojekt M2:
Markus Fischer, Peter Kemper, Zenghui Wu
Informatik 4,Universität Dortmund

Teilprojekt M1:
Carsten Tepper
Informatik 4,Universität Dortmund

Dortmund, 01.12.2003

Inhaltsverzeichnis

Kapitel 1

Einleitung

Abbildungen zwischen verschiedenen Modellwelten dienen dem Ziel, bereits vorhandene und erprobte Analysetechniken für neue Modellwelten verfügbar zu machen. Innerhalb des Sonderforschungsbereichs 559 werden Prozessketten-Modelle - nachfolgend kurz als *ProC/B* Modelle notiert - in Petri Netz (PN) Modelle transformiert, um damit funktionale und quantitative Analysetechniken, die bereits für PNs implementiert sind, auch für *ProC/B* Modelle anwenden zu können. Die Abbildung ist automatisiert, d.h. wird durch eine Software geleistet. Die erste Version der Abbildung wurde in den Jahren 2000 bis Ende 2001 implementiert und ist in [1] dokumentiert. Bedingt durch den Wunsch, die Klasse abbildbarer *ProC/B* Modelle zu vergrößern, neu eingeführte *ProC/B* Modellierungskonstrukte (Lager-Baustein, Schleifen) behandeln zu können und bei der Anwendung der Software aufgetretene Fehler zu beheben, wurde im Jahr 2003 die Abbildungs-Software erweitert (Version 2).
Die grundlegende Idee der Abbildung ist es, für Modellierungskonstrukte der *ProC/B* Notation verhaltensäquivalente PN Substitute zu definieren. Die Semantik zahlreicher *ProC/B* Konstrukte ermöglicht eine recht einfache Übersetzung, allerdings unterliegt die Abbildung auch Restriktionen, die entweder aus der begrenzten Ausdruckskraft der PN Notation oder aus Bedingungen der Analysetechniken (behandelbare Zustandsraumgröße) resultieren. Technisch wird die Abbildung durch einen Parser realisiert, der über eine *ProC/B* Schnittstelle das gegebene *ProC/B* Modell einliest und das resultierende PN Modell in einer C++ Klassenhierarchie als Datenstruktur schrittweise aufbaut bzw. übersetzt. Die PN Datenstruktur wiederum besitzt einen Parser, der das aufgebaute PN Modell in die APNN Notation (textuelle Austauschformat für PNs) überführt.
Dieser Bericht dokumentiert Erweiterungen und behobene Fehler der Version 1. Dies betrifft die folgenden Konstrukte der *ProC/B* Notation: `Oder-Konnektor`,

`Prozessketten-Konnektor` zur Synchronisation von Prozessen, mehrdimensionale `Counter` bzw. `Lager` als passive Ressourcen, `LOOP` und `ENDLOOP` als Schleifenkonstrukte und zeitbehaftete Prozesskettenelemente (PKEs) als einfache Zeitverzögerungen. Die Tabelle 1.1 gibt eine Übersicht zur Klasse übersetzbarer *ProC/B* Modelle.

ProC/B Element	Abbildbarkeit auf Petri-Netz
`FE`	Strukturierungselement ohne Einfluss auf Modell-Dynamik
`Server`	$\oplus$ "Random" Bediendisziplin, Mehrbediener $\otimes$ "Infinite Server" Semantik durch Mehrbediener approximiert $\ominus$ FCFS, PS Bediendisziplin, Bedienprioritäten, zustandsabh. Bediengeschwindigkeit
`Counter`	$\oplus$ ein- und mehrdimensionale Zähler mit konstanter (In/De)krementierung $\ominus$ dynamische, datenabhängige Zählermanipulation
`Quelle`	$\oplus$ Poisson-artige und singuläre Prozesserzeugung (auch "bulk"-Prozesse) $\ominus$ boolsche (datenabhängige) Prozesserzeugung
`Senke`	$\otimes$ "offene" Prozesse werden via Kurzschluss durch "geschlossene Prozesse" ersetzt
`PK`	beinhaltet (potentiell) alle unten beschriebene Konstrukte
`PK-Interface`	$\ominus$ alle Typen von Aufrufparametern
`Delay-PKE`	$\oplus$ konstante Verzögerung mit Markov'schen Zeitverbrauch $\otimes$ "Infinite Server" Semantik durch Mehrbediener approximiert $\ominus$ datenabhängige Zeitverbräuche
`Code-PKE`	$\oplus$ HiSlang Code zur Addition/Subtraktion von Konstanten auf/von `FE-Variable` $\ominus$ sonstiger HiSlang Code
`Loop-PKE`	$\oplus$ Endlosschleife, boolsche Abbruchbedingung (Test auf `FE-Variablen` Werte) probabilistische Abbruchbedingung (geometrische Verteilung der Durchläufe) $\otimes$ boolsche Abbruchbedingung durch probabilistische Abbruchbedingung approxim. $\ominus$ komplexe boolsche Ausdrücke als Abbruchbedingung
`UND-Konnektor`	$\oplus$ "Fork"-Operator zur Erzeugung von Parallel-Prozessen $\otimes$ "Join-Operator" rekombiniert+synchronisiert jeweils terminierte Parallel-Prozesse, die aus u.U. verschiedenen "Fork" hervorgehen ("Min-Match" Semantik)
`ODER-Konnektor`	$\oplus$ probabilistischer "Branch"-Operator, boolsche Variante für einfache Ausdrücke wie Test auf `FE-Variablen` Werte $\ominus$ komplexe boolsche Ausdrücke für "Branch"-Auswahl
`PK-Konnektor`	$\oplus$ Synchronisation, Erzeugung und Replikation von Prozessen
`FE-Variable`	$\oplus$ Integer-Variablen mit endlichem Definitionsbereich (auch mehrdimensional) $\ominus$ Variablen $\neq$ Integer-Typ
`PK-Parameter`	$\ominus$ alle Ausprägungen

Tabelle 1.1: *ProC/B* Modellelemente (Auswahl) und ihre Abbildbarkeit auf Petri Netze: Ausprägungen von *ProC/B*-Konstrukten, die ohne Einschränkung abbildbar sind ($= \oplus$) , bedingt (approximativ) abbildbar sind ($= \otimes$) und nicht behandelbar sind ($= \ominus$).

Kapitel 2

Die Abbildung von *ProC/B* nach Petri Netzen

Die Erweiterungen der Implementierung in der Version 2 werden nachfolgend anhand von 'Fallbeispielen' vorgestellt. Die Implementierung basiert auf der Version 1, deren Funktionalität hier nicht wiederholt wird, siehe [1].

2.1 Schachtelung von Oder-Konnektoren

Das Modell in Abbildung A.1 zeigt zwei verschachtelte `Oder-Konnektoren`. Die Situation, dass der Ausgang eines `öffnenden Oder-Konnektors` direkt mit dem Eingang eines inneren, verschachtelten `öffnenden Oder-Konnektors` verbunden ist, führte in Version 1 zu einem Laufzeitfehler. Die Schachtelung von `Oder-Konnektoren` funktionierte nur für den Spezialfall, wenn zwischen Ausgang und Eingang verschachtelter `Oder-Konnektoren` mindestens ein Prozessketten-Element existiert.
Wie im Bericht [1] beschrieben, wird ein `öffnender Oder-Konnektor` durch eine Stelle und pro Verzweigung durch eine zeitlose Transition beschrieben, die das probabilistische Routing auf die einzelnen Zweige realisieren. Der `schliessende Oder-Konnektor` besteht aus einer Stelle und einer zeitlosen Transition, welche die Zweige wieder zu einem Zweig zusammenführt. Eine Synchronisation findet nicht statt, da immer nur eine Aktion (Zweig) ausgeführt wird. Die Abbildung A.2 zeigt die vollständige Petri-Netz Beschreibung des *ProC/B* Modells aus Abbildung A.1.

2.2 Sequenzen von PK-Konnektoren mit Prozess Replikation

Im Modell aus Abbildung A.3 werden zwei direkt verbundene `PK-Konnektoren` beschrieben. In der ersten Version der Abbildung wurde die Quelle (links) mit dem rechten `PK-Konnektor` verbunden, weil im Petri Netz die Eingangsstelle, welche die Anbindung des `PK-Konnektor` modelliert, nicht eindeutig war. Die im Ausgang eines `PK-Konnektors` neu erzeugten Prozesse werden im Petri Netz durch einen Kurzschluss mit endlicher Population modelliert, so dass die Anzahl dort befindlicher Prozesse und damit auch der gesamte Zustandsraum beschränkt ist. In der ersten Version konnte die maximale Anzahl gleichzeitig existierender Prozesse nicht parametrisiert werden. Nun kann die Tokenanzahl der Stelle *Env_PKKonnetor* durch die Option '@@ENV_TOKEN' in der Ansteuerungsdatei festgelegt werden, z.B. in diesem Modell ist die Tokenanzahl der Stelle *Env_PKKonnektor* gleich 4.
Eine nächste Erweiterung betrifft die Multiplizität der an den Eingängen und Ausgängen zu synchronisierender Prozesse. In der ersten Version wurde über jeden eingehenden und ausgehenden Strang am `PK-Konnektor` standardmäßig stets genau ein Prozess synchronisiert.
Im Modell in Abbildung A.3 ist die Anzahl der eingehenden Prozesse am ersten `PK-Konnektor` gleich 3 und die Anzahl der gleichzeitig zu startenden Prozesse von `PKQuelle` PID3 gleich 2, die anderen haben den Wert 1. Dementsprechend nimmt die Transition *PKKonnektor_8* (linker `PK-Konnektor`) 3 Token aus seiner Eingabestelle und erzeugt 1 Token für die Eingabestelle des oberen Strangs und 2 Token für die Eingabestelle des unteren Strangs. Es obliegt der Verantwortung des Modellierers, vervielfachte Prozesse später wieder so zu "rekombinieren", dass die maximale Anzahl der Prozesse und damit auch die Größe des Zustandsraums endlich ist.
Die Abbildung A.4 zeigt die vollständige Petri-Netz Beschreibung des *ProC/B* Modells aus Abbildung A.3. Man beachte, dass die Inzidenzmatrizen der Transitionen des Petri Netzes nicht visualisiert sind und somit eine Verifikation des korrekten "kombiniererens" und "rekombinierens" der Prozesse am `PK-Konnektor` nicht durch die Abbildung A.3 möglich ist (das Petri Netz ist aber korrekt).

2.3 Sequenzen von PK-Konnektoren mit Prozess-Rekombination

Die Abbildung A.5 zeigt ein *ProC/B* Modell, in dem an einem `PK-Konnektor` (links) beginnende Prozesskette mit dem Eingang eines nachfolgenden `PK-Konnektors` (rechts) verbunden ist. Wie bereits oben erwähnt, sind die Prozesse der unteren, zwischen den `PK-Konnektoren` verlaufenden Prozesskette im PN kurz geschlossen, d.h. nach einer Synchronisation werden sie in die Quelle zurückgeführt. In der Version 1 wurde diese Konstellation der `PK-Konnektoren` fehlerhaft übersetzt bzw. es trat ein Laufzeitfehler auf.
Die Abbildung A.6 zeigt das zugehörige PN Modell. Der Kurzschluss wird über die abgerundete Kante realisiert.

2.4 Mehrdimensionale Counter (Lager)

Im Modell aus Abbildung A.7 wird ein zweidimensionales `Lager` beschrieben. Die `Lager-Funktionseinheit` ist eine Erweiterung der *ProC/B* Notation, die in der ersten Version der Abbildung noch nicht identifiziert und verarbeitet werden konnte. Die Semantik der `Lager-Funktionseinheit` (unterer Teil) stimmt mit der des `Counter` überein, sie bietet aber zusätzliche Dienste zur Modellierung von Zugriffen an. In der Version 2 wird nur der Dienst `change` unterstützt, so dass `Lager-Funktionseinheit` und `Counter` identisch und somit analog behandelbar sind. Ein bekannter Fehler aus der Version 1 (Initialwert des `Counter` wurde falsch behandelt wenn ungleich von Null) wurde beseitigt.
Im Modell A.7 werden Zugriffe auf ein zwei-dimensionales `Lager` modelliert, das mit (3,2) initialisiert ist. Die zulässige Untergrenze ist (0,0), die Obergrenze (10,10). Im PN (Abbildung A.8) wird ein `Lager` durch eine Stelle (Token zeigt Zugriffwunsch an), zeitlose Transition (Zugriff) und zwei weitere Stellen, die den Lagerzustand kodieren (freie und belegte Kapazität), beschrieben. Die Stellen *AutoLager_2dim1Diff* und *AutoLager_2dim2Diff* erhalten als Anfangsmarkierung den Wert (Obergrenze-Initialisierung-Untergrenze), d.h. die Stelle *AutoLager_2dim1Diff* erhält den Wert 7=10-3-0. Die Stellen *AutoLager_2dim1* und *AutoLager_2dim2* erhalten den Wert 3 und 2 (Initialisierung-Untergrenze).

2.5 Schleifen (LOOP und ENDLOOP)

Im Modell aus Abbildung A.9 werden zwei ineinander verschachtelte Schleifen modelliert. Der Start bzw. das Ende einer Schleife werden durch das `LOOP` bzw. `ENDLOOP` Prozesskettenelement (PKE) markiert. Der Schleifenrumpf wird wiederholt ausgeführt, bis die (boolesche) Abbruchbedingung erfüllt ist.
Ein `LOOP PKE` bzw. ein `ENDLOOP PKE` wird durch eine Stelle und eine zeitlose Transition als PN modelliert. Boolesche Abbruchbedingungen können in relativ einfachen Fällen exakt übersetzt werden (siehe Abschnitt "Datenabhängiges Verhalten"), anderenfalls wird der Test der booleschen Abbruchbedingung randomisiert, indem die Anzahl der Schleifendurchläufe durch eine geometrische Verteilung approximiert wird. Die probabilistische Auswahl zwischen Beendigung und Fortführung der Schleife wird durch zwei konkurrierende Transitionen modelliert. Der Parameter der zugrundeliegenden geometrischen Verteilung muss durch den Anwender in der Ansteuerungsdatei der Abbildung via der Option '@@LOOP_REPEAT_PROB' gesetzt werden.
Die Abbildung A.10 zeigt das vollständige Petri-Netz des *ProC/B* Modells aus Abbildung A.9.

2.6 Zeitbehaftete Prozesskettenelemente

`Delay PKEs` modellieren die Dynamik eines `Infinite Server`, d.h. Prozesse erfahren eine (stochastische) Verzögerung, die unabhängig von der Anzahl gleichzeitig im `Delay PKE` befindlicher Prozesse ist. In der ersten Version der Abbildung wurde ein `Delay PKE` im PN durch eine 1-Bediener Semantik approximiert. In der zweiten Version wird die Approximation durch eine k-Bediener Semantik verbessert, wobei die Anzahl der Bediener k durch die Option '@@DELAY_MODI' in der Ansteuerungsdatei festgelegt wird. Dadurch ist es möglich, die `Infinite Server` Semantik beliebig genau zu approximieren.
Die Abbildung A.12 zeigt die vollständige PN Beschreibung des *ProC/B* Modells aus Abbildung A.11. Die approximierte Infinite-Server Semantik ist hier in verschiedenen Modi der Transitionen *Drehen_a* und *Transport_a* "versteckt" und nicht explizit visualisiert.

2.7 Quelle mit Typ AT

Die Abbildung A.13 beinhaltet eine `Quelle` vom Typ 'AT'. Die einmalige Instantiierung der Prozesse wird durch die Transition *Quelle_PID* in Abbil-

dung A.14 modelliert. Der Zeitpunkt der Instantiierung ist die Feuerungszeit der Transition, im Beispiel eine zeitlose Transition. Die Anzahl der Token in der Stelle *At_PID* stimmt mit der Anzahl zu erzeugender Prozesse überein. Im Gegensatz zur `Quelle` vom Typ 'EVERY' hat die Stelle *At_PID* keine einlaufende Kante, die sonst ein geschlossenes System durch Kurzschluss von Senke mit Quelle modelliert.

2.8 Variablen

Variablen mit diskretem und endlichem Definitionsbereich und bestimmte, auf Addition und Subtraktion beschränkte Zugriffe werden durch die Abbildung unterstützt. Skalare Werte der Variablen werden durch Tokenpopulationen einzelner Stellen modelliert, wobei über zwei Farben zwischen dem eigentlichen, aktuellen Wert der Variablen und seiner Differenz zum maximal erlaubten Wert (Definitionsbereich) unterschieden wird. Die (redundante) Differenz-Information erleichtert die Auswertung von Vergleichtests (der aktuelle Wert ist mindestens/höchstens ...). Das Modell in Abbildung A.15 beinhaltet zwei Integer-`Variablen` "X" und "Y". "X" ist eindimensional und hat den Initialwert 0. "Y" ist zweidimensional mit Initialwert (1,0). Die Abbildung A.16 zeigt das zugehörige PN Modell. Die Stelle "X" modelliert die Variable "X", die Stelle "Y0" modelliert die erste Komponente der Variablen "Y" usw. Für die Spezifikation des maximal erlaubten Wertes der Variablen gibt es 3 Möglichkeiten: weil kein Attributfeld "maximal" in der *ProC/B* Variablenspezifikation vorgesehen ist, kann ersatzweise das Attributfeld "Kommentar" zur Spezifikation der Obergrenze verwendet werden. Findet der Parser dort keinen auswertbaren Ausdruck, verwendet der Parser eine global gültige Obergrenze, die in der Ansteuerungsdatei mit der Option '@@VAR_MAX_TOKEN' optional definierbar ist. Wenn in der Ansteuerungsdatei keine globale Obergrenze angegeben wurde, verwendet der Parser als Default-Wert den Wert 2. Im betrachten Beispiel finden weder Zugriffe auf die Variablen noch werden deren Werte getestet. Deswegen sind die zugehörigen Stellen isoliert von den Stellen und Transitionen, welche die Dynamik der Prozesskette modellieren. Zugriffe auf Variablen werden im nachfolgenden Abschnitt betrachtet.

2.9 Code PKE

Variablenmanipulationen werden in der *ProC/B* Notation mit `Code PKE` modelliert. Das *ProC/B* Modell in Abbildung A.17 hat zwei `Code PKE`, die Wer-

te einer zweidimensionalen Variablen "T" verändern. Das linke `Code PKE` "Plus" inkrementiert den Wert der ersten Komponenten, die rechte `Code PKE` "Minus" dekrementiert den Wert der zweiten Komponenten. Als Variablenmanipulationen sind in der vorliegenden Version der Abbildung nur die Addition und Subtraktion von Konstanten zulässig, weil diese Operationen direkt durch Transitionen des PN und damit auf einfache Weise abbildbar sind. Die Abbildung A.18 zeigt das zugehörige PN Modell. Ein `Code PKE` wird durch eine Stelle und eine zeitlose Transition (Zugriff) beschrieben. Die zeitlose Transition wird mit der Stelle der zu manipulierenden `Variablen` verbunden.

2.10 Datenabhängiges Verhalten

Datenabhängiges Verhalten in *ProC/B* Modellen wird in wenigen und sehr beschränkten Fällen durch die Abbildung auf PNs unterstützt. Die Restriktion auf bestimmte Ausprägungen von *ProC/B* Modellen geschieht vor dem Hintergrund des Zustandsraums, der in Modellen mit exzessiv-datenabhängigen Verhalten extrem groß wird und damit eine zustandsraum-basierte Analyse, die das eigentliche Ziel der Abbildung von *ProC/B* Modellen auf PN Modelle ist, ausschließt. Behandelbare Fälle beinhalten Variablen mit endlichem und diskretem Wertebereich und deren Manipulation durch Addition und Subtraktion konstanter Werte, boolesche ODER-Konnektoren mit Test auf Gleichheit und Vergleiche mit konstanten Werten und eine boolesche Abbruchbedingung bei Schleifen (ENDLOOP).

2.11 Abbildungs-Restriktionen für Petri-Netze

In diesem Abschnitt werden einige vorhandene technische Schwierigkeiten der Implementierung anhand von Beispielen dokumentiert. Die *ProC/B* Modelle aus Abbildung A.23 und A.24 können nicht abgebildet werden.

1. Im Modell aus Abbildung A.23 sind ein `PK-Konnektor` und ein `Öffender Oder-Konnektor` direkt miteinander verbunden. In der *ProC/B* Schnittstelle wird der `Öffnende Oder-Konnektor` als ein Element vom Typ "PKTeil" klassifiziert, die Abbildung kann jedoch am Ausgang eines `PK-Konnektors` aus technischen Gründen nur *ProC/B* Elemente zulassen, die in der *ProC/B* Schnittstelle unter den Typ "LinearElement" (z.B. `Delay PKE`) subsumiert sind. Dies bedeutet, dass wenn zum Beispiel am Ausgang des `PK-Konnektors` ein `Delay PKE` folgt, ist das *ProC/B* Modell problemlos abbildbar.

2. Das Modell aus Abbildung A.24 ist sehr ähnlich zum Modell aus Abbildung A.5, nur zwischen dem zweiten `PK-Konnektor` und der `virtuellen Senke` fehlt ein `Delay PKE`. Dieses Modell kann nicht in PNs übersetzt werden (Laufzeitfehler). Der Grund hierfür ist ein methodisches Defizit beim Entwurf des Parsers, der prinzipiell bekannt ist, aber noch nicht behoben werden konnte.

2.12 Ansteuerungsdatei

Parameter der Abbildung können über eine Ansteuerungsdatei spezifiziert werden. Insgesamt können bis zu 14 Parameter definiert werden. Die Bedeutung der Parameter ist nachfolgend beschrieben:

1. `@@MODELFILE` vollständiger Pfad der B1-Datei (textuelles Austauschformat für *ProC/B* Modelle)
2. `@@PARTITION` für Erweiterung reserviert, aktuell ohne Bedeutung
3. `@@AGGREGATE` "YES" oder "NO", bei "YES" wird nur eine bestimmte Funktionseinheit übersetzt (siehe `@@FE_NAME`) und für diese ein Aggregat (aktuell nur flußäquivalente Aggregierung) mit der in `@@ANALYSIS_TOOL` angegebenen Technik berechnet
4. `@@DELAY_MODI` "x:int" Approximation für IS-Semantik bei `Delay PKE`, x=Anzahl der Bediener
5. `@@ENV_TOKEN` "x:int" Initiale Token-Belegung von Stellen, die eine Kurzschlussstelle (=Umgebung) beschreiben
6. `@@LOOP_REPEAT_PROB` "x:double" Parameter der Cox-Verteilung für probabilistische Anzahl von Schleifendurchläufen (`ENDLOOP`)
7. `@@VAR_MAX_TOKEN` "x:int" Definitionbereich von Integer-Variablen ist [0,x]
8. `@@ANALYSIS_TOOL` "supgspn" (=zustandsraumbasiert+numerisch) , "simulator" (Simulation) Analysetechnik für Aggregatberechnung
9. `@@FE_NAME` Name zu aggregierende Funktionseinheit
10. `@@NO_OF_SERVICES` "x:int" Anzahl Dienste der Funktionseinheit definiert durch `@@FE_NAME`
11. `@@NAME_OF_SERVICE1` Name Dienst 1

12. `@@TRANS_OF_SERVICE1` Mess-Transition im Petri-Netz zur Ermittlung des Durchsatzes

13. `@@POP_OF_SERVICE1` "x:int" Obergrenze für Population

14. `@@OUTPUT_FORMAT` "Normal" oder "HIT" (Format der textuellen Aggregatbeschreibung): "Normal" ist Format zur Weiterverarbeitung in ein explizit in *ProC/B* spezifiziertes Aggregat, "HIT" ist HiSlang-Format zur Weiterverarbeitung durch HIT Analysetool

Nicht alle 14 Parameter müssen in der Ansteuerungsdatei enthalten werden, wenn der Parameter `@@AGGREGATE` mit "NO" gesetzt wurde, dann ist nur `@@MODELFILE` obligatorisch, alle weiteren Parameter sind obsolet oder werden mit Default-Werten initialisiert.
Eine typische Ansteuerungsdatei sieht wie folgt aus:

```
@@MODELFILE /tmp/aggregate/SimCheck.B1
@@PARTITION modular
@@AGGREGATE YES
@@DELAY_MODI 5
@@ENV_TOKEN 5
@@LOOP_REPEAT_PROB 0.3
@@VAR_MAX_TOKEN 3
@@ANALYSIS_TOOL supgspn
@@FE_NAME MM2
@@NO_OF_SERVICES 1
@@NAME_OF_SERVICE1 PID1
@@TRANS_OF_SERVICE1 MeasurePoint1
@@POP_OF_SERVICE1 2
@@OUTPUT_FORMAT HIT
```

2.13 Aufruf des Übersetzers

Weil der Übersetzer in zwei Software-Tools eingebunden ist (APNN-Toolbox und ProC/B-Toolset), werden zwei verschiedene Aufrufvarianten unterstützt:

1. pk2pn `B1-Datei Ausgabeverzeichnis`

2. pk2pn `Ansteuerungsdatei` (siehe Abschnitt 2.12)

In der ersten Variante wird als erster Parameter die B1-Spezifikation des *ProC/B* Modells und als zweiter Parameter ein Ausgabeverzeichnis angegeben. In das Ausgabeverzeichnis wird das erzeugte Petri-Netz Modell (in textueller apnn-Notation) geschrieben. Dieser Aufruf wird in der graphischen Benutzeroberfläche der APNN-Toolbox durch die Schritte "File"→ "Import"→ "ProC/B" in der Menüleiste aktiviert.
In der zweiten Variante wird als erster und einziger Parameter eine Ansteuerungsdatei übergeben, deren Inhalt im Abschnitt 2.12 beschrieben wird. Dieser Aufruf wird in der graphischen Benutzeroberfläche des ProC/B-Toolset durch Selektion einer zu aggregierenden Funktionseinheit und anschließend durch Auswahl des Menüpunkts "Bearbeiten" → "Aggregieren von.." aktiviert. Dabei können die in der Ansteuerungsdatei des Übersetzers enthaltenen Informationen über eine graphische Benutzeroberfläche definiert werden. Das Eingabefenster erscheint nachdem die oben angegebenen Schritte ausgeführt wurden. Gleichzeitg dient das Eingabefenster zum Aufruf des Übersetzers. Das erzeugte Petri-Netz Modell (in textueller apnn-Notation) wird in das Verzeichnis `\tmp\aggregate` geschrieben.

Anhang A

Bespielmodelle *ProC/B* und Petri-Netze

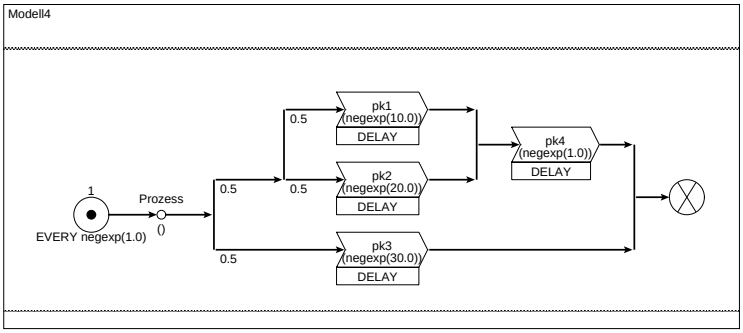


Abbildung A.1: *ProC/B* Modell: direkte Verbindung bzw. Schachtelung von `Oder-Konnektoren`

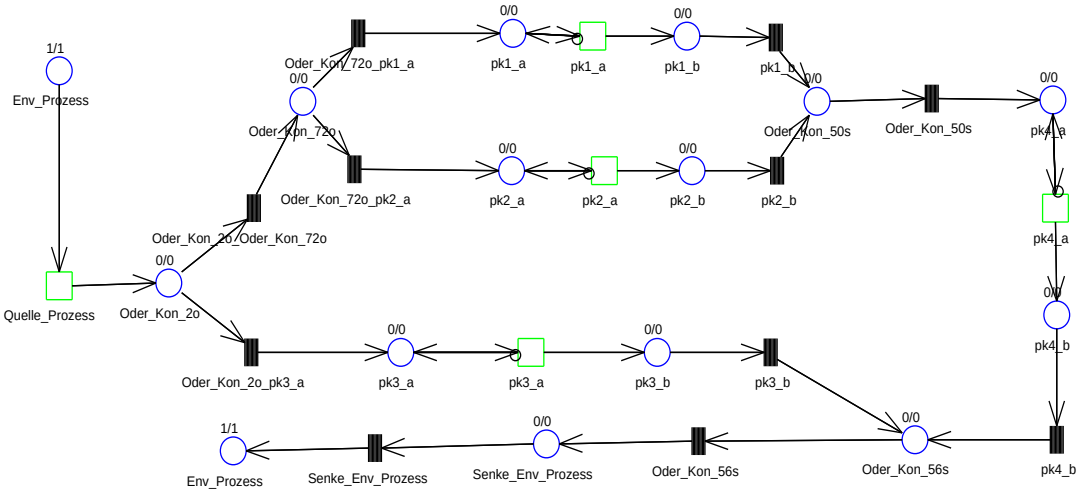


Abbildung A.2: PN Modell: direkte Verbindung bzw. Schachtelung von `Oder-Konnektoren`

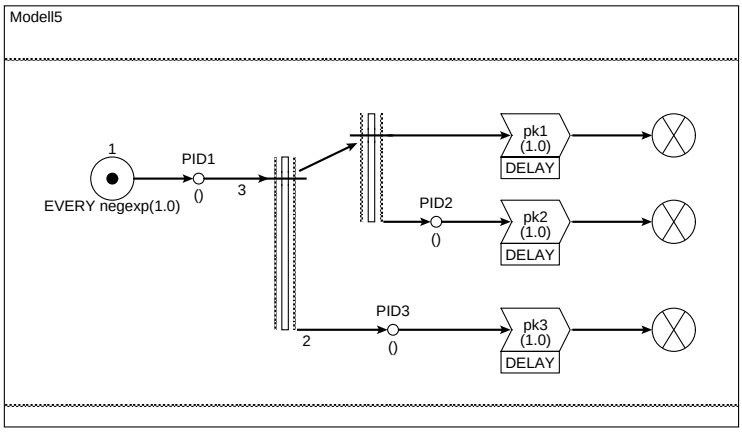


Abbildung A.3: *ProC/B* Modell: direkte Verbindung von Fork-Operatoren

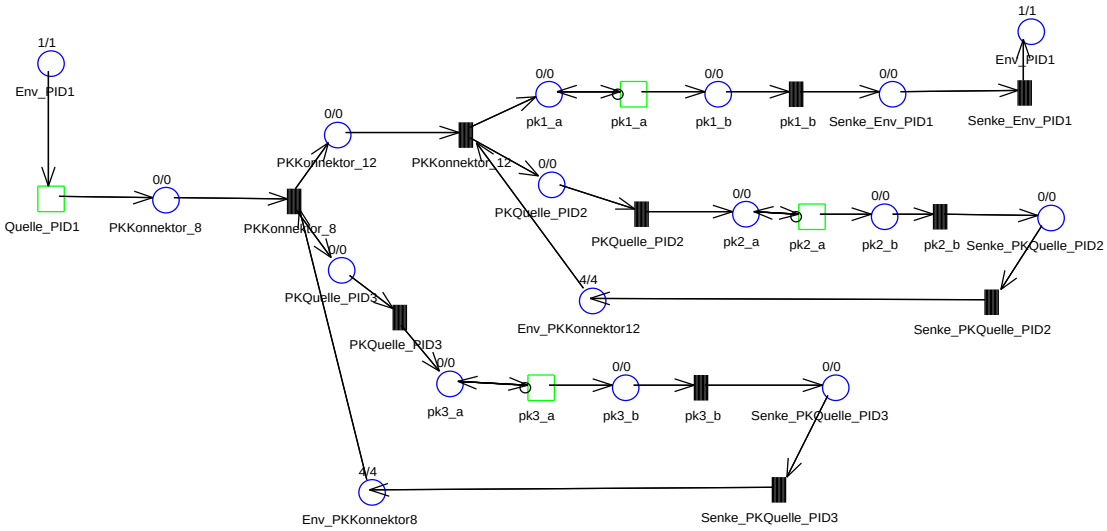


Abbildung A.4: PN Modell: direkte Verbindung von Fork-Operatoren

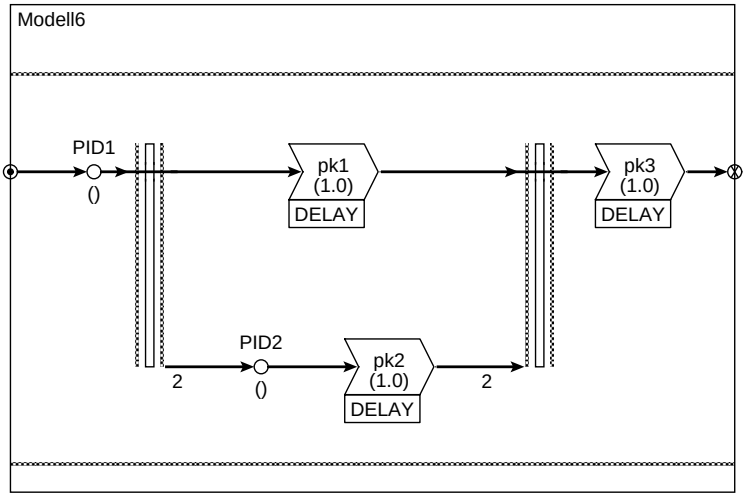


Abbildung A.5: *ProC/B* Modell: Fork-Join Operatoren

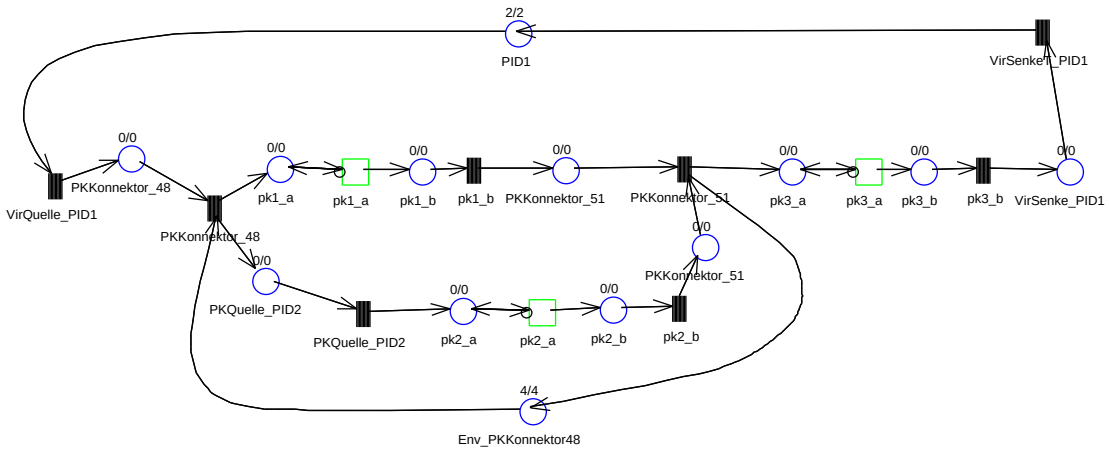


Abbildung A.6: PN Modell: Fork-Join Operatoren

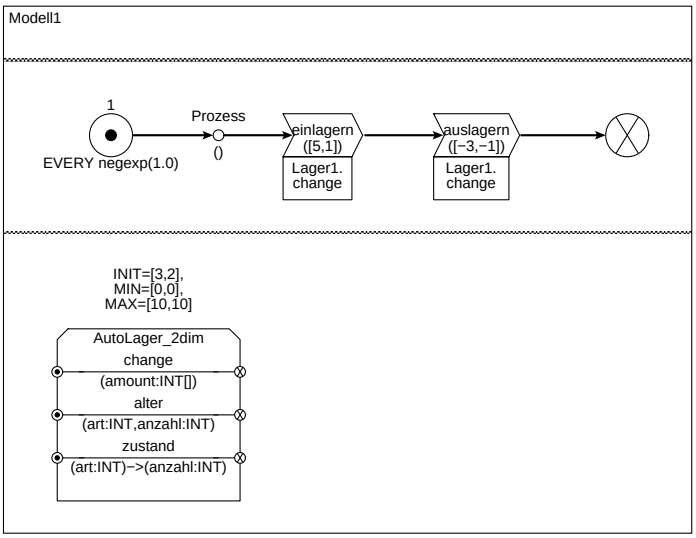


Abbildung A.7: *ProC/B* Modell: Lager

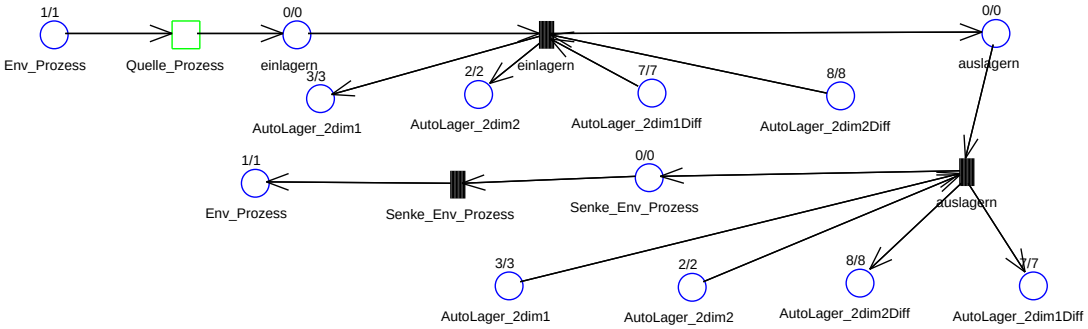


Abbildung A.8: PN Modell: Lager

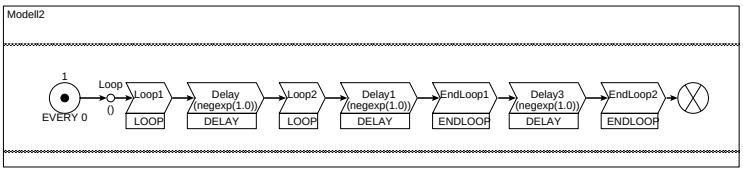


Abbildung A.9: *ProC/B* Modell: geschachtelte Schleifen

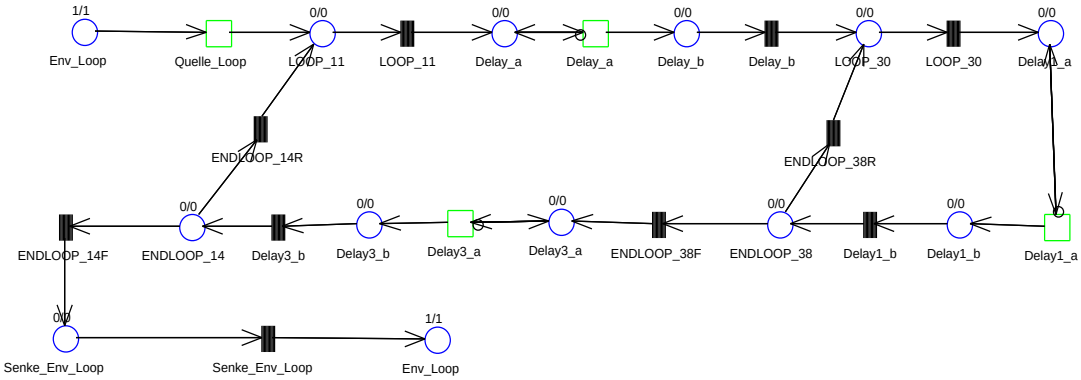


Abbildung A.10: PN Modell: geschachtelte Schleifen

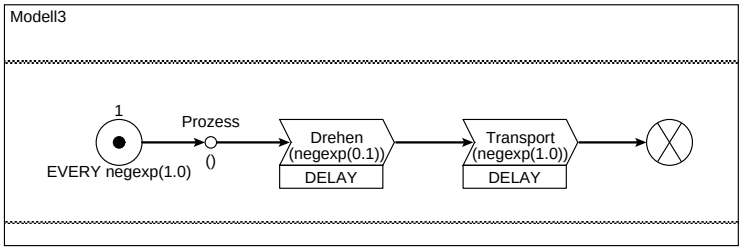


Abbildung A.11: *ProC/B* Modell: lastunabhängige Verzögerungen

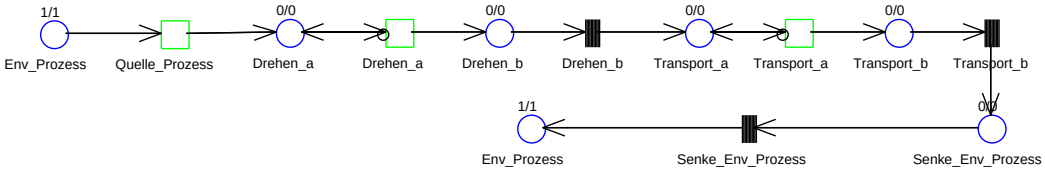


Abbildung A.12: PN Modell: lastunabhängige Verzögerungen

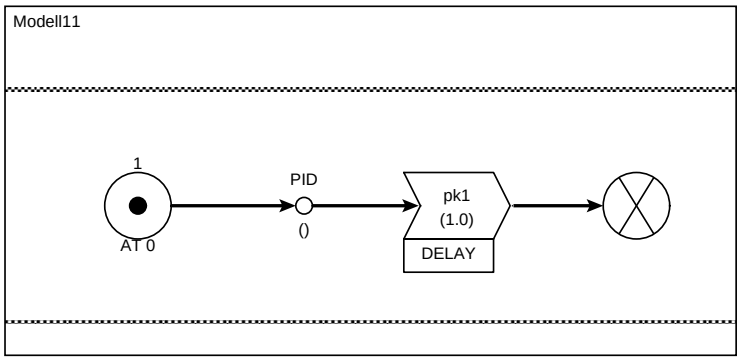


Abbildung A.13: *ProC/B* Modell: singuläre Prozesserzeugung

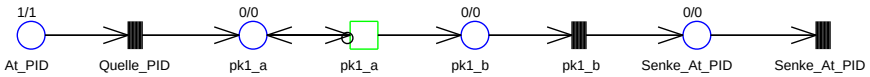


Abbildung A.14: PN Modell: singuläre Prozesserzeugung

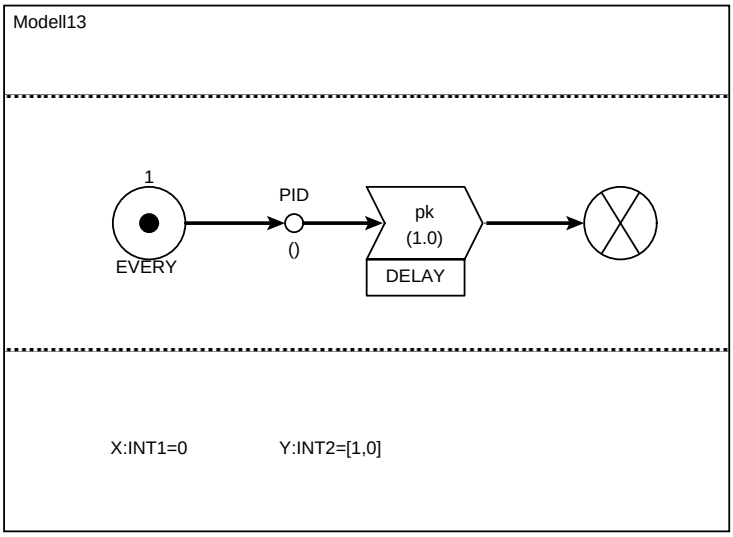


Abbildung A.15: *ProC/B* Modell: Variablen-Deklaration

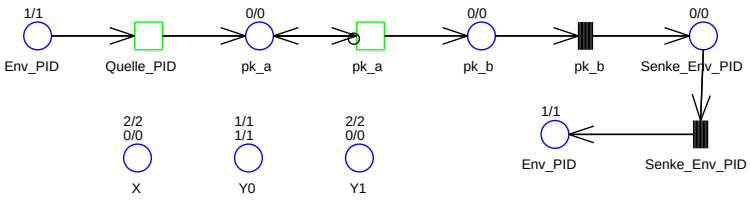


Abbildung A.16: PN Modell: Variablen-Deklaration

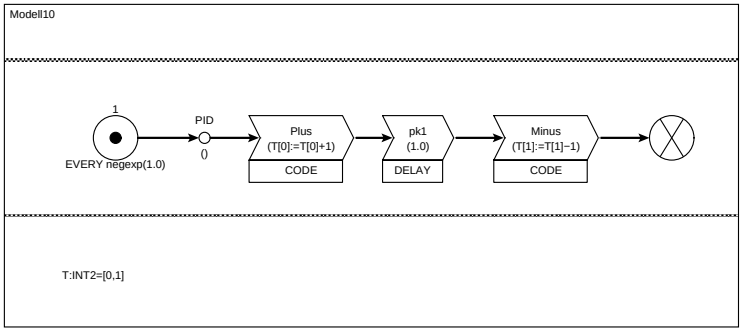


Abbildung A.17: *ProC/B* Modell: Variablenmanipulation

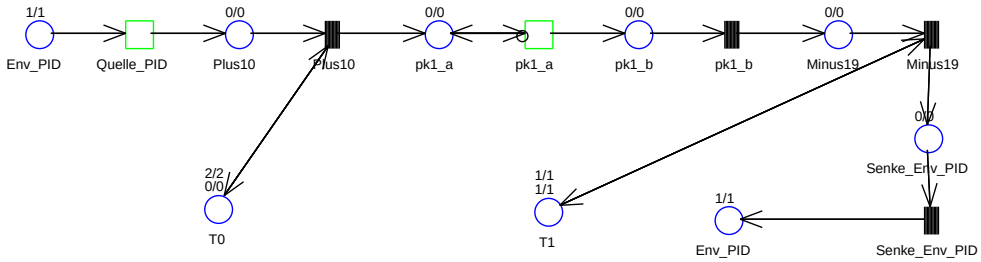


Abbildung A.18: PN Modell: Variablenmanipulation

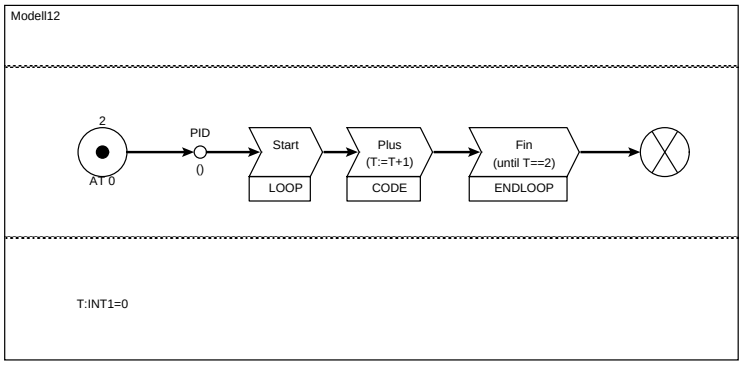


Abbildung A.19: *ProC/B* Modell: Schleife mit boolescher Abbruchbedingung

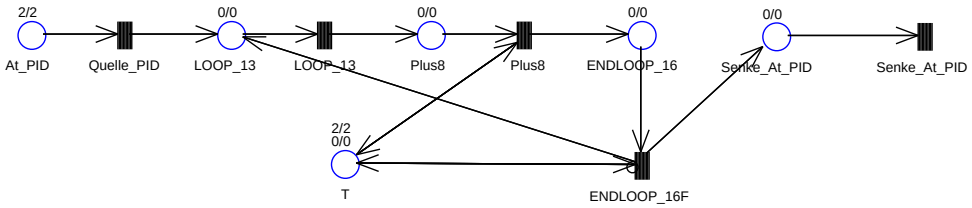


Abbildung A.20: PN Modell: Schleife mit boolescher Abbruchbedingung

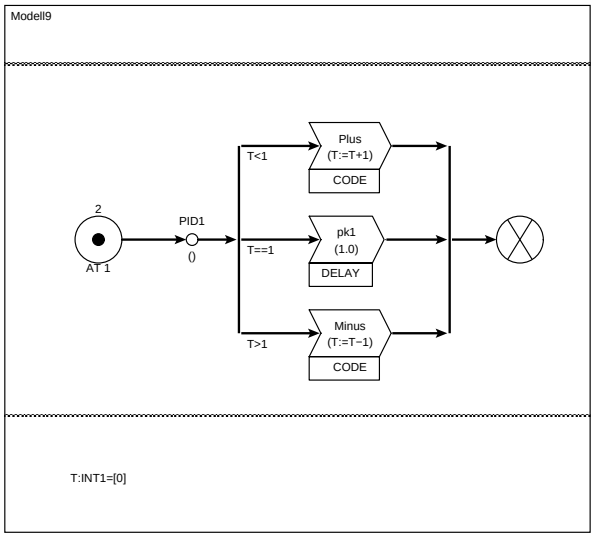


Abbildung A.21: *ProC/B* Modell: öffender, boolescher Oder-Konnektor

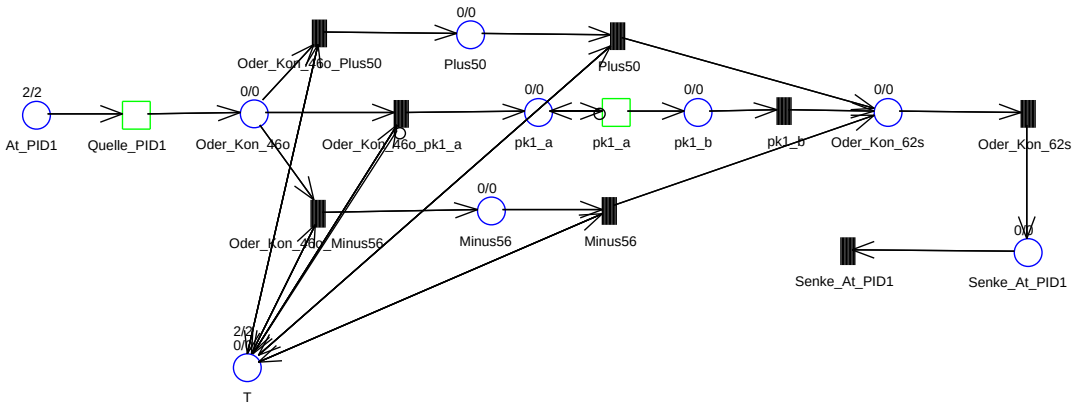


Abbildung A.22: PN Modell: öffender, boolescher Oder-Konnektor

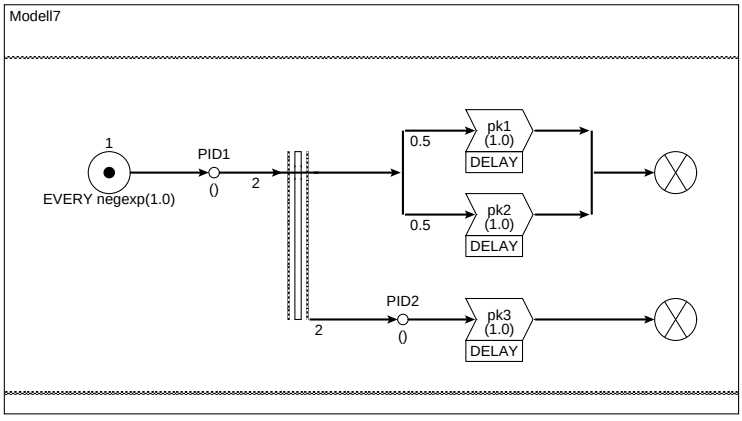


Abbildung A.23: *ProC/B* Modell: Sequenz PK-Konnektor und öffnender ODER-Konnektor

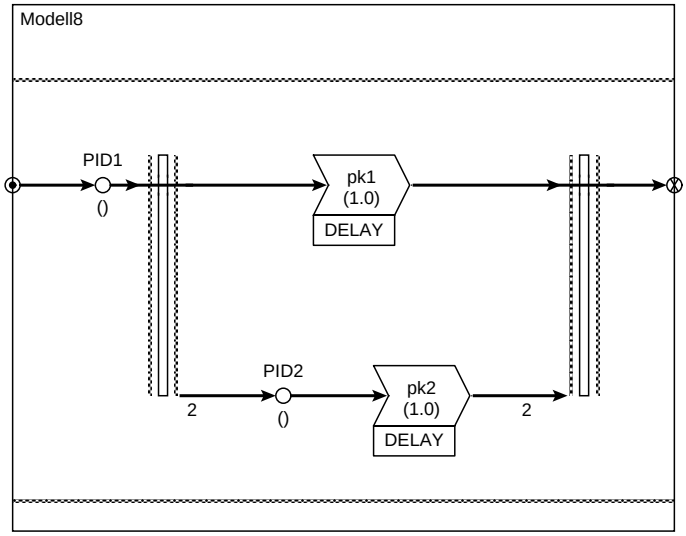


Abbildung A.24: *ProC/B* Modell: Sequenz PK-Konnektor und virtuelle Senke

Anhang B

ProC/B Elemente

ProC/B Element	Darstellung	Semantik
`Funktionseinheit` (`FE`)	Rechteck	Teilmodell
`Server`	Rechteck abgerundet	Bedienstation
`Counter`	Trapez	Semaphor, Zähler
`Quelle`	$\odot$	(Prozess/Last)-generierung
`Senke`	$\otimes$	Ende (Teil)Prozess
`Prozess-Kette` (`PK`)	horizontaler "Flow-Chart"	Prozess-Spezifikation
`PK-Interface`	Kringel	Schnittstelle (Service/Prozess)-Aufruf
`PK-Element` (`PKE`)	pfeilartiges Hexagon	Aktivität in Prozess
`Delay-PKE`	pfeilartiges Hexagon	Zeitverzögerung im Prozess
`Code-PKE`	pfeilartiges Hexagon	Programm-Code basierend auf HIT Syntax
`Loop-PKE`	pfeilartiges Hexagon	kapselt Prozesse in Schleife
`UND-Konnektor`	vertikaler Balken und gestrichelte Linien parallel	Prozess-Aufsplittung ("fork")
`ODER-Konnektor`	vertikaler Balken	Prozess-Alternativen ("branch")
`PK-Konnektor`	vertikaler Balken und gestrichelte Linien parallel	Prozess-Synchronisation
`FE-Variable`	textuell	Variable mit Geltungsbereich in `FE`
`PK-Parameter`	textuell	Aufrufparameter für `PK` (=Geltungsbereich)

Tabelle B.1: *ProC/B* Modellelemente (Auswahl): Graphische Darstellung und Semantik

Literaturverzeichnis

[1] M.Arns, M.Fischer, P.Kemper, C.Tepper. Softwareentwurf eines Übersetzers von B1-PK nach Petri-Netze -Version 1.00 - Interner SFB 559 Bericht 00000, 28.02.2001