

Brandenburgische Technische Universität Cottbus
Institut für Informatik
LS „Datenstrukturen und Softwarezuverlässigkeit“

Dokumentation PED Version 4.3 Benutzerleitfaden

Reimo Tiedemann

4. Juni 1997

Reimo Tiedemann
Dokumentation PED Version 4.3
Benutzerleitfaden

Brandenburgische Technische Universität Cottbus
Institut für Informatik
LS „Datenstrukturen und Softwarezuverlässigkeit“
Postfach 10 13 44
D - 03013 Cottbus
Tel: (+49-355) 69 3884
Fax: (+49-355) 69 3820
Email: ped@informatik.tu-cottbus.de
WWW: <http://www-dssz.informatik.tu-cottbus.de/~wwwdssz>

Inhaltsverzeichnis

1. Einleitung	5
2. Installation	9
2.1. Systemvoraussetzungen	9
2.2. Softwarepaket	9
2.3. Konfiguration und Kommandozeilenparameter	10
3. Editor	11
3.1. Grundeinstellungen	11
3.1.1. Basics-Dialog	11
3.1.2. Show-Dialog	12
3.1.3. Transition-Dialog	13
3.2. Elementare Funktionen	13
3.2.1. Erzeugen von Netzelementen	13
3.2.2. Selektieren und Bewegen	14
3.2.3. Modifizieren der Attribute	15
3.2.4. Rotieren von Transitionen	15
3.3. Hierarchiekonzept	15
3.3.1. Vergrößern von Teilnetzen (Coarse)	16
3.3.2. Einebnen von Subnetzen (Flat)	16
3.3.3. Das Konzept der Kopplungsknoten	16
3.3.4. Zeichnen von Bögen zwischen verschiedenen Teilnetzen	17
3.3.5. Bottom-Up Entwurf	18
3.3.6. Top-Down Entwurf	18
3.4. Das Konzept der logischen Knoten	19
3.5. Spezielle Funktionen	19
3.5.1. Clipboard Funktionen	19
3.5.2. Undo	20
3.5.3. Split	20

3.5.4.	Merge	21
3.5.5.	Squeeze Numbers	21
3.5.6.	Align Vertical/Horizontal	22
3.5.7.	Set of	22
3.5.8.	Search	23
3.6.	Exportfunktionen	24
3.6.1.	Graphische Exportfunktionen	24
3.6.2.	Export zu weiteren Petrinetzwerkzeugen	25
3.6.3.	Generische Exportfunktion	27
4.	Hierarchiebrowser	29
4.1.	Grundeinstellungen	29
4.2.	Navigation	30
4.3.	Selektieren	30
4.4.	Wrap - Unwrap	30
4.5.	Editierfunktionen	30
4.6.	Speichern von Teilnetzen	31
4.7.	Export zu FrameMaker	31
A.	Quickreferenz	33
A.1.	Übersicht aller Funktionen im Editor	33
A.2.	Übersicht aller Funktionen im Hierarchiebrowser	36
B.	Beispiel <i>.pedrc</i> File	37
C.	Beispiele generischer Exportroutinen	39
C.1.	Testexport	39
C.2.	Export zu INA	40

1. Einleitung

Der graphische Petrinetz**ED**itor PED liegt mittlerweile in der Version 4.3 vor. Er ist aus einer Diplomarbeit von Gunnar Czichy entstanden [Czi93] und wurde seitdem an der BTU Cottbus, Lehrstuhl Datenstrukturen und Softwarezuverlässigkeit, bei Prof. Monika Heiner ständig weiterentwickelt. PED ist zentraler Bestandteil einer Petrinetz-Werkzeugumgebung für Unix-Workstations (Abbildung 1.1) mit Anbindung an einen Animator (PEDVisor) und eine Reihe von Analysewerkzeugen (INA, PROD, PEP).¹ Alle Schnittstellen zwischen den einzelnen Komponenten werden über File-Import/Exportfunktionen realisiert.

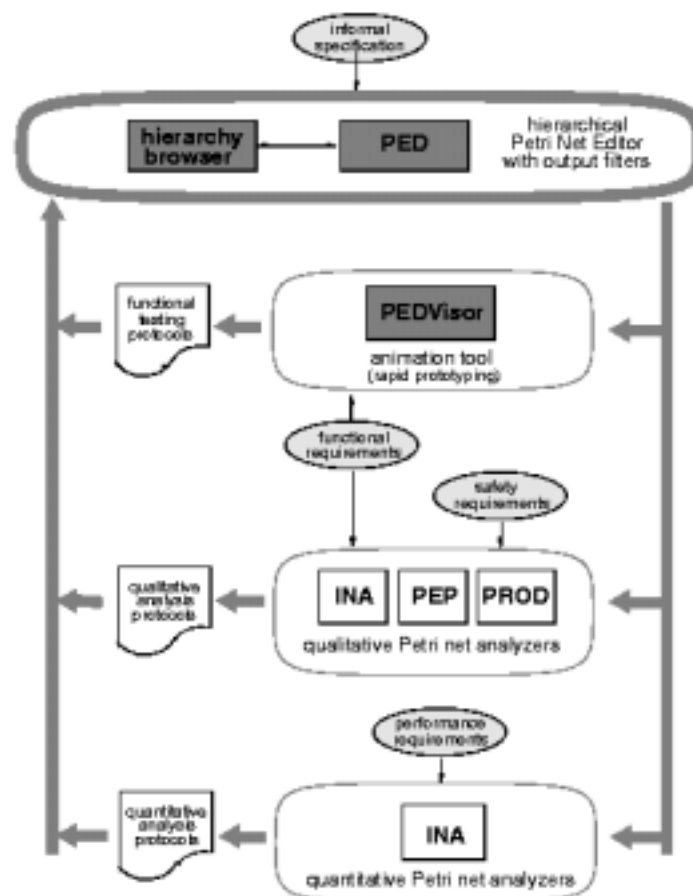


Abbildung 1.1.: Petrinetz-Werkzeugumgebung

¹Einen Einblick in den Entwurfs- und Analyseprozess bietet eine Fallstudie in [HDSml].

Netzklassen PED unterstützt als graphischer Petrinetzeditor neben den klassischen Platz-Transitions-Netzen eine Reihe von Netzklassen aus dem Bereich der Low-Level Netze (LL-Netze). Diese Netzklassen sind Erweiterungen des klassischen Modells.

Das klassische Modell geht von einem bipartiten gerichteten Graphen aus, mit disjunkten, nicht-leeren und endlichen Platz- und Transitions Mengen. Bögen verbinden je ein Element aus der einen Menge mit einem aus der anderen Menge. Nicht unterscheidbare Marken können auf Plätzen mit unbeschränkter Kapazität liegen. Kanten besitzen Vielfachheiten.

Diese Klasse der PT-Netze kann in PED durch Hinzunahme weiterer Element-Attribute erweitert werden. Netzklassen werden nicht explizit eingestellt, sondern sie ergeben sich aus der Menge der benutzten Attribute. Attribute, die man in einem Netz nicht benötigt, können ausgeblendet werden.

Element-Attribute Natürlich ist es nur sinnvoll von „unterstützten Netzklassen“ zu sprechen, wenn die benutzten Attribute auch in die Analyse einbezogen werden können. Aus dem Grund möchte ich an dieser Stelle eine Übersicht aller in PED vorhandenen strukturellen Element-Attribute geben, mit Benennung der in Frage kommenden Netzklasse und einer Angabe der Analysetools, bei denen die Exportfunktion ein derartiges Attribut unterstützt.

Plätze

<i>Attribute</i>	<i>Netzklasse</i>	<i>Analyse</i>
Markierung	PT-Netze	INA, PROD, PEP,
Kapazität	PT-Netze mit Kapazitäten	INA, PROD, PEP,
Zeitdauer	PT-Netze mit Verweilzeiten	–
Zeitintervall		–

Transitionen

<i>Attribute</i>	<i>Netzklasse</i>	<i>Analyse</i>
Fakt	PT-Netze	INA
Priorität	PT-Netze mit Prioritäten	INA
Zeitdauer	Schaltdauer-Netze bzw. D-Netze (<i>timed nets</i>)	INA
Zeitintervall	Schaltdauer-Intervall-Netze bzw. DI-Netze und Intervall-Netze bzw. I-Netze (<i>time nets</i>)	INA

Bögen

<i>Attribute</i>	<i>Netzklasse</i>	<i>Analyse</i>
Inhibitorkante	Inibitor-Netze	–
Resetkante		–
Testkante		–
Vielfachheit	PT-Netze	INA, PROD, PEP
Zeitdauer	PT-Netze mit Kantenverzögerung (<i>arc timed nets</i>)	–
Zeitintervall		–

Zusätzlich gibt es für jedes Element ein "Algebratext"-Attribut. Dabei handelt es sich um einen elementbezogenen, mehrzeiligen Kommentar. Ursprünglich sollten damit algebraische Netze unterstützt werden. Diese Entwicklung wurde aber nicht weiter verfolgt, so daß eine anderweitige Verwendung dieses Attributes möglich ist.

Exportfunktionen PED bietet einige Schnittstellen zu weiteren Petrinetzwerkzeugen standardmäßig an (Abschnitt 3.6). Diese Schnittstellen sind sowohl der vorhergehenden Tabelle als auch der Abbildung 1.1 zu entnehmen.

Wie man sieht, gibt es einige Attribute, die derzeit nicht oder nur teilweise für die Analyse verwendet werden. PED stellt nun eine Möglichkeit zur Verfügung, die Exportfunktionalität dynamisch zu erweitern. Benutzerdefinierte Exportroutinen können in den Editor integriert werden, um weitere Exportformate zu unterstützen (Abschnitt 3.6.3). Dabei hat man Zugriff auf alle vorhandenen strukturellen und graphischen Attribute. Insbesondere das "Algebratext"-Attribut kann als universelles Attribut dienen. Die Interpretation des Textes innerhalb der Exportfunktion bzw. im Analysewerkzeug kann beliebig erfolgen.

Neben den Exportmöglichkeiten zu Petrinetzwerkzeugen gibt es weitere Exportfunktionen zu verschiedenen Grafikformaten. Damit können die im Editor gezeichneten Netze in anderen Grafikprogrammen weiterbearbeitet (FrameMaker, XFig), ausgedruckt (PostScript) und in eigene Publikationen eingebunden werden.

Hierarchiekonzept Der Petrinetzeditor erlaubt die Erstellung von hierarchischen Platz-Transitions-Netzen (Abschnitt 3.3). Mehrere Netzelemente können zu Subnetzen zusammengefaßt und einem Grobknoten (Coarse-Knoten) zugeordnet werden. In den einzelnen Subnetzen sind die Schnittstellen zum Gesamtnetz durch sogenannte Kopplungsknoten sichtbar.

Die bei der Modellierung komplexer Systeme mit klassischen PT-Netzen entstehenden Netze können sehr groß werden. Das von PED unterstützte Hierarchiekonzept dient einem strukturierten Entwurf derartiger Netze. Mit PED genießt der Benutzer ein hohes Maß an Freiheit während des Entwurfsprozesses. Neben dem standardmäßigen Bottom-Up Entwurf liegen die Stärken von PED vor allem in der Unterstützung des Top-Down Entwurfs. Subnetze können auf sehr einfache Weise nachträglich verfeinert werden. Die Wiederverwendbarkeit von Netzbausteinen erleichtert diesen Prozeß.

Logische Knoten Eine weiteres Mittel zur übersichtlicheren Darstellung von Petrinetzen sind logische Knoten. Knoten können in PED mehrere graphische Ausprägungen besitzen, die alle den selben Knoten repräsentieren.

Übersicht PED besteht aus zwei eigenständigen Komponenten, die zusammengenommen die Arbeitsumgebung bilden (siehe Abbildung 1.2). Die Hauptkomponente stellt die Zeichenfläche zur Verfügung. Die Funktionalität der als "Editor" bezeichneten Komponente wird im Abschnitt 3 beschrieben. Parallel dazu unterstützt der Hierarchiebrowser die Arbeit mit hierarchischen Petrinetzen. Seine Funktionalität wird im Abschnitt 4 erläutert. Eine vollständige Auflistung aller in PED zur Verfügung stehenden Funktionen ist im Anhang A innerhalb einer Quickreferenz zu finden.

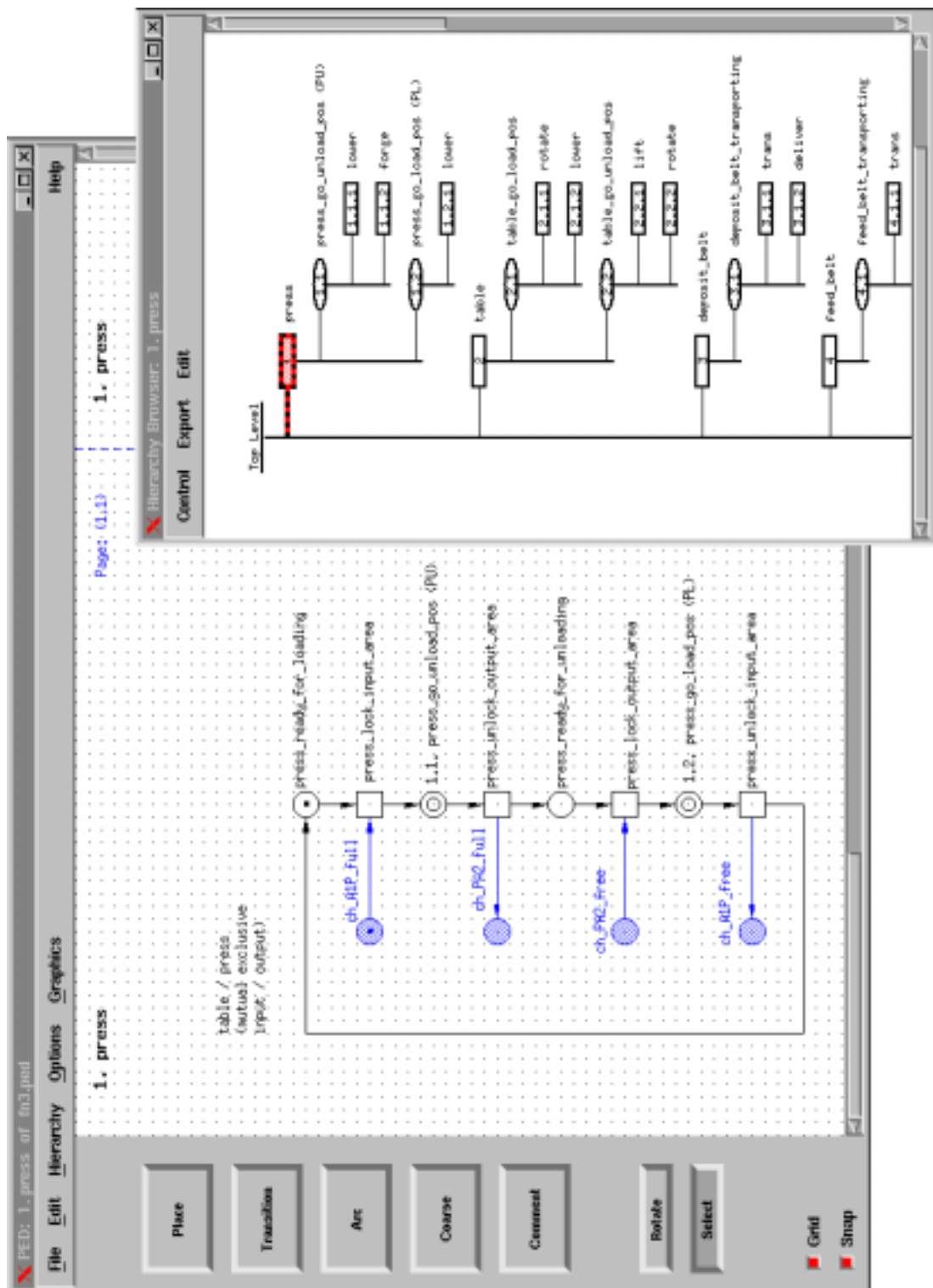


Abbildung 1.2.: Snapshot der Arbeitsumgebung, Editor und Hierarchiebrowser

2. Installation

2.1. Systemvoraussetzungen

Der Petrinetzeditor PED wurde auf einer SUN-Workstation unter Solaris entwickelt. Außerdem gibt es eine Portierung nach Linux. Portierungen auf weitere Unix-Plattformen sind möglich (siehe unten).

Neben dem X-Windows-System sollte die OSF/Motif-Bibliothek (ab Version 1.2x) vorhanden sein. In der Distribution ist für beide Plattformen (Solaris, Linux) entweder eine dynamisch oder eine statisch gelinkte Version enthalten, bei der alle X11-, Xt- und Motif-Bibliotheken dazugelinkt wurden. Im Normalfall (bei vorhandenem Motif) sollte aber die dynamisch gelinkte Version ausreichen.

2.2. Softwarepaket

Copyrights Der Petrinetzeditor PED Version 4.3 wird zur nichtkommerziellen Nutzung frei zur Verfügung gestellt. Die Distribution enthält das ausführbare Programm. Die Programmquellen hingegen sind von dieser Freigabe ausgeschlossen.

Sollten Probleme mit den erzeugten *binaries* auftreten oder der Wunsch nach Portierung auf eine weitere Unix-Plattform bestehen, dann können Sie jederzeit zu uns Kontakt aufnehmen.

Postadresse: BTU Cottbus
Institut für Informatik, LS „Datenstrukturen u. Softwarezuverlässigkeit“
Postfach 10 13 14
D - 03013 Cottbus

Email: ped@informatik.tu-cottbus.de

WWW: <http://www-dssz.informatik.tu-cottbus.de/~wwwdssz/index.html>

Distribution Für die Distribution stehen vier verschiedene gepackte Verzeichnisarchive zur Verfügung. Je nach Hardwareplattform und vorhandener Motif-Bibliothek reicht es aus, eine der vier Dateien zu laden.

<i>pedV4.3_solaris_dynamic.tar.gz</i>	Dynamische Version für Solaris
<i>pedV4.3_solaris_static.tar.gz</i>	Statische Version (X11, Xt, Xm) für Solaris (bis 2.4)
<i>pedV4.3_linux_dynamic.tar.gz</i>	Dynamische Version für Linux (Elf, Motif 2.0)
<i>pedV4.3_linux_static.tar.gz</i>	Statische Version für Linux (Elf)

Diese Dateien können mit dem Kommando "tar zxvf <filename>" in einem geeigneten Verzeichnis entpackt werden.

Verzeichnisbaum Eine vollständige Distribution enthält folgende Unterverzeichnisse:

<i>bin</i>	Ausführbares Programm (<i>ped</i> File)
<i>config</i>	Konfigurationsdatei (<i>.pedrc</i> File), Ressourcedatei (<i>XmPed</i> File)
<i>doc</i>	Dokumentation (PostScript)
<i>examples</i>	Beispielnetze
<i>lib</i>	Dynamische Exportroutinen ("*.lua" Files)

2.3. Konfiguration und Kommandozeilenparameter

Der Distribution liegen zwei Konfigurationsdateien bei (im Verzeichnis *config*), die noch an eine geeignete Stelle kopiert werden müssen.

.pedrc File Das *.pedrc* File aus dem Verzeichnis *config* kann optional ins eigene Heimatverzeichnis kopiert werden. Falls eine solche Datei beim Programmstart vorhanden ist, wird die generische Exportfunktionalität von PED aktiviert (siehe Abschnitt 3.6.3). In dieser Datei werden die zusätzlichen Menüeinträge im *File - Export* - Menü konfiguriert. In diesem Fall sind das zwei weitere Einträge, ein Testexport und ein Beispielexport zu INA. Zusätzlich muß noch innerhalb dieser Datei der Pfad zu den eigentlichen Exportroutinen ("*.lua" Files) angegeben werden. Die Syntax dieser Konfigurationsdatei dürfte keine Probleme bereiten, so daß das im Anhang B angegebene Beispiel ausreichen sollte.

"*.lua" Files Im Verzeichnis *lib* stehen zwei Exportroutinen, die zur Laufzeit des Programms interpretiert werden können. Sie sollen als Beispiel dienen, um eigene Exportroutinen schreiben zu können (siehe Abschnitt 3.6.3). Diese Dateien mit der Endung ".lua" können an beliebiger Stelle stehen, entweder systemweit einheitlich in einem Verzeichnis oder privat für eigene Exportroutinen. Wichtig ist nur, daß in der Datei *.pedrc* der eingestellte Pfad auf diese Dateien verweist.

XmPed File Weiterhin befindet sich im Verzeichnis *config* die Ressourcedatei. In ihr werden Programmeinstellungen wie Farben, Fonts, Tastaturkürzel, Buttonbeschriftungen u.ä. definiert. Für eine korrekte Programminitialisierung ist sie in eines der unter X-Windows vorgesehenen Standardverzeichnisse zu kopieren (z.B. "/usr/lib/X11/app-defaults"). Falls dafür keine Rechte vorhanden sind, bzw. falls man seine eigenen Einstellungen bevorzugt, genügt ein Kopieren in das Heimatverzeichnis.

Kommandozeilenparameter Es gibt eine einzige Kommandozeilenoption für PED. Beim Aufruf des Programms kann gleich eine zu ladende Datei angegeben werden. PED startet und lädt das entsprechende Netz in die Arbeitsfläche. Die Dateierdung ".ped" kann weggelassen werden.

```
ped [<filename>]
```

3. Editor

3.1. Grundeinstellungen

3.1.1. Basics-Dialog

Einige elementare Einstellungen im Editor können über den Menüeintrag *Options - Basics* vorgenommen werden (Abbildung 3.1).

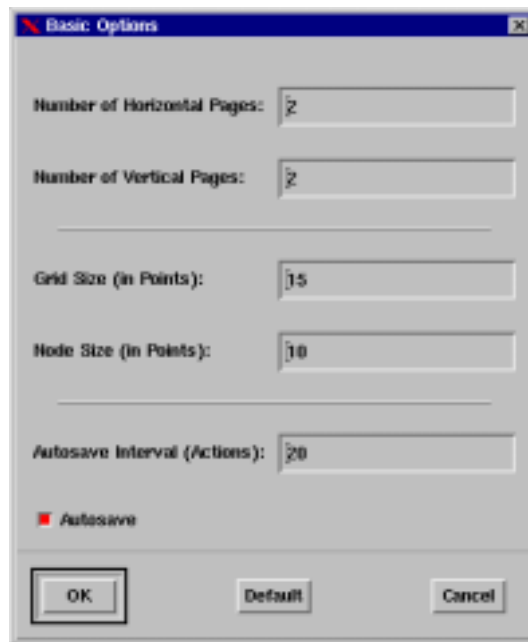


Abbildung 3.1.: Bildschirmausschnitt Basics - Dialog

Größe der Arbeitsfläche Die Größe der Arbeitsfläche wird durch die Anzahl der horizontalen Seiten mal Anzahl der vertikalen Seiten angegeben. Eine Seite hat DIN-A4-Format. Die Arbeitsfläche ist auf 5x5 DIN-A4-Seiten beschränkt.

Grid Als Hilfsmittel zur Positionierung der Netzelemente kann ein Grid eingeschaltet werden (Toggle-Button im Hauptfenster links unten). Bei zusätzlich eingeschalteter *Snap*-Funktion werden alle Netzelemente (inklusive der Knickpunkte der Bögen) an diesem Gitter ausgerichtet. Der gewünschte Gitterabstand wird in Pixel angegeben. Der Mindestabstand beträgt 5 Pixel.

Knotengröße Die Größe der Knoten wird global festgelegt und ist somit für alle Knoten gleich. Sie wird ebenfalls in Pixel angegeben und legt den Radius (bzw. Seitenlänge/2) der Knoten fest. Die Mindestknotengröße beträgt 5 Pixel.

Autosave-Funktion Der Editor besitzt eine Autosave-Funktion. Die Speicherzeitpunkte der Sicherungskopie (“*.ped.save” Datei) werden durch die Anzahl der verstrichenen Aktionen im Editor bestimmt.

3.1.2. Show-Dialog

Der *Options - Show* - Dialog bietet die Möglichkeit, einzelne Komponenten im Editor hervorzuheben bzw. auszublenden (Abbildung 3.2).



Abbildung 3.2.: Bildschirmausschnitt Show - Dialog

Attribute von Netzelementen Die Sichtbarkeit der Attribute kann für jeden Attributtyp einzeln festgelegt werden. Damit kann man nicht benötigte Attributtypen ausblenden. Diese Festlegungen werden ebenfalls bei den Exportfunktionen für die graphischen Dateiformate beachtet.

Freie Kommentare Kommentare können ausgeblendet werden.

Basisobjekte und Konflikt-Cluster Der Editor bietet die Fähigkeit, Basisobjekte (Basisteilnetze [Wik91]) oder Konflikt-Cluster (statische Konflikte) des editierten Netzes hervorzuheben. Diese werden im Editor rot dargestellt.

3.1.3. Transition-Dialog

Im *Options - Transition* - Dialogfenster wird der Standard-Transitionstyp festgelegt (Abbildung 3.3). Defaultmäßig werden Transitionen quadratisch gezeichnet. Es gibt aber auch die Möglichkeit, Transitionen rechteckig oder linienförmig und gefüllt darzustellen. Die Darstellungsform einzelner Transitionen kann nachträglich über den Attribut-Dialog des Knotens geändert werden (siehe Abschnitt 3.2.3).



Abbildung 3.3.: Bildschirmausschnitt Transition-Form - Dialog

3.2. Elementare Funktionen

Der Editor arbeitet in verschiedenen Eingabemodi, die auf der linken Button-Leiste angewählt werden können. Befindet sich der Mauszeiger über der Arbeitsfläche, wird der momentan aktivierte Modus durch eine entsprechend veränderte Cursor-Form verdeutlicht.

Die speziell für einen Modus bestimmten Aktionen sind über die linke Maustaste zu erreichen. Die mittlere und die rechte Maustaste hingegen ist in allen Modi einheitlich belegt. Mit der mittleren Maustaste können einzelne Netzelemente (de)selektiert werden. Die rechte Maustaste bietet das Edit-Menü als Popup-Fenster auf der Arbeitsfläche.

3.2.1. Erzeugen von Netzelementen

Die oberen, größeren Buttons führen zu den Erzeugungsmodi der einzelnen Netzelemente. Plätze, Transitionen, Coarse-Knoten und Kommentare können durch Klick mit der linken Maustaste auf die gewünschte Position erzeugt werden. Kommentare werden in dem folgenden Dialogfenster eingegeben. Falls auf diese Art ein Coarse-Knoten erzeugt wurde, so ist das darunterliegende, leere Subnetz miterstellt worden (siehe Abschnitt 3.3.6).

Befindet sich ein schon vorhandenes Netzelement unter dem Mauszeiger, wird kein neues Element erzeugt, sondern der Editor verhält sich entsprechend dem Selektmodus. Damit ist ein bequeres Arbeiten möglich, da ein Umschalten bei den typischen aufeinanderfolgenden Arbeiten wie: Erzeugen, Ändern der Attribute, Bewegen des Elementes oder der Attribute, Erzeugen eines weiteren Elementes ... in den meisten Fällen nicht nötig ist.

Zeichnen von Bögen Dazu muß der linke Mausbutton auf dem Quellknoten niedergedrückt und über dem Zielknoten wieder losgelassen werden. Damit wird der Bogen als geradlinige Verbindung zwischen den beiden Knoten gezeichnet. Bögen können darüberhinaus beliebig viele Knickpunkte besitzen. Zusätzliche Knickpunkte werden erzeugt, indem ein Bögen selektiert wird und die sichtbaren Knickpunkte verschoben werden.

Soll ein Bogen gleich in mehreren Bogenzügen gezeichnet werden, kann der Mausbutton auch über einer freien Fläche losgelassen werden. Die Kante wird dann bis zu diesem ersten Knickpunkt gezeichnet und der Editor begibt sich in einen speziellen Kanteneingabemodus, der auch abgeschlossen werden muß. Dazu können nun in einer Folge von Klicks mit der linken Maustaste weitere Knickpunkte hinzugefügt werden. Bei der Anwahl eines Zielknotens endet dieser Kanteneingabemodus. Ist die Verbindung erlaubt, wird der Kantenzug vollendet, im anderen Fall wird der schon gezeichnete Kantenzug gelöscht (wenn man sich entscheidet, das Zeichnen des Bogens abubrechen, genügt ein Klick auf den Quellknoten).

Das Zeichnen von Kantenzügen auf die eben beschriebene Art ist sehr hilfreich, wenn man den ersten und den letzten auf einer Linie ausgerichteter Knoten miteinander verbinden möchte. Damit wird verhindert, daß der Bogen quer durch alle anderen Knoten verläuft. Wollte man die Knickpunkte solch eines Bogens verschieben, wäre dies mit einiger Mühe verbunden.

3.2.2. Selektieren und Bewegen

Selektieren Im Selektmodus lassen sich allgemein Elemente selektieren, verschieben oder modifizieren. Wie oben schon erwähnt, tritt dieser Modus auch in Kraft, wenn sich innerhalb der verschiedenen Erzeugungsmodi ein Element unter dem Mauszeiger befindet.

Wenn ein Element mit der linken Maustaste angeklickt wird, so ist es exklusiv selektiert und hervorgehoben. Selektionen anderer Elemente werden damit rückgängig gemacht. Mit der mittleren Maustaste können zusätzlich weitere Elemente einzeln (de)selektiert werden.

Zeigt der Mauszeiger beim Klick mit der linken Maustaste auf ein freies Feld, kann durch die gedrückt gehaltene Maustaste eine rechteckige Selektionsbox aufgezogen werden. Alle innerhalb dieser Box liegenden Elemente sind damit selektiert. Alle vorherigen Selektionen sind aufgehoben.

Eine Selektion kann zusätzlich durch die Menüpunkte *Edit - Select All Nodes* und *Edit - Select Logical Nodes* erreicht werden.

Bewegen Elemente müssen selektiert sein, bevor sie mit gedrückt gehaltener linker Maustaste an eine neue Position verschoben werden können. Dazu ist eines der selektierten Elemente anzuwählen. Alle anderen Elemente werden entsprechend verschoben. Falls Bögen existieren, deren Vorgänger- und Nachfolgerknoten auf diese Weise bewegt werden, werden auch alle Knickpunkte entsprechend verschoben. Ein einzelnes Element oder ein Attributblock kann auf die gleiche Art bewegt werden, die vorherige Selektion kann aber entfallen, da sie implizit geschieht.

Plätze, Transitionen und Bögen besitzen zwei Attributblöcke, die unabhängig voneinander frei positioniert werden können. Der eine Block beinhaltet das Algebra-Attribut, der andere die restlichen Attribute. Da Coarse-Knoten kein Algebra-Attribut besitzen, haben sie nur einen Attributblock.

Die Knickpunkte der Bögen werden auf die schon beschriebene Weise verschoben. Auch dazu muß die Kante selektiert sein, wodurch die verschiebbaren Knickpunkte sichtbar werden.

3.2.3. Modifizieren der Attribute

Die Attribute der Elemente können in allen schon beschriebenen Modi durch Doppelklick mit der linken Maustaste auf ein solches Element oder dessen Attribute geändert werden. Dazu wird ein Attribut-Dialogfenster verwendet (Abbildung 3.4). Kommentare werden auf die gleiche Art editiert.



Abbildung 3.4.: Beispiel: Bildschirmausschnitt Arc-Attributes - Dialog

Für ein komfortables Editieren der Attribute kann auch eine ganze Gruppe von Elementen über einen der Menüeinträge *Edit - Set of...* gleichzeitig bearbeitet werden. Dies ist beispielsweise vorteilhaft, wenn man nur den Namen der Knoten einer Bildschirmseite (Hierarchieebene) einen Prä- oder Suffix zuordnen möchte (siehe Abschnitt 3.5.7).

3.2.4. Rotieren von Transitionen

Der Rotatemodus ermöglicht das Rotieren von Transitionen. Durch Klick auf eine Transition wird diese um 45° gedreht.

3.3. Hierarchiekonzept

Der Petrinetzeditor erlaubt die Erstellung von hierarchischen Platz-Transitions-Netzen. Das von PED unterstützte Hierarchiekonzept ist rein syntaktischer Natur. Mehrere Netzelemente können zu Subnetzen zusammengefaßt werden. Solch ein Subnetz wird durch ein Grobknoten (Coarse-Knoten) ersetzt und diesem zugeordnet. Durch wiederholtes Zusammenfassen von Netzelementen und Teilnetzen entsteht eine hierarchische Struktur. Die eigentliche Netzsemantik wird davon

nicht berührt. Da die Modellierung mit Platz-Transitions-Netzen schnell recht groß und unübersichtlich werden kann, dient dieses Konzept einzig einem strukturierten Entwurf (Bottom-Up, Top-Down) und einer übersichtlicheren Darstellung der Netze.

3.3.1. Vergrößern von Teilnetzen (Coarse)

Um eine Menge von Netzelementen in ein Subnetz zu bewegen, müssen diese zuvor selektiert werden. Danach wird durch Anwahl des Menüpunktes *Hierarchy - Coarse* ein neues Subnetz erstellt und alle selektierten Knoten, inklusiv ihrer internen Bögen, werden in das erstellte Subnetz bewegt. Die selektierten Knoten und ihre internen Bögen verschwinden aus dem aktuellen Teilnetz. An ihre Stelle tritt ein Coarse-Knoten (im Schwerpunkt der zuvor selektierten Knoten), der alle Kantenverbindungen in das erstellte Subnetz auf sich vereint.

Coarse-Knotentyp Als Erscheinungsform des Coarse-Knoten wird in jedem Fall erst einmal die Standardvariante gewählt. Damit wird keine Aussage darüber getroffen, ob das entstandene Subnetz transitionsberandet oder platzberandet ist. Falls alle hinein- und hinausführenden Bögen innerhalb des Subnetzes einheitlich mit Transitionen oder Plätzen verbunden sind, kann eine speziellere Form für den Coarse-Knoten gewählt werden. Für derartige Grobtransitionen bzw. Grobplätze gibt es eine entsprechende Darstellungsvariante, die auf den Typ der Randknoten hindeutet. Die Standardvariante kann somit gemischtberandeten Subnetzen vorbehalten bleiben.

Diese Einstellung muß grundsätzlich vom Benutzer vorgenommen werden, sie wird durch den Editor nicht überprüft. Die Auswahl ist innerhalb des Attribut-Diologs für den Coarse-Knoten möglich, der durch Doppelklick mit der linken Maustaste auf den Knoten erreicht werden kann.

Wechsel des Subnetzes Neben der Auswahl des Coarse-Knotentyps bietet dieser Dialog die Möglichkeit, einen Namen für das Subnetz zu vergeben und auf bequeme Weise in das darunterliegende Subnetz zu wechseln. Die selbe Aktion kann über den Menüpunkt *Hierarchy - Level Down* erreicht werden. Für die entgegengesetzte Richtung, um in der Hierarchie nach oben zu wandern, ist der Menüpunkt *Hierarchy - Level Up* vorgesehen. Für beide recht häufig vorkommenden Aktionen sind auch Tastaturkürzel vergeben (siehe Abschnitt A). Eine noch komfortablere Möglichkeit, zwischen verschiedenen Subnetzen zu wechseln, bietet der Hierarchiebrowser (siehe Abschnitt 4.2).

3.3.2. Einebnen von Subnetzen (Flat)

Die reverse Operation von *Coarse* ist *Hierarchy - Flat*. Damit werden die Elemente eines Subnetzes wieder in das aktuelle Teilnetz eingefügt und der selektierte Coarse-Knoten verschwindet. Im aktuellen Teilnetz wird dazu ein wenig Platz geschaffen und die dazukommenden Netzelemente werden an gleicher Stelle wie im Subnetz positioniert. Das Ergebnis dieser Aktion muß in den meisten Fällen nachbearbeitet werden, da die Positionierung i. d. R. nicht befriedigend ausfällt.

3.3.3. Das Konzept der Kopplungsknoten

Neben den Coarse-Knoten als Hilfsmittel für einen strukturierten Entwurf, werden zusätzlich Kopplungsknoten in die erstellten Subnetze eingefügt. Diese und die zu bzw. von ihnen verlau-

fenden Bögen sind im Editor blau gezeichnet. Sie repräsentieren die Verbindungen zu weiteren Teilnetzen. Coarse-Knoten scheiden als Kopplungsknoten aus. An ihrer Stelle werden die vom Coarse-Knoten vertretenen Transitionen bzw. Plätze verwendet.

Da es sich hierbei um ein logisches Konzept handelt, sind diese Elemente nur weitere graphische Ausprägungen der Originalelemente. Sie dienen der übersichtlicheren Darstellung der Umgebung von Teilnetzen und vor allem, um weitere Bögen zwischen verschiedenen Teilnetzen einfügen zu können (siehe nächster Abschnitt). Erlaubte Aktionen auf diesen Elementen wirken sich direkt auf den Originalknoten aus (Änderung der Attribute, Einfügen von Kanten, Verschmelzen von Knoten).

Löschen von Kopplungselementen Kopplungsknoten können nicht direkt gelöscht werden. Solch eine Aktion sollte mit dem Originalknoten vorgenommen werden. Löscht man hingegen eine Kante zu einem Kopplungsknoten, so verschwindet dieser aus dem Teilnetz, sofern er nicht noch weitere Verbindungen zu einem Randknoten dieses Teilnetzes unterhält. Das Verschwinden eines Kopplungsknoten verändert aber nur die Schnittstelle dieses Teilnetzes (und möglicherweise weiterer) und hat ansonsten keinen Einfluß auf den Originalknoten selbst. Anders hingegen verhält es sich mit der gelöschten Kante. Diese und mit ihr auch die Originalkante (die zwischen diesen Teilnetzen verläuft) und alle weiteren graphischen Repräsentationen werden komplett gelöscht.

3.3.4. Zeichnen von Bögen zwischen verschiedenen Teilnetzen

Das Zeichnen von Bögen zwischen verschiedenen Teilnetzen ist auf verschiedene Weise möglich. Natürlich können Teilnetze auch wieder eingeebnet werden, um derartige Kanten zu zeichnen. Das sollte aber nur als letztes Mittel in Betracht gezogen werden müssen.

Kopplungsknoten ist schon vorhanden Im einfachsten Fall kann ein weiterer Bogen zwischen einem schon existierenden Kopplungsknoten und einem Knoten des aktuellen Teilnetzes (Nicht-Kopplungsknoten) gezeichnet werden. Damit wird eine Kante zwischen dem vom Kopplungsknoten repräsentierten Knoten (dem Originalknoten) und dem Knoten des aktuellen Teilnetzes in das Petrinetz aufgenommen. Außerdem werden alle zusätzlich benötigten Kopplungselemente erstellt. Solch eine Kante kann aber nur gezeichnet werden, wenn sich schon beide zu verbindenden Knoten (ein Originalknoten und ein Kopplungsknoten) im aktuellen Teilnetz befinden.

Schwieriger ist es, wenn das aktuelle Teilnetz keinen passenden Kopplungsknoten enthält. Dafür können zwei Strategien angewendet bzw. kombiniert werden.

Kopplungsknoten nach "unten" kopieren Zum einen gibt es immer die Möglichkeit, zwischen einer Transition oder einem Platz (egal ob Original- oder Kopplungsknoten) und einem Coarse-Knoten einen Bogen zu zeichnen. Damit wird implizit in dem Subnetz ein neuer Knoten erstellt (ob es sich dabei um einen Platz oder eine Transition handelt hängt davon ab, mit welchem Knotentyp die Verbindung zum Coarse-Knoten gezeichnet wurde). Zusätzlich werden die entsprechenden Kopplungselemente in das Subnetz eingefügt, das heißt, der "obere" Knoten bekommt eine weitere graphische Ausprägung (einen Kopplungsknoten) in dem Subnetz. Wenn man dieses Verfahren fortsetzt, kann man den gewünschten Kopplungsknoten bis in das Subnetz kopieren, in dem die Kante gezeichnet werden soll. Die Knoten, die "unterwegs" unötigerweise

erstellt wurden, werden gelöscht. Den untersten neu erstellten Knoten könnte man gleich dazu benutzen, ihn mit dem gewünschten Knoten zu verschmelzen (siehe Abschnitt 3.5.4) oder man zeichnet die Verbindung neu. Für dieses Verfahren gilt aber die Einschränkung, daß beide Knoten auf einem Pfad zur Wurzel der Hierarchie (Toplevel) liegen müssen.

Originalknoten nach “unten” schieben Wenn man eine Transition oder einen Platz auf eine Coarse-Knoten “wirft”, wird dieser Knoten in das Subnetz verschoben. Um nun einen Bogen zwischen Knoten verschiedener Teilnetze zu zeichnen, erstellt man in einem Teilnetz, das beide Knoten innerhalb von Subnetzen enthält, einen neuen Platz und eine neue Transition und verbindet diese mit einer Kante. Die Strategie läuft nun darauf hinaus, beide neu erstellten Knoten bis in die gewünschten Subnetze “hinunterzudrücken” und dort mit den eigentlichen Knoten zu verschmelzen (siehe Abschnitt 3.5.4).

3.3.5. Bottom-Up Entwurf

Der Bottom-Up Entwurf wird automatisch durch die Möglichkeit der Vergrößerung von Teilnetzen (*Coarse*) unterstützt. Dabei werden Netze erst einmal flach gezeichnet. Teilbereiche können dann beliebig oft zu Grobknoten zusammengefaßt werden. Diese Entwurfsvariante stößt bei großen Netzen an ihre Grenzen, da man ein solches Netz meist nicht komplett flach zeichnen möchte.

Erstellung von Netzbausteinen Trotzdem wird diese Methodik häufig verwendet, um einzelne Unterroutinen (Teilnetze) in Form von Modulen vollständig zu entwerfen. Diese können separat erstellt und dann durch die *File - Import* - Funktion in das Gesamtnetz integriert werden. Damit ist die Erstellung einer Bibliothek von Unterroutinen möglich, die als Bausteine im Gesamtnetz Verwendung finden.

3.3.6. Top-Down Entwurf

Die besonderen Stärken von PED liegen in der Unterstützung eines Top-Down Entwurfs von Petrinetzen. Dieser wird häufig verwendet, um eine Abstraktionsebene vollständig zu entwerfen, ohne jede einzelne Funktion modulieren zu müssen. Derartige Unterroutinen können nachträglich verfeinert werden. Die dafür notwendigen Konzepte wurden bereits vorgestellt.

Sehr hilfreich ist die Möglichkeit, einzelne Coarse-Knoten zu erzeugen, ohne ein schon vorhandenes Teilnetz vergrößern zu müssen. Die diesen Coarse-Knoten zugeordneten Subnetze sind leer. Im Anschluß daran können beliebig viele Verbindungen mit dem später zu verfeinernden Subnetz geschaffen werden. Für eine solche Verbindung kann ein Bogen direkt zwischen einer Transition oder einem Platz (Originalknoten oder Kopplungsknoten) und dem Coarse-Knoten gezeichnet werden. Damit werden die später für die Verfeinerung benutzbaren Schnittstellen in dem Subnetz erschaffen (ein Knoten zu bzw. von dem die Kante verlaufen soll und die entsprechenden Kopplungselemente, siehe auch Abschnitt 3.3.4).

Verwendung von Netzbausteinen Für die Verfeinerung der Subnetze können wahlweise vorhandene Komponenten genutzt werden, indem sie über die *File - Import* - Funktion (siehe vorherigen Abschnitt) oder den *Copy and Paste* - Mechanismus in das entsprechende Subnetz kopiert werden. Die Verbindungen zwischen den Randknoten dieser Komponenten und dem

übergeordneten Netz werden im allgemeinen durch das Verschmelzen dieser Knoten mit den vorhandenen Schnittstellenknoten erreicht (siehe Abschnitt 3.5.4).

3.4. Das Konzept der logischen Knoten

Oft hat man in einem Petrinetz Knoten, die mit sehr vielen Bögen verbunden sind und in mehreren Teilnetzen benötigt werden (z.B. zur Modellierung von globalen Variablen). Das führt zu einer unübersichtlichen Darstellung im Editor. Derartige Knoten können besser durch mehrere logischen Ausprägungen modelliert werden. Semantisch gehören sie zusammen und bilden einen einzigen Knoten.

Erstellung Logische Knoten werden durch ihre Namensgleichheit referenziert. Wählt man in dem Attribut-Dialog den entsprechenden Toggle-Button und setzt damit das *logical flag*, dann werden alle Knoten des selben Typs und mit gleichem Namen (es muß ein Name vorhanden sein!) zu einem logischen Knoten verbunden. Im Editor werden sie grau-schattiert dargestellt. Über den Menüeintrag *Edit - Select Logical Nodes* können weitere zu einer logischen Ausprägung gehörende Knoten innerhalb einer Seite selektiert werden. Die logische Beziehung zwischen mehreren Ausprägungen kann rückgängig gemacht werden, indem die Namensgleichheit aufgehoben und das *logical flag* zurückgesetzt wird.

Löschen Löscht man eine Ausprägung eines logischen Knotens über den Menüpunkt *Edit - Delete*, dann wird nur diese eine Ausprägung des Knotens mit all seinen verbundenen Kanten entfernt. Will man hingegen alle Ausprägungen eines logischen Knotens löschen, so kann dies mit *Edit - Delete Logically* erreicht werden.

Auswirkungen Alle Änderungen der Knoten-Attribute innerhalb einer logischen Ausprägung wirken sich einheitlich in allen weiteren logischen Ausprägungen aus. Für den Entwurfsprozeß und die Visualisierung im Editor ist diese Aufspaltung der logischen Knoten in mehrere Ausprägungen von Vorteil, für die Analyse des Netzes sollen sie aber wieder aufeinander abgebildet werden. Bei den Exportfunktionen zu den Analysewerkzeugen werden diese logischen Ausprägungen berücksichtigt, d. h. sie werden als ein einzelner Knoten exportiert (siehe Abschnitt 3.6.2).

3.5. Spezielle Funktionen

In diesem Abschnitt sollen nur einige Editorfunktionen näher erläutert werden, bei deren Erklärung es einiger Worte mehr bedarf. Ansonsten sei auf den Anhang A verwiesen, dort findet sich eine komplette Zusammenstellung aller Funktionen des Editors.

3.5.1. Clipboard Funktionen

Der Editor stellt ein eigenes Clipboard zu Verfügung. Damit hat man eine weitere Möglichkeit Elemente oder Teilnetze zu erzeugen bzw. zu vervielfältigen. Elemente werden solange im Clipboard gespeichert, bis sie durch eine neue Aktion, die Elemente im Clipboard ablegt, überschrieben werden.

Cut Mit *Edit - Cut* werden alle selektierte Elemente in das Clipboard kopiert und aus der Arbeitsfläche entfernt. Dabei werden alle mit den selektierten Knoten verbundenen Bögen einbezogen. Ein Wiederherstellen des ursprünglichen Netzes kann mit *Edit - Undo* erfolgen.

Copy Ein *Edit - Copy* arbeitet analog zu *Cut*, nur werden die Elemente nicht aus der Arbeitsfläche gelöscht.

Paste Der Menüeintrag *Edit - Paste* kopiert den Inhalt des Clipboards auf die aktuelle Seite im Editor, mit der Einschränkung, daß nur Kanten kopiert werden, die zwischen Knoten verlaufen, die sich auch im Clipboard befinden. Will man das ursprüngliche Netz wiederherstellen, sollte man *Undo* benutzen.

Mit *Paste* können Teilnetze beliebig oft kopiert werden. Die auf die aktuelle Seite kopierten Elemente sind selektiert. Sie sollten sofort in einen freien Bereich verschoben werden, da jede neue Aktion die vorherige Selektion aufhebt und man so diese Elemente schwerer verschieben kann (auch wenn man weitere Kopien innerhalb dieses Teilnetzes erstellen will).

3.5.2. Undo

Ein *Edit - Undo* kann *Cut*- und *Delete*-Aktionen rückgängig machen. Es arbeitet einstufig, das heißt, es kann immer nur der Zustand vor der letzten *Cut*- oder *Delete*-Aktion wiederhergestellt werden. Durch bestimmte weitere Aktionen, wie *Coarse*, *Flat* oder *Merge* geht die *Undo*-Fähigkeit verloren.

3.5.3. Split

Durch den Menüpunkt *Edit - Split* können Transitionen oder Plätze vervielfältigt werden. Dabei müssen zwei Fälle unterschieden werden.

Anwendung auf einen Kopplungsknoten Der wohl häufigste Fall einer Anwendung der *Split*-Funktion bezieht sich auf einen Kopplungsknoten innerhalb eines Subnetzes. Damit können weitere graphische Ausprägungen des durch den Kopplungsknoten repräsentierten Originalknotens innerhalb dieses Subnetzes geschaffen werden. Für jeden *Pre*- und *Postnode* innerhalb dieser Seite wird eine Kopie des Kopplungsknotens angelegt. Danach unterhält jede Kopie des Kopplungsknotens nur zu genau einem weiteren Knoten dieser Seite Kantenverbindungen (als *Pre*- und/oder *Postnode*). Gibt es nur zu einem Knoten derartige Kantenverbindungen, bleibt die Aktion ohne Erfolg.

Alle entstandenen Kopien repräsentieren weiterhin den selben Originalknoten. Sie müssen nun nur noch an die gewünschten Stellen verschoben werden. Diese Anwendung führt in einigen Fällen (bei vielen Kantenverbindungen) zu einer übersichtlicheren Darstellung des Teilnetzes. Diese Aktion kann durch Anwendung der *Merge*-Funktion einzeln wieder rückgängig gemacht werden.

Anwendung auf einen Originalknoten Die Anwendung der *Split*-Funktion auf einen Originalknoten führt zu einer Kopie des Knotens, wobei alle Kantenverbindungen zu bzw. von diesem Knoten mitkopiert werden. Besitzt der zu spaltende Knoten einen Namen, so werden beide

entstandenen Knoten zu einem logischen gewandelt. Da alle Kantenverbindungen kopiert werden, können nun noch redundante Kanten per Hand gelöscht und die Knoten an den gewünschten Stellen positioniert werden. Auch diese Funktion kann die Übersichtlichkeit eines Teilnetzes erhöhen (siehe auch Abschnitt 3.4).

3.5.4. Merge

Verschiebt man einen Knoten derart (im folgenden ist das der “bewegte” Knoten), daß er über einem weiteren Knoten (im folgenden der “ruhende” Knoten) zum liegen kommt, dann wird bei einer erlaubten Verschmelzung dieser beiden Knoten (insbesondere müssen sie gleichen Knotentyps sein) ein Dialogfenster zur Bestätigung einer tatsächlich vorzunehmende Verschmelzung der Knoten angeboten.

Ist eine derartige Verschmelzung nicht erlaubt oder wird die Abfrage mit “No” beantwortet, dann werden beide Knoten nebeneinander positioniert. Für die erlaubten Fälle, müssen drei Szenarios unterschieden werden. Für alle drei Fälle gilt, daß nur Transitionen und Plätze untereinander verschmolzen werden können. Als Grundregel sollte man den Knoten, dessen Eigenschaften man bewahren möchte, nicht transportieren, sondern als ruhenden Knoten verwenden (im 2. und 3. Fall spielt das natürlich keine Rolle).

Verschmelzung zweier Originalknoten Durch diese Aktion wird der bewegte Knoten gelöscht und dessen Bögen werden durch den ruhenden Knoten übernommen. Die Attribute des bewegten Knotens gehen verloren. Alle graphischen Ausprägungen beider Knoten bleiben erhalten. Sie repräsentieren nun den selben Knoten. Bei Bedarf können diese Kopplungsknoten einzeln verschmolzen werden (siehe nächster Anwendungsfall).

Verschmelzung zweier Kopplungsknoten Diese Aktion ist nur erlaubt, wenn es sich bei beiden Knoten um graphische Ausprägungen des selben Originalknoten handelt. Es muß also eine *Split*-Aktion vorausgegangen sein (siehe vorherigen Abschnitt). Alle vorhandenen Kantenverbindungen werden vom resultierenden Knoten übernommen.

Verschmelzung eines Original- mit einem Kopplungsknoten Bei dieser Aktion ist es unerheblich, welcher der beiden Knoten transportiert wurde. In jedem Fall “gewinnt” der Kopplungsknoten. Eigentlich bedeutet dieser Vorgang eine Verschmelzung vom Original des Kopplungsknotens mit dem Knoten der aktuellen Seite (gewöhnlich ist das der bewegte Knoten), wobei dessen Eigenschaften verloren gehen. Alle Kantenverbindungen des Knotens der aktuellen Seite gehen auf den vom Kopplungsknoten repräsentierten Knoten über.

3.5.5. Squeeze Numbers

Mit dem Menüpunkt *Edit - Squeeze Numbers* können die vergebenen Knotennummern auf Wunsch des Benutzers in eine fortlaufende Reihenfolge gebracht werden. Durch Editierfunktionen wie *Delete* und *Cut* entstehen Lücken in der Nummerierung, die auf die Art geschlossen werden. Aus Performance-Gründen geschieht das nicht nach jeder Einzelaktion.

Für einige Analysetools ist es wichtig, daß die Knotennummern “dicht” sind. Der Benutzer braucht diese Aktion vor einem Export zu derartigen Analysetools aber nicht per Hand anstoßen, da sie implizit durchgeführt wird.

3.5.6. Align Vertical/Horizontal

Neben der Möglichkeit, beim Editiervorgang mit eingeschalteter *Snap*-Funktion zu arbeiten, gibt es noch zwei Hilfsfunktionen, die ein exaktes Ausrichten der Netzelemente auf einer Linie erleichtern.

Graphics - Align Vertical richtet alle selektierten Knoten in vertikaler Richtung auf einer Linie aus.

Graphics - Align Horizontal erledigt das gleiche in horizontaler Richtung. Für beide Funktionen sind Tastaturkürzel vergeben, die den Editiervorgang sehr erleichtern (siehe Quickreferenz).

3.5.7. Set of ...

Oft ist es wünschenswert, die Attribute mehrerer Elemente gleichzeitig ändern zu können. Dazu dienen die *Edit - Set of ...* Funktionen. Es wird nach Elementtypen unterschieden. Man erhält ein abgewandeltes Dialogfenster zur Eingabe der Attribute für den jeweiligen Elementtyp (Abbildung 3.5). Die Änderungen beziehen sich auf selektierte Elemente, wobei nur die Elemente des jeweiligen Typs einbezogen werden. Alle anderen werden ignoriert. Dadurch können mit *Select All* durchaus alle Elemente einer Seite selektiert sein, um beispielsweise den Namen aller Transitionen einer Seite einen Präfix voranzustellen.



Abbildung 3.5.: Bildschirmausschnitt Set of Transitions Attributes - Dialog

In den Eingabefeldern des Dialogfensters sind standardmäßig "*" Zeichen eingetragen. Bei der Eingabe eines "*" wird an dieser Stelle der alte Wert verwendet. Bei Eingabe eines anderen Wertes, wird dieser in allen Elementen gesetzt. Wenn es sich um einen String als Wert handelt, symbolisiert der "*" den alten String, der beliebig erweitert werden kann. Sei der alte Wert "v", dann wird "p*c*s" zu "pvcvs" erweitert.

3.5.8. Search

Die Suchfunktion *Edit - Search* ist ein nützliches Werkzeug, um in großen Netzen direkt in bestimmte Teilnetze oder zu bestimmten Knoten zu springen (Abbildung 3.6).

Über die oberen Toggle-Buttons wird der Suchbereich auf einen Knotentyp oder auf Teilnetze eingegrenzt. Falls nach Teilnetzen gesucht werden soll, wird der *Number*-Parameter als Knotennummer des übergeordneten Coarse-Knotens interpretiert, gleiches gilt für den *Name*-Parameter. Im *Net*-Eingabefeld kann die Netznummer in Form einer Dezimalklassifikation angegeben werden.

Die vergebenen Netznummern sind im Editor auf der Arbeitsfläche links oben (für das aktuelle Teilnetz) und in der Coarse-Knotenbezeichnung (für das darunterliegende Teilnetz) angegeben. Außerdem sind sie dem Hierrachiebrower zu entnehmen (siehe Abschnitt 4).

Falls nach Transitionen oder Plätzen gesucht wird, kann noch ein Attributwert in die Anfrage aufgenommen werden. Der Eingabewert wird entsprechend des ausgewählten Toggle-Buttons (Attributtyp) interpretiert.

Alle Eingabewerte werden mit einem logischen UND verknüpft. Als Wildcard kann das "*" - Zeichen verwendet werden.



Abbildung 3.6.: Bildschirmausschnitt Search - Dialog

In dem Beispiel in Abbildung 3.6 wird ein Platz gesucht, dessen Nummer beliebig ist, dessen Name den Präfix "Synchro" besitzt, der sich innerhalb des Teilnetzes "2.*" befindet und auf dem eine Marke liegt.

Öffnet man den Suchdialog und ist ein Knoten gerade selektiert, so wird der Name des selektierten Knotens gleich in das *Name*-Eingabefeld übernommen. Da der Suchdialog permanent auf dem Desktop stehen bleibt und man sich durch Anklicken von *Search* immer zum nächsten gefunde-

nen Knoten hangelt, ist diese Suchfunktion sehr gut geeignet, beispielsweise alle Ausprägungen eines logischen Knotens aufzusuchen und die entsprechenden Seiten zu laden.

3.6. Exportfunktionen

PED bietet einige Schnittstellen zu externen Programmen standardmäßig an. Darüberhinaus wird eine Möglichkeit unterstützt, eigene Exportfunktionen in den Editor einzuhängen (siehe Abschnitt 3.6.3). Die standardmäßig angebotenen Schnittstellen lassen sich im wesentlichen in zwei Kategorien unterteilen, zum einen sind das Exportfunktionen zu verschiedenen Grafikformaten und zum anderen die Schnittstellen zu weiteren Petrinetzwerkzeugen.

3.6.1. Graphische Exportfunktionen

Durch den Export zu verschiedenen Grafikformaten können die im Editor gezeichneten Netze in anderen Grafikprogrammen weiterbearbeitet (FrameMaker, XFig), ausgedruckt (PostScript) und in eigene Publikationen eingebunden werden.

Die im *Options - Show* - Dialogfenster eingestellte Sichtbarkeit der Attribute wird auch bei den Exportfunktionen berücksichtigt. Ein Export in ein Grafikformat überträgt nur die aktuelle, im Editor geladene Seite. Um ein hierarchisches Netz komplett zu exportieren, müssen alle Seiten geladen und einzeln exportiert werden.

Export zu FrameMaker

Bei der Exportfunktion zu FrameMaker (*File - Export - Maker Interchange Format*) wird ein MIF-File erzeugt. Das *Maker Interchange Format* ist ein ASCII-Format und kann damit plattformunabhängig verwendet werden. Das DTP-Programm FrameMaker gibt es für verschiedene Plattformen (unter anderem auch für die Windows-Welt). Über die entsprechende Importfunktion von FrameMaker kann das exportierte File geladen und weiterbearbeitet werden.

Das MIF-Format ist z. Z. das einzige Grafikformat, welches auch aus dem Hierarchybrowser heraus exportiert werden kann. Dort hat man zusätzlich die Möglichkeit, mehrere bzw. auch alle Seiten eines hierarchischen Netzes in einem Schritt zu exportieren. Damit entfällt der Aufwand, alle Seiten einzeln zu laden (siehe auch Abschnitt 4.7).

Export zu XFig

Bei der Exportfunktion zu XFig (*File - Export - XFig*) wird ein FIG-File erzeugt (Version 3.1). Dieses Format wurde vor allem deshalb zur Verfügung gestellt, weil XFig selbst ein Konvertieren in viele weitere Grafikformate erlaubt (latex, pic, epic, eepic, pictex, ps, eps . . .). Bei XFig handelt es sich um ein Zeichenprogramm, das für diverse Unix-Plattformen (unter anderem für Linux) existiert und als *Public Domain Software* frei verfügbar ist.

Einbinden von Grafiken in \LaTeX -Files Bei dem Versuch, ein mit PED gezeichnetes Netz in ein \LaTeX -File einzubinden, lassen sich über den Umweg "XFig" gute Ergebnisse erzielen. Es können einige Makropakete für \TeX verwendet werden (z.B. EPIC und EEPIC). In Zukunft wird es durch Angebot einer PED-PSTricks-Schnittstelle noch eine weitere Möglichkeit geben.

Optionen im XFig-Dialogfenster Neben dem zu verwendenden Font können weitere Einstellungen im XFig-Dialogfenster vorgenommen werden. Diese sind für das Einbinden der Netze in \LaTeX -Files gedacht. Über die Toggle-Buttons kann für Kommentare oder bestimmte Attribute (Zeitintervall, Knotennummer, . . .) festgelegt werden, ob sie in die Formelumgebung ($\$. . . \$$) gesetzt werden sollen, um dann später in \TeX als solche interpretiert zu werden. Insbesondere ist schon bei der Eingabe der Strings zu beachten (Kommentare oder sonstiges), daß das \TeX -like geschehen muß. Beispielsweise sollten für bestimmte Sonderzeichen die entsprechenden Befehle eingegeben werden. XFig selbst reicht diese Zeichenketten unverändert an \TeX weiter.

Export zu PostScript

Bei der Exportfunktion zu PostScript (*File - Export - PostScript*) wird ein PS-File erzeugt. Dieses kann direkt auf einem PostScript-fähigen Drucker ausgegeben werden. Ist das Netz auf der aktuellen Editorseite größer als eine DIN-A4-Seite, werden automatisch mehrere Seiten angelegt.

3.6.2. Export zu weiteren Petrinetzwerkzeugen

Export zu INA (Integrierter Netzanalysator)

Der Integrierte Netzanalysator (entwickelt von P. Starke, Humboldt-Universität zu Berlin) ist ein leistungsfähiges Werkzeug zur Analyse von Platz-Transitions-Netzen [INA], das nahezu alle Verfahren der „klassischen“ Petrinetz-Theorie implementiert, wie z. B. die Berechnung von Invarianten, die Überprüfung struktureller Eigenschaften wie die Deadlock-Falle-Eigenschaft, strukturelle Netzreduktion, die Erzeugung von Erreichbarkeitsgraphen (symmetrisch und *stur*-reduziert) und Überdeckbarkeitsgraphen.

Sind in dem Netz Inhibitor- bzw. Resetkanten enthalten, so wird ein Export nach INA nicht erlaubt. Testkanten hingegen werden als eine Kante, die in beide Richtungen führt, interpretiert.

Durch die Exportfunktion *File - Export - Petri Net Maschine* können drei verschiedenen Dateien erzeugt werden.

PNT-File Im PNT-File werden die strukturellen Netzinformationen in einem platzorientierten Format abgelegt. Mehrere logische Ausprägungen eines Knotens sind auf einen einzelnen Knoten abgebildet. Ein Knoten wird über seine Nummer identifiziert. An Attributen werden Anfangsmarkierung, Platzkapazität, Transitionspriorität und Bogenvielfalt unterstützt.

Zusätzlich kann INA zeitbewertete Netze und Intervallnetze analysieren. Dafür kann man optional zwei weitere Dateien exportieren, die zusammen mit dem PNT-File für die Analyse verwendet werden.

TMD-File Das TMD-File enthält die Zeit(dauer)-Attribute von Transitionen (für alle Transitionen, deren Zeitattribut ungleich 0 ist).

TIM-File Das TIM-File speichert für Transitionen die entsprechenden Intervalle, sofern diese verwendet wurden.

Export zum PROD-Analyzer

Der PROD-Analyzer ist ein Werkzeug zur Analyse von Prädikat-Transitions-Netzen auf der Basis des Erreichbarkeitsgraphen. PROD implementiert in einer Anfragesprache differenzierte Strategien zur Traversierung und Analyse des Erreichbarkeitsgraphen, die mächtig genug sind, um in ihr eine temporale Logik (*computational tree logic, CTL*) zu formulieren¹. Daneben stellt PROD einen *model checker* zur Validierung temporallogischer Formeln (*linear time temporal logic, LTL*) zur Verfügung, der auf der Konstruktion eines reduzierten Erreichbarkeitsgraphens auf der Basis der Methode der *sturen Mengen* beruht.

PROD wurde an der Technischen Universität Helsinki entwickelt [PRO] und ist frei verfügbar (*Public Domain*).

Sind in dem Netz Inhibitor- bzw. Resetkanten enthalten, so wird ein Export nach PROD nicht erlaubt. Testkanten hingegen werden als eine Kante, die in beide Richtungen führt, interpretiert.

Durch die Exportfunktion *File - Export - PROD-Analyzer* kann ein Net-File erzeugt werden.

NET-File Beim Export in das PROD-Format wird nicht die volle Funktionalität von PROD ausgenutzt. Die mit PED erzeugbaren Netze stellen aber eine Untermenge der von PROD analysierbaren Prädikat-Transitions-Netze dar. Die Netzstruktur wird transitionsorientiert innerhalb eines NET-Files gespeichert. Mehrere logische Ausprägungen eines Knotens werden auf einen einzelnen Knoten abgebildet. Die unterstützten Attribute beschränken sich auf die Anfangsmarkierung und die Bogenvielfalt.

Eindeutigkeit der Knotennamen Innerhalb der Exportfunktion werden alle für Plätze und Transitionen vergebenen Namen auf Eindeutigkeit überprüft. Ist diese gegeben, so werden die Namen als *Identifier* innerhalb des NET-Files benutzt, wenn nicht, sind den Namen zusätzlich die Knotennummern vorangestellt (T<num>_Name bzw. P<num>_Name). Eine entsprechende Meldung erscheint auf der Standardausgabe.

Export zu PEP

Das im Rahmen der ESPRIT-Projekte DEMON und CALIBAN an der Universität Hildesheim entstandene Werkzeug PEP [PEP] unterstützt neben Platz-Transitions-Netzen eine als M-Netze bezeichnete High-Level-Netzklasse, sowie die OCCAM-ähnliche Programmiersprache $B(PN)$ ² und ein prozeßalgebraisches Kalkül namens PBC.²

Für Platz-Transitionsnetze stellt PEP ein ebenfalls auf *model checking* beruhendes Verfahren zur Überprüfung von Formeln einer eingeschränkten CTL-Variante für eine eingeschränkte Petrinetzklasse (1-beschränkte Netze) zur Verfügung.

Sind in dem Netz Inhibitor- bzw. Resetkanten enthalten, so wird ein Export nach PEP nicht erlaubt. Testkanten hingegen werden als eine Kante, die in beide Richtungen führt, interpretiert.

Durch die Exportfunktion *File - Export - PEPTOOL* werden zwei verschiedenen Dateien erzeugt.

¹Eine gute Einführung in temporale Logik kann etwa in [Eme90] gefunden werden.

²Für weitere Informationen sei der geneigte Leser auf [PEP] verwiesen.

LL_NET-File Das LL_NET-File enthält die strukturellen Netzinformation im kantenorientierten Format. Von den Attributen werden die Anfangsmarkierung und die Bogenvielfachheiten unterstützt. Mehrere logische Ausprägungen eines Knotens werden auf einen einzelnen Knoten abgebildet.

PROJEKT-File Das PROJEKT-File enthält Verwaltungsdaten wie etwa den Namen des LL_NET-Files.

3.6.3. Generische Exportfunktion

PED bietet eine dynamische Möglichkeit an, die Exportfunktionalität zu erweitern. Zu diesem Zweck wurde in PED ein Interpreter integriert. Die Exportfunktionalität beschränkt sich allerdings auf flache strukturelle Netzinformationen, d.h. es können keine graphischen Informationen zu den Netzelementen abgefragt, bzw. Hierarchien exportiert werden. Die von PED unterstützten Element-Attribute stehen alle zur Verfügung. Am besten eignet sich dieser generische Ansatz für den Export zu Analysetools, die flache Netze verarbeiten.

Der Interpreter selbst und die dazugehörige Skriptsprache Lua wurden an der *Pontifical Catholic University of Rio de Janeiro* entwickelt und eignen sich ausgezeichnet, um bestehende Applikationen dynamisch zu erweitern (ohne das Programm neu kompilieren zu müssen). Der sogenannte Lua-Interpreter steht für die nichtkommerzielle Nutzung frei zur Verfügung und wurde zu dem Programm "ped" statisch hinzugebunden.

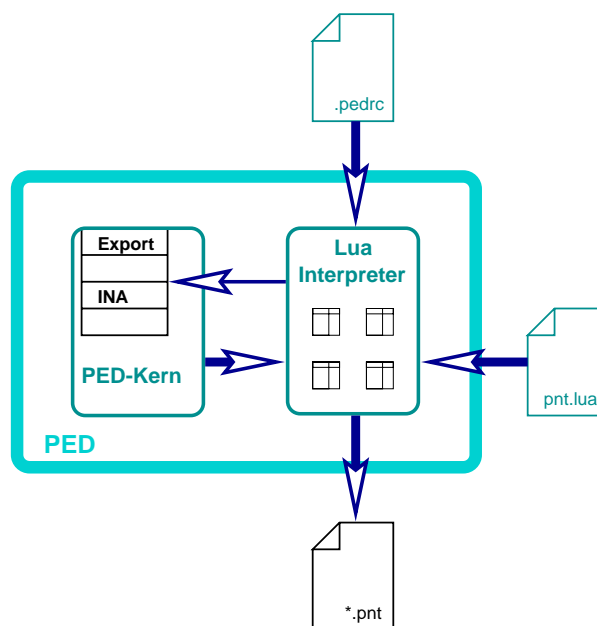


Abbildung 3.7.: Generische Exportfunktionalität in PED

Einbinden von Exportfunktionen Für jede Exportfunktion muß ein entsprechendes Lua-Skript geschrieben werden. Das Verzeichnis, in dem diese "*.lua" Skripts dann stehen, wird im *.pedrc* File eingestellt. Ebenso werden die zusätzlichen Menüeinträge in PED und die Zuordnung

zu den einzelnen Exportroutinen (“*.lua” Skripts) über dieses File konfiguriert. Abbildung 3.7 zeigt eine Übersicht der beteiligten Komponenten (in diesem Fall für den Export nach INA). Ein Beispiel eines *.pedrc* Files ist im Anhang B enthalten, die Syntax dürfte keine Probleme bereiten.

Erstellung der Exportroutinen Lua hat eine PASCAL-ähnliche Syntax. Bei der Programmierung der Exportroutinen kann man auf Tabellen zugreifen, die innerhalb von PED angelegt werden. Für Plätze, Transitionen, Bögen und Kommentare gibt es jeweils eine Tabelle, in der alle Elemente eines Typs enthalten sind. Das Netz ist darin kantenorientiert gespeichert, d. h. *Pre-* und *Postnodes* sind bei den Kanten vermerkt.

Im Anhang C sind zwei Beispiele derartiger “*.lua” Files angegeben. Im ersten Beispiel (zu Testzwecken erstellt) wird der Inhalt aller angelegten Tabellen in eine Datei geschrieben. Auf die Art können alle verfügbaren Informationen ermittelt werden.

Das zweite Beispiel demonstriert den Export zu INA. In INA sind Netze platzorientiert gespeichert. Dementsprechend wird die kantenorientierte Sicht von PED in eine platzorientierte Struktur gewandelt. Dazu werden wiederum Tabellen zur Speicherung der *Pre-* und *Posttransitionen* angelegt, die durch das Durchlaufen der Kantentabelle gefüllt werden. Die Netzinformationen werden dann in formatspezifischer Syntax aus den Tabellen in eine Datei geschrieben. Weiterführende Information zu Lua finden sich unter [LUA].

4. Hierarchiebrowser

Der Hierarchiebrowser unterstützt die Arbeit mit hierarchischen Petrinetzen. Neben seiner Funktion der Darstellung der Netzhierarchie, erleichtert er die Navigation zwischen verschiedenen Teilnetzen. Er ist als eigenständige Komponente ausgelegt und kann somit parallel zum Editor benutzt werden.

Der Hierarchiebrowser wird über den Menüeintrag *Hierarchie - Browser* gestartet. In einigen Punkten erweitert er die Funktionalität des Editors. Das betrifft vor allem das Abspeichern von Teilnetzen (Abschnitt 4.6) und die Exportfunktion zu FrameMaker (Abschnitt 4.7).

Die Darstellungsform der Netz-Icons deutet auf den Typ des zugeordneten Coarse-Knotens hin. Falls es sich um eine Grobtransition handelt (Coarse-Knoten mit transitionsberandetem Subnetz), so ist das entsprechende Netz-Icon als Rechteck gezeichnet. Grobplätze erhalten ein ellipsenförmiges Netz-Icon und gemischtberandete Subnetze (Standardvariante) vereinen beide Formen.

4.1. Grundeinstellungen

Über den Menüeintrag *Control - Options* können einige Grundeinstellungen im Hierarchiebrowser vorgenommen werden (Abbildung 4.1). Diese betreffen die Darstellung der Netz-Icons und deren Anordnung im Browser.

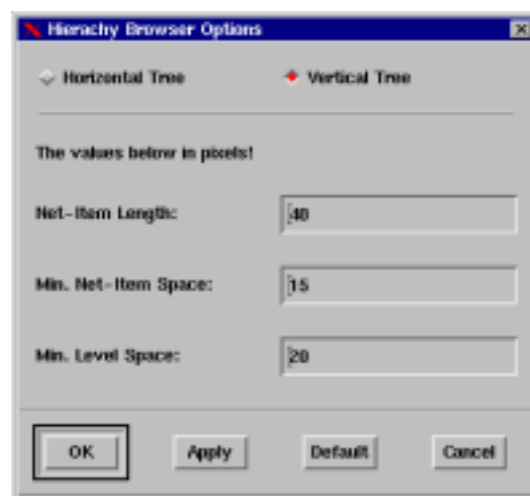


Abbildung 4.1.: Bildschirmausschnitt Options - Dialog

Grundsätzlich kann man zwischen der Darstellung der Netzhierarchie in Form eines Baums (*horizontal tree*) oder einer verzeichnisstruktur-ähnlichen Darstellungsvariante (*vertical tree*) wählen. Die Netz-Icons haben immer eine einheitliche Größe und können nicht frei positioniert werden. Es lassen sich aber dennoch innerhalb des *Options*-Dialogfensters Größe und Abstände zwischen den Netz-Icons einstellen. Die Parameter werden in Pixel angegeben.

4.2. Navigation

Der Hierarchiebrowser ermöglicht es auf sehr einfache Art zwischen Teilnetzen zu wechseln. Durch Doppelklick mit der linken Maustaste auf ein Netz-Icon wird dieses Teilnetz in den Editor geladen. Das aktuelle Teilnetz ist stets durch einen roten Hintergrund hervorgehoben.

4.3. Selektieren

Das Selektieren von Netz-Icons stimmt mit der vom Editor bekannten Funktionalität überein. Mit der linken Maustaste kann man ein Netz-Icon exklusiv selektieren oder durch Aufziehen einer Selektionsbox eine Gruppe von Netz-Icons. Mit der mittleren Maustaste können einzelne Netz-Icons (de)selektiert werden. Der Menüeintrag *Edit - Select All* bewirkt die Selection aller Netz-Icons. Über die rechte Maustaste steht das Editmenü zur Verfügung.

4.4. Wrap - Unwrap

Ähnlich der Funktionsweise eines Dateimanagers, können ganze Hierarchiezweige eingepackt (*wrap*) und ausgepackt (*unwrap*) werden. Auf diese Weise läßt sich die Netzhierarchie in gewünschter Tiefe darstellen (gegebenfalls nur bis zu einer bestimmten Abstraktionsstufe).

Ein Doppelklick mit der mittleren Maustaste auf ein Netz-Icon bewirkt das "Entpacken" einer weiteren Hierarchiestufe, sofern dessen Subnetze noch nicht angezeigt waren. Sind dagegen schon Subnetze des Netz-Icons dargestellt, so verschwinden alle tieferliegenden Netz-Icons. Die gleiche Funktionalität wird über die Menüeinträge *Edit - Unwrap* und *Edit - Wrap* erreicht. Diese Aktion bezieht sich auf alle selektierten Netz-Icons.

Über den Menüpunkt *Edit - Unwrap All* werden alle vorhandenen Teilnetze dargestellt. Ein vollständiges "Einpacken" bewirkt *Edit - Wrap All* (bis auf Subnetze unterhalb vom Toplevel).

4.5. Editierfunktionen

Clipboardfunktionen, Undo Im Hierarchiebrowser stehen auch die vom Editor bekannten Clipboardfunktionen *Cut*, *Copy*, *Paste* (siehe Abschnitt 3.5.1) und das einstufige *Undo* (siehe Abschnitt 3.5.2) zur Verfügung. Die Aktionen beziehen sich auf selektierte Netz-Icons.

Das Löschen, Verschieben oder Kopieren eines Netz-Icons entspricht einer derartigen Aktion mit einem einzeln selektierten Coarse-Knoten auf der Arbeitsfläche im Editor. Die Umgebung des Coarse-Knoten geht dabei verloren (alle Kantenverbindungen in das Subnetz). Wenn beispielsweise ein Netz-Icon in ein anderes kopiert wird, dann erscheint in dem Subnetz des Ziel-Icons ein einzelner Coarse-Knoten mit vollständigem Subnetz, jedoch ohne eine Kantenverbindung nach außen.

Das Kopieren eines Teilnetzes (bzw. einer selektierten Gruppe von Teilnetzen) innerhalb des Hierarchiebrowsers kann auch durch eine *Drag and Drop*-Aktion mit der linken Maustaste erreicht werden.

Modifizieren der Netz-Icon-Attribute Im Hierarchiebrowser stehen auch die vom Editor bekannten Dialogfenster zum Modifizieren der Element-Attribute zur Verfügung. Da die Eigenschaften der Netz-Icons durch die zugeordneten Coarse-Knoten bestimmt werden, entsprechen diese Dialogfenster exakt den schon im Editor benutzen Dialogfenstern zum Modifizieren von Coarse-Knoten-Attributen. Entsprechend gibt es auch hier wieder eine Variante, die sich auf ein einzelnes Netz-Icon bezieht (*Edit - Net-Item Attributes*) und eine weitere, für eine Gruppe von Netz-Icons (*Edit - Set of Net-Items*). Damit kann der Name des Teilnetzes und der Coarse-Knotentyp modifiziert werden (siehe auch Abschnitte 3.2.3 und 3.5.7).

4.6. Speichern von Teilnetzen

Mit dem Hierarchiebrowser können Teile eines Netzes abgespeichert werden. Der Menüpunkt *Export - Save Subnet As* speichert alle selektierten Teilnetze in eine Datei.

Mit dieser Funktion ist es beispielsweise möglich, ein Netz nur bis zu einer bestimmten Abstraktionstufe (Tiefe) zu speichern. Die nicht selektierten, darunterliegenden Teilnetze werden, sofern das möglich ist, zu Transitionen (für Grobtransitionen) oder zu Plätzen (für Grobplätze) reduziert. Voraussetzung dafür ist, daß diese Subnetze wirklich nur Randknoten eines Typs besitzen. Diese Netze können dann wieder geladen werden. Damit steht auch ein allgemeiner Mechanismus zur Verfügung, um derartige Teilnetze zu analysieren.

Außerdem eignet sich diese Funktion zum Abspeichern einzelner Netzbausteine durch die Selektion ganzer Teilbäume. Diese können dann in anderen Netzen wiederverwendet werden (Anlegen einer Netzbibliothek).

4.7. Export zu FrameMaker

Es gibt zwei verschiedene Funktionen, mit denen im Hierarchiebrowser MIF-Files erzeugt werden können (siehe auch Abschnitt 3.6.1).

Mehrere Teilnetze Die erste erweitert die schon vom Editor bekannte Exportfunktion um die Möglichkeit, mehrere bzw. auch alle Seiten eines hierarchischen Netzes in einem Schritt zu exportieren. Diese Funktion wird durch den Menüpunkt *Export - FrameMaker - Set of SubNets* aufgerufen. Sie bezieht sich auf alle selektierten Teilnetze. Für jedes Teilnetz wird eine neue Seite innerhalb des MIF-Files angelegt.

Hierarchiebaum Die zweite Funktion (*Export - FrameMaker - Hierarchy Tree*) erzeugt eine Grafik des Hierarchiebaums im MIF-Format. Dies gelingt aber nur, wenn die Verzeichnisstruktur als Darstellungsvariante gewählt wurde. Der horizontale Baum ist in den meisten Fällen zu breit für eine Darstellung auf einer DIN-A4-Seite.

A. Quickreferenz

A.1. Übersicht aller Funktionen im Editor

Das File - Menü

<i>Funktion</i>	<i>short cut</i>	<i>Beschreibung</i>
New	<i>Ctrl+N</i>	Neues Netz zeichnen, altes Netz verwerfen
Load	<i>Ctrl+L</i>	Netz aus einer Datei laden
Reload	<i>Ctrl+R</i>	Letzten gespeicherten Zustand wiederherstellen
Import	<i>Ctrl+I</i>	Netz aus einer Datei laden und in die aktuelle Seite einfügen
Save	<i>Ctrl+S</i>	Netz speichern
Save As		Netz unter neuem Dateinamen speichern
Exit	<i>Ctrl+X</i>	Programm verlassen

Das File-Export - Menü

<i>Funktion</i>	<i>short cut</i>	<i>Beschreibung</i>
PostScript		Export zu PostScript (siehe 3.6.1)
XFig		Export zu XFig (siehe 3.6.1)
Maker Interchange		Export zu FrameMaker (siehe 3.6.1)
Petri Net Maschine		Export zu INA (siehe 3.6.2)
PROD-Analyzer		Export zu PROD (siehe 3.6.2)
PEPTOOL		Export zu PEP (siehe 3.6.2)

Das Edit - Menü

<i>Funktion</i>	<i>short cut</i>	<i>Beschreibung</i>
Refresh		Neuzeichnen der Arbeitsfläche
Undo	<i>Ctrl+U</i>	Einstufiges Undo (siehe 3.5.2)
Cut	<i>Ctrl+C</i>	Selektiertes Teilnetz ausschneiden und im Clipboard ablegen (siehe 3.5.1)
Copy	<i>Ctrl+O</i>	Selektiertes Teilnetz ins Clipboard kopieren (siehe 3.5.1)
Paste	<i>Ctrl+P</i>	Inhalt des Clipboards auf die aktuelle Seite kopieren (siehe 3.5.1)

Delete	<i>Ctrl+P</i>	Selektierte Elemente löschen
Delete Logically		Wie Delete, nur logische Knoten werden komplett gelöscht (siehe 3.4)
Split	<i>Shift+T</i>	Knoten aufspalten, bzw. graphische Ausprägungen vervielfältigen (siehe 3.5.3)
Set of Places		Attribute aller selektierten Plätze ändern (siehe 3.5.7)
Set of Transitions		Attribute aller selektierten Transitionen ändern (siehe 3.5.7)
Set of Coarse Nodes		Attribute aller selektierten Coarse-Knoten ändern (siehe 3.5.7)
Set of Arcs		Attribute aller selektierten Bögen ändern (siehe 3.5.7)
Search	<i>Shift+S</i>	Suche nach Teilnetzen, Plätzen, Transitionen (siehe 3.5.8)
Squeeze Numbers	<i>Shift+N</i>	Lücken bei der Knotennummerierung schließen (siehe 3.5.5)
Select Logical Nodes	<i>Shift+L</i>	Weitere logische Ausprägungen eines Knotens selektieren (siehe 3.4)
Select All Nodes	<i>Shift+A</i>	Alle Elemente einer Seite selektieren

Das Hierarchy - Menü

<i>Funktion</i>	<i>short cut</i>	<i>Beschreibung</i>
Coarse		Zusammenfassen eines Teilnetzes zu einem Grobknoten (siehe 3.3.1)
Flat		Einebnen eines Subnetzes (siehe 3.3.2)
Level Up	<i>Shift+U</i>	In das übergeordnete Teilnetz wechseln (siehe 3.3.1)
Level Down	<i>Shift+D</i>	In das Subnetz eines selektierten Coarse-Knotens wechseln (siehe 3.3.1)
Browser	<i>Shift+D</i>	Hierarchiebrowser starten (siehe 4)

Das Options - Menü

<i>Funktion</i>	<i>short cut</i>	<i>Beschreibung</i>
Basics	<i>Ctrl+B</i>	Grundeinstellungen im Editor (siehe 3.1.1)
Show	<i>Ctrl+W</i>	Einstellung der Sichtbarkeit von Attributen, Kommentaren, Basisobjekten und Konflikt-Clustern (siehe 3.1.2)
Transition	<i>Ctrl+T</i>	Bestimmen der Standard-Transitionsform (siehe 3.1.3)

Das Graphics - Menü

<i>Funktion</i>	<i>short cut</i>	<i>Beschreibung</i>
Justify (Left, Top)	<i>Ctrl+J</i>	Das aktuelle Teilnetz wird in die linke obere Ecke verschoben (ausgerichtet)
Zoom	<i>Ctrl+Z</i>	Verkleinern/Vergrößern des aktuellen Teilnetzes
Rotate		Ein selektiertes Teilnetz wird um 90 ° gedreht
Flip Left-Right		Ein selektiertes Teilnetz wird auf einer vertikalen Achse gespiegelt
Flip Up-Down		Ein selektiertes Teilnetz wird auf einer horizontalen Achse gespiegelt
Align Vertical	<i>Ctrl+V</i>	Alle selektierten Elemente werden auf einer vertikalen Linie ausgerichtet (siehe 3.5.6)
Align Horizontal	<i>Ctrl+H</i>	Alle selektierten Elemente werden auf einer horizontalen Linie ausgerichtet (siehe 3.5.6)

Das Help - Menü

<i>Funktion</i>	<i>short cut</i>	<i>Beschreibung</i>
Verify	<i>Ctrl+V</i>	Selbsttest zur Überprüfung einiger interner Konsistenzbedingungen (aus Implementierungssicht, <i>keine</i> Verifizierung des erstellten Netzmodells!)
About		Informationen zu PED (Versions-, stempel, Copyrights)

A.2. Übersicht aller Funktionen im Hierarchiebrowser

Das Control - Menü

<i>Funktion</i>	<i>Beschreibung</i>
Options	Grundeinstellungen zur Darstellung und Anordnung der Netz-Icons (siehe 4.1)
Close	Schließen des Hierarchiebrowsers

Das Export - Menü

<i>Funktion</i>	<i>Beschreibung</i>
Save Subnet As	Selektierte Teilnetze werden in eine Datei gespeichert (siehe 4.6)
Hierarchy Tree	Graphischer Export des Hierarchiebaums zu FrameMaker (siehe 4.7)
Set of Subnets	Graphischer Export aller selektierten Teilnetze zu FrameMaker (siehe 4.7)

Das Edit - Menü

<i>Funktion</i>	<i>Beschreibung</i>
New Arrange	Der Hierarchiebaum wird neu aufgebaut
Unwrap	Entfalten eines Netz-Icons (siehe 4.4)
Unwrap All	Entfalten des gesamten Hierarchiebaums (siehe 4.4)
Wrap	Zusammenfalten eines Netz-Icons (siehe 4.4)
Wrap All	Zusammenfalten des gesamten Hierarchiebaums (siehe 4.4)
Undo	Einstufiges Undo (siehe 4.5)
Cut	Ausschneiden von selektierten Teilnetzen und Ablage im Clipboard (siehe 4.5)
Copy	Kopieren von selektierten Teilnetzen ins Clipboard (siehe 4.5)
Paste	Kopieren des Clipboard-Inhaltes in ein selektiertes Teilnetz (siehe 4.5)
Delete	Löschen von selektierten Teilnetzen (siehe 4.5)
Net-Item Attributes	Modifizieren der Attribute eines selektierten Netz-Icons (siehe 4.5)
Set of Net-Items	Modifizieren der Attribute mehrerer selektierten Netz-Icons (siehe 4.5)

B. Beispiel *.pedrc* File

```
-- .pedrc

-- User OutputRoutines Directory

_UserGenRoutineDir_ = "/home/rt/ped/pedenv/lib/"

-- Number of Menu Entries

_NrOfMenuEntries_ = 2

-- Export Menu Definition

_ExportMenu_ = {}

_ExportMenu_[1] = {}
_ExportMenu_[1].MenuEntry = "GEN - Generic Output..."
_ExportMenu_[1].PopupEntry = "Export GEN File"
_ExportMenu_[1].PopupEntryLabel = "GEN File:"
_ExportMenu_[1].RoutineName = "gen.lua"
_ExportMenu_[1].FileExtension = ".gen"

_ExportMenu_[2] = {}
_ExportMenu_[2].MenuEntry = "PNT - Integrated Net Analyzer..."
_ExportMenu_[2].PopupEntry = "Export PNT File"
_ExportMenu_[2].PopupEntryLabel = "PNT File:"
_ExportMenu_[2].RoutineName = "pnt.lua"
_ExportMenu_[2].FileExtension = ".pnt"
```


C. Beispiele generischer Exportroutinen

C.1. Testexport

```
-- Generic Output Routines

-- Start Declarations

-- Routines To Print Structured Data

function write_record(t, depth)
    local i, v = next(t, nil)
local j = depth

    write("{\n")
    while i do
        store(i, v, depth + 1)
        i, v = next(t, i)
    end

while j ~= 0 do
write('\t')
j = j - 1
end
    write("}")
end

function write_value(value, depth)
    local t = type(value)
        if t == 'nil' then write("nil")
        elseif t == 'number' then write(value)
        elseif t == 'string' then write("'",value, "'")
        elseif t == 'table' then write_record(value, depth)
        end
write('\n')
if depth == 0 then
write('\n')
end
end

function store(name, value, depth)
local i = depth
```

```

while i ~= 0 do
write('\t')
i = i - 1
end

        write(format("%s = ", name))
        write_value(value, depth)
end

-- Start Code

FileDesc = writeto(_FileName_)

write("\n-- Aktiv Settings\n\n")

store("_CurrentMenuEntry_", _CurrentMenuEntry_, 0)
store("_FileName_", _FileName_, 0)
store("_FileNameHead_", _FileNameHead_, 0)
store("_FileNameTail_", _FileNameTail_, 0)

write("\n-- Net Settings\n\n")

store("_PlaceTable_", _PlaceTable_, 0)
store("_TransitionTable_", _TransitionTable_, 0)
store("_ArcTable_", _ArcTable_, 0)
store("_CommentTable_", _CommentTable_, 0)

writeto()

-- End Of Code

```

C.2. Export zu INA

```

-- INA Output Routines

-- Start Declarations

-- Create Tables

PreT = {}
PostT = {}

-- Init Table

function init_table(t, table)
    local i, _ = next(table, nil)

    while i do
        t[i] = {}
        i, _ = next(table, i)
    end
end
end

```

```
-- Make Pre- or PostTransition Relation

function make_PRel(t, f, atable)
local i, v = next(atable, nil)

while i do
if v[f] then
t[v["Place"]][v["Trans"]] = 1
end
i, v = next(atable, i)
end
end

-- Output Routines

function writeListOfPlaces(PlaceTable, PreT, PostT)
local i, v = next(PlaceTable, nil)

while i do
write(" ",i," ",v.Tokens," ")

local j, _ = next(PreT[i], nil)

while j do
write(j)
j, _ = next(PreT[i], j)
if j then
write(" ")
end
end

write(", ")

j, _ = next(PostT[i], nil)

while j do
write(j)
j, _ = next(PostT[i], j)
if j then
write(" ")
end
end

write("\n")

i, v = next(PlaceTable, i)

end
end

function writeListOfT(TransitionTable)
local i, v = next(TransitionTable, nil)

while i do
```

```

write(format("%8d: %-20.16s    0    0\n",i,v.Name))

i, v = next(TransitionTable, i)
      end
end

function writeListOfP(PlaceTable)
local i, v = next(PlaceTable, nil)

while i do
write(format("%8d: %-20.16s65535    0\n",i,v.Name))

i, v = next(PlaceTable, i)
      end
end

-- Start Code

init_table(PreT, _PlaceTable_)
init_table(PostT, _PlaceTable_)

make_PRel(PreT, "TP", _ArcTable_)
make_PRel(PostT, "PT", _ArcTable_)

-- Begin Output

FileDesc = writeto(_FileName_)

write("P    M    PRE,POST NETZ 0:",_FileNameTail_,"\n")

writeListOfPlaces(_PlaceTable_, PreT, PostT)

write("@\n")

write("place nr.      name capacity time\n")

writeListOfP(_PlaceTable_)

write("@\n")

write("trans nr.      name priority time\n")

writeListOfT(_TransitionTable_)

write("@\n")

writeto()

-- End Of Code

```

Literaturverzeichnis

- [Czi93] CZICHY, G.: *Entwurf und Implementierung eines graphischen Editors für hierarchische Petrinetzmodelle*. Diplomarbeit, TU Dresden, 1993.
- [Eme90] EMERSON, E. A.: *Handbook of Theoretical Computer Science*, Kapitel Temporal and modular logic, Seiten 996–1072. J. van Leeuwen, Amsterdam, Netherland, 1990.
- [HDSml] HEINER, M., P. DEUSSEN und J. SPRANGER: *A Case Study in Developing Control Software of Manufacturing Systems with Hierarchical Petri Nets*. Proceedings of the 1st Int. Workshop on Manufacturing and Petri Nets held at Int. Conf. on Application and Theory of Petri Nets (ICATPN '96), Osaka/Japan, June 1996 (<http://www-dssz.informatik.tu-cottbus.de/~wwwdssz/docs/publikationen.html>).
- [INA] INA: *Homepage*. <http://www.informatik.hu-berlin.de/~starke/ina.html>, April 1997.
- [LUA] LUA: *Homepage*. <http://www.inf.puc-rio.br/~roberto/lua.html>, Februar 1997.
- [PEP] PEP: *Homepage*. <http://www.informatik.uni-hildesheim.de/~pep/HomePage.html>, Juni 1996.
- [PRO] PROD: *Homepage*. <http://topos.hut.fi/~petrinet/prod.html>, Januar 1997.
- [Wik91] WIKARSKI, D.: *Zur Modellierung des Verhaltens verteilter Systeme auf der Basis von Netzen*. Interner Arbeitsbericht, Institut für Informatik und Rechentechnik, Berlin, 1991.