Anforderungsanalyse eines medizinischen Informationssystems mit Algebraischen High-Level-Netzen

Claudia Ermel

17. April 1996

Inhaltsverzeichnis

Kapitel 1

Einleitung

Eine wichtige Rolle bei der Erprobung neuer theoretischer Konzepte und Modellierungstechniken in der Software-Entwicklung spielen Fallstudien. Sie bieten die Möglichkeit, die entwickelten Konzepte im Hinblick auf ihre Anwendbarkeit auf größere Systeme zu prüfen, den Nutzen der Konzepte zu veranschaulichen und eventuell die Notwendigkeit von Ergänzungen des Formalismus im Hinblick auf seine praktische Anwendbarkeit festzustellen.

Wir entwickeln in dieser Arbeit als Fallstudie die Anforderungsanalyse eines medizinischen Informationssystems mit Algebraischen High-Level-Netzen, einer Weiterentwicklung von Petrinetzen. Wir demonstrieren anhand der Fallstudie die Konzepte der horizontalen Strukturierung und der regelbasierten Abstraktion/Verfeinerung von Netzen und prüfen ihre praktische Bedeutung für formale Systemspezifikation.

Von Petrinetzen zu Algebraischen High-Level-Netzen

Petrinetze gelten seit langem als einfache Modellierungstechnik für parallele bzw. verteilte Systeme, in denen mehrere Komponenten unabhängig voneinander arbeiten, aber miteinander kommunizieren können. Die Petrinetztechnik erlaubt es, verteilte Systeme anschaulich (graphisch) und mathematisch fundiert zu spezifizieren ([Pet62], [Rei85] und viele andere). Dies ist hauptsächlich auf folgende Eigenschaften von Petrinetzen zurückzuführen:

- *Einfache Beschreibungssprache*: Petrinetzspezifikationen enthalten einige einfache, aber mächtige Beschreibungskonzepte mit wenigen Symbolen und einer geringen Anzahl von Regeln.
- *Anschaulichkeit*: Die graphische Repräsentation des spezifizierten Systems als Petrinetz erlaubt es, sowohl die statische Struktur als auch die dynamischen Aspekte des Systems in natürlicher Weise zu veranschaulichen und trägt so zum Verständnis der Netzspezifikation bei.
- *Universalität*: Petrinetze werden zur Modellierung von Systemen aus den verschiedensten Bereichen eingesetzt, von Softwareentwicklung über Elektrotechnik bis hin zu Verfahrenstechnik und Maschinenbau.

- *Wohlfundierte Theorie*: Die mengentheoretische Fundierung der Petrinetztheorie liefert wohldefinierte semantische Modelle und erlaubt exakte mathematische Analysen von Netzeigenschaften (z.B. Invarianten- oder Erreichbarkeitsanalyse).
- *Simulationsfähigkeit*: Die Möglichkeit der Simulation des Prozeßablaufs in einer Petrinetzspezifikation erlaubt die Prüfung der (informellen) Anforderungen an das System.
- *Existenz von Werkzeugen*: Es gibt eine reichhaltige Auswahl von Tools für die verschiedensten Petrinetz-Klassen, die die Spezifikation, Simulation und Analyse von Petrinetzen unterstützen.

Für die Modellierung großer, komplexer Systeme fehlen der klassischen Petrinetztechnik allerdings formale Konzepte zur Kompositionalität, also zum Zerlegen größerer Netze in kleinere bzw. zum Komponieren kleinerer Netze zu größeren, wobei sich die Semantik dann aus der Semantik der Komponenten bestimmen läßt. Auch Konzepte zur Abstraktion, ähnlich dem Information-Hiding-Konzept bei Programmiersprachen, sind bei der Modellierung komplexer Systeme unentbehrlich.

Es wurde daher in den letzten Jahren ein algebraischer Ansatz von Petrinetzen entwickelt, der Petrinetze als grundlegende algebraische Struktur auffaßt (z.B. [MM90]). Damit werden Sätze und Methoden aus der Kategorientheorie auf Petrinetze anwendbar, die eine algebraische Definition der Netzsemantik gestatten. Auch Konstruktionen zur Komposition von Netzen lassen sich kategoriell definieren (z.B. [DH90], [Erm94]).

Dennoch ist das Petrinetz-Konzept weiterhin ein Basis-Konzept, daß sich nur für die Beschreibung des Datenflusses des Systems eignet, aber nicht, um die Daten selbst zu spezifizieren.

Im Bereich der formalen Spezifikation von Datentypen sind seit vielen Jahren Sprachen und Methoden entwickelt worden, mit denen sich auch Konsistenz- und Korrektheitsnachweise für Datentypspezifikationen durchführen lassen, wie z.B. algebraische Spezifikationen [EM85].

Der erste wichtige Schritt, Datentypspezifikationen mit Petrinetzen zu verbinden und höhere Abstraktionsebenen mit Petrinetzen zu erreichen, war die Entwicklung sogenannter *Coloured Petrinetze* von Jensen in [Jen81]. Das Konzept sieht verschiedene Typen von Token auf den einzelnen Stellen vor, sowie verschiedene „Betriebsmodi", unter denen Transitionen schalten können.

In [Vau86] werden zum ersten Mal Coloured Petrinetze mit algebraischen Spezifikationen kombiniert. Die Weiterentwicklung dieser Idee, das Konzept der *Algebraischen High-Level- Netze* (AHL-Netze, siehe z.B. [Hum89], [DHP91], [EPR94]) kombiniert kategorielle Stellen/Transitionen-Netze mit algebraischen Spezifikationen: Token werden durch Datenelemente eines algebraischen Datentyps spezifiziert, die Kanten des Netzes sind mit Termen mit Variablen der entsprechenden algebraischen Spezifikation beschriftet, und die Transitionsmodi werden von den Belegungen der Variablen der Terme der hinein- bzw. hinausführenden Kanten bestimmt. Als Strukturierungstechniken wurden die Konzepte *Union* und *Fusion* für horizontale Strukturierung (Komposition – Zerlegung) und Transformationssysteme für vertikale Strukturierung (Abstraktion – Verfeinerung) für AHL-Netze entwickelt. Dabei werden AHL-Netze als Erweiterung kategorieller PT-Netze definiert.

Es wird daran gearbeitet, die kategorielle Semantik von P/T-Netzen, sowie Netzanalysemethoden wie Invarianten und Erreichbarkeits- und Markierungsgraphen auf AHL-Netze zu übertragen. Auch die Entwicklung von Tools zur Unterstützung bei der Konstruktion, Simulation, Verfeinerung, Komposition und Verifikation von AHL-Netzen, verbunden mit größeren Fallbeispielen, wird ein wichtiger Schritt bei der Weiterentwicklung der Theorie, sowie der AHL-Netze-Anwendung in der Praxis sein.

Die Fallstudie: Ein medizinisches Informationssystem

Informationstechnik in der Medizin

Der Einsatz der Informationstechnik in der Medizin hat für Patienten und Ärzte eine Fülle neuer technikgestützter Diagnoseverfahren eröffnet. Wie wichtig der Einsatz von Computern ist, zeigen die umfangreichen und aufwendigen Analysetechniken, die im Labor zur Probenanalyse verwendet werden, sowie die vielseitige Sensortechnik und die komplexe Bildverarbeitung, die arztgerecht aufbereitet werden muß, und in der Verknüpfung, Darstellung und Speicherung spezifischen zeitkritischen Vorgaben unterliegt. Die rasche Entwicklung der computergestützten Bildgebung von Teilen des menschlichen Körpers erweitert die diagnostischen Möglichkeiten beachtlich. Die Belastungen des Patienten z.B. durch Kontrastmittelinjektion oder die Strahlungsdosis werden dadurch deutlich gemindert. Dynamische Vorgänge, wie z.B. das schlagende Herz und das Verhalten der Herzkranzgefäße können phasenaufgelöst abgebildet und klinisch beurteilt werden.

Die Anwendung moderner Technik in der Medizin führt allerdings zu einem hohen Aufwand, um alle Meßdaten, Ultraschallaufnahmen, Endoskopien, Röntgen- und Computertomographie-Bilder usw. für jeden Patienten zu verwalten. Kliniken und Arztpraxen stehen vor dem Problem, mit einem immer schneller wachsenden Datenvolumen umzugehen. Vernetzte Kommunikationssysteme und Online-Datenbanken sind erforderlich, damit Patientendaten „telepräsent" und dem behandelnden Arzt jederzeit bei Einhaltung des Datenschutzes verfügbar gemacht werden können.

Das Informationsmanagementsystem HDMS

Am Deutschen Herzzentrum Berlin (DHZB) arbeitet aus diesem Grund die Projektgruppe Medizin/Informatik (PMI) seit 1989 an dem medizinischen Informationssystem HDMS (Heterogeneous Distributed Information Management System). Dessen erklärtes Ziel ist „die Realisierung der allgemeinen rechnergestützten und kontrollierten Integration und Kommunikation von Patientendaten und Dokumenten. Es dient dem Zweck, die medizinische Versorgung zu verbessern, zu vereinfachen und die Nutzung analytischer und diagnostischer Leistungen über organisatorische und staatliche Grenzen hinweg in direkter Weise zu ermöglichen" [FHMO91].

Der Schwerpunkt des Vorhabens liegt also auf Integration und Kommunikation einer Vielzahl von Daten äußerst unterschiedlichen Typs, z.B. Textdaten wie personenbezogene Patientendaten oder Krankheitsgeschichten, Befundberichte oder Diagnosen, Bilddaten in Form von Röntgenbildern oder Ultraschallechos bis hin zu bewegten Bildern wie Herzkatheterfilmen. Zur Heterogenität der Daten kommt ihre Verteiltheit hinzu: Bilddaten werden an verschiedenen Orten mit unterschiedlichen Geräten erzeugt, Textdaten (rechnergestützt erstellt oder auch handschriftlich bzw. per Schreibmaschine) können vom behandelnden Hausarzt oder Fachärzten stammen und in verschiedenen Praxen aufbewahrt werden oder beim Krankenhausaufenthalt des Patienten anfallen.

Die Problemstellung von HDMS ist insofern typisch für die meisten modernen Systementwicklungen, als daß die Systemkonfiguration, in welche das neu zu entwickelnde Produkt einzubetten ist, einen großen Teil der Anforderungsbeschreibung der Gesamtsystem-Spezifikation ausmachen.
Das HDMS-Projekt eignete sich daher auch gut als Feld für die Erprobung theoretischer Konzepte innerhalb des Projektes Korrekte Software (KORSO, 1991-1994), dessen Projektziel es war, bereits vorhandene Methoden zur formalen Spezifikation, Entwicklung und Verifikation von Software so weiterzuentwickeln, daß sie für die Praxis verwendbar werden.

Da es sich bei HDMS um ein sehr großes Projekt handelt, wurde für KORSO eine angemessene Einschränkung und Abstraktionsebene des gesamten HDMS als Fallstudie gewählt, das sogenannte HDMS-A (abstraktes HDMS). Diese Einschränkungen und die Spezifikation von HDMS-A im KORSO-Projekt wird konkret im 3. Kapitel dieser Arbeit beschrieben. Sie verwendet informelle, semi-formale und formale Darstellungstechniken bei der schrittweisen Entwicklung einer Anforderungsanalyse für den Kern von HDMS-A, der sogenannten *Elektronischen Patientenakte*. Die Schwierigkeit bei KORSO bestand darin, die verschiedenen, nur auf informeller Ebene zusammenhängenden Spezifikationsteile in einer konsistenten, einheitlichen, formalen Anforderungsdefinition zu integrieren.

Zielsetzung der Fallstudie

Das Problem, für HDMS-A eine angemessene formale Anforderungsanalyse zu entwickeln, bildet den Ausgangspunkt für die vorliegende Fallstudie: Wir stellen eine Modellierung der IST-Analyse von HDMS-A mit AHL-Netzen vor und integrieren damit die Beschreibung von Dokumenten und Arbeitsabläufen in einer einzigen formalen Spezifikation. Mit Hilfe des auf High-Level-Replacement-Systemen [EHKP91], [Pad96] und Graphgrammatiken [Ehr79] beruhenden formalen Konzepts der *Transformation* von AHL-Netzen [PER95] [REP93] modellieren wir dann regelbasierte, mit horizontalen Struktrurierungstechniken verträgliche Abstraktionsschritte vom IST-System über die Essenz des IST-Systems zur funktionalen Essenz, wo von Technologieabhängigkeiten und zufälligen Gegebenheiten traditioneller Arbeitsabläufe abstrahiert und das System um zusätzliche Leistungen ergänzt wird. Erste Ideen zu dieser Arbeit liefert der Bericht „*Formal Requirements Analysis Using Algebraic High-Level Nets and Transformation: A KORSO Case Study improving parts of HDMS-A*“ [PREC93], wo erstmals der Einsatz von AHL-Netzen zur Spezifikation eines Teils der IST-Analyse (Herzkatheteruntersuchung und -diagnose) von HDMS-A beschrieben wird.

Ziel der Fallstudie ist es darüberhinaus, die Strukturierungs- und Transformationskonzepte von AHL-Netzen zu demonstrieren und im Hinblick auf ihre Anwendbarkeit für komplexe Systeme die Notwendigkeit von Ergänzungen des Formalismus um weitere formale, semantisch fundierte Modellierungskonzepte (z. B. um ein Modulkonzept) zu begründen.

Übersicht über die Kapitel

Wir geben an dieser Stelle eine kurze Übersicht über den Inhalt der Kapitel dieser Arbeit an:

Im zweiten Kapitel *Algebraische High-Level-Netze: Eine Einführung* führen wir den Leser zunächst in das Vorgehen bei der Modellierung mit AHL-Netzen anhand eines Teils unserer Fallstudie ein, um damit das Grundwissen bereitzustellen für die im dritten Kapitel folgende gesamte Fallstudie, die mit AHL-Netzen spezifizierte Anforderungsanalyse des medizinischen Informationsystems, im folgenden HDMS-AHL genannt. Die Fallstudie umfaßt die IST-Analyse, also die Spezifikation von Dokumenten und Arbeitsabläufen des gesamten IST-Systems und enthält Regeln für das Auslösen essentieller Bestandteile des IST-Systems (IST-Essenz) und für Teile der Fallstudie die Transformation von der IST-Essenz zur funktionalen Essenz des Systems. Ein Vergleich mit den im KORSO-Projekt verwendeten Spezifikationsmethoden rundet das Kapitel *Fallstudie* ab.

Im vierten Kapitel folgt die formale Definition von AHL-Netzen. Wir definieren und demonstrieren das Schaltverhalten von AHL-Netzen und zeigen die Konstruktion von verhaltensäquivalenten Stellen/Transitionen-Netzen (*Flattening*). Anschließend führen wir die Kategorie **AHLNET** der AHL-Netze ein, deren Kovollständigkeit eine wichtige theoretische Voraussetzung für die formale Komposition von AHL-Netzen ist. Die horizontalen Strukturierungskonzepte *Union* und *Fusion* werden als Kolimiten in der Kategorie AHLNET formalisiert. Außerdem beschäftigen wir uns mit dem vertikalen Strukturierungskonzept der regelbasierten Abstraktions- und Verfeinerungsmechanismen, die wir auf ihre formale Basis, sogenannte High-Level-Replacement-Systeme, zurückführen. Beide Strukturierungsprinzipien sind miteinander verträglich, was wiederum anhand von Netzen aus der Fallstudie demonstriert wird.

Eine Zusammenfassung der Arbeit, sowie Hinweise auf Möglichkeiten für die Erweiterung der Fallstudie, aber auch für den Ausbau des AHL-Netze-Formalismus (Modulkonzept, hierarchische AHL-Netze) erfolgen in Kapitel 5.

Kapitel 2

Algebraische High-Level-Netze - Eine Einführung

Dieses Kapitel soll den Leser in die Lage versetzen, das Fallbeispiel auch ohne Vorkenntnisse über AHL-Netze verstehen zu können. Es enthält wenig technische Details und formalen Hintergrund, setzt aber grundlegende Kenntnisse über Stellen/Transitionen-Netze voraus (siehe z.B. [Rei85], [Erm94]).

Algebraische Spezifikation und Netzstruktur

Die Theorie der AHL-Netze baut auf Teilen der Spezifikationsmethoden Petrinetze, algebraische Spezifikationen und High-Level-Replacement-Systemen auf. AHL-Netze gehören zur Klasse der sogenannten High-Level-Petrinetze, wie auch Coloured Petrinetze [Jen92] und Predicate/Transition-Netze [GL81], [Gen91], die seit etwa 15 Jahren untersucht werden. High-Level-Netze entstanden aus der Idee heraus, das Konzept der Petrinetze um Abstraktionsmöglichkeiten zu erweitern.

In Stellen/Transitionen-Petrinetzen gibt es nur eine Art von Token, was bedeutet, daß der Zustand einer Stelle mithilfe einer natürlichen Zahl (ihrer Tokenanzahl) beschrieben wird. In High-Level-Netzen dagegen kann jedes Token komplex strukturierte Daten enthalten, die z.B. den Zustand eines Prozesses oder eine gesamte Datenbank beschreiben.

AHL-Netze werden bei der Spezifikation verteilter und nebenläufiger Systeme eingesetzt. Die Kombination von Petrinetzen mit algebraischen Spezifikationen gestattet die Beschreibung der statischen und der dynamischen Aspekte des spezifizierten Systems innerhalb desselben Formalismus. Der statische Teil eines AHL-Netzes besteht aus algebraischen Datentypspezifikationen, die die vom System benötigten bzw. generierten Daten beschreiben. Der dynamische Teil verarbeitet die spezifizierten Daten in wohldefinierten Schritten. So kann dargestellt werden, wie die Daten im System verteilt sind und welche Daten bei Ablauf des Prozesses berechnet werden.

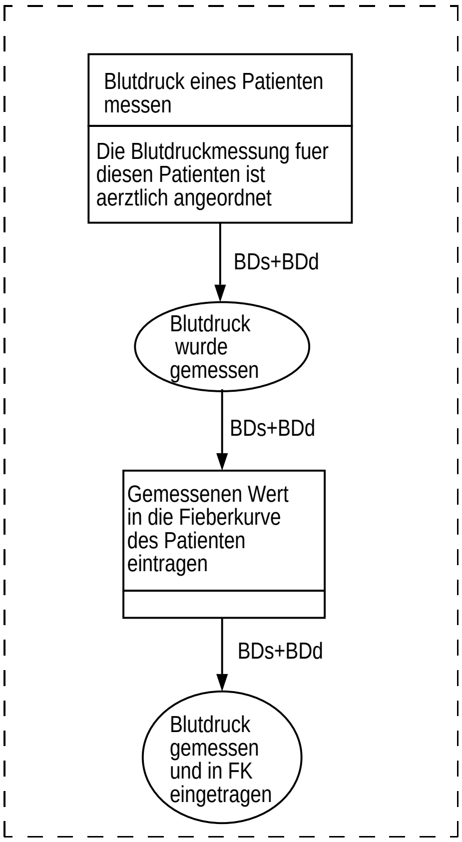
Die Netzstruktur eines AHL-Netzes besteht aus Stellen (Systemzustände) und Transitionen (Zustandsübergänge), die gegenseitig über gerichtete Kanten verbunden sind, und entspricht insoweit der Struktur klassischer Petrinetze. Allerdings ist eine Kante nicht nur mit einem Gewicht beschriftet, also einer Zahl, die die Anzahl der zu bewegenden Token vorschreibt, sondern mit einem Term zur Spezifikation, der auch Variablen enthalten kann, und der die Daten beschreibt, die den entsprechenden Stellen abgezogen bzw. hinzugefügt werden sollen. Das bedeutet, ein Zustandsübergang im System, auch Aktion genannt, der durch das Schalten einer Transition modelliert wird, entfernt Daten aus bestimmten definierten Systembereichen und berechnet daraus neue Daten, die anderen (oder den gleichen) Systembereichen hinzugefügt werden.

Betrachten wir als einführendes Beispiel ein Teilnetz aus unserer HDMS-A-Fallstudie:

Es wird folgender Sachverhalt modelliert: Der Patient befindet sich auf der Pflegestation. Der für ihn zuständige Pfleger ermittelt seinen Blutdruck, wenn dies vom Arzt auf dem Anordnungsbogen gefordert wurde. Der gemessene Blutdruck (zwei Werte: systolisch und diastolisch) wird vom Pfleger in das Dokument Fieberkurve eingetragen.

Wir sehen, daß im wesentlichen zwei Aktionen ausgeführt werden: das Messen und das Eintragen. Dementsprechend enthält unser Netz auch zwei Transitionen. Die erste Transition überführt den Systemzustand vor der Messung in den Zustand nach der Messung und vor dem Eintragen der Werte in die Fieberkurve. Die zweite überführt diesen Zustand in den Endzustand (Messung und Eintragung ausgeführt).

Ein Grobentwurf unseres Netzes könnte also zunächst einmal so aussehen:

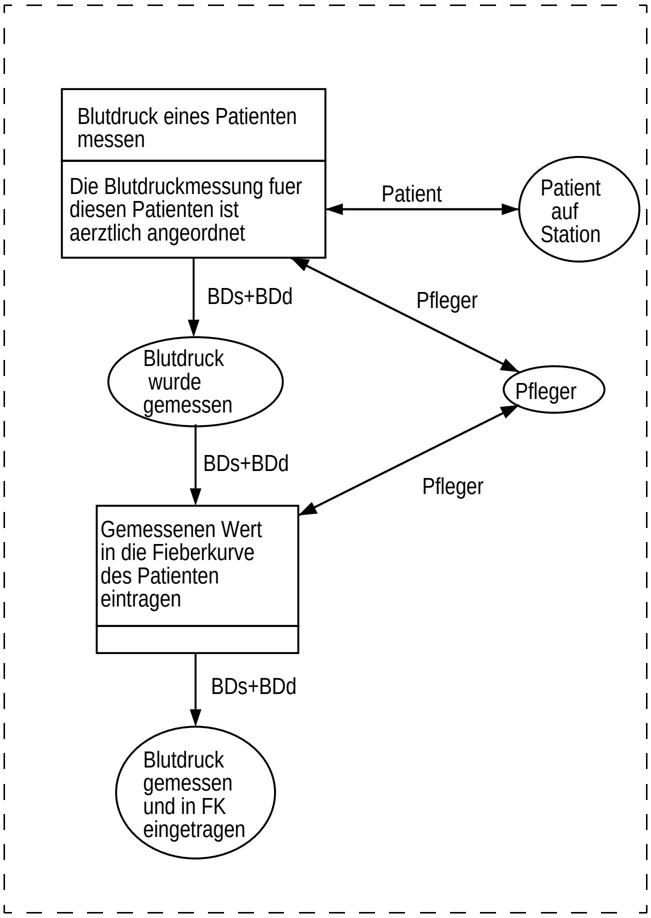


B0: Grobentwurf Blutdruckmessung

Der gestrichelte Rahmen deutet an, daß es sich hierbei noch nicht um ein syntaktisch korrektes AHL-Netz handelt.

Die konkreten Blutdruckwerte werden als Belegungen der entsprechenden Variablen (erkennbar an den großen Anfangsbuchstaben) über die Kantenbeschriftungen weitergereicht, und befinden sich nach Eintreten beider Ereignisse (im Modell: dem Schalten beider Transitionen) auf der Stelle *Blutdruck gemessen und in FK eingetragen.*
Wir geben die Schaltbedingung der ersten Transition (Blutdruckmessung ist angeordnet) zunächst informell an, da wir die zugrundeliegende formale Spezifikation der Dokumente noch entwickeln müssen.

Zunächst fällt auf, daß die agierenden Personen (Patient auf Station und Pfleger) in unserem Modell noch fehlen. Ein Token der Sorte *Patient* (wie komplex auch immer es spezifiziert ist) wird zum Schalten der ersten Transition gebraucht, und Token der Sorte *Pfleger* sind bei beiden Transitionen beteiligt. Wir fügen also dem bisherigen Netz zwei Stellen hinzu: die Stelle *Patient auf Station*, die sozusagen alle Patienten enthält, die sich momentan auf der Station aufhalten, und die Stelle *Pfleger*, die alle Pfleger beinhaltet. Je ein Exemplar (modelliert als Variable der entsprechenden Sorte) wird beim Schalten einer Transition abgezogen und von derselben Transition wieder hinzugefügt, denn Patient und Pfleger befinden sich nach der Blutdruckmessung immer noch auf der Station. Das wird durch den mit einer Variablen beschrifteten Doppelpfeil modelliert. Wir erhalten folgendes erweitertes Netz:



B1: Erweiterung um beteiligte Personen

Betrachten wir das Schaltverhalten von B1: Beide Transitionen können parallel schalten, wenn genügend Patienten und Pfleger auf den entsprechenden Stellen vorhanden sind. So kann der erste Pfleger den beim ersten Patienten gemessenen Wert in die Fieberkurve eintragen, während ein zweiter Pfleger bei einem anderen Patienten die Messung ausführt. Es gibt nun allerdings ein Problem. Betrachten wir folgenden konkreten Fall: Auf der Stelle *Pfleger* befindet sich der Pfleger *hans* (Datenelement zur Sorte $Pfleger$ einer passenden Algebra zur Spezifikation BEH-Spec, siehe S. 49). Auf der Stelle *Patient auf Station* befinden sich die beiden Patienten *otto* und $fritz$ (Daten der Sorte *Patient*). Für beide ist die Blutdruckmessung angeordnet, daß heißt für einen von ihnen kann die Transition *Blutdruck messen* schalten. Belegen wir die Variable *Patient* zunächst mit *otto* und die Variable $Pfleger$ mit *hans*. Die Transition schaltet: *otto* und *hans* werden aus ihrem Vorbereich entfernt, aber, da dieselben Stellen auch in ihrem Nachbereich liegen, wieder hinzugefügt. Der gemessene Blutdruck wird als Belegung der Variablen $BDs + BDd$ auf der Stelle *Blutdruck gemessen* abgelegt. Nun trägt der Pfleger die gemessenen Werte aber nicht sofort in die Fieberkurve ein, sondern mißt zunächst bei einem anderen Patienten den Blutdruck: Die Transition *Blutdruck messen* schaltet zum zweiten Mal, aber diesmal wird die Variable *Patient auf Station* mit $fritz$ belegt. Der Blutdruck von $fritz$ landet als zweiter $BDs + BDd$-Wert auf der Stelle *Blutdruck gemessen*. Will der Pfleger nun die Werte in die entsprechenden Fieberkurven eintragen, was durch das Schalten der zweiten Transition modelliert wird, so kommt er ganz durcheinander, weil die reinen Blutdruckwerte, so wie wir sie bisher modelliert haben, keine Zuordnung zu den entsprechenden Patienten ermöglichen. Auch die Fieberkurven müssen eindeutig identifiziert werden können.

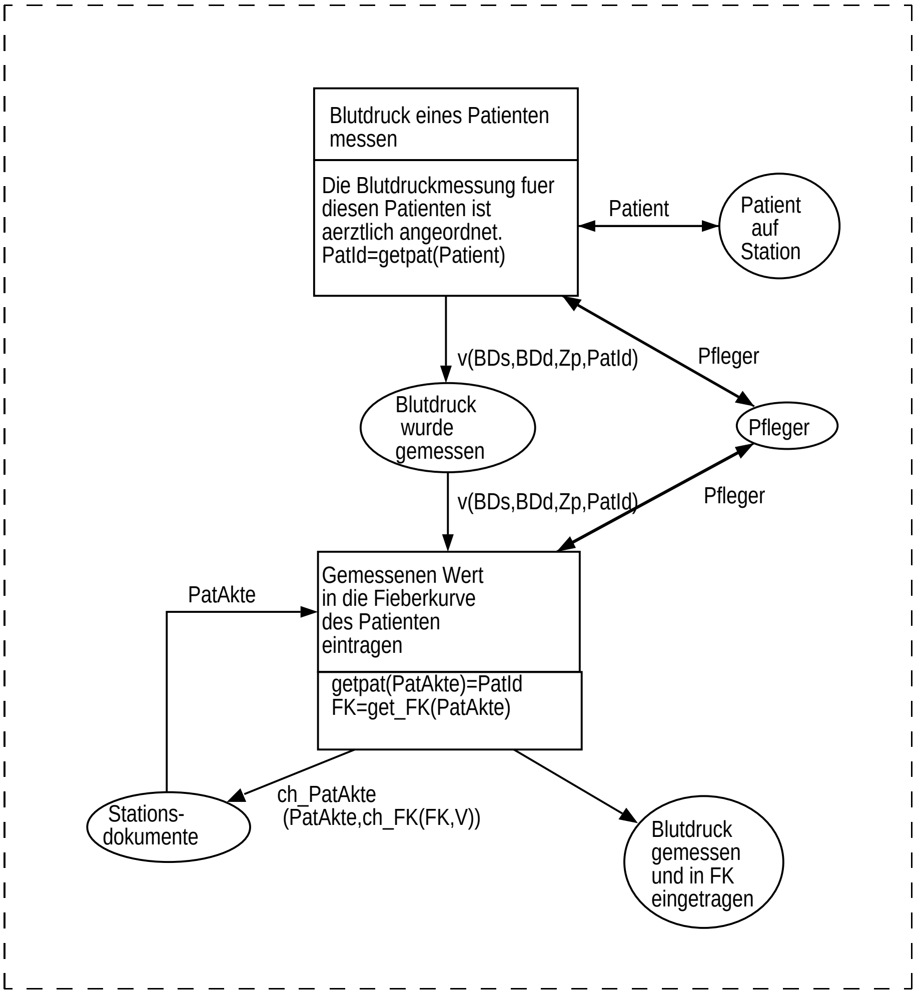
Wir führen aus diesem Grund die eindeutige Identifizierung $PatId$ ein, die man sich als eine Art Patientennummer vorstellen kann. Jedes Dokument, das sich auf einen bestimmten Patienten bezieht, enthält auch dessen $PatId$. Den Blutdruckwerten auf der Stelle *Blutdruck gemessen.* wird die $PatId$ über eine tupelbildende Operation hinzugefügt, so daß die Transition *Blutdruckwerte in FK eintragen* in ihrer Bedingung prüfen kann, ob die Blutdruckwerte und die Fieberkurve sich auch auf den gleichen Patienten beziehen. Die tupelbildende Operation v (wegen Sorte $Vitalwert$, siehe BEH-Spec, S. 63) gibt auch gleich noch den Zeitpunkt Zp der Messung mit an.

Die Fieberkurve selbst existiert bisher in unserer Spezifikation auch nur informell, da die Transition *Blutdruckwerte in FK eintragen* nicht beschreibt, wie das eigentlich geschehen soll. Wir erweitern daher unser Netz um eine Stelle $Stationsdokumente$. Dort befinden sich die sogenannten Patientenakten, die jeweils eine Sammlung aller patientenbezogenen Dokumente darstellen siehe BEH-Spec, S. 49), und aus denen somit die Fieberkurve zum Eintragen der Werte entnommen werden müssen. Der Ausdruck $Fieberkurve$ ist übrigens eine nicht ganz zutreffende Bezeichnung für eine Sammlung von Meßreihen bzw. deren Kurvendarstellungen (siehe BEH-Spec, S. 57). Wir werden später unser Netz auf andere sogenannte Vitalwertemessungen erweitern.

Soll nun ein Wert eingetragen werden, so wird die passende Fieberkurve der passenden Patientenakte entnommen und um den Eintrag erweitert der Stelle wieder hinzugefügt. Der Eintrag wird über eine Funktion $ch\_PatAkte(FK, V)$ (für *change PatAkte*) in der Schaltbedingung der

Transition realisiert (siehe BEH-Spec, S. 50) Die Variable V hat dabei die Sorte $Vitalwert$ und wird mit dem Term $v(BDs, BDd, Zp, PatId)$ belegt. Generell schreiben wir Funktionen und Konstanten in den Kantenbeschriftungen und Transitionsschaltbedingungen mit kleinen Anfangsbuchstaben, Variablen dagegen groß.

Es ergibt sich das Netz:



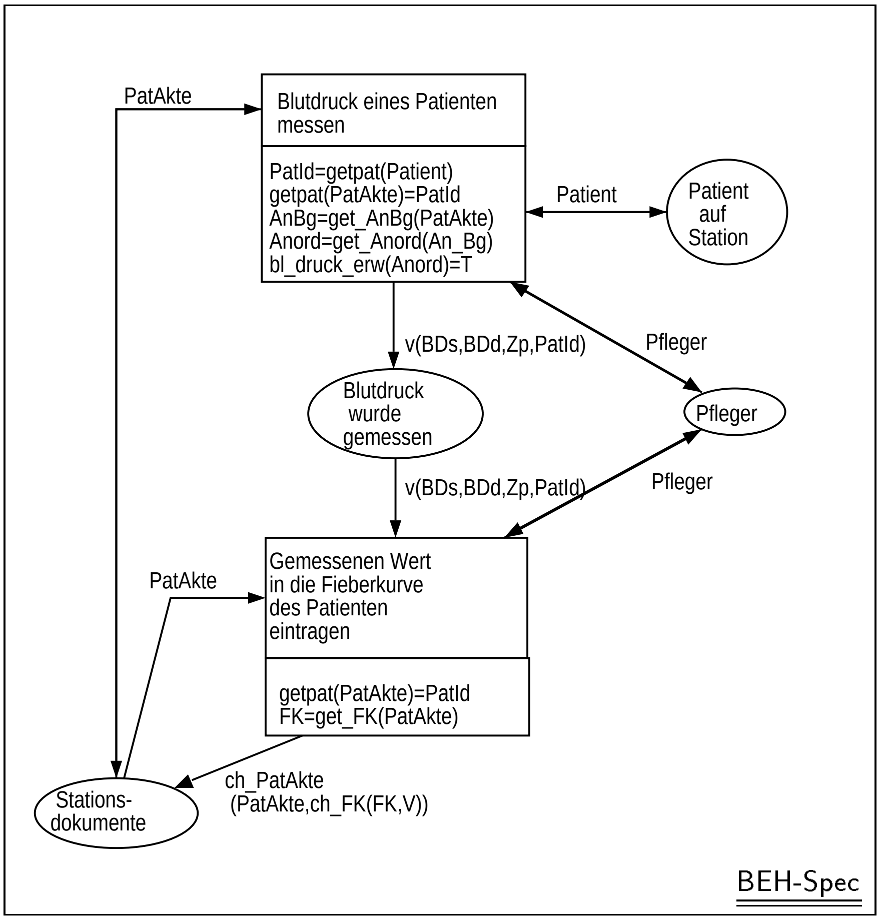
B2: Erweiterung um Stationsdokumente

Wir sehen, daß wir die Stelle *Blutdruck gemessen und eingetragen* nun nicht mehr brauchen, da sie nur dazu diente, den Endzustand unseres kleinen Systems zu modellieren. Dieser Endzustand (der Eintrag in der Fieberkurve) ist nun auf der Stelle *Stationsdokumente* repräsentiert.

Wir wollen nun noch die Schaltbedingung der Transition *Blutdruck messen* formal als Term angeben. Dazu müssen wir das Dokument *Anordnungsbogen* aus der Patientenakte entnehmen, das alle Anordnungen des Arztes als Tupel enthält Die Einträge des Tupels lassen sich als boolesche Werte spezifizieren, wobei der Wert *Blutdruck_erwünscht* auf dem entsprechenden

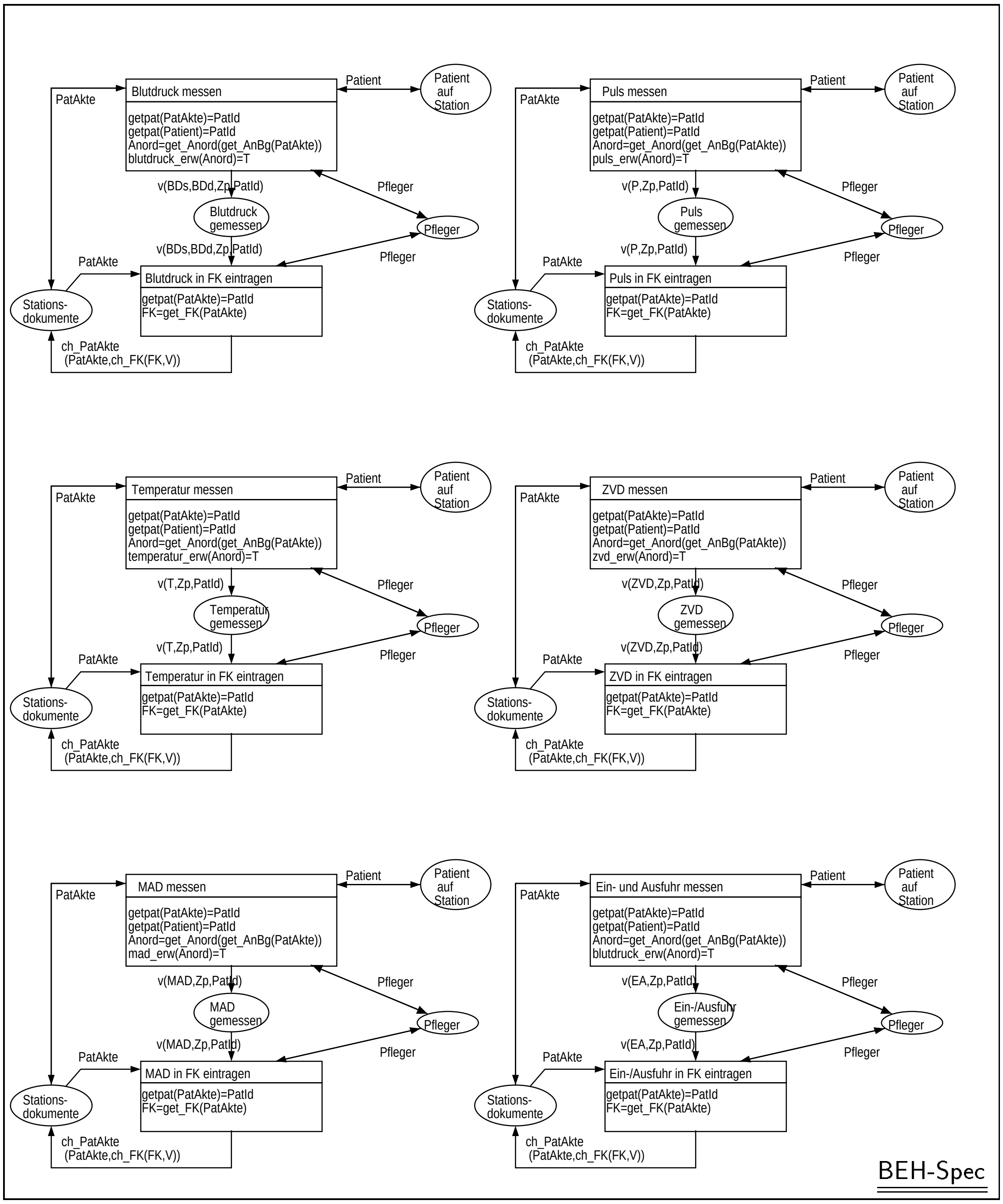
Anordnungsbogen *true* sein muß, damit die Bedingung erfüllt ist.

Der Anordnungsbogen wird von der Transition *Blutdruck messen* „gelesen", um die Bedingung zu prüfen (siehe BEH-Spec, S. 52). Die Patientenakte mit dem Anordnungsbogen wird dann unverändert wieder auf der Stelle *Stationsdokumente* abgelegt, was wir im Netz wiederum als Doppelpfeil darstellen:



B3: Formale Beschriftungen und Schaltbedingungen

In Netz B3 haben wir die Blutdruckmessung erstmals syntaktisch korrekt als AHL-Netz spezifiziert und dabei auch die zum Netz gehörende algebraische Spezifikation BEH-Spec mit angegeben. Im Rahmen der Behandlung eines Patienten werden allerdings noch weitere Vitalwerte des Patienten laut Anordnung oder nach Routine regelmäßig vom Pfleger gemessen und in die Fieberkurve eingetragen. Wir können nun zunächst der Einfachheit halber für jeden anderen Vitalwert (Puls, Temperatur, Mittlerer Arterieller Druck, Zentralvenöser Druck, Ein-/Ausfuhr) je ein Netz mit der Struktur angeben, wie wir es für die Blutdruckmessung entwickelt haben, und diese Teilnetze nebeneinandersetzen, wobei wir das Gesamtnetz als eine mögliche Modellierung der Vitalwertemessung auffassen. Wir erhalten:



B4: Einzelne Vitalwertemessungen

Es fällt auf, daß die Komplexität dieses Netzes, die Anzahl seiner Stellen und Transitionen, schon ziemlich hoch ist. Außerdem sind einige Stellen mehrfach vorhanden. Da die Stellen *Patient auf Station* nicht miteinander verbunden sind, ist das System so auch nicht korrekt modelliert, da an den Patienten, die sich auf einer Stelle befinden, auch nur eine bestimmte Messung vorgenommen werden kann. Notwendig ist also eine Verknüpfung der Teilnetze derart, daß das Ergebnis ein zusammenhängendes Netz liefert, das sämtliche Meßvorgänge für alle Patienten modelliert. Sehen wir uns dazu die Strukturierungsmöglichkeiten an, die es für AHL-Netze gibt.

Strukturierungsmöglichkeiten für AHL-Netze

Weil die Komplexität industrieller Softwareprodukte immer mehr zunimmt, wächst auch der Bedarf nach adäquaten Strukturierungstechniken, die schon in der Spezifikationsphase eine strukturierte Entwicklung von Softwaresystemen gestatten. Wir unterscheiden zwei Arten von Strukturierungstechniken: zum einen *Horizontale Strukturierung*, die die Teile des Systems untereinander und mit dem gesamten System in Beziehung setzt (Komposition/Dekomposition), und zum anderen *Vertikale Strukturierung*, die die Beziehung der verschiedenen Stufen einer Abstraktion bzw. Verfeinerung des Systems beschreibt.

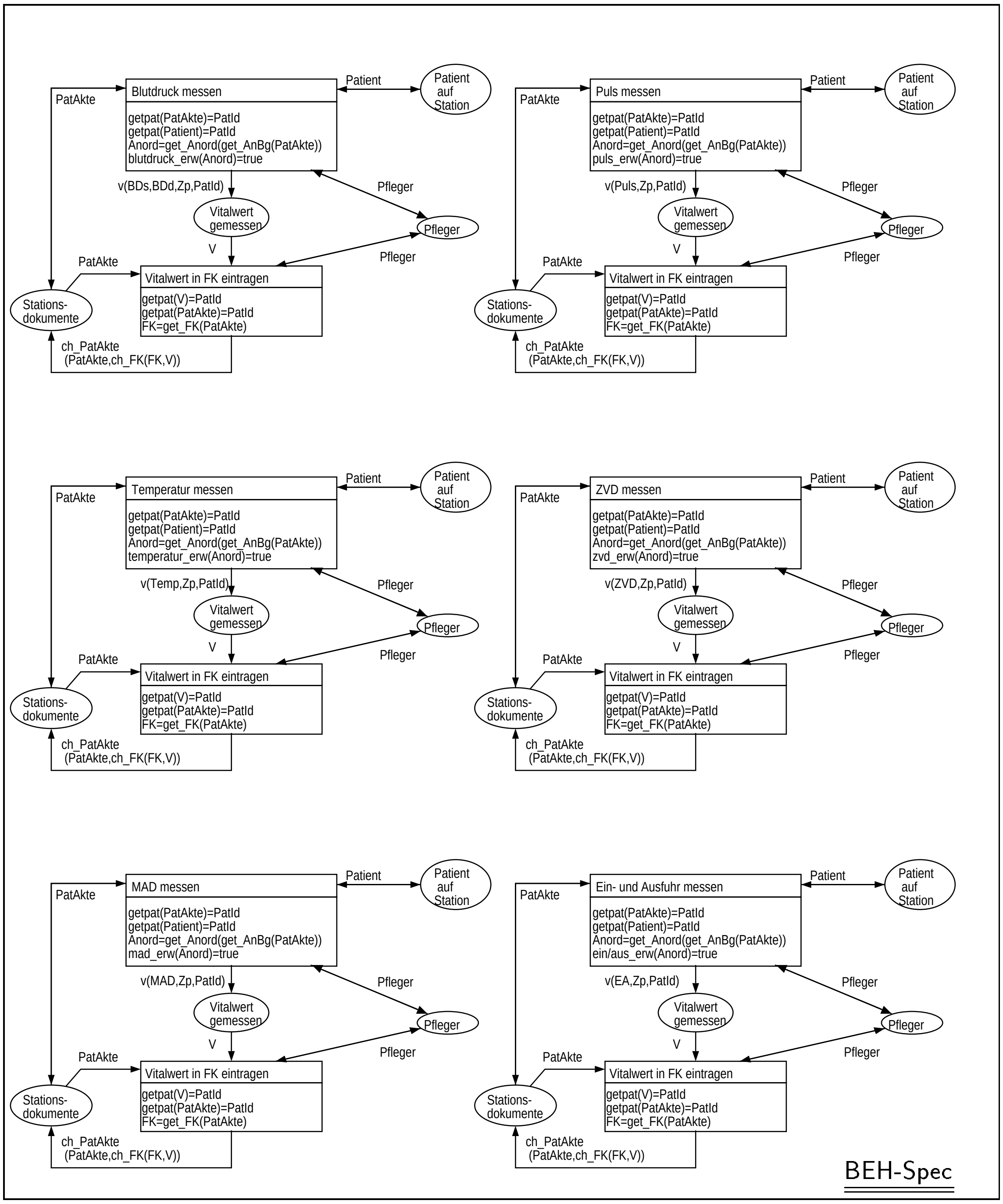
Wir wollen an dieser Stelle die Techniken *Union* und *Fusion* zur horizontalen Struktrurierung unseres Beispielnetzes einsetzen.

Fusion

Die Technik *Fusion* kann dabei als Identifikation von Teilnetzen eines Gesamtnetzes aufgefaßt werden. Für die Lesbarkeit von AHL-Netzen kann es z.B. durchaus sinnvoll sein, dasselbe Teilnetz mehrmals darzustellen, wobei eine virtuelle Verbindung zwischen den Teilnetzen besteht, sodaß auf semantischer Ebene das Teilnetz nur einmal vorhanden ist. Die Strukturierungstechnik *Fusion* ist dabei der formale Ausdruck dieser virtuellen Verbindung zwischen gleichen Teilnetzen.

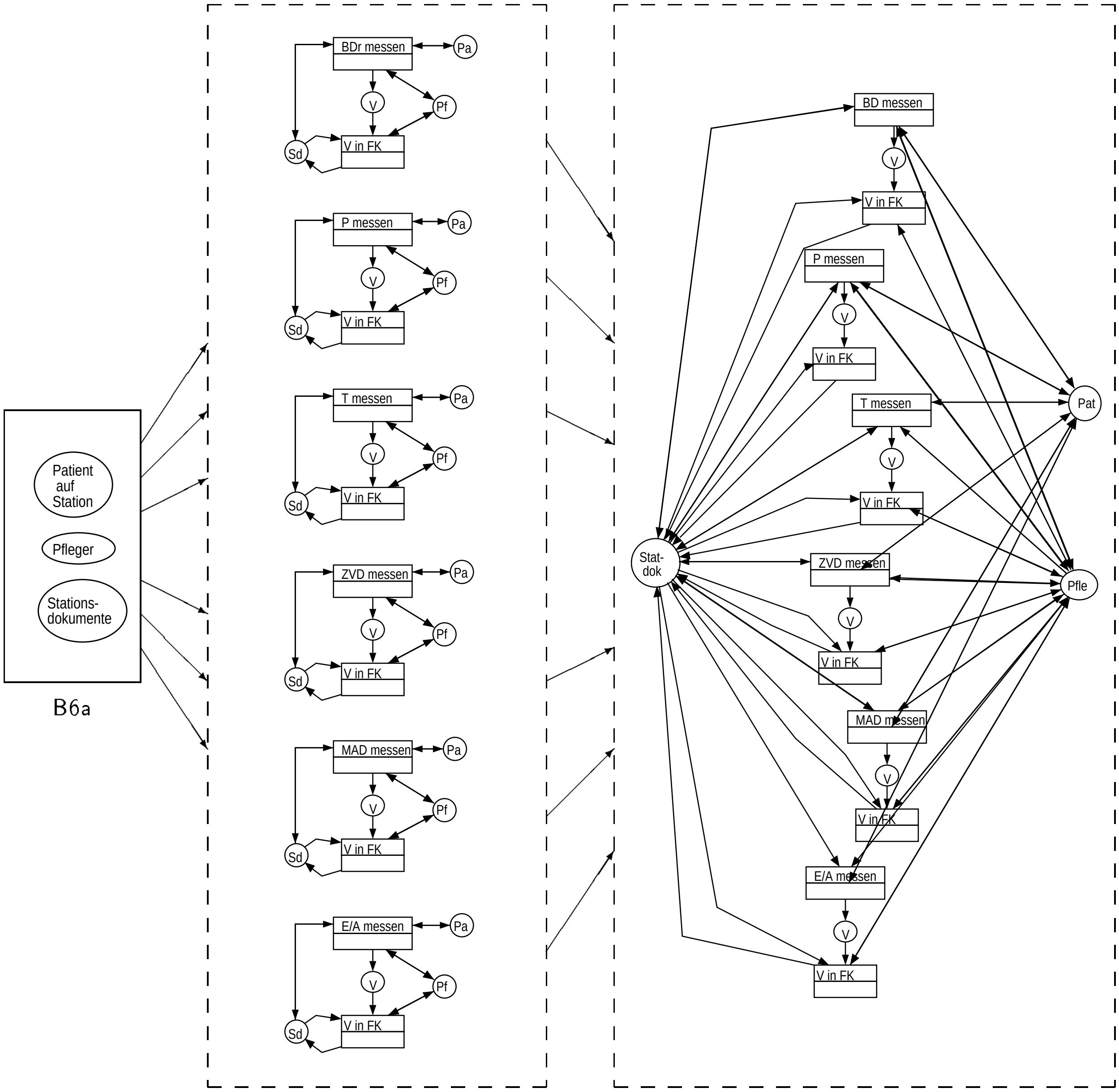
Bevor wir die Operation $Fusion$ auf unser bisheriges AHL-Netz anwenden, sehen wir uns das Netz B4 noch einmal genauer an: Allen Teilnetzen ist gemeinsam, daß der gemessene Wert nach der Messung in die Fieberkurve eingetragen wird. Allerdings läßt sich die Operation $Fusion$ über den Teilnetzen so nicht anwenden, da die Stellen unterschiedliche Namen haben (*Blutdruck gemessen*, *Puls gemessen*, *Temperatur gemessen*, ...) und ihre ausgehenden Kanten mit unterschiedlichen Termen beschriftet sind, nämlich mit Variablen zu unterschiedlichen Sorten (BD: Sorte Blutdruck, P: Sorte Puls, T:Sorte Temperatur, ...). Unser Ziel ist es nun, diese ähnlichen Teilnetze so einander anzugleichen, daß wir darüber die Operation $Fusion$ anwenden können. Wir benennen also die Stellen *Blutdruck gemessen*, *Puls gemessen*, *Temperatur gemessen* usw. um in *Vitalwert gemessen*. Dadurch verlieren wir keine Information, denn was genau gemessen wurde, sagt die jeweilige Transition im Vorbereich der neuen Stelle. Die Kantenbeschriftungen der jeweils in die Stelle *Vitalwert gemessen* eingehenden Kanten (z.B. *tupel(BD,PatId)* oder *tupel(Puls,PatId)*) können so spezifiziert werden, daß sie für alle derartigen Kanten derselben Sorte (z.B. $Vitalwert$) entsprechen. Dann ist die Kantenbeschriftung aller jeweils aus der Stelle *Vitalwert gemessen* herausführenden Kanten immer eine Variable V der Sorte $Vitalwert$. Die Transitionen *Puls in FK eintragen*, *Temperatur in FK eintragen* usw. modellieren jetzt alle den gleichen Vorgang *Vitalwert in FK eintragen.*

Aus diesen Überlegungen resultiert das mit B4 gleichbedeutende, aber einfacher zu strukturierende Netz B5:



B5: Einzelne Vitalwertemessungen angepaßt

Ausgehend von Netz B5 wenden wir die *Fusion* zunächst der Übersicht halber als Identifikation der mehrfach auftretenden Stellen *Patient auf Station*, *Pfleger* und *Stationsdokumente* an (Schnittstellennetz B6a) und erhalten das nunmehr zusammenhängende Netz B6:

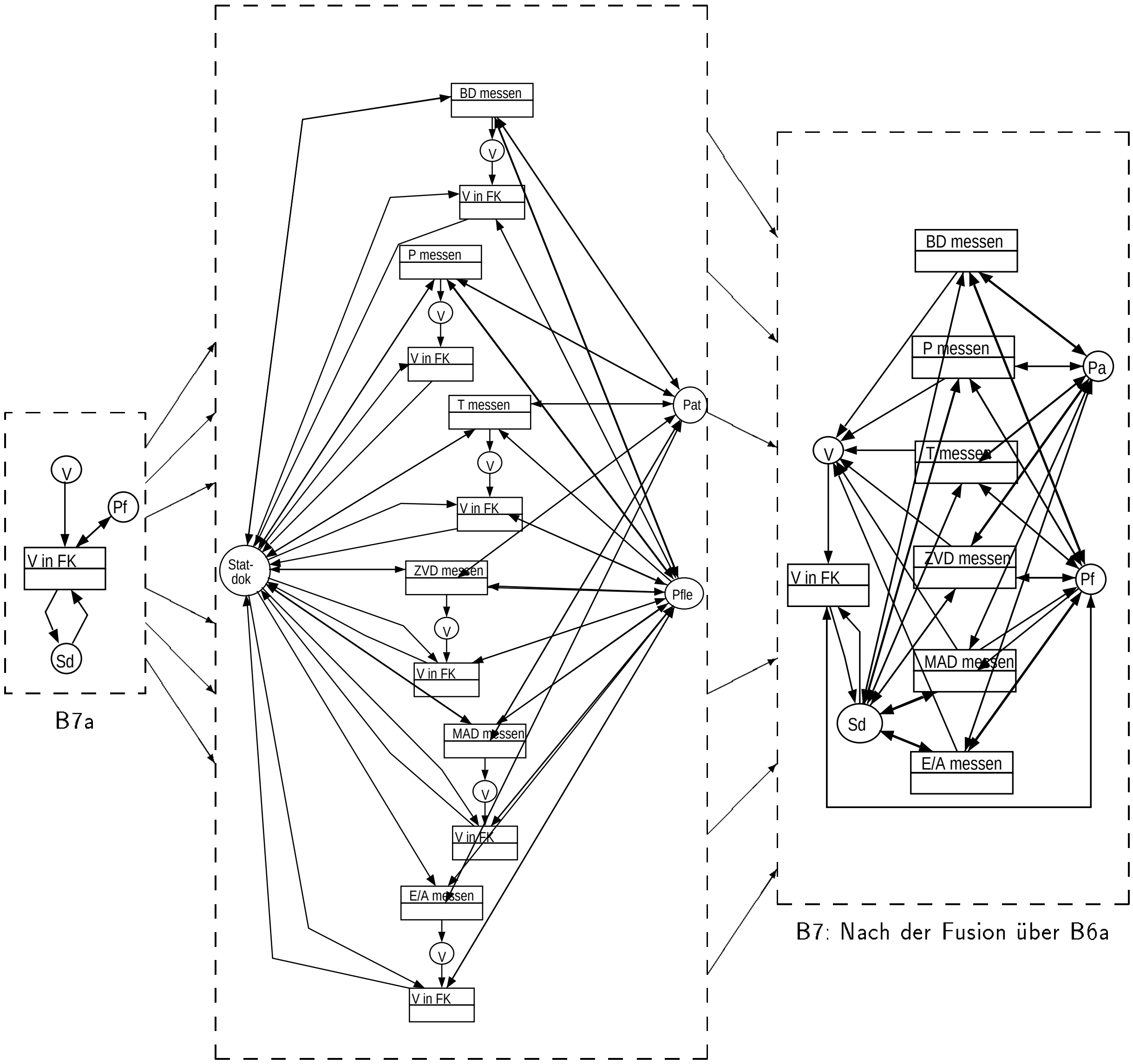


B6a

B5: Vor der Fusion über B6a

B6: Nach der Fusion über B6a

Wenden wir auf Netz B6 nun ein zweites Mal die Operation *Fusion* an, diesmal über dem Teilnetz B7a. Die Stellen *Pfleger* und *Stationsdokumente* werden hier zwar nicht „verklebt", da sie wegen der ersten *Fusion* sowieso nur noch einmal vorhanden sind, aber sie müssen im Schnittstellennetz angegeben werden, da die Transition nur eindeutig identifiziert werden kann, wenn ihr gesamter Vor- und Nachbereich bei einer Strukturierungsoperation angegeben wird. Wir erhalten das nun wesentlich weniger komplexe Netz B7:

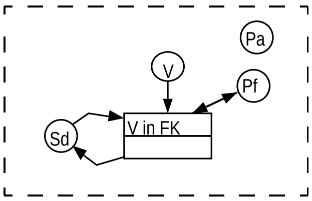


B7a

B7: Nach der Fusion über B6a

B6: Vor der Fusion über B7a

Beide *Fusion*-Operationen kann man auch in anderer Reihenfolge, oder aber in einem Schritt vollziehen, und zwar ausgehend von Netz B5, diesmal über dem Teilnetz B7b:



B7b

Das Ergebnis ist in beiden Fällen wieder B7, wie leicht nachzuvollziehen ist.

Union

Die Technik *Union* wird verwendet, um mehrere verschiedene Netze über einer gemeinsamen Schnittstelle zu komponieren. Sollte die Schnittstelle leer sein, erhalten wir als Union zweier Netze deren disjunkte Vereinigung (zwei getrennte Unternetze als Gesamtnetz).

Um die Technik *Union* anhand unseres Beispiels Vitalwertemessung demonstrieren zu können, stellen wir uns vor, wir hätten anstelle von B5 nicht ein Gesamtnetz von disjunkten Teilnetzen, sondern wir hätten eben diese Teilnetze B5a, ..., B5e als getrennte AHL-Netze. Diese können nun über dem gemeinsamen Schnittstellennetz B7b komponiert werden. Als Ergebnis erhalten wir wiederum B7, dasselbe Netz wie nach der (Gesamt-)*Fusion* der Teilnetze. Abb. 2.1 zeigt diese Komposition.

Formal ist dies die zusammenfassende Darstellung der sukzessiven Pushoutbildung folgender Pushouts in der Reihenfolge (1) bis (5):

$$
\begin{tikzcd}
\mathsf{B7b} \arrow[r] \arrow[d] \arrow[dr, phantom, "(1)"] & \mathsf{B5a} \arrow[d] & \mathsf{B7b} \arrow[r] \arrow[d] \arrow[dr, phantom, "(3)"] & \mathsf{B5d} \arrow[d] & \mathsf{B7b} \arrow[r] \arrow[d] \arrow[dr, phantom, "(5)"] & \mathsf{B5f} \arrow[d] \\
\mathsf{B5b} \arrow[r] & \mathsf{AB} \arrow[r] & \mathsf{ABC} \arrow[r] & \mathsf{ABCD} \arrow[r] & \mathsf{ABCDE} \arrow[r] & \mathsf{B7} \\
 & \mathsf{B7b} \arrow[r] \arrow[u] \arrow[ur, phantom, "(2)"] & \mathsf{B5c} \arrow[u] & \mathsf{B7b} \arrow[r] \arrow[u] \arrow[ur, phantom, "(4)"] & \mathsf{B5e} \arrow[u] &
\end{tikzcd}
$$

Allgemein besteht eine Beziehung zwischen *Union* und *Fusion* insofern, als daß wir äquivalente Netze erhalten, wenn wir entweder die *Union* der Netze $N1$ und $N2$ über dem Interface I bilden, oder aber zunächst die disjunkte Vereinigung $(N1 + N2)$ von $N1$ und $N2$ und dann die *Fusion* der mehrfach auftretenden Vorkommen von I in $N1+N2$. Beide Möglichkeiten haben wir in unserem Beispiel demonstriert.

Auf die kategorielle Basis horizontaler Strukturierungstechniken (*Fusion* als Koegalisator-Konstruktion und *Union* als Pushout) gehen wir ein in Kapitel 4 („Formale Definition von AHL-Netzen“).

Wie man in einem Netz verschiedene Aktionen zusammenfaßt, wie man Aktionen in Teilaktionen aufspaltet, oder wie man zusätzliche Systemeigenschaften dem Netz formal hinzufügt, das betrachten wir im folgenden Abschnitt.

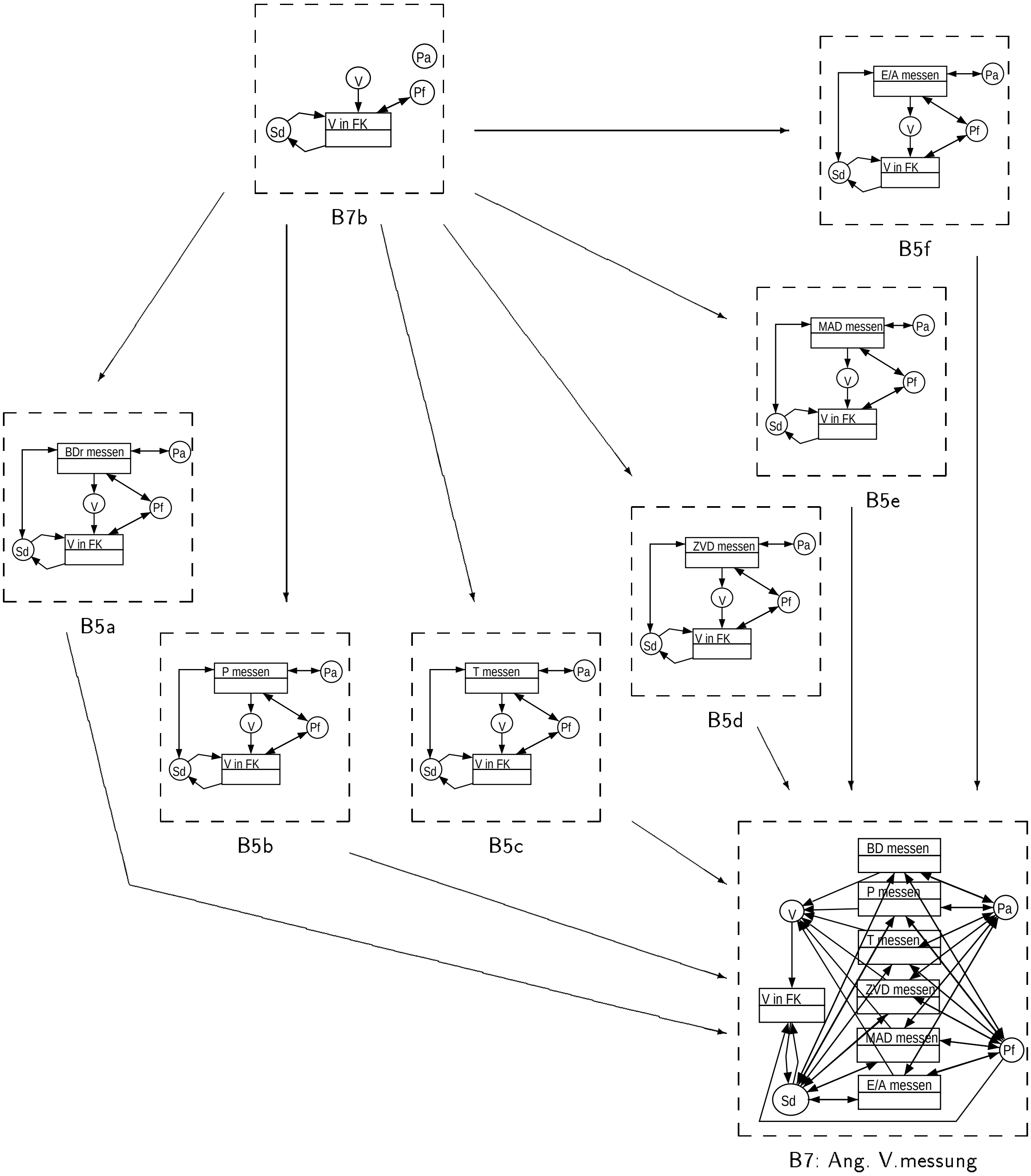


Abbildung 2.1: Komposition von Netzen über die Operation *Union*

Regelbasierte Abstraktion/Verfeinerung

Wir kommen nun zur vertikalen Strukturierung von AHL-Netzen, der Regelbasierten Abstraktion/Verfeinerung, auch Transformation genannt. Die Idee bei der regelbasierten Verfeinerung ist es, ein Teilnetz innerhalb des gesamten Netzes durch ein anderes Teilnetz zu ersetzen. Dies muß auf konsistente Weise geschehen, da ein Ersetzungsschritt das gesamte Netz verändert. Ausgehend von der Idee der Graphtransformation mittels Graphgrammatiken [Ehr79], formulieren wir die Ersetzungsschritte lokal, das heißt wir stellen Regeln auf, die die Ersetzung eines Teilnetzes durch ein anderes angeben, ohne dabei das ganze Netz zu berücksichtigen. Dieses Vorgehen hat den Vorteil einer modularen, überschaubaren Formulierung aller Ersetzungsschritte, auch wenn das Gesamtsystem sehr komplex ist. Außerdem erlaubt es die mehrmalige Anwendung der gleichen Ersetzungsschritte, ohne sie mehrmals formulieren zu müssen.

Eine Ersetzungsregel p (production) besteht nun aus einer linken Seite L, die das zu ersetzende Teilnetz enthält, und einer rechten Seite R, die das neue Teilnetz enthält, durch das ein Teil von L ersetzt werden soll. Diese Regel kann auf ein Netz N angewendet werden und liefert nach ihrer Anwendung das Netz M . Diese Anwendung einer Regel wird ausgedrückt durch $N \stackrel{p}{\Longrightarrow} M$ und heißt Ableitung oder Transformation. Die Beziehung zwischen linker und rechter Regelseite wird wie bei algebraischen Graphgrammatiken über Morphismen und Pushouts definiert. Die exakte Definition der Regeln und ihrer Anwendungsbedingungen, sowie ihrer kategoriellen Basis in High-Level-Replacement-Systemen erfolgt in Kapitel 4. An dieser Stelle wollen wir zunächst die formale Konstruktion und ihre Anwendung intuitiv verständlich darstellen.

Eine Regel kann in zwei Schritte unterteilt werden: Löschen und Hinzufügen von Teilnetzen. Deshalb gibt es als Zwischenschritt bei einer Regel auf dem Weg von L nach R noch das Schnittstellennetz K, das aus den Netzteilen besteht, die bei Anwendung der Regel erhalten bleiben, also sowohl in L als auch in R vorhanden sind. Eine Regel ist also gegeben durch $L \stackrel{l}{\longleftarrow} K \stackrel{r}{\longrightarrow} R$, wobei l und r AHL-Netz- Morphismen sind, also Abbildungen zwischen AHL-Netzen. Gelöscht werden diejenigen Teile aus L, die nicht im Bildbereich des Morphismus $l : K \to L$ liegen. Das Hinzufügen von Teilen funktioniert entsprechend: Es werden die Teile hinzugefügt, die zwar in R, aber nicht im Bildbereich des Morphismus $r : K \to R$ liegen. Dadurch, daß AHL-Netz-Morphismen verwendet werden, wird die syntaktische Korrektheit der durch Regelanwendung resultierenden Netze gewährleistet.

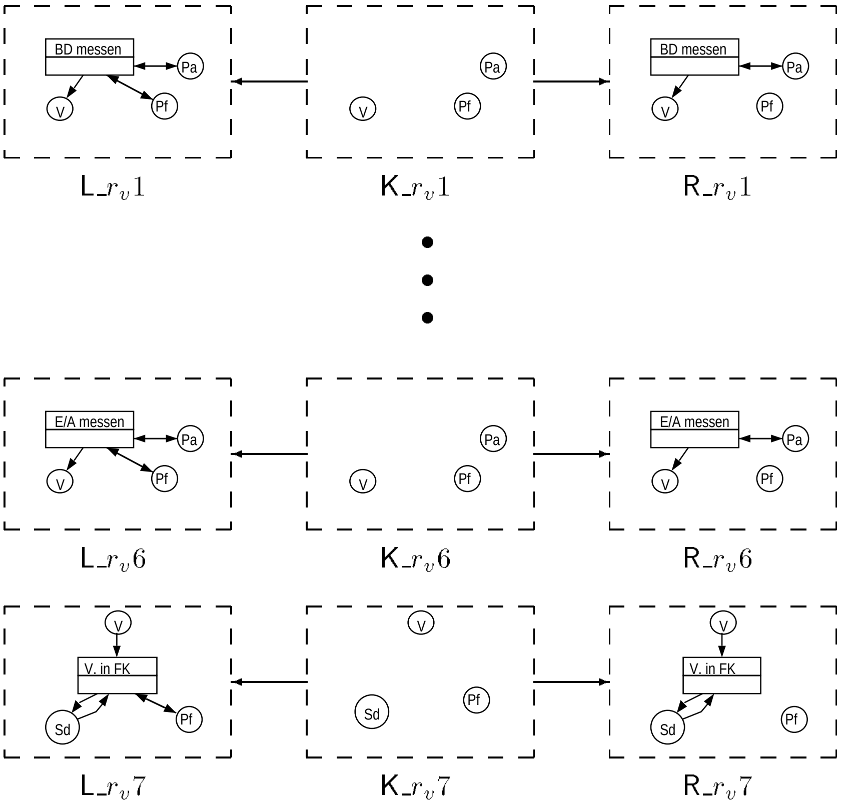
Betrachten wir nun die Anwendung einer Regel $L \stackrel{l}{\longleftarrow} K \stackrel{r}{\longrightarrow} R$ auf ein AHL-Netz N mit dem resultierenden AHL-Netz M. Die Transformation von N nach M wird erreicht über die Bildung der beiden Pushouts (1) und (2):

$$\begin{tikzcd}
L \arrow[d, "m1"'] \arrow[dr, phantom, "(1)"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "m2"'] \arrow[dr, phantom, "(2)"] & R \arrow[d, "m3"] \\
N & C \arrow[l, "c1"] \arrow[r, "c2"'] & M
\end{tikzcd}$$

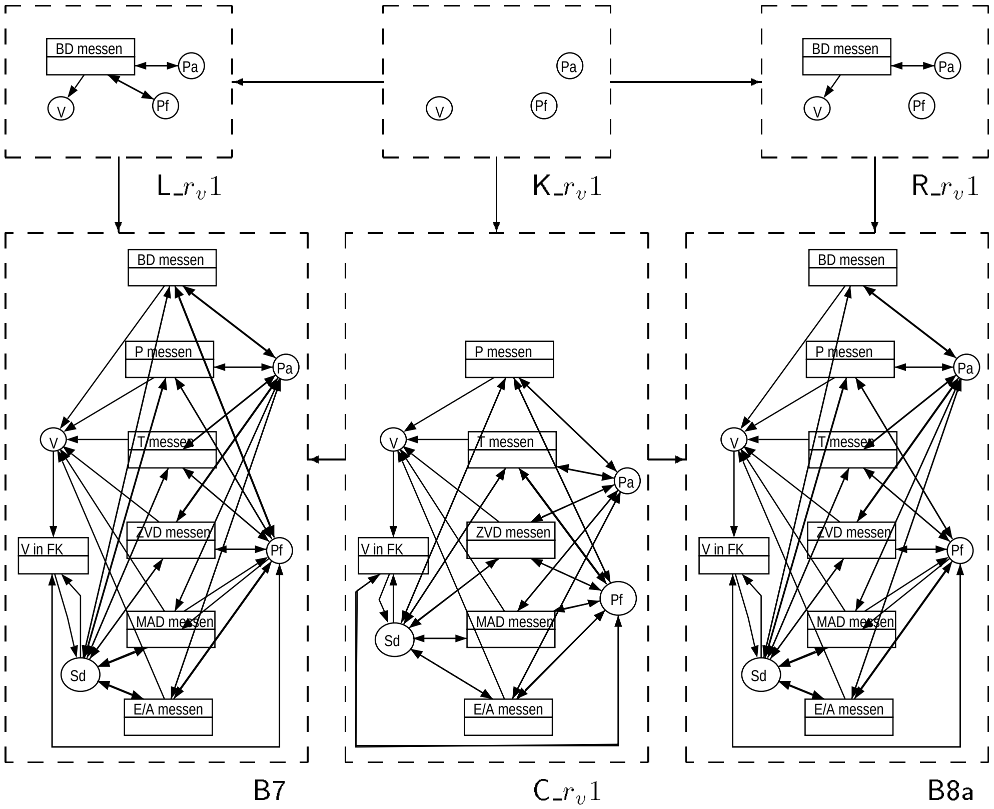
Der löschende Teil der Regel $l : K \to L$ wird auf N angewendet durch den Morphismus $c1 : C \to N$. Das Netz C (context net) entspricht also dem Netz N ohne den „ Unterschied“ zwischen den Teilnetzen L und K. Der hinzufügende Teil der Regel $r : K \to R$ wird entsprechend auf C angewendet durch den Morphismus $c2 : C \to R$.

Wir illustrieren die Anwendung von Transformationsregeln an unserem Beispielnetz:

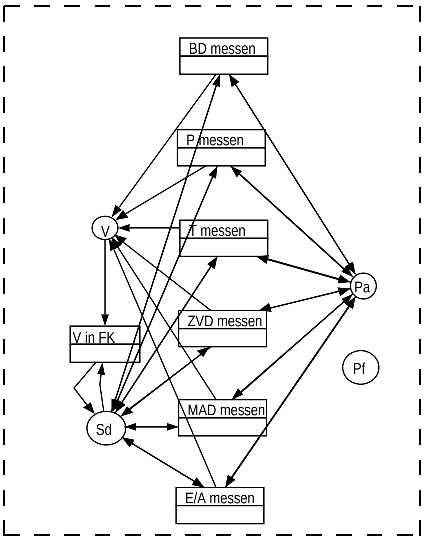
Wir wollen auf dem Weg zu einem datenbankunterstützen Modell der Vitalwertemessung zunächst von den beteiligten Pflegern abstrahieren, da diese im elektronischen System nicht mehr relevant sind. Wir löschen dazu mit den Regeln r_v1 bis r_v7 die Kanten, die mit $Pfleger$ beschriftet sind. Die Stelle $Pfleger$ soll erhalten bleiben, da sie eventuell noch für horizontale Strukturierung gebraucht werden:



Wir löschen dabei die Transitionen und erzeugen sie quasi neu, da sich ihr Vor- bzw. Nachbereich ändert. Damit ist es auch eine veränderte Transition. Wenden wir die Regeln in beliebiger Reihenfolge oder auch parallel auf unser Netz B7 an, so bilden wir Pushouts der Form



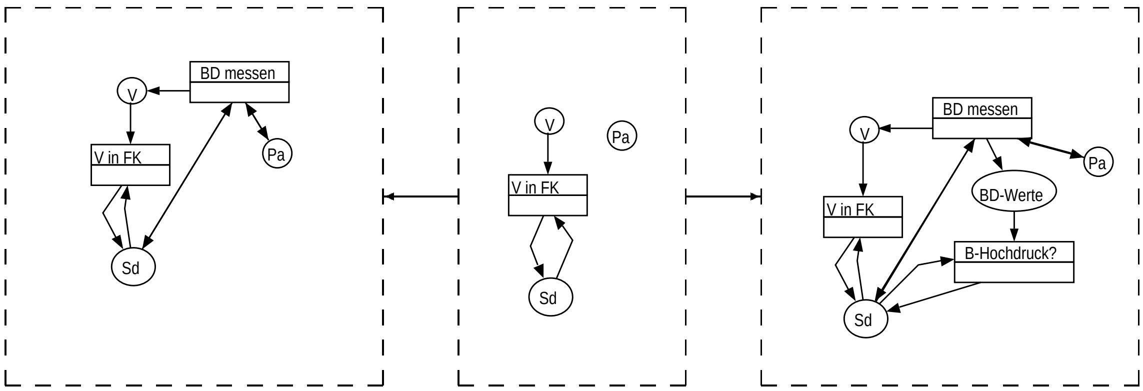
und erhalten nach einmaliger Anwendung aller Regeln in beliebiger Reihenfolge



B8: Abstraktion von Pfleger

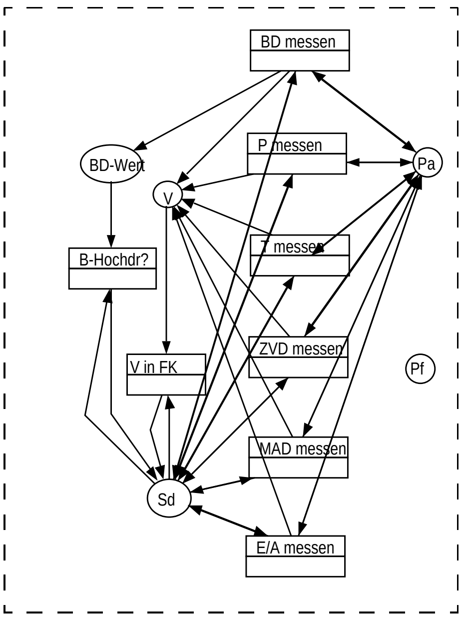
Als nächsten Schritt demonstrieren wir das Hinzufügen eines Teilnetzes über eine Regel. Wir wollen nach jeder Blutdruckmessung prüfen, ob es sich hierbei um zu hohen Blutdruck handelt. Dies kann als Wertebereichsprüfung spezifiziert werden. Im Fall des Bluthochdrucks wird der

Arzt durch einen deutlichen Warnvermerk in der Fieberkurve davon in Kenntnis gesetzt:



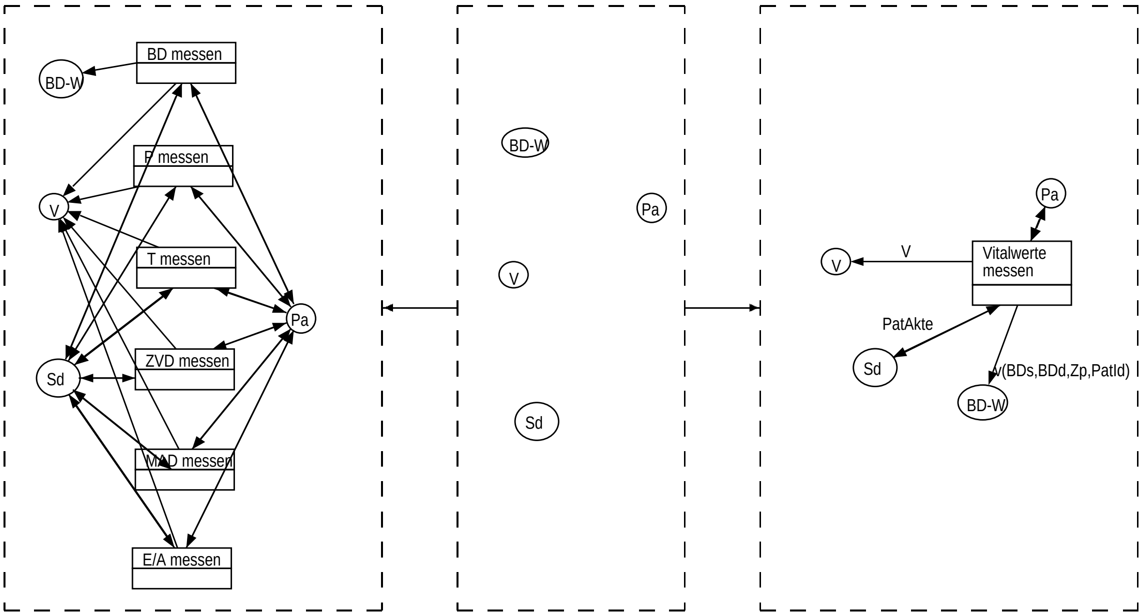
L_r_v8 K_r_v8 R_r_v8

In diesem Fall enthält das Kontextnetz mehr als nur eine Stellenmenge, da die Regel z.B. auch auf das noch nicht fusionierte Netz B5 angewendet werden kann. Dann ist es nötig, den Warnvermerk auch auf der „richtigen“ Stelle *Stationsdokumente* abzulegen, was durch die eindeutige Verbindung der Stelle *Vitalwert gemessen* mit der Stelle *Stationsdokumente* über die Transition im Kontextnetz gewährleistet wird.
In diesem Fall wird über die Regel auch die Spezifikation verändert: die neue Spezifikation nach Anwendung der Regel enthält zusätzliche Sorten und Operationen, die das Eintragen des Bluthochdruck-Warnvermerks in die Fieberkurve betreffen (siehe S.99).
Wenden wir Regel r_v8 auf Netz B8 an, erhalten wir Netz B9:



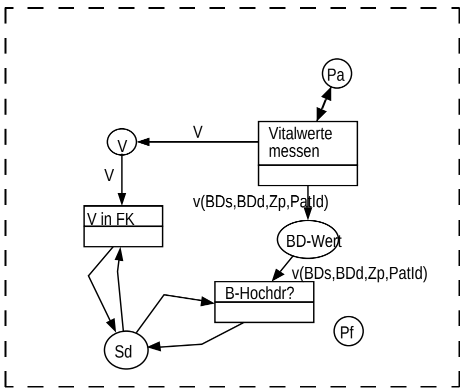
B9: Bluthochdruckwarnung

Wir wollen nun noch eine Regel betrachten, die zu den Regeln r_v1 bis r_v7 nicht parallel ausgeführt werden kann, da sie wesentliche Teile des Netzes löscht. Wir fassen alle bisherigen Meß-Transitionen in einer einzigen Transition *Angeordnete Vitalwertemessung* zusammen und abstrahieren somit von der Nebenläufigkeit der einzelnen Messungen:



L_$r_v 9$ K_$r_v 9$ R_$r_v 9$

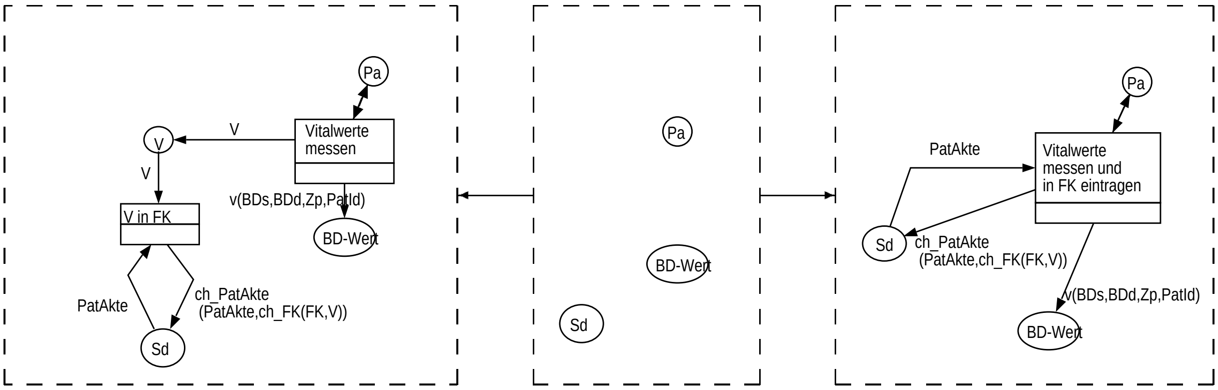
Die Regel, angewandt auf das Netz B9, ergibt B10:



B10: Vitalwertemessungen zusammengefaßt

Ebenso können wir erst $r_v 9$ auf B8, und dann $r_v 8$ auf das resultierende Netz, das B10 ohne der Bluthochdruckprüfung entspricht, anwenden. Als Ergebnis erhalten wir wieder B10.

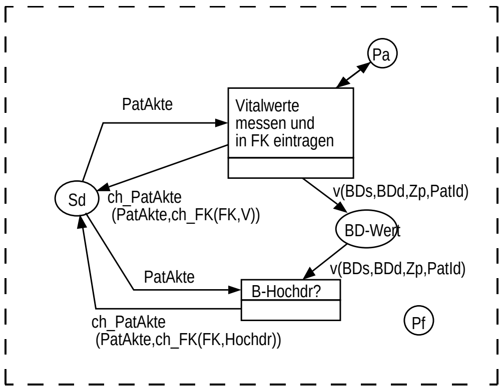
Als letzte Regel in diesem Beispiel betrachten wir die Zusammenfassung der Aktionen *Vitalwerte messen* und *Vitalwert in FK eintragen*:



L_r_v10 K_r_v10 R_r_v10

Die Regel r_v10 läßt sich nicht parallel zu einer bisherigen Regel anwenden, da sie auf die in der vorigen Regel neu erzeugte Transition *Vitalwerte messen* aufsetzt.

Als Ergebnis haben wir in Netz B11 eine Abstraktion von der tatsächlichen Vitalwertemessung B7:



B11: Vitalwertemessungen abstrakt

Dieses abstrahierte Netz enthält alle relevanten Daten des tatsächlichen Ablaufes der Behandlung des Patienten auf Station (IST-Essenz) und kann nun wiederum als Grundlage für regelbasierte Verfeinerungen dienen, die das Modell des angeforderten computergestützten Systems liefern (Funktionale Essenz).

Wir haben somit in diesem Kapitel einen kleinen Ausschnitt aus unserer Fallstudie, der Anforderungsanalyse von HDMS-A, kennengelernt. Die vollständigen Netze und Regeln finden sich in Kapitel 3.2.2. In der im folgenden Kapitel beschriebenen Gesamt-Fallstudie gehen wir mit den einzelnen Netzen analog zu unserem Beispiel vor, und vereinigen dann die Teilnetze über *Union*-Bildung zum Gesamtnetz DHZB-Aufenthalt-IST-System (Kapitel 3.2) bzw. nach Anwendung regelbasierter Abstraktion/Verfeinerung zum Netz DHZB-Aufenthalt-IST-Essenz (Kapitel 3.3) und erhalten nach weiteren regelbasierten Verfeinerungsschritten das Netz DHZB-Aufenthalt-Funktionale_Essenz(Kapitel 3.4).

Kapitel 3

Die Fallstudie HDMS-AHL

3.1 Vorstellung des Gesamtprojekts

Grundlage unseres Fallbeispiels ist die im Projekt *Korrekte Software* (KORSO, 1991-1994) entwickelte Fallstudie HDMS-A (*Heterogeneous Distributed Management System - Abstract Version*), die auf dem konkreten, seit 1989 in der Entwicklung befindlichen HDMS-System der *Projektgruppe Medizin/Informatik* (PMI) am *Deutschen Herzzentrum Berlin* (DHZB) basiert. Ziel des Projekts HDMS ist die rechnergestützte Integration und Kommunikation einer Vielzahl medizinischer und verwaltungstechnischer Daten, die beim Aufenthalt von Patienten im DHZB anfallen (siehe auch Kapitel 1.2: *Die Fallstudie: Ein medizinisches Informationssystem*).

Allerdings war es innerhalb von KORSO weder sinnvoll noch möglich, die Arbeit der PMI in vollem Umfang „nachzuentwickeln". Insbesondere eine formale Spezifikation der konkreten Hardware- und Softwareschnittstellen, auf denen HDMS basiert, also der sogenannten Systemrandbedingungen, hätten den Rahmen von KORSO gesprengt, denn die Heterogenität der Daten, die Verteiltheit der Systemkomponenten und die Heterogenität der schon vorhandenen beteiligten Hard- und Netware, sowie der zu integrierenden Softwaresysteme machen eine konkrete HDMS-Gesamtlösung äußerst komplex.

HDMS-A: Die Spezifikation von HDMS im KORSO-Projekt

Innerhalb von KORSO wurden die Systemrandbedingungen daher in abstrakter Form spezifiziert, d.h. die Systemkomponenten Datenbanken, Betriebssystem, Kommunikationsmodell, Programmiersprachenumgebung und Benutzerschnittstelle wurden unabhängig von konkreten Sprachen und Semantiken beschrieben. Durch diese Simulation wurde HDMS zu HDMS-A (abstraktes HDMS). Einerseits gestattet eine solche Abstraktion die Wiederverwendung des zu entwickelnden Systems unter veränderten Umweltbedingungen, z.B. mit einer anderen als der konkreten, heute im DHZB anzutreffenden Hardwarezusammenstellung; der Systemkern, die elektronische Patientenakte, kann konzeptionell klar entworfen und entwickelt werden, ohne auf Details und Veränderungen der wirklichen Konfiguration Rücksicht nehmen zu müssen. Andererseits wird sichergestellt, daß die Fallstudie die wesentlichen Rahmenbedingungen von HDMS, nämlich die Entwicklung einer elektronischen Patientenakte im Hinblick auf ein heterogenes verteiltes System, nachvollzieht [LCFW92].

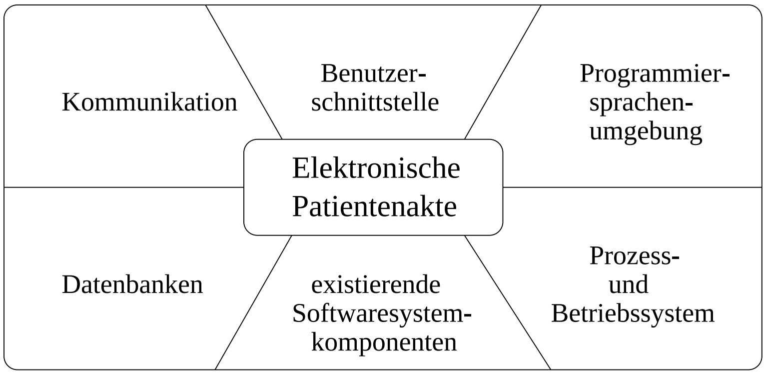


Abbildung 3.1: Komponenten der HDMS-A-Systemkonfiguration

Abbildung 3.1 zeigt die Einbettung des Kerns des HDMS-A-Systems in die (abstrakt spezifizierte) Systemkonfiguration des neu zu entwickelnden Produkts:

- *Der Kern des HDMS-Systems — Die Elektronische Patientenakte*:

  Der Kern der Fallstudie ist die Realisierung eines rechnergestützten Systems anstelle der bisher in Papierform vorliegenden Patientenakte, also alle für den medizinischen Bereich wichtigen Daten zu einer Person allen an dem System angeschlossenen Ärzten in einheitlicher Form verfügbar und bearbeitbar zu machen. Wir werden im nächsten Abschnitt auf die Modellierung des Systemkerns mit Hilfe von AHL-Netzen eingehen, die Gegenstand dieser Arbeit ist.

- *Die Rahmenbedingungen der HDMS-Systementwicklung*:

  Wir beschreiben hier die Systemeinbettung der elektronischen Patientenakte nur kurz, da die Systemrahmen-Komponenten in dieser Arbeit nicht modelliert werden, und geben dafür die relevanten Literaturstellen aus dem KORSO-Projekt an.

  - *Die Datenbanken*:

    Es werden ein abstraktes hierarchisches Datenbankmodell und ein abstraktes relationales Datenbankmodell spezifiziert [Het93]. Die spezifizierten Datenbanken sind parametrisch, und es wird ein Mechanismus angegeben, wie eine Instantiierung des Systems mit einen konkreten Datenschema aussieht und wie das Ergebnis dieser Instantiierung semantisch zu interpretieren ist [Con93a], [Con93b], [Huß93]. Sicherheitsanforderungen, die die Konsistenz und Korrektheit der Daten [Fuc94], sowie die Regelungen der Zugriffsrechte betreffen, müssen beachtet werden [GH93], [Ste93], [Ren94].

  - *Die Benutzerschnittstelle*:

    Es werden Window-, Menu- und Mouse-Konzepte modelliert, unabhängig vom konkreten graphischen Erscheinungsbild der Schnittstelle [MZ94]. Die abstrakte Beschreibung enthält auch den Teil der Schnittstelle, der die Korrektheit bzw. Plausibilität der Eingaben garantiert. Ein konkreter Entwurf einer Benutzerschnittstelle für die Herzkatheteruntersuchung, der auf [MZ94] und den konkreten Daten (Entities) in [SNM$^+$93] basiert, findet man in [KBS93], [Shi94].

– *Das Prozeß- und Kommunikationsmodell*:

  Es wird ein Mechanismus spezifiziert, der es erlaubt, Verbindungen zwischen aktiven Komponenten (Programmen) herzustellen, über diese Verbindungen geregelt Botschaften auszutauschen und sie wieder abzubauen. Ein Format für den Botschaftenaustausch wird festgelegt, in das die aktiven Komponenten ihre Botschaften abzubilden haben, damit die Kommunikation systemweit einheitlich funktioniert [Dam93].

– *Die Zielsprachen*:

  Als programmiersprachliche Umgebung werden in Anlehnung an die im Herzzentrum verwendeten konkreten Implementierungssprachen ML, PROLOG und C drei abstrakte Sprachkerne (funktional, logisch und imperativ) spezifiziert.

Kommen wir nun zum Kern der Fallstudie, der elektronischen Patientenakte. Um sich die Anforderungen an das System klarzumachen, ist es hilfreich, die bisherige Realisierung der Patientenakte in Papierform näher zu betrachten. Die Akte besteht aus einer Ansammlung einzelner Akteneinträge, meist einzelner Blätter. Die Anordnung der Blätter wird von der chronologischen Abfolge des Entstehens der Einträge bestimmt. Dadurch, daß einzelne Blätter leicht aus der Akte herausgenommen und wieder zurückgelegt werden können, ist es aber auch möglich, chronologisch weit auseinanderliegende Einträge in einen näheren Kontext zu bringen. Ein Kontext könnte z.B. die Menge aller Röntgenbilder in der Akte sein. Solche Kontexte sind Voraussetzung für diagnostische Leistungen der behandelnden Ärzte.

Der Weg von den in einer Akte vorhandenen Patientendaten zu einer Diagnose, die dann natürlich selbst einen Eintrag in der Akte ergibt, sieht daher typischerweise so aus: Grundlage der Diagnose sind eine Menge von primären Analysedaten, die allerdings in der Regel nicht in der Urform in die Diagnose einfließen, sondern in interpretierter Form. Die erste Stufe dieser Interpretation sind interpretierende Analysen, z.B. Falschfarbendarstellungen von Ultraschallaufnahmen oder Kontrastanreicherungen für Röntgenbilder. Die letzte Stufe solcher Interpretationen sind Befundberichte, die von hinzugezogenen Ärzten wiederum auf der Grundlage primärer und bereits vorinterpretierter Daten verfertigt werden. Mitunter fließen auch bereits vorhandene diagnostische Leistungen in die Diagnosestellung ein. Die Patientenakte muß natürlich diese vielfältigen Querbezüge dokumentieren und, soweit möglich, organisatorisch unterstützen; in Papierform geht das nur durch eine hierarchische Unterstruktur, wie wir sie von fast allen Druckerzeugnissen kennen.

Diese kurze Beschreibung eines Teils des IST-Zustands der Patientenakte macht deutlich, welche Erwartungen an eine rechnergestützte Patientenakte gestellt werden:

1. Gleichzeitige Verfügbarmachung von Daten für mehr als eine Person und somit Vermeidung der Belastungen des Patienten durch Mehrfachuntersuchungen.

2. Verbesserter Umgang durch einfacherern Zugriff auf vielfältige Unterstrukturen.

Die erste Erwartung an eine Rechnerunterstützung resultiert aus der Beschränkung, daß sich eine vollständige Papierakte immer nur an einem Ort aufhalten kann und somit nur einem Arzt zu einer Zeit zur Verfügung steht. Sollen Informationen aus einer solchen Akte weiteren Ärzten zur Verfügung gestellt werden, so führt das entweder zur Unvollständigkeit der Akte oder zu einer (teilweisen) Kopie. Beides ist nicht wünschenswert. Insbesondere bei Kopien stellt sich das Problem, alle Kopien auf dem neuesten Stand zu halten. Aber auch schon die Vollständigkeit einer solchen Akte ist ein Problem. In der Regel wird ein Patient nicht nur von einem Arzt behandelt. Also ist im allgemeinen die Information zu seiner Krankengeschichte auf mehrere Aktenschränke verteilt. Ist nun die Vollständigkeit seiner Akte z.B. bei einem Krankenhausaufenthalt nötig, so müßten viele Schriftstücke verschickt werden. Daß das unpraktikabel ist, wird dadurch deutlich, daß sich Krankenhäuser in der Regel alle nötigen Daten dadurch beschaffen, daß sie die zugehörigen Untersuchungen und Analysen noch einmal selber durchführen. Es ist klar, daß die damit verbundenen Kosten und Belastungen der Patienten vermieden werden könnten, wenn medizinische Daten leichter und schneller kommunizierbar wären.

Die zweite Erwartung könnte unter dem Stichwort „Die Patientenakte als multimedialer Hypertext" zusammengefaßt werden. Sie betrifft Leistungen der Bilddatenverarbeitung, Multimedialität und Benutzerschnittstellengestaltung und wird in der Entwicklung der funktionalen Essenz des Systems in dieser Arbeit noch nicht berücksichtigt.

Anforderungsanalyse mit formalen Methoden

Wir beschreiben an dieser Stelle unser Vorgehen zur formalen Anforderungsanalyse (Requirements Engineering) mit AHL-Netzen im Unterschied zum Vorgehen im KORSO-Modell.

Allgemein versteht man unter der Anforderungsanalyse die „systematische Erfassung von Anforderungen, mit dem Ziel einer Anforderungsdefinition, die einerseits die Wünsche der Systemnutzer richtig und vollständig wiedergibt und andererseits eine ausreichende Grundlage für die spätere Konstruktion des Systems liefert“. Das Ziel, die Anforderungsdefinition, wiederum ist „ein Dokument, das eine vollständige Beschreibung dessen enthält, was ein Softwaresystem tun soll, aber nicht erklärt, wie es das tun soll“ [Huß93].

Anforderungen an ein System lassen sich klassifizieren in

- Funktionale Anforderungen, also Anforderungen, die das Ein-/Ausgabeverhalten des Systems beschreiben, und zwar
  - statische Aspekte (Datenstrukturen)
  - dynamische Aspekte (Ablaufreihenfolge von Aktionen)
  - Wertverlaufs-Aspekte
  - funktionale Sicherheits-Anforderungen (Spezialfall aller Aspekte)
- Nicht-Funktionale Anforderungen, und zwar
  - Verteiltheitsanforderungen
  - technologieabhängige Sicherheits-Anforderungen
  - Leistungsanforderungen (Performance)
  - Aspekte des Entwicklungsprozesses

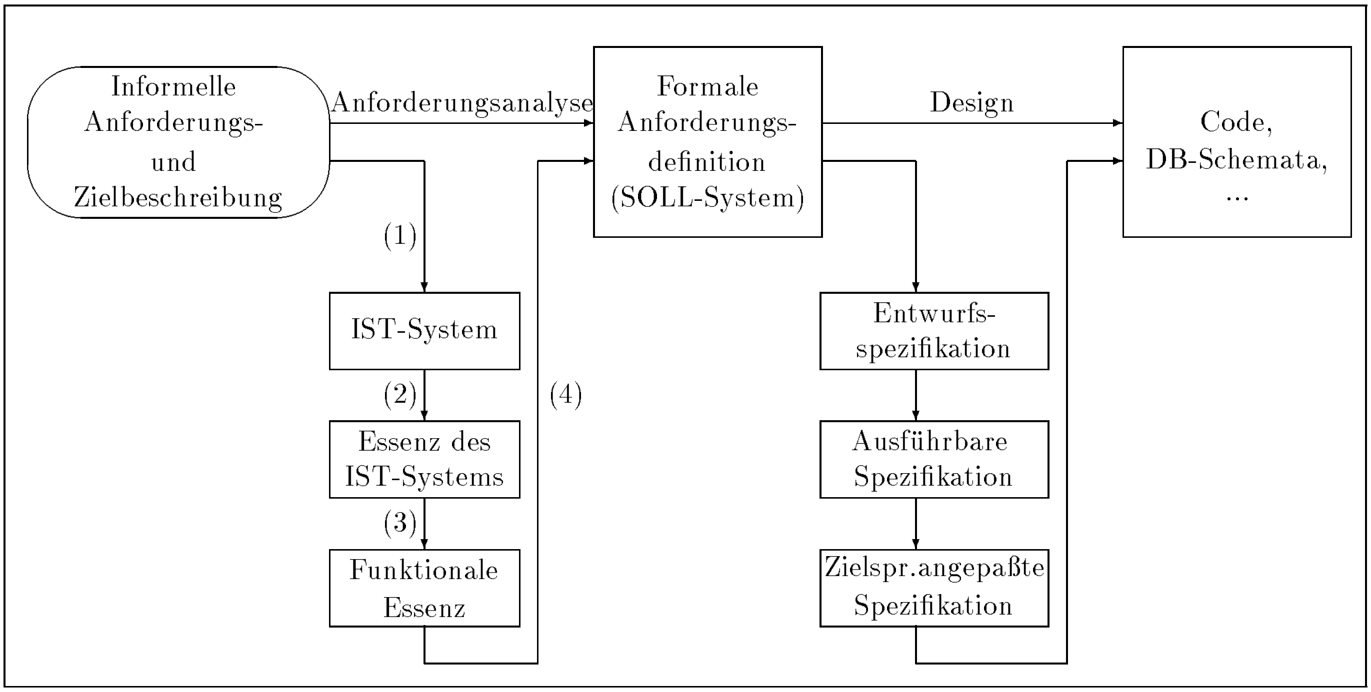


Abbildung 3.2: Formale Anforderungsanalyse und Software Engineering

In dieser Arbeit befassen wir uns ausschließlich mit funktionalen Anforderungen eines typischen datenbank-gestützten, nicht realzeit-orientierten Systems.

Abbildung 3.2 gibt einen Überblick über die Rolle der Anforderungsanalyse innerhalb der klassischen Phaseneinteilung des Software-Engineerings, angepaßt an die Bedürfnisse der Entwicklung korrekter Software:

Schritt (1) der Anforderungsanalyse , die IST-Analyse, erfaßt die fachlichen Abläufe und die statische Datenstruktur und liefert die Modellierung des IST-Systems.
In Schritt (2), bei der Ableitung der IST-Essenz, werden Technologieabhängigkeiten und zufällige Gegebenheiten eliminiert, und die Beschreibung wird auf ihren wesentlichen Kern reduziert.
Schritt (3) ist die Entwicklung der funktionalen Essenz des Systems, die im wesentlichen die IST-Essenz umfaßt. Abweichungen sind Vereinheitlichung von Datenstrukturen und Ergänzungen um zusätzliche Leistungen des zu entwickelnden Systems.
In Schritt (4) schließlich werden die technologische Basis und Systemrahmenkomponenten mit einbezogen. Typische Beispiele sind:

- die Einbeziehung vorhandener Systeme über eine formal beschriebene Schnittstelle,
- die Einhaltung einer bestimmten Verteilungsstruktur der Datenhaltung (aus organisatorischen oder rechtlichen Gründen),
- die Berücksichtigung von Benutzerrollen und Zugriffsrechten,

- die Verwendung einer Kommunikations-Architektur mit Konsequenzen für die Bündelung und Entkoppelung von Transaktionen.

Die Vorgehensweise in den hier als *Design* bezeichneten späteren Phasen der Entwicklung ist genauer beschrieben im KORSO-Papier [PW94]; sie ist nicht Gegenstand der vorliegenden Arbeit.

In dieser Arbeit befassen wir uns anhand unserer Fallstudie mit den ersten drei Schritten der Anforderungsanalyse, der IST-Analyse, der Ableitung der IST-Essenz und der Entwicklung der funktionalen Essenz. Üblicherweise arbeitet man in diesen Phasen häufig mit informellen Hilfsmitteln; erst das Endprodukt, die formale Anforderungsdefinition, markiert den vollständigen Übergang in die Welt der formalen Programmentwicklung. Dagegen arbeitet man im Design korrekter Software fast ausschließlich mit formalen Hilfsmitteln. Wir wollen zeigen, daß es durchaus Sinn macht, formale Methoden bereits in den frühen Phasen der Anforderungsermittlung einzusetzen.

Betrachten wir zunächst das Vorgehen im KORSO-Projekt, wo mit informellen und semiformalen Methoden gearbeitet wurde:

Im KORSO-Projekt wurden zunächst der IST-Zustand der wesentlichen Dokumente der existierenden Papier-Patientenakte (statische Sicht), sowie die typischen Arbeitsabläufe, wie sie auf der Grundlage von Papierdokumenten ablaufen (dynamische Sicht), modelliert [CKL93b]. Die bei der IST-Analyse auftretende Schwierigkeit bestand darin, die statische und die dynamische Sicht des IST-Systems einheitlich zu modellieren. Die Spezifikationen der Dokumente, d.h. ihrer statischen, logischen Datenstruktur erfolgte in der algebraischen Spezifikationssprache SPECTRUM [BFG+93]. Die Arbeitsabläufe dagegen wurden als einfache Bedingungs/Ereignis-Netze beschrieben, die nur grob die Sequentialisierung und Parallelität der Abläufe wiedergeben. Auch die Veränderung einzelner Daten bei einem Arbeitsablauf (z.B. ein Eintrag im Dokument Fieberkurve) läßt sich in diesem Modell nicht darstellen. Zudem beschränkt sich die Sicht auf jeweils einen vorhandenen Patienten im System und läßt die Parallelität der Krankenhausaufenthalte mehrerer Patienten völlig außer acht.

Diese Lösung ist aus Gründen der Konsistenz und Korrektheit nicht befriedigend und wird daher in dieser Arbeit durch die Integration der statischen und dynamischen Sicht verbessert.

Auch der Übergang vom IST-System zur funktionalen Essenz sollte nach Möglichkeit innerhalb eines Formalismus erfolgen.

Innerhalb von KORSO wird dieser Übergang jedoch durch verschiedene informelle Übergänge beschrieben. Für die Beschreibung der statischen Daten und ihrer Organisation wurden Entity-Relationship-Diagramme verwendet, Datenflußdiagramme enthalten die essentiellen Arbeitsabläufe und Transaktionen. Um zu einer formalen Anforderungsspezifikation zu gelangen, wurden nun die semiformalen E/R- und Datenflußdiagramme in formale SPECTRUM- Spezifikationen übersetzt. Der Nachteil einer derartigen algebraischen Spezifikation liegt auf der Hand: Erstens können Modifikationen der Spezifikation nicht in eine benutzerfreundlichere Form, z.B. in E/R- und Datenflußdiagramme, zurückübersetzt werden. Zweitens kann das spezifizierte System schlecht bzw. gar nicht simuliert und prototypisch getestet werden. Eine umfassende Übersicht über die im KORSO-Projekt verwendeten Spezifikationsmethoden findet sich in [CHL95].

In dieser Arbeit soll dieser Übergang vom IST-System zur funktionalen Essenz innerhalb des AHL-Netz-Formalismus erfolgen und so einerseits die Vorteile eines einheitlichen Entwicklungsformalismus aufzeigen; andererseits soll die Fallstudie dazu dienen, wesentliche Merkmale von AHL-Netzen herauszustellen und Anregungen zu einer praxisorientierten Weiterentwicklung des AHL-Netz-Formalismus geben.

Die Entwicklung der funktionalen Essenz mit AHL-Netzen

Im folgenden wird die IST-Analyse und daraus die Entwicklung der funktionalen Essenz der vorliegenden KORSO-Fallstudie HDMS-A mit Hilfe von AHL-Netzen neu konzipiert. Der Zusammenhang der involvierten Dokumente und Arbeitsabläufe wird zunächst im IST-Zustand mit Hilfe von AHL-Netzen spezifiziert, um die statische und die dynamische Sicht des Systems in einen einheitlichen Formalismus zu integrieren und die Veränderung von Daten und Werten bei einem Arbeitsablauf modellieren zu können. Untersuchungen zum Schaltverhalten und die Anwendung horizontaler Strukturierungstechniken sind wegen der Komplexität und des Umfangs des IST-Systems bereits in dieser Spezifikationsphase notwendig.

Anschließende regelbasierte Abstraktionsschritte modellieren dann den Weg vom IST-System zur funktionalen Essenz, bilden also vor dem formalen Hintergrund der Instantiierung von High-Level-Replacement-Systemen mit Petrinetzen einen konsistenten und mit Strukturierungstechniken kompatiblen Übergang, ohne daß der Formalismus gewechselt werden muß. Lokale Abstraktionen sollen dabei von für das System irrelevanten Arbeitsabläufen (wie z.B. Blutprobe schütteln) und -umgebungen (wie z.B. verwendete Laborgeräte) abstrahieren und nur die Merkmale und Daten herausarbeiten, die für das zu spezifizierende System wesentlich sind.

Solche, für Verfeinerung bzw. Abstraktion verwendeten regelbasierten Ansätze eignen sich besonders gut für größere Systeme, da Seiteneffekte, also Auswirkungen von Änderungen bestimmter Spezifikationsteile in Form von nicht beabsichtigen Änderungen anderer Spezifikationsteile, vermieden werden: Systemteile, die durch Regeln veränderte Systemteile benutzen, müssen selbst auch durch diese Regeln entsprechend verändert werden. Lokal definierte Änderungen ergeben also konsistent globale Änderungen.

Beschreibung der ausgewählten Dokumente und Arbeitsabläufe

Die IST-Analyse greift in Anlehnung an das KORSO-Projekt aus der großen Menge real existierender Dokumente eine Teilmenge heraus, die sowohl in sich abgeschlossen ist, als auch einen kompletten Krankenhausaufenthalt modelliert.
Dabei handelt es sich um folgende Dokumente:

- das *Stammdatendokument* enthält alle patientenbezogenen Daten, die bei der Aufnahme ins Krankenhaus festgehalten werden, und die sich selten oder gar nicht ändern wie Name, Geburtsdatum und -ort, Krankenkasse etc.
- die *Fieberkurve*, die Werte der täglich am Patienten durchgeführten Untersuchungen wie Blutdruck, Fieber, Puls, Medikation usw. enthält,
- den *Notfallabor-Anforderungs- und Befundbogen (NAB)* für Werte, die der Laborbereich aus verschiedenen Analysen von Blut und Harn ermittelt, und

- den *Herzkatheterbefund*, der alle im Rahmen einer Herzkatheter-Untersuchung anfallenden Daten bis hin zum ärztlichen Befund enthält, also Röntgenaufnahmen, Druckkurven, Laboruntersuchungswerte, Blutströmungsdaten und den Befundtext.

Die ausgewählten Dokumente decken ein weites Spektrum möglicher Funktionalitäten ab. Es ergeben sich für die Modellierung im Zusammenhang mit den beschriebenen Dokumenten folgende Arbeitsabläufe:

- Die *Aufnahme*, bei der sich der ankommende Patient ausweist und Einweisungs- und Kostenübernahmeschein übergibt. Auf dieser Basis wird ein Krankenhaus-Behandlungsvertrag geschlossen. Im Dialog werden die Stammdaten des Patienten erfragt, die sich auf allen Dokumentvordrucken und Etikettensätzen wiederfinden, die an die Kostenstelle und die erste Station geschickt werden, sowie auf denen, die in der Anmeldung verbleiben (Aufnahmekarte).

- Die *Behandlung*, bei der auf der Station täglich sogenannte Vitalwerte wie Puls, Blutdruck und Temperatur gemessen werden; dazu können je nach Situation des Patienten weitere Werte kommen: auf der Intensivstation werden z.B. bisweilen stündlich der mittlere arterielle Druck sowie der zentralvenöse Druck gemessen. Diese Werte, sowie Art und Menge der verabreichten Medikamente bzw. Infusionen werden auf der Fieberkurve vermerkt.

- Den *Labor-Ablauf*, wo verschiedene Analysen von Blut und Harn durchgeführt werden, deren Ergebnisse die MTA in den NAB einträgt. Die Dokumente NAB und Fieberkurve bedingen sich gegenseitig: Einerseits werden die relevanten Laborwerte aus dem NAB zur weiteren Behandlung in die Fieberkurve übertragen; andererseits braucht die MTA zur Analyse Angaben über die bisherige Medikation aus der Fieberkurve.

- Die *Herzkatheter-Untersuchung*, bei der ein Herzkatheter (ein Schlauch, eine Leitung) in eine im Oberarm oder Oberschenkel befindliche Arterie eingeführt und von dort aus in die Herzhöhlen vorgeschoben wird. Bei dieser Untersuchung, die aus verschiedenen Röntgenuntersuchungen und Druckmessungen besteht, können wichtige Rückschlüsse auf Erkrankungen im Bereich des Herzens und der großen Gefäße gezogen werden. Auch das Erweitern verengter Gefäße mit Hilfe des Katheters ist möglich. Alle anfallenden Daten der Untersuchung werden in einem Protokoll festgehalten, das zusammen mit den Röntgenfilmen die Grundlage des vom Arzt erstellten Herzkatheterbefundes bildet.

- Die *Entlassung*, die der Anmeldung von der den Patienten zuletzt behandelnden Station über eine Abgangsmeldung mitgeteilt wird. Die Abgangsmeldung enthält Informationen über Entlassungszeitpunkt und -ziel, sowie die Entlassungsdiagnose, die wiederum auf der Aufnahmekarte vermerkt werden.

Eine genauere Beschreibung der modellierten Dokumente und Arbeitsabläufe findet sich bei den entsprechenden Netzen und Spezifikationen im folgenden Abschnitt *IST-Analyse mit AHL-Netzen*.

3.2 IST-Analyse mit AHL-Netzen

In diesem Kapitel beschreiben wir den IST-Zustand des HDMS-A-Systems, also die Daten und Arbeitsabläufe, die vor Einführung eines Computersystems existieren. Wir spezifizieren das IST-System mit AHL-Netzen, die wir den Arbeitsabläufen Aufnahme, Behandlung, Labor-Ablauf, Herzkatheterisierung und Entlassung zuordnen.

Wir beschreiben, nach Arbeitsabläufen gegliedert, die horizontale Strukturierung der einzelnen Netze, wo über die Operationen *Union* (Zusammenfassung von zwei Netzen zu einem) und *Fusion* (Identifizierung von Teilnetzen innerhalb eines Netzes, siehe auch Kap. 4.3) für jeden Bereich genau ein Netz konstruiert wird. Diese fünf Teilnetze werden dann anschließend über *Union* zum Gesamtnetz DHZB-Aufenthalt-IST-System vereinigt. Wir stellen allerdings nur die einzelnen Teilnetze dar, da das Gesamtnetz sehr groß wird. Für jeden Bereich wird die dazugehörige algbraische Spezifikation angegeben. Da sich die verwendeten Datentypen in den für einen Bereich angegebenen Netzen stark überschneiden, gibt es nicht zu jedem Netz eine eigene Spezifikation. Vielmehr beziehen sich alle Netze eines Bereiches auf dieselbe Spezifikation, wobei auch Abhängigkeiten von Teilspezifikationen aus anderen Bereichen vorhanden sind (die Aufnahmekarte, zum Beispiel, wird sowohl bei der Aufnahme als auch bei der Entlassung verwendet; der Datentyp *PatId* zur Patientenidentifizierung wird in jedem Bereich gebraucht).

In Abb. 3.3 stellen wir die Konstruktion des Gesamtnetzes DHZB-AUFENTHALT durch die wiederholte Anwendung der Operationen *Union* und *Fusion* dar. Auf die jeweiligen Schnittstellennetze gehen wir ein, wenn wir die einzelnen Netze eines Bereiches beschreiben.

Abb. 3.4 stellt in einem Benutzt-Hierarchie-Diagramm die Abhängigkeiten der einzelnen Teilspezifikationen dar. Diese werden anschließend zu einer Gesamtspezifikation des Netzes DHZB-AUFENTHALT-Spec vereinigt, wobei die mehrfach auftretenden Teilspezifikationen dann nur noch einmal vorhanden sind. Die Spezifikationen sind mit den üblichen Strukturierungsmethoden *Actualization, Renaming* und *Union* strukturiert, siehe [CEW93].

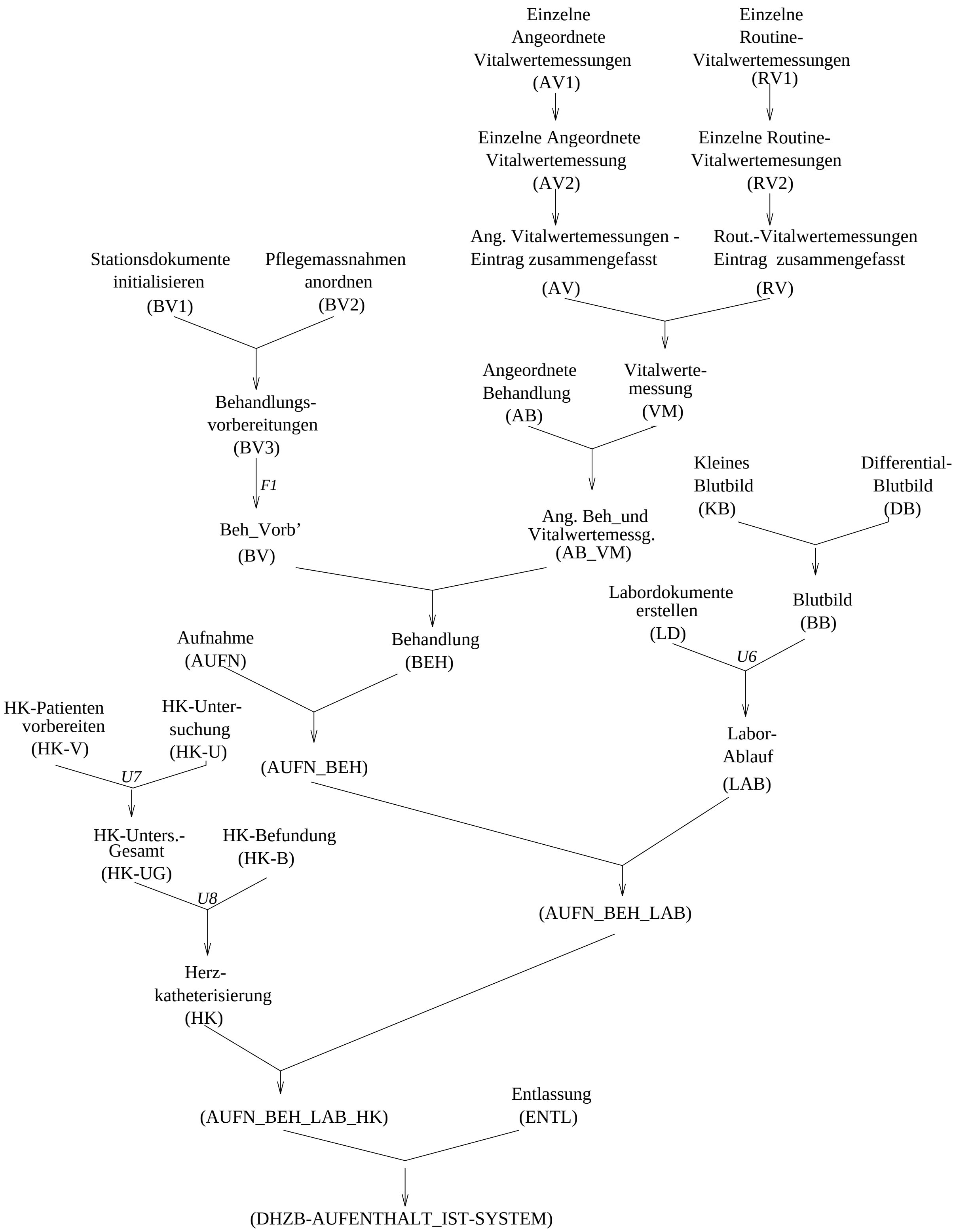
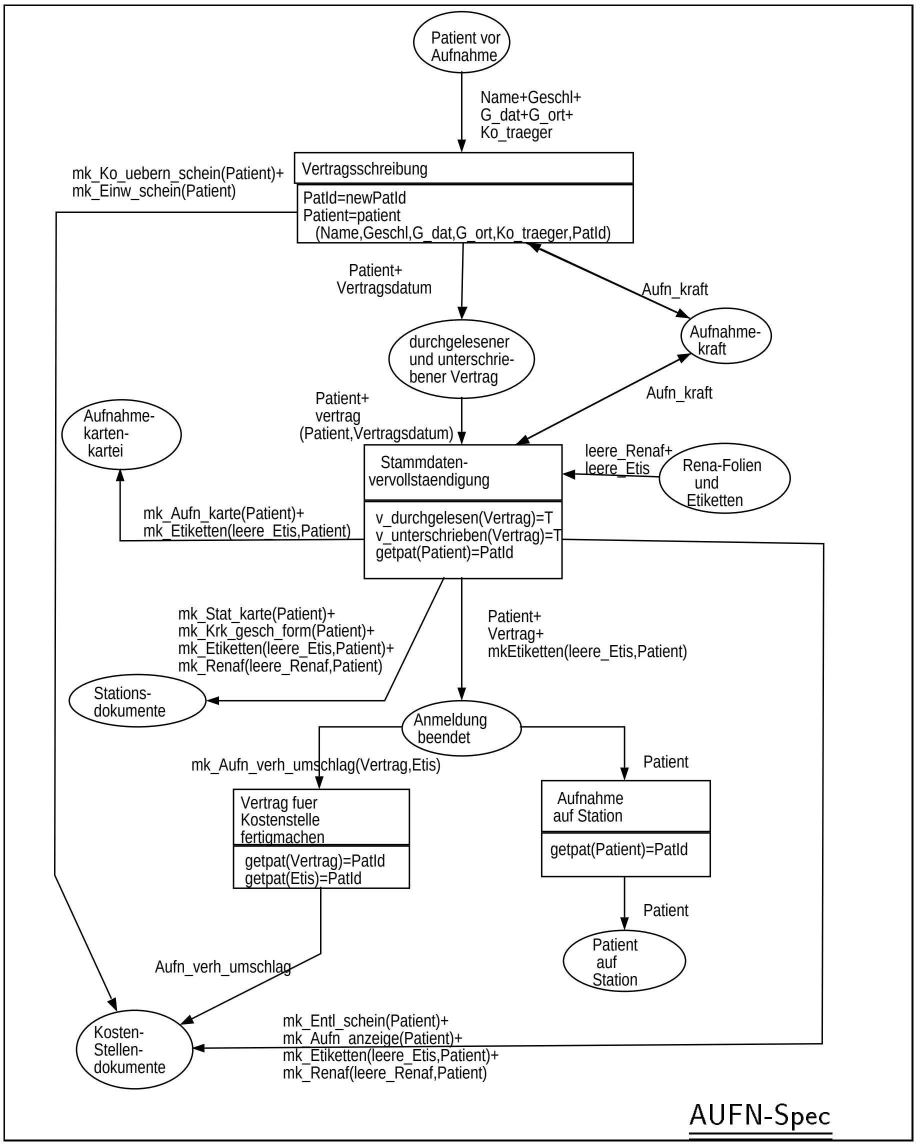


Abbildung 3.3: DHZB-Aufenthalt: Horizontale Strukturierung der Netze des IST-Systems

Abbildung 3.4: Benutzt-Hierarchie der Teilspezifikationen des IST-Systems

3.2.1 Aufnahme

Für den Arbeitsablauf *Aufnahme* gibt es nur ein Netz. Es beschreibt die Vertragsschreibung, sowie die Erzeugung einer Patientenidentifikation, die auf allen patientenbezogenen Dokumenten beim Krankenhausaufent halt vorhanden sein muß. Außerdem wird beschrieben, wie bestimmten Dokumenten Etiketten und Rena-Folien beigefügt werden, und an welche Stellen welche Dokumente nach Vertragsabschluß verteilt werden. Der Patient wird dann an seine erste Station überwiesen.



AUFN: Aufnahme_IST-System

Es folgt die algebraische Spezifikation AUFN-Spec zum Netz AUFN. Wir schreiben in allen Spezifikationen Sorten und Variablen mit großen und Operationen mit kleinen Anfangsbuchstaben.

AUFN-Spec: Spezifikation zum Netz Aufnahme

AUFN-Spec = PATIENT + VERTRAG + AUFNAHMEKARTE + STATIONSDOKUMENTE + KOSTENSTELLENDOKUMENTE + PERSONAL

Bei der Aufnahme im DHZB weist sich der ankommende Patient aus und übergibt Einweisungs- und Kostenübernahmeschein. Auf dieser Basis wird ein Krankenhaus-Behandlungsvertrag geschlossen. Im Dialog werden Stammdaten des Patienten erfragt: Name, Vorname, Adresse, Krankenkasse usw. Sie finden sich auf allen Dokumentvordrucken und Etikettensätzen wieder, die an die Kostenstelle und die erste Station geschickt werden, sowie auf denen, die in der Anmeldung verbleiben. Dazu gehört vorallem die Aufnahmekarte, auf der bei der Entlassung zusätzlich zu den Stammdatennoch Entlassungsdatum, -ziel und -diagnose vermerkt werden.
Die Spezifikation der Aufnahme enthält die Spezifikationen der verwendeten Dokumente.

PATIENT = STAMMDATEN +
sorts: Patient, Adresse, Pat_Ergänzung
opns: getpat: Patient $\rightarrow$ PatId
get_patient: PatId $\rightarrow$ Patient
pat_adresse: $\rightarrow$ Adresse
hausarzt_name:$\rightarrow$ Name
patient: Name $\times$ Geschlecht $\times$ Datum $\times$ Ort $\times$ Ko_träger $\times$ PatId $\rightarrow$ Patient
pat_erg: Adresse $\times$ Name $\rightarrow$ Pat_Ergänzung
eqns: getpat (patient (N, G, D, O, T, Id)) = Id;
get_patient(getpat(Patient))= Patient;

STAMMDATEN = PATID +
sorts: Name, Geschl, G_dat, G_ort, Ko_träger

PATID =
sorts: PatId
opns: – *newPatId generiert eindeutige PatId für neuen Patienten*
newPatId: $\rightarrow$ PatId

VERTRAG = STAMMDATEN $+_{PATID}$ ETIKETTEN +

sorts: Vertrag, Aufn_verh_umschlag
opns: vertr_datum: $\rightarrow$ Datum
getpat: Vertrag $\rightarrow$ PatId
mk_Vertrag: Patient, Datum $\rightarrow$ Vertrag
v_durchgelesen: Vertrag $\rightarrow$ Bool
v_unterschrieben: Vertrag $\rightarrow$ Bool
mk_Aufn_verh_umschlag: Vertrag $\times$ Etiketten $\rightarrow$ Aufn_verh_umschlag
eqns: getpat (mk_Vertrag(Pat, D)) = getpat(Pat);

ETIKETTEN = PATID +
sorts: Etiketten
opns: leere_Etis: $\rightarrow$ Etiketten
mk_Etiketten: Etiketten $\times$ Patient $\rightarrow$ Etiketten
getpat: Etiketten $\rightarrow$ PatId
eqns: getpat (mk_Etiketten(E, Pat)) = getpat(Pat);

AUFNAHMEKARTE = ETIKETTEN $+_{PATID}$ + RENA-FOLIEN + DIAGNOSE +

Die Aufnahmekarte enthält über die Stammdaten hinausgehende Angaben (Pat_Ergänzung), z.B. Namen und Adresse des Hausarztes. Entlassungsziel und -diagnose werden bei der Entlassung des Patienten auf der Aufnahmekarte vermerkt, bevor sie archiviert wird. Die Spezifikation DIAGNOSE ist Teil von HK-Spec

sorts: Aufn_karte, Entlassungsziel
opns:
– *Aufnahmekarte mit Stammdaten und -ergänzungen anlegen*
aufn_karte : Patient $\times$ Pat_Ergänzung $\times$ Zeitpunkt $\times$
Entlassungsziel $\times$ Diagnose $\rightarrow$ Aufn_karte
keine_E_zp: $\rightarrow$ Zeitpunkt
kein_E_ziel: $\rightarrow$ Entlassungsziel
keine_E_diag: $\rightarrow$ Diagnose
mk_Aufn_karte : Patient $\times$ Pat_Ergänzung $\rightarrow$ Aufn_karte

– *Aufnahmekarte identifizieren*
getpat: Aufn_karte $\rightarrow$ PatId
get_Patient: Aufn_karte $\rightarrow$ Patient;
get_Pat_Erg: Aufn_karte $\rightarrow$ Pat_Ergänzung;
– *Aufnahmekarte bei der Entlassung um Entl.daten vervollständigen*
ch_Aufn_karte: Aufn_karte $\times$ Zeitpunkt $\times$ Entlassungsziel $\times$
Diagnose $\rightarrow$ Aufn_karte

eqns: – *Aufnahmekarte mit Stammdaten und -ergänzungen anlegen*

mk_Aufn_karte(Pat,Pat_Erg) =
aufn_karte(Pat,Pat_Erg,keine_E_zp,kein_E_ziel, keine_E_diag);

– *Aufnahmekarte identifizieren*

getpat (mk_Aufn_karte (Id)) = Id;
getpat (ch_Aufn_karte (A,Zp,Ez,Diag)) = getpat(A);
get_Patient(aufn_karte(Pat,Pat_Erg,E_zp,E_ziel, E_diag) = Pat;
get_Pat_Erg(aufn_karte(Pat,Pat_Erg,E_zp,E_ziel, E_diag) = Pat_Erg;

– *Aufnahmekarte bei der Entlassung um Entl.daten vervollständigen*

ch_Aufn_karte(Aufnk,Zp,E_zp,E_ziel,E_diag) =
aufn_karte(get_Patient(Aufnk),
get_Pat_Erg(Aufnk),E_zp,E_ziel, E_diag);

RENA-FOLIEN = PATID +

sorts: Rena-Folien

opns: leere_Renaf: → Rena-Folien
mk_Renaf: Rena-Folien× Patient → Rena-Folien

STATIONSDOKUMENTE = ETIKETTEN $+_{PATID}$ RENA-FOLIEN +

sorts: Stationskarte, Krankengesch_formular

opns: getpat: Stationskarte → PatId
getpat: Krankengesch_formular → PatId
mk_Stationskarte: Patient → Stationskarte
mk_Krankengesch_formular: Patient
→ Krankengesch_formular

eqns: getpat (mk_Stationskarte (Pat)) = getpat(Pat);
getpat (mk_Krankengesch_formular(Pat)) = getpat(Pat);

KOSTENSTELLENDOKUMENTE = ETIKETTEN $+_{PATID}$ RENA-FOLIEN +

sorts: Einw_schein, Ko_übernahmeschein, Aufn_anzeige,
Entl_schein

opns: –*Dokumente identifizieren*

getpat: Einw_schein → PatId
getpat: Ko_übernahmeschein→ PatId
getpat: Aufn_anzeige→ PatId
getpat: Entl_schein→ PatId

–*Dokumente anlegen*

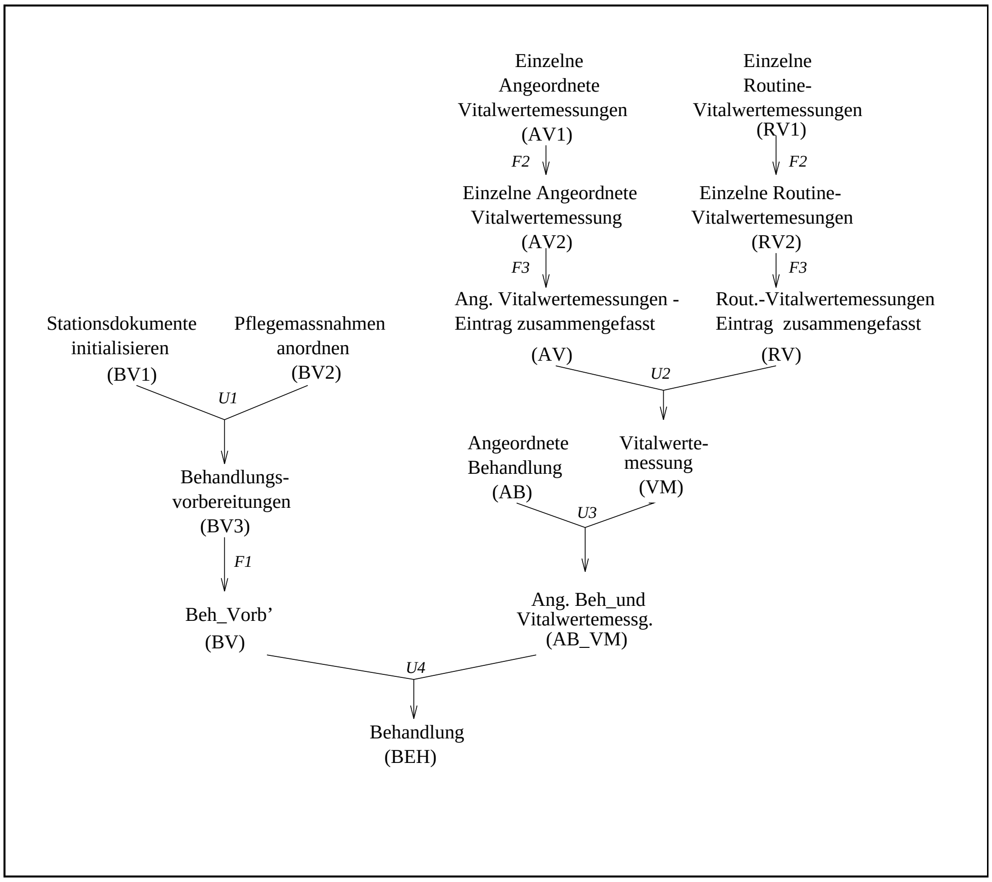
mk_Einw_schein: Patient → Einw_schein

mk_Ko_übernahmeschein: PatId → Ko_übernahmeschein
mk_Aufn_anzeige: Patient→ Aufn_anzeige
mk_Entl_schein: Patient → Entl_schein
eqns: getpat (mk_Einw_schein(Pat)) = getpat(Pat);
getpat (mk_Ko_übernahmeschein(Pat)) = getpat(Pat);
getpat (mk_Aufn_anzeige(Pat)) = getpat(Pat);
getpat (mk_Entl_schein(Pat)) = getpat(Pat);

PERSONAL = *–In den Netzen erscheinen nur Variablen zu den jeweiligen Sorten.*
sorts: Aufn_kraft, Arzt, Pfleger, Schwester, Stat_kraft, MTA

3.2.2 Behandlung

Der Behandlung umfaßt den Umgang von Arzt und Pflegepersonal mit dem Patienten auf der Pflege- und der Intensivstation und ist in folgende Teilnetze untergliedert:

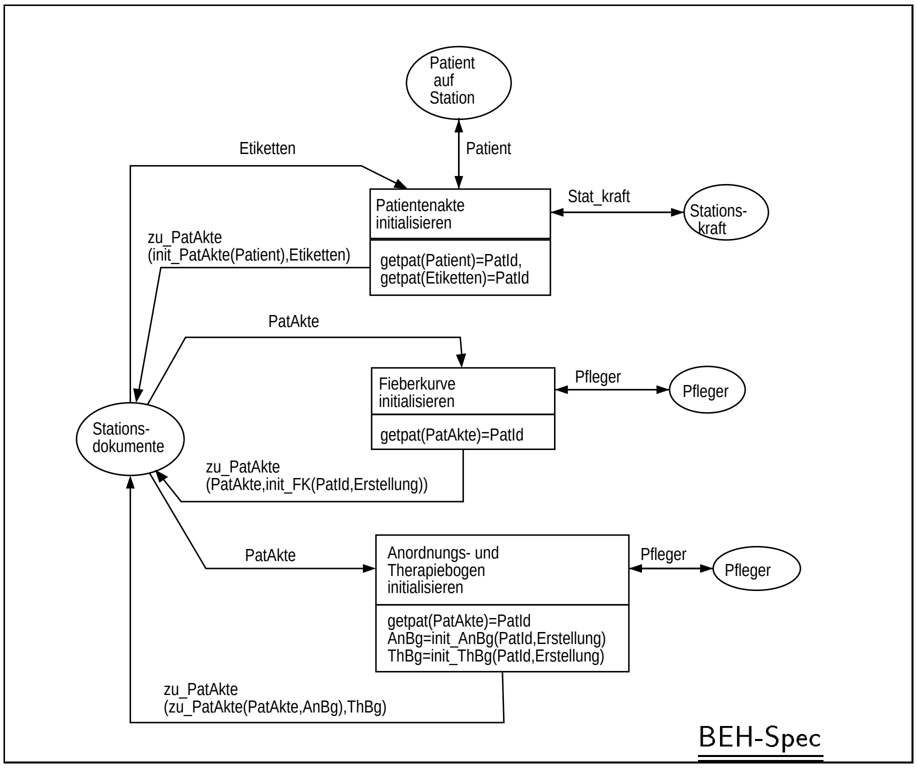


Behandlung: Strukturierung der Netze

Hier dargestellt werden die Teilnetze Stationsdokumente initialisieren (BV1), Pflegemaßnahmen anordnen (BV2), Behandlungsvorbereitungen (BV3), Einzelne_Angeordnete_Vitalwertemessungen (AV1, AV2, AV), Einzelne_Routine_Vitalwertemessungen (RV1) und Angeordnete_Behandlung (AB), sowie die für die Anwendung der jeweiligen *Union*- bzw. *Fusion*-Operationen notwendigen Schnittstellennetze.

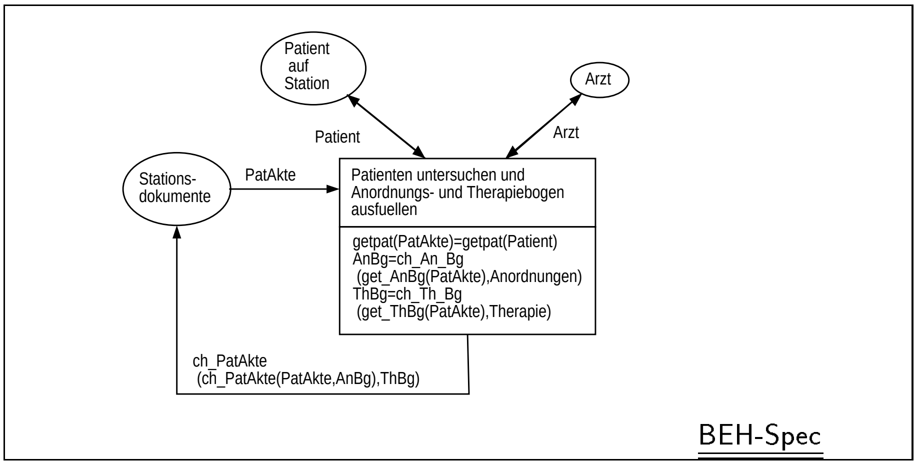
Die Spezifikation BEH-Spec ist nicht entsprechend der Netze strukturiert, da es bei den verwendeten Datentypen (hauptsächlich beim Typ *PatAkte*) zu viele Überschneidungen gibt.

Im Netz BV1 werden die wichtigsten Dokumente für die Betreuung des Patienten auf der Station angelegt: die Patientenakte, die diverse patientenrelevante Dokumente enthalten soll; die Fieberkurve, in die Meßwerte der täglich am Patienten vorgenommenen Untersuchungen eingetragen werden; der Anordnungsbogen, auf dem Anordnungen des Arztes an das Pflegepersonal eingetragen werden; der Therapiebogen, auf dem der Arzt die verschriebenen medizinischen und chirurgischen Therapiemaßnahmen angeben wird.

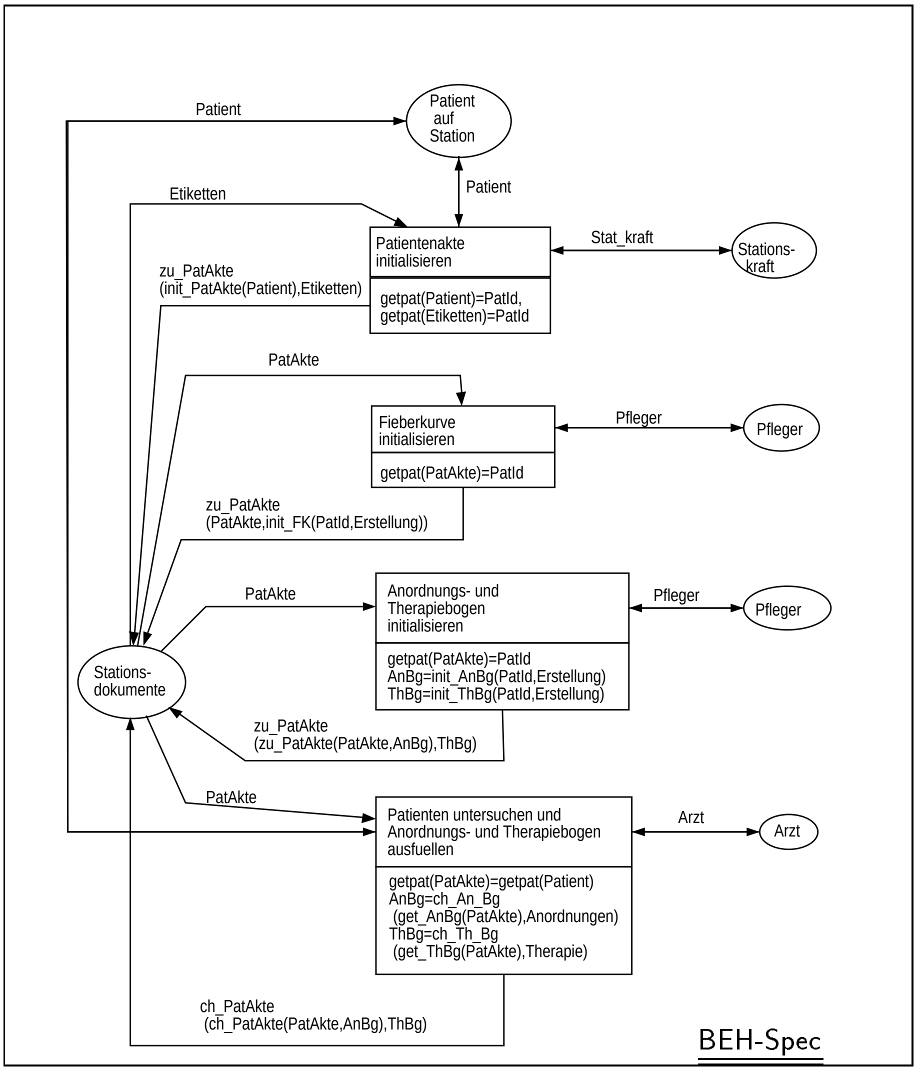


BV1: Stationsdokumente initialisieren

Das Netz BV2 modelliert die Untersuchung des Patienten, deren Ergebnisse der Arzt in den Anordnungsbogen bzw. Therapiebogen einträgt. Die Bögen werden also der Patientenakte entnommen und ihr dann, mit den eingetragenen Anordnungen und Therapiemaßnahmen, wieder hinzugefügt. Die Anwendung der *Union* über dem Netz U1, das nur die Stellen *Stationsdokumente* und *Patient auf Station* enthält, ergibt das vereinigte Netz BV3 und die Anwendung der *Fusion* über das Netz $F1$, bestehend aus der Stelle *Pfleger*, ergibt Netz BV.

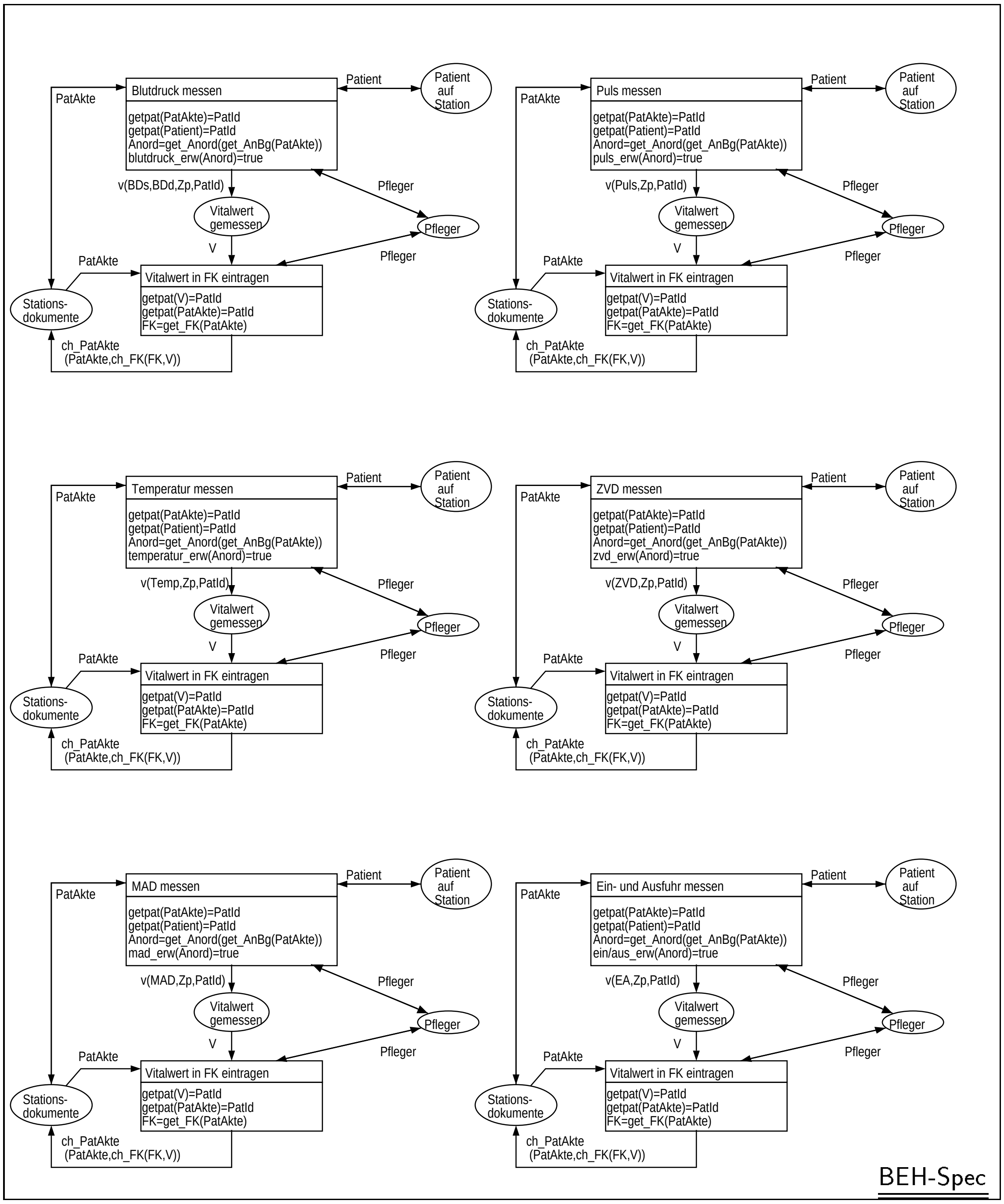


BV2: Pflegemaßnahmen anordnen



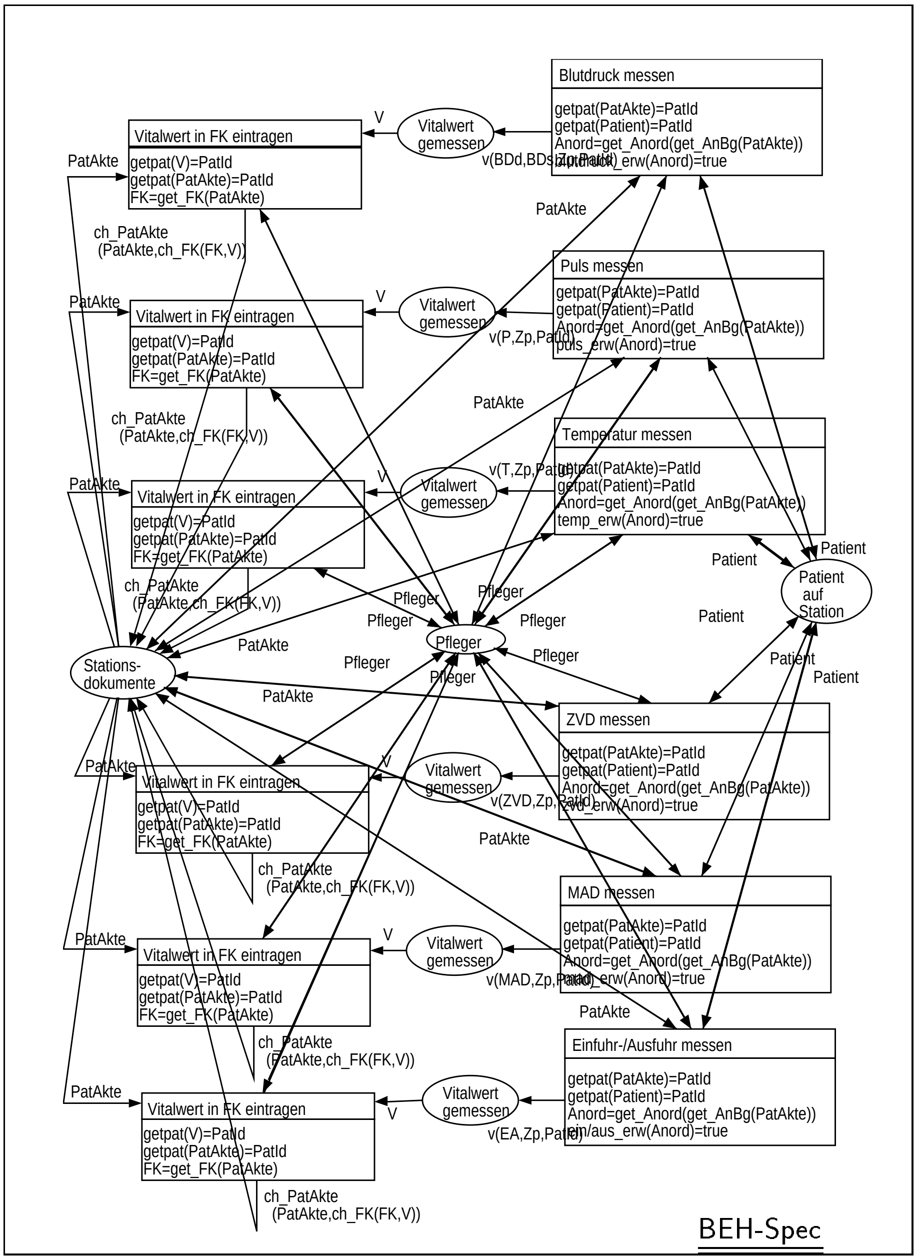
BV3:Behandlungsvorbereitungen

Das folgende Netz (AV1) zeigt sechs von der Struktur her gleiche Teilnetze, die den Messungen unterschiedlicher Vitalwerte (Puls, Temperatur, Blutdruck, Zentralvenöser Druck (ZVD), Mittlerer Arterieller Druck (MAD), Ein-/Ausfuhr) entsprechen. Die Werte werden in die entsprechenden Meßwertetabellen der Fieberkurve aus der passenden Patientenakte eingetragen.



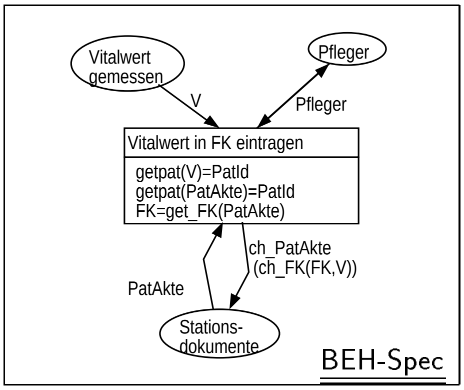
AV1:Einzelne_Angeordnete_Vitalwertemessungen

Wir fassen die in den einzelnen Teilnetzen gleichen Stellen nun über eine *Fusion*-Operation zusammen, wobei das Teilnetz F2 die Stellen *Stationsdokumente, Pfleger* und *Patient auf Station* enthält, und erhalten das zusammenhängende Netz AV2:

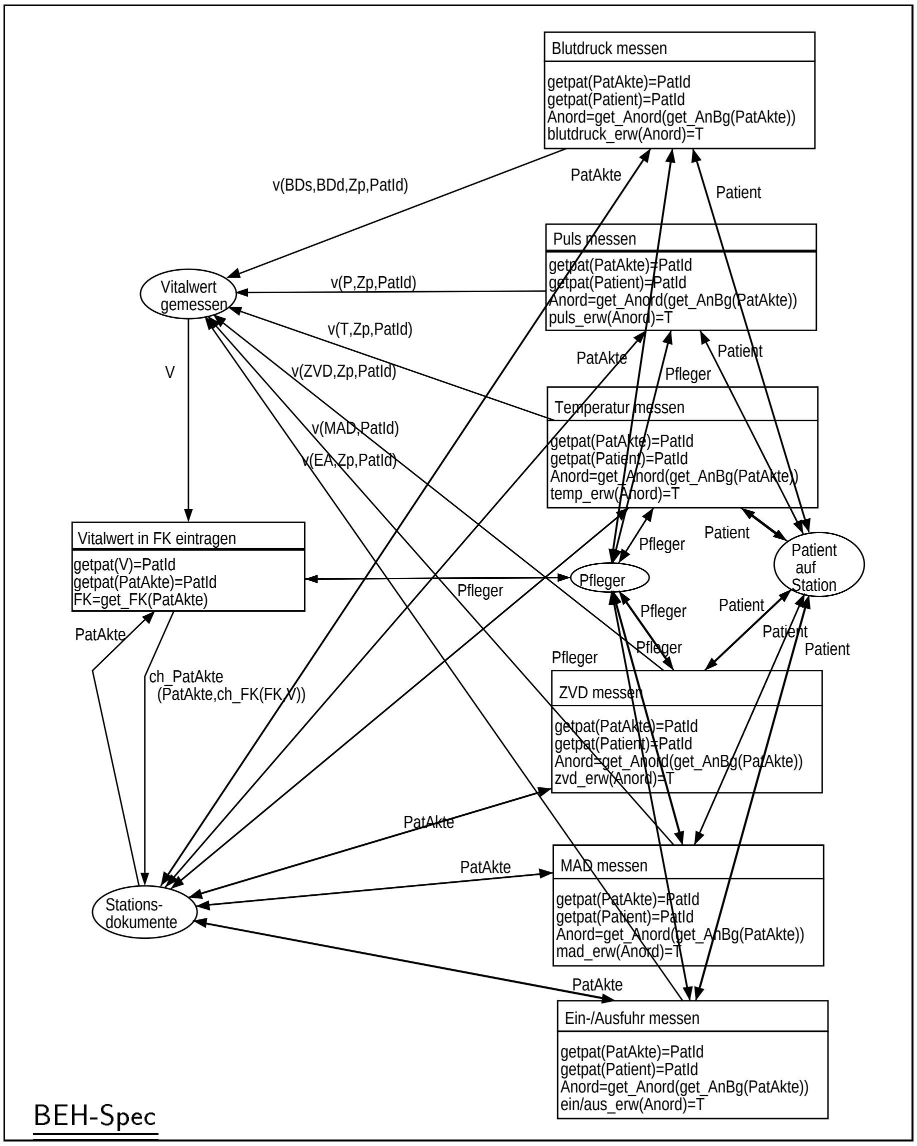


AV2:Einzelne_Angeordnete_Vitalwertemessungen, nach Fusion F2

Eine weitere Fusion über dem Teilnetz (F3) liefert das Netz AV, wo die Eintragung der Vitalwerte in die entsprechende Fieberkurve nur noch einmal modelliert wird. Das Netz wird dadurch wesentlich übersichtlicher, ohne daß inhaltliche Aussagen verlorengehen.



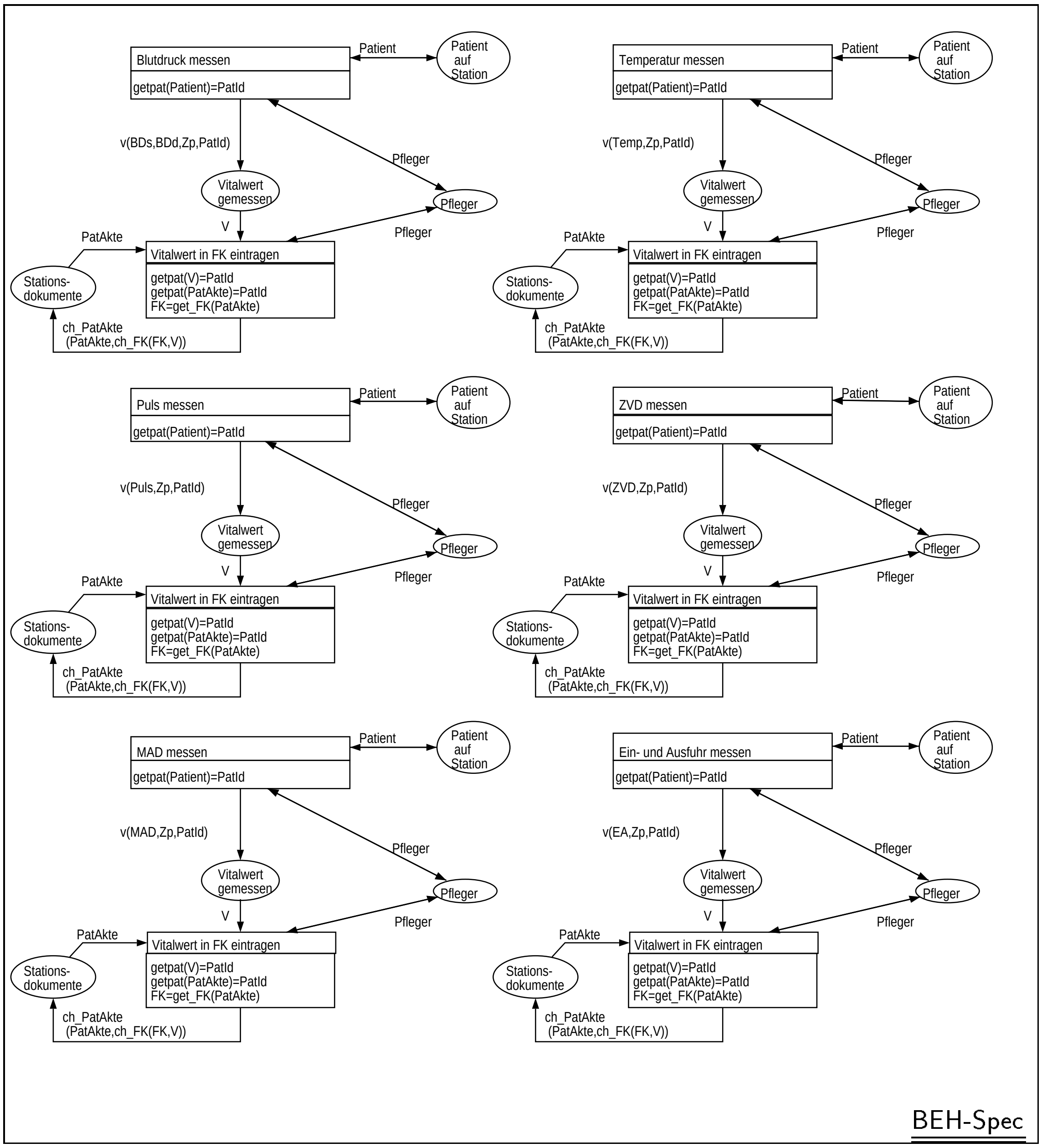
F3: Gemeinsames Teilnetz zur Fusion von AV2 zu AV



AV: Ang.Vitalwertemessungen - Eintrag zusammengefaßt

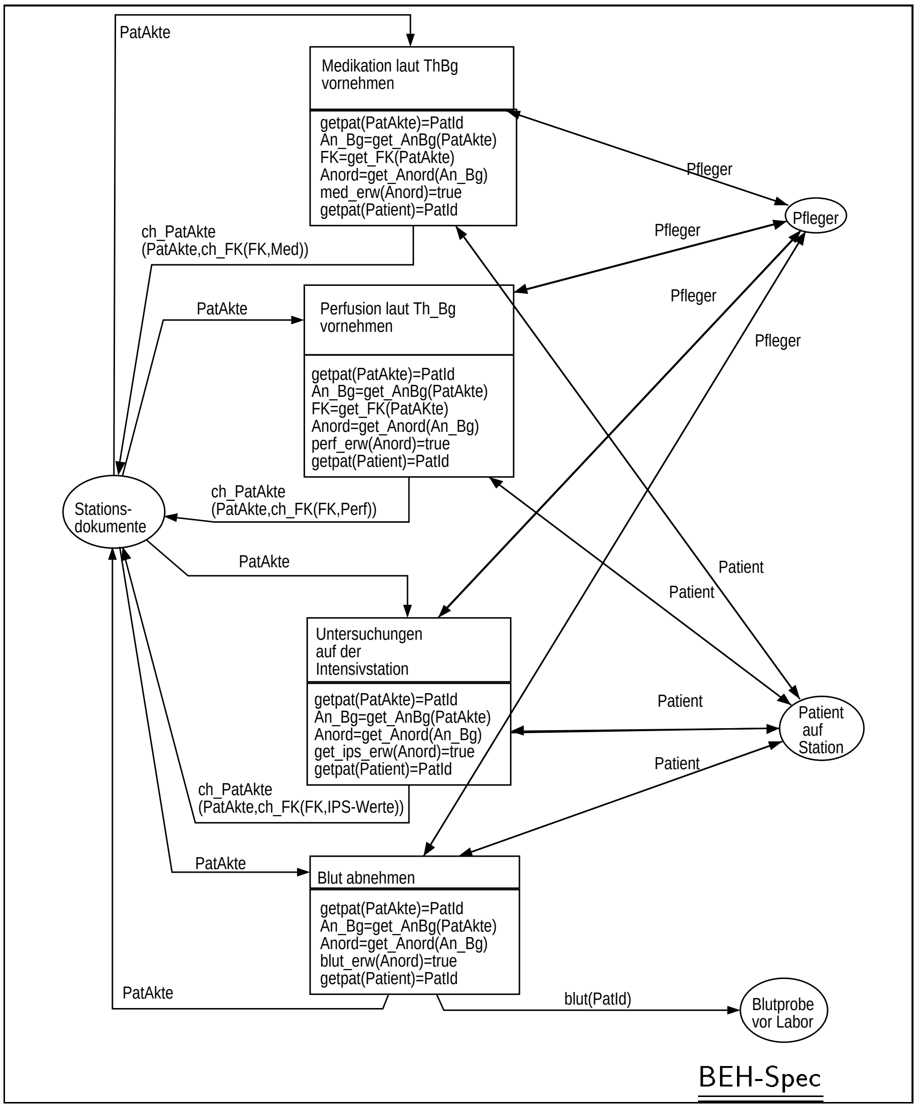
Die im Netz RV1 modellierten Routine-Vitalwertemessungen unterscheiden sich von den Angeordneten Vitalwertemessungen nur insofern, als daß der Anordnungsbogen nicht überprüft wird. Die Patientenakte wird daher erst zum Eintragen der Messungen benötigt.

Die Fusionen zu den Netzen RV2 und RV entsprechen den Fusionen zu AV2 und AV bei den Angeordneten Vitalwertemessungen und werden daher hier nicht explizit dargestellt.



RV1:Einzelne_Routine-Vitalwertemessungen

Weitere angeordnete Pflegemaßnahmen sind im Netz Angeordnete Behandlung (AB) zusammengefaßt und betreffen die Verabreichung von Medikamenten bzw. Perfusionen, sowie Untersuchungen auf der Intensivstation und die Entnahme von Blut für eine Laboruntersuchung:



AB: Angeordnete_Behandlung

Führt man nun mit den angegebenen Netzen die Operationen *Fusion* und *Union* so durch, wie im Übersichtsdiagramm (Abb. 3.3) dargestellt, so erhält man das Netz Behandlung (BEH), zu dem, ebenso wie zu den Teilnetzen, die Spezifikation BEH-Spec gehört:

BEH-Spec: Spezifikation zum Netz Behandlung

BEH-Spec = PAT_AKTE + PERSONAL

PAT_AKTE = PA-DATEN + PATIENT $+_{PAT\_ID}$ + ETIKETTEN $+_{PAT\_ID}$ ANORDN_BOGEN $+_{PAT\_ID}$ THERAPIEBOGEN $+_{PAT\_ID}$ FIEBERKURVE + DHZB_NAB + HK-ÜBERWEISUNG +

Der Datentyp PatAkte entspricht der Patientenakte aus Papier, die alle relevanten Dokumente zu einem Patienten enthält. Der Datentyp modelliert somit den statischen Kern eines DHZB-Aufenthalts und wird als Liste realisiert (nicht als Tupel, da die PatAkte z.B. mehrere HK-Überweisungen oder Therapiebogen enthalten darf).
Die Liste enthält normalerweise folgende Dokumente als Elemente:
Patient: Stammdaten (spezifiziert in AUFN-Spec)
Etiketten: (spezifiziert in AUFN-Spec)
Anordnungsbogen: Liste ärztlicher Pflegeanordnungen
Therapiebogen: Liste ärztlicher Therapiemaßnahmen (auch Medikation)
Fieberkurve: enthält sämtliche medizinischen Meßwerte als Listen
NAB: Notfall-Labor- Anforderungs- und Befundbogenbogen,
–(spezifiziert in LAB-Spec)
HK-Überweisung: Überweisung des Patienten zur Herzkatheterisierung
–(spezifiziert in HK-Spec)

actualize LISTE by PA-DATEN
using
sortnames PA-Daten for D
PatAkte for Liste(PA-Daten)
opnames list_PA for list: PA-Daten → PatAkte
keine_PA_Daten for nodata: → PA-Daten
zur_PatAkte for radd: PatAkte × PA-Daten → PatAkte
head_PA for head: PatAkte → PA-Daten
search_PA for search: PA-Daten × PatAkte → PA-Daten
ladd_PA for ladd: PA-Daten × PatAkte → PatAkte
ch_PA-Daten for change: PatAkte × PA-Daten × PA-Daten → PatAkte
endact +

– Umwandlung verschiedener Typen in PA-Daten und zurück
opns: patient2PA-Daten: Patient→ PA-Daten
pa-Daten2Patient: PA-Daten → Patient
etis2PA-Daten: Etiketten → PA-Daten
pa-Daten2Etis: PA-Daten → Etiketten

an_bg2PA-Daten: Anordn_bogen → PA-Daten
pa-daten2An_bg: PA-Daten → Anordn_bogen
th_bg2PA-Daten: Therapiebogen → PA-Daten
pa-daten2Th_bg: PA-Daten → Therapiebogen
fk2PA-Daten: Fieberkurve → PA-Daten
pa-Daten2Fk: PA-Daten → Fieberkurve
nab2PA-Daten: NAB → PA-Daten
pa-Daten2NAB: PA-Daten → NAB
überw2PA-Daten: HK_Überw → PA-Daten
pa-Daten2Überw: PA-Daten → HK_Überw

– *PatAkte identifizieren*
getpat: PatAkte → PatId

– *PatAkte nach bestimmten Einträgen durchsuchen*
get_Patient: PatAkte → Patient
get_Etis: PatAkte → Etiketten
get_AnBg: PatAkte → Anordn_bogen
get_ThBg: PatAkte → Therapiebogen
get_FK: PatAkte → Fieberkurve
get_NAB: PatAkte → NAB
get_Überw: PatAkte → HK_Überw

– *PatAkte initialisieren*
init_PatAkte:Patient → PatAkte

– *Dokument in die PatAkte einlegen*
zu_PatAkte: PatAkte × Fieberkurve → PatAkte
zu_PatAkte: PatAkte × Anordn_bogen → PatAkte
zu_PatAkte: PatAkte × Therapiebogen → PatAkte
zu_PatAkte: PatAkte × Etiketten → PatAkte
zu_PatAkte: PatAkte × NAB → PatAkte
zu_PatAkte: PatAkte × HK-Überweisung → PatAkte
zu_PatAkte: PatAkte × Druckkurve → PatAkte
– *Eintrag in ein Dokument der PatAkte vornehmen*
ch_PatAkte: PatAkte × Fieberkurve → PatAkte
ch_PatAkte: PatAkte × Anordn_bogen → PatAkte
ch_PatAkte: PatAkte × Therapiebogen→ PatAkte

eqns:

– *Umwandlung verschiedener Typen in PA-Daten und zurück*
patient2PA-Daten(pa-Daten2Patient(PD)) = PD;
pa-Daten2Patient(patient2PA-Daten(PT)) = PT;
etis2PA-Daten(pa-Daten2Etis(PD)) = PD;
pa-Daten2Etis(etis2PA-Daten(E)) = E;

an_bg2PA-Daten(pa-Daten2An_bg(PD)) = PD;
pa-Daten2An_bg(an_bg2PA-Daten(AB)) = AB;
th_bg2PA-Daten(pa-Daten2Th_bg(PD)) = PD;
pa-Daten2Th_bg(th_bg2PA-Daten(TB)) = TB;
fk2PA-Daten(pa-Daten2Fk(PD)) = PD;
pa-Daten2Fk(fk2PA-Daten(FK)) = FK;
nab2PA-Daten(pa-Daten2NAB(PD)) = PD;
pa-Daten2NAB(nab2PA-Daten(NB)) = FK;
überw2PA-Daten(pa-Daten2Überw(PD)) = PD;
pa-Daten2Überw(überw2PA-Daten(ÜB)) = FK;

– *PatAkte identifizieren*
getpat (ch_PatAkte(PA,FK)) = getpat(PA);
getpat (ch_PatAkte(PA,AnBg)) = getpat(PA);
getpat (ch_PatAkte(PA,ThBg)) = getpat(PA);
getpat (init_PatAkte(Pat)) = getpat(Pat);
getpat (PA) = getpat(get_Patient(PA));

– *Dokument in die PatAkte einlegen*
zu_PatAkte(PA,FK) = zur_PatAkte(PA,fk2PA-Daten(FK));
zu_PatAkte(PA,AnBg) = zur_PatAkte(PA,an_bg2PA-Daten(Anbg));
zu_PatAkte(PA,ThBg) = zur_PatAkte(PA,th_bg2PA-Daten(ThBg));
zu_PatAkte(PA,Etis) = zur_PatAkte(PA,etis2PA-Daten(Etis));
zu_PatAkte(PA,NAB) = zur_PatAkte(PA,nab2PA-Daten(NAB));
zu_PatAkte(PA,ÜB) = zur_PatAkte(PA,über2PA-Daten(Üb));

– *PatAkte nach bestimmten Einträgen durchsuchen*
get_Patient (PA) = pa-Daten2Patient(head_PA(PA)) ;
get_Etis (PA) = pa-Daten2Etis(search_PA(etis2PA-Daten(Etis)));
get_AnBg (PA)= pa-Daten2An_bg(search_PA(an_bg2PA-Daten(AnBg)));
get_ThBg (PA)= pa-daten2Th_bg(search_PA(th_bg2PA-Daten(ThBg)));
get_Fk (PA)= pa-daten2Fk(search_PA(fk2PA-Daten(FK)));
get_NAB (PA)= pa-daten2NAB(search_PA(nab2PA-Daten(NB)));
get_Überw (PA)= pa-daten2Überw(search_PA(überw2PA-Daten(ÜB)));

– *PatAkte initialisieren*
init_PatAkte(Pat) = list_PA(patient2PA-Daten(Pat));

– *Eintrag in ein Dokument der PatAkte vornehmen*
ch_PatAkte(PA,FK_neu) =
    ch_PA-Daten (PA,fk2PA-Daten(FK_neu),
    fk2PA-Daten(get_FK(PA)))

ch_PatAkte(PA,An_Bg_neu) =
    ch_PA-Daten (PA,an_bg2PA-Daten(An_Bg_neu),

an_bg2PA-Daten(get_AnBg(PA)))

ch_PatAkte(PA,Th_Bg_neu) =
ch_PA-Daten (PA,th_bg2PA-Daten(Th_Bg_neu),
th_bg2PA-Daten(get_ThBg(PA)))

PA-DATEN =
Hilfsdatentyp zur Umwandlung aller Daten in PatAkten-Einträge
sorts: PA-Daten

ANORDN_BOGEN = PATID +

Der Anordnungsbogen wird vom Arzt nach einer Untersuchung des Patienten ausgefüllt und enthält Anordnungen an das Pflegepersonal auf der Station. Die Anordnungen betreffen Messungen bestimmter Werte (Vitalwerte: Blutdruck, Puls, Temperatur, ...), aber auch Untersuchungen auf der Intensivstation, die Verabreichung von Medikamenten und Infusionen, sowie die Entnahme einer Blutprobe für eine Laboranalyse.
Art, Dauer und Dosis der zu verabreichenden Medikamente finden sich auf dem Therapiebogen (Datentyp THERAPIEBOGEN).
Die laut Anordnung (bzw. Routine) gemessenen Werte und verabreichten Medikamente werden in das Dokument Fieberkurve eingetragen.

sorts: Anordn_bogen, An_Bg_Id, Anordnungen
opns:
– *Anordnungsbogen initialisieren*
keine_Anordnungen:$\rightarrow$ Anordnungen
init_An_Bg: PatId $\times$ Datum $\rightarrow$ Anordn_bogen
mk_An_Bg_Id: PatId $\times$ Datum $\rightarrow$ An_Bg_Id
an_bg: An_Bg_Id $\times$ Anordnungen $\rightarrow$ Anordn_bogen

– *Anordnungsbogen identifizieren*
getpat: Anordn_bogen$\rightarrow$ PatId
get_An_Bg_Id: Anordn_bogen$\rightarrow$ An_Bg_Id

– *Anordnungsbogen ausfüllen*
ch_An_Bg: Anordn_bogen $\times$ Anordnungen $\rightarrow$ Anordn_bogen

– *Anordnungen auf dem Anordnungsbogen lesen*
– *(Anordnungen: Tupel vom Arzt gewünschter Pflegemaßnahmen)*
get_Anord: Anordn_bogen$\rightarrow$ Anordnungen

– *12 Anordnungen gibt es maximal auf dem Anordnungsbogen:*

– *1. Kleines Blutbild, 2. Diff.blutbild, 3. Blutabnahme,*
– *4. Blutdruckmessg, 5. MAD-Messg., 6. Puls-Messg.,*
– *7. Temp.-Messg, 8. ZVD-Messg., 9. Ein-/Ausfuhr,*
– *10.Intensivstat.messg., 11. Medikation, 12. Perfusion*

anord: $\text{Bool}^{12} \rightarrow$ Anordnungen

– *Anordnungen lesen*

klBB_erw, diffBB_erw, blut_erw, blutdruck_erw, mad_erw,
puls_erw, temp_erw, zvd_erw, ein/aus_erw, ips_erw,
med_erw, perf_erw: Anordnungen $\rightarrow$ Bool

eqns:

– *Anordnungsbogen initialisieren*

init_An_Bg(Id,Dat) = an_bg(mk_An_Bg_Id(Id,Dat),keine_Anordnungen);

– *Anordnungsbogen identifizieren*

getpat(init_An_Bg(Id,D) = Id;
getpat(ch_An_Bg(AB,A) = getpat(AB);

– *Anordnungsbogen ausfüllen*

ch_An_Bg(AB,A) = an_bg(get_An_Bg_Id(AB),A);

– *Einträge auf dem Anordnungsbogen lesen*

get_Anord(init_An_Bg(Id,Dat)) = keine_Anordnungen;
get_Anord(ch_An_Bg(AB,A)) = A;

– *Anordnungen lesen*

klBB_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = K;
diffBB_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = D;
blut_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = B;
blutdruck_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = BD;
mad_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = MAD;
puls_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = P;
temp_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = T;
zvd_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = ZVD;
ein/aus_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = EA;
ips_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = IPS;
med_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = MED;
perf_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = PER;

<u>THERAPIEBOGEN</u> = <u>PATID</u> + <u>THERAPIE</u> +

Der Therapiebogen wird vom Arzt nach einer Untersuchung, bzw. nach einer Herzkatheter-Untersuchung des Patienten ausgefüllt. Er enthält Angaben über notwendige chirurgische Therapiemaßnahmen (Bypass,

Klappenersatz), aber auch über die medikamentorische Weiterbehandlung des Patienten. Ist eine Medikation oder Perfusion nötig, so wird das auch auf dem Anordnungsbogen für das Pflegepersonal vermerkt.
Art, Dauer und Dosis der zu verabreichenden Medikamente finden sich aber nur auf dem Therapiebogen.
Die laut Anordnung (bzw. Routine) und Therapiebogen verabreichten Medikamente werden in das Dokument Fieberkurve eingetragen.

sorts: Therapiebogen, Th_Bg_Id
opns:

– Therapiebogen initialisieren
init_Th_Bg: PatId $\times$ Datum $\rightarrow$ Therapiebogen
keine_Therapie:$\rightarrow$ Therapie
mk_Th_Bg_Id: PatId $\times$ Datum $\rightarrow$ Th_Bg_Id
th_bg: Th_Bg_Id, Therapie $\rightarrow$ Therapiebogen

– Therapiebogen identifizieren
getpat: Therapiebogen $\rightarrow$ PatId
get_Th_Bg_Id: Therapie_bogen$\rightarrow$ Th_Bg_Id

– Therapiebogen ausfüllen
– (Therapie: Tupel (Chirurg.Th., Medizin. Th))
ch_Th_Bg: Therapiebogen $\times$ Therapie$\rightarrow$ Therapiebogen

– Therapieangaben auf dem Therapiebogen lesen
get_Th: Therapiebogen $\rightarrow$ Therapie

eqns:

– Therapiebogen initialisieren
init_Th_Bg(Id,Dat) = th_bg(mk_Th_Bg_Id(Id,Dat),keine_Therapie);

– Therapiebogen identifizieren
getpat(init_Th_Bg(Id,D) = Id;
getpat(ch_Th_Bg(TB,Th) = getpat(TB);

– Therapiebogen ausfüllen
ch_Th_Bg(TB,Th) = th_bg(get_Th_Bg_Id(TB),Th);

– Einträge auf dem Therapiebogen lesen
get_Th(init_Th_Bg(Id,Dat)) = keine_Therapie;
get_Th(ch_Th_Bg(TB,Th)) = Th;

$\underline{\underline{\text{THERAPIE}}} = \underline{\underline{\text{MEDIKATIONEN}}} +_{\underline{\underline{MEDIKAMENT,DOSIS}}} \underline{\underline{\text{PERFUSIONEN}}} +$

Therapie wird als Tupel (Chirurgische Therapie, Medizinische Therapie) realisiert, wobei die chirurg. Therapie angibt, ob und welche Art von Bypass bzw. Klappenersatz notwendig ist, und die medizinische Therapie zu verabreichende Medikamente (bzw. Perfusoren), deren Dosierung und Medikationszeitraum beschreibt.

sorts: Klappenersatz, Chirurgische_Th, Medizinische_Th, Therapie
opns:

– Chirurgische Therapie
kein_Klappenersatz: → Klappenersatz
plastik, Bio_prothese, ...(?) ...: → Klappenersatz

lad_bypass, d1_bypass, d2_bypass, circ_bypass, m1_bypass, m2_bypass, m3_bypass, rka_bypass, r_desc_post_bypass, ptca_ct: → Bool
keine_Chirurg_Th:→ Chirurgische_Th

chirg_Th: Bool^{10} × Klappenersatz → Chirurgische_Th

– Medizinische Therapie
kardiale_Th:→ Bool
zusatz: → Text
med_Th: Bool × Medikationen × Perfusionen × Text → Medizinische_Th

–Therapie insgesamt
therapie: Chirurgische_Th × Medizinische_Th → Therapie
get_chirurg_Th: Therapie → Chirurgische_Th
get_medizin_Th: Therapie → Medizinische_Th

eqns:

get_chirurg_Th (therapie(CT,MT) = CT;
get_medizin_Th (therapie(CT,MT) = MT;

MEDIKATIONEN = actualize LISTE by MEDIKATION
using
sortnames Medikation for D
Medikationen for Liste(Medikation)
opnames list_Med for list: Medikation → Medikationen
keine_Medikation for nil: → Medikationen
keine_Med for nodata: → Medikation
ch_Medikationen for radd: Medikationen × Medikation → Medikationen
head_Med for head: Medikationen → Medikation
search_Med for search: Medikation × Medikationen → Medikation

ladd_Med for ladd: Medikation $\times$ Medikationen $\rightarrow$ Medikationen
endact

<u>MEDIKATION</u> = <u>MEDIKAMENT</u> +<u>DOSIS</u> +
sorts: Medikation
opns: beginne_Med, beende_Med: $\rightarrow$ Zeitpunkt
keine_Med: $\rightarrow$ Medikation
mk_Medikation: Medikament $\times$ Zeitpunkt^2 $\times$ Dosis $\rightarrow$ Medikation
getMedBeginn, getMedEnde: Medikation $\rightarrow$ Zeitpunkt
eqns: getMedBeginn(mk_Medikation(M, Z1, Z2)) = Z1;
getMedEnde(mk_Medikation(M, Z1, Z2)) = Z2;
getMedBeginn(M) $\leq$ getMedEnde(M);

<u>MEDIKAMENT</u> =

Medikamente werden die Arzneimittel genannt, die jeweils zu einem definierten Zeitpunkt und in einer genauen Menge an Tabletten, Zäpfchen, Tropfen bzw. per Injektion verabreicht werden. Stoffe, die dem Körper während eines Zeitintervalls zugeführt werden, heißen auf der Fieberkurve Infusion bzw. Perfusion und werden getrennt aufgeführt.

sorts: Medikament
opns: vasodilatanzien, beta_blocker, ca_antagonisten, diuretika,
pos_inotrope_subst, antiarrhythmika, antihypertensiva,
endokarditisprophylaxe, antikoagulantium: Text $\rightarrow$ Medikament

<u>DOSIS</u> =
sorts: Dosis
opns: med_dosis_ml/h, med_dosis_TabAnz, med_dosis_Tropfen: Wert $\rightarrow$ Dosis
perf_dosis_ml/h, perf_dosis_Tropfen: Wert $\rightarrow$ Dosis

<u>PERFUSIONEN</u> = actualize <u>LISTE</u> by <u>MEDIKATION</u>
using
sortnames Perfusion for D
Perfusionen for Liste(Perfusion)
opnames list_Perf for list: Perfusion $\rightarrow$ Perfusionen
keine_Perfusion for nil: $\rightarrow$ Perfusionen
keine_Perf for nodata: $\rightarrow$ Perfusion
zu_Perfusionen for radd: Perfusionen $\times$ Perfusion $\rightarrow$ Perfusionen
head_Perf for head: Perfusionen $\rightarrow$ Perfusion
search_Perf for search: Perfusion $\times$ Perfusionen $\rightarrow$ Perfusion
ladd_Perf for ladd: Perfusion $\times$ Perfusionen $\rightarrow$ Perfusionen
endact

<u>PERFUSION</u> = <u>MEDIKAMENT</u> +<u>DOSIS</u> +

sorts: Perfusion
opns: beginne_Perf, beende_Perf: $\rightarrow$ Zeitpunkt
keine_Perf: $\rightarrow$ Perfusion
mk_Perfusion: Medikament $\times$ Zeitpunkt^2 $\times$ Dosis $\rightarrow$ Perfusion
perfusion: Perfusion$\rightarrow$ FK_Daten
getPerfBeginn, getPerfEnde: Perfusion$\rightarrow$ Zeitpunkt
eqns: getPerfBeginn(mk_Perfusion(P, Z1, Z2)) = Z1;
getPerfEnde(mk_Perfusion(P, Z1, Z2)) = Z2;
getPerfBeginn(P) $\leq$ getPerfEnde(P) = Z2;

$\underline{\underline{\text{FIEBERKURVE}}} = \underline{\text{PATID}} + \underline{\text{VITALWERTE}} + \underline{\text{MEDIKATIONEN}} + \underline{\text{PERFUSIONEN}} + \underline{\text{INTENSIVSTATION\_WERTE}} + \underline{\text{DHZB\_NAB}} +$

Das vielseitigste Dokument der Patientenakte ist die Fieberkurve.
Basis ist eine Tabelle, deren Spalten für verschiedene Zeitpunkte während des Klinikaufenthaltes stehen und deren Zeilen zeit- und patientenbezogene Daten enthalten.
Die Fieberkurve wird als 6-Tupel mit folgenden Einträgen spezifiziert:
1. PatId: eindeutiger Identifikator der Fieberkurve, ermöglicht die korrekte Zuordnung anderer Dokumente, die sich auf denselben Patienten beziehen.
2. Vitalwerte: Täglich gemessen werden sogenannte Vitalwerte wie Puls, Blutdruck, Körpertemperatur; dazu können je nach Situation des Patienten weitere Werte kommen: auf der Intensivstation werden z.B. bisweilen stündlich der mittlere arterielle Druck (MAD) sowie der zentralvenöse Druck (ZVD) gemessen. Außerdem werden die täglichen Mengen und Ausprägungen an Ein-und Ausfuhr eingetragen.
3., 4. Medikation, Perfusion: Die nach Anweisungs des Arztes (Therapiebogen) verabreichten Medikamente dokumentiert das Pflegepersonal durch Eintragung von Name, Art, Konzentration der Arznei, sowie des genauen Zeitpunktes der Einnahme auf der Fieberkurve
5. Intensivstationwerte: Liegt der Patient/die Patientin auf der Intensivstation, so wird eine zusätzliche Blutuntersuchung gemacht und deren Meßwerte in eine Untersuchungstabelle der Fieberkurve eingetragen. Außerdem kommen dann Angaben über Operationen, die Diagnose und spezielle Verabreichungen hinzu.
6. NAB-Werte aus dem DHZB-NAB (auch in der PatAkte) dienen zur Kommunikation mit dem Labor. Der Arzt kreuzt auf dem Bogen Untersuchungen an, die er wünscht. Diese werden im Labor gemacht. Dort wird der DHZB-NAB ausgefüllt und geht zurück an die Station,wo die Aufkleber in die Fieberkurve geklebt werden .

sorts: Fieberkurve, FK_Ident
opns:
– *Fieberkurve initialisieren: noch keine Einträge*
keine_Vitalwerte : $\rightarrow$ Vitalwerte

keine_Intensivstat_werte: $\rightarrow$ Intensivstat_werte
keine_NAB_Werte: $\rightarrow$ NAB_Werte

init_FK: PatId $\times$ Datum $\rightarrow$ Fieberkurve
mk_FK_Ident: PatId $\times$ Datum $\rightarrow$ FK_Ident
fk: FK_Ident $\times$ Vitalwerte $\times$ Medikationen $\times$ Perfusionen $\times$
Intensivstat_werte $\times$ NAB_Werte $\rightarrow$ Fieberkurve

– *Werte aus der Fieberkurve lesen*
getpat: Fieberkurve $\rightarrow$ PatId
get_Ident: Fieberkurve $\rightarrow$ FK_Ident
get_Vitalwerte: Fieberkurve $\rightarrow$ Vitalwerte
get_Medikationen: Fieberkurve $\rightarrow$ Medikationen
get_Perfusionen: Fieberkurve $\rightarrow$ Perfusionen
get_Intensivstat_werte: Fieberkurve $\rightarrow$ Intensivstat_werte
get_NAB_Werte: Fieberkurve $\rightarrow$ NAB_Werte

– *Werte in die Fieberkurve eintragen*
ch_FK: FK $\times$ Vitalwerte $\rightarrow$ Fieberkurve
ch_FK: FK $\times$ Medikationen $\rightarrow$ Fieberkurve
ch_FK: FK $\times$ Perfusionen $\rightarrow$ Fieberkurve
ch_FK: FK $\times$ Intensivstat_werte $\rightarrow$ Fieberkurve
ch_FK: FK $\times$ NAB_Werte $\rightarrow$ Fieberkurve

eqns:

– *Fieberkurve initialisieren: noch keine Einträge*
init_FK(Id,Erstg) = fk(mk_FK_Ident(Id,Erstg), keine_Vitalwerte,
keine_Medikation, keine_Perfusion, keine_Intensivstat_werte,
keine_NAB_Werte)

– *Werte aus der Fieberkurve lesen*
getpat(init_FK(Id, D)) = Id;
getpat(fk(mk_FK_Ident(Id,D),V,M,P,Int,NAB)) = Id;
get_Ident(fk (Id,V,M,P,Int,NAB)) = Id;
get_Vitalwerte (fk (Id,V,M,P,Int,NAB)) = V;
get_Medikationen (fk (Id,V,M,P,Int,NAB)) = M;
get_Perfusionen (fk (Id,V,M,P,Int,NAB))= P;
get_Intensivstat_werte (fk (Id,V,M,P,Int,NAB)) =Int;
get_NAB_Werte(fk (Id,V,M,P,Int,NAB))= NAB;

– *Werte in die Fieberkurve eintragen: Vitalwerte*
ch_FK(FK,v(Puls,Zp,Id)) = fk (get_Ident(FK),
ch_Vitalw(get_Vitalwerte(FK),v(Puls,Zp)),
get_Medikationen (FK),
get_Perfusionen(FK),

```
        get_Intensivstat_werte (FK),
        get_NAB_Werte(FK))

ch_FK(FK,v(Temp,Zp,Id)) = fk (get_Ident(FK),
        ch_Vitalw(get_Vitalwerte(FK),v(Temp,Zp)),
        get_Medikationen (FK),
        get_Perfusionen(FK),
        get_Intensivstat_werte (FK),
        get_NAB_Werte(FK))

ch_FK(FK,v(BDsyst, BDdiast,Zp,Id)) = fk (get_Ident(FK),
        ch_Vitalw(get_Vitalwerte(FK),v(BDsyst,BDdiast,Zp)),
        get_Medikationen (FK),
        get_Perfusionen(FK),
        get_Intensivstat_werte (FK),
        get_NAB_Werte(FK))

ch_FK(FK,v(ZVD,Zp,Id)) = fk (get_Ident(FK),
        ch_Vitalw(get_Vitalwerte(FK),v(ZVD,Zp)),
        get_Medikationen (FK),
        get_Perfusionen(FK),
        get_Intensivstat_werte (FK),
        get_NAB_Werte(FK))

ch_FK(FK,v(MAD,Zp,Id)) = fk (get_Ident(FK),
        ch_Vitalw(get_Vitalwerte(FK),v(MAD,Zp)),
        get_Medikationen (FK),
        get_Perfusionen(FK),
        get_Intensivstat_werte (FK),
        get_NAB_Werte(FK))

ch_FK(FK,v(Einf,Ausf,Zp,Id)) = fk (get_Ident(FK),
        ch_Vitalw(get_Vitalwerte(FK),v(Einf,Ausf,Zp)),
        get_Medikationen (FK),
        get_Perfusionen(FK),
        get_Intensivstat_werte (FK),
        get_NAB_Werte(FK))

        – Werte in die Fieberkurve eintragen: Medikationen
ch_FK(FK,Med) = fk (get_Ident(FK),
        get_Vitalwerte(FK),
        ch_Medikationen (get_Medikationen(FK), Med),
        get_Perfusion(FK),
        get_Intensivstat_werte (FK),
        get_NAB_Werte(FK))
```

– *Werte in die Fieberkurve eintragen: Perfusionen*

```
ch_FK(FK,Perf) = fk (get_Ident(FK),
          get_Vitalwerte(FK),
          get_Medikationen (FK),
          ch_Perfusionen(get_Perfusionen(FK), Perf),
          get_Intensivstat_werte (FK),
          get_NAB_Werte(FK))
```

– *Werte in die Fieberkurve eintragen: Intensivstationswerte*

```
ch_FK(FK,Intensiv) = fk (get_Ident(FK),
          get_Vitalwerte(FK),
          get_Medikationen (FK),
          get_Perfusionen(FK),
          ch_Intensivstat_werte (get_Intensivstat_werte (FK),Intensiv)
          get_NAB_Werte(FK))
```

– *Werte in die Fieberkurve eintragen: NAB-Werte*

```
ch_FK(FK,NAB) = fk (get_Ident(FK),
          get_Vitalwerte(FK),
          get_Medikationen (FK),
          ch_Perfusionen(get_Perfusionen(FK), Perf),
          get_Intensivstat_werte (FK),
          ch_NAB_Werte (get_NAB_Werte (FK),NAB)
```

VITALWERTE =

Der Eintrag „Vitalwerte“ in der Fieberkurve wird als 6-Tupel folgender Wertetabellen (Listen) spezifiziert:

1. Die Puls_Liste enthält Pulsschlag-Werte in der Einheit 1/min,
2. Die Temp_Liste enthält Körpertemperaturmessungen in °C ,
3. Die BD_Liste enthält den systol. und den diastol. Blutdruck in mmHG.
4. Die ZVD-Liste enthält zentralvenöse Druckwerte,
5. die MAD-Liste mittlere arterielle Druckwerte in mmHG;
6. die E-/A-Liste enthält Angaben über Ein- und Ausfuhr in ml.

– *Puls_Liste*

```
actualize LISTE by VITALWERT
using
     sortnames Vitalwert for D
               Puls_Liste for Liste(Vitalwert)
     opnames list_Puls for list: Vitalwert → Puls_Liste
               kein_Pulswert for nil: Puls_Liste
               append_Puls for radd: Puls_Liste × Vitalwert → Puls_Liste
endact +
```

– *Temp_Liste*

```
actualize LISTE by VITALWERT
using
        sortnames Vitalwert for D
                  Temp_Liste for Liste(Vitalwert)
        opnames list_Temp for list: Vitalwert → Temp_Liste
                  kein_Tempwert for nil: Temp_Liste
                  append_Temp for radd: Temp_Liste × Vitalwert → Temp_Liste
endact +
```

– *BD_Liste*

```
actualize LISTE by VITALWERT
using
        sortnames Vitalwert for D
                  BD_Liste for Liste(Vitalwert )
        opnames list_BD for list: Vitalwert → BD_Liste
                  kein_BD for nil: BD_Liste
                  append_BD for radd: BD_Liste × Vitalwert → BD_Liste
endact +
```

– *ZVD_Liste*

```
actualize LISTE by VITALWERT
using
        sortnames Vitalwert for D
                  ZVD_Liste for Liste(Vitalwert )
        opnames list_ZVD for list: Vitalwert → ZVD_Liste
                  kein_ZVD for nil: ZVD_Liste
                  append_ZVD for radd: ZVD_Liste × Vitalwert → ZVD_Liste
endact +
```

– *MAD_Liste*

```
actualize LISTE by VITALWERT
using
        sortnames Vitalwert for D
                  MAD_Liste for Liste(Vitalwert )
        opnames list_MAD for list: Vitalwert → MAD_Liste
                  kein_MAD for nil: MAD_Liste
                  append_MAD for radd: MAD_Liste × Vitalwert → MAD_Liste
endact +
```

– *E-/A_Liste*

```
actualize LISTE by VITALWERT
using
        sortnames Vitalwert for D
```

E-/A_Liste for Liste(Vitalwert)
opnames list_E-/A for list: Vitalwert $\rightarrow$ E-/A_Liste
kein_E-/A for nil: E-/A_Liste
append_E-/A for radd: E-/A_Liste $\times$ Vitalwert $\rightarrow$ E-/A_Liste
endact +

sorts: Vitalwerte
opns:
– *Vitalwerte-Konstruktor.*
vitalwerte: Puls_Liste $\times$ Temp_Liste $\times$ BD_Liste $\times$
ZVD_Liste $\times$ MAD_Liste $\times$ E-/A_Liste $\rightarrow$ Vitalwerte

– *auf bestimmte Vitalwertetabellen zugreifen*
get_Puls_Liste: Vitalwerte $\rightarrow$ Puls_Liste
get_Temp_Liste: Vitalwerte $\rightarrow$ Temp_Liste
get_BD_Liste: Vitalwerte $\rightarrow$ BD_Liste
get_ZVD_Liste: Vitalwerte $\rightarrow$ ZVD_Liste
get_MAD_Liste: Vitalwerte $\rightarrow$ MAD_Liste
get_E-/A_Liste: Vitalwerte $\rightarrow$ E-/A_Liste

– *Vitalwerte hinzufügen (Messungen eintragen)*
ch_Vitalwerte: Vitalwerte $\times$ Vitalwert $\rightarrow$ Vitalwerte

eqns:
– *auf bestimmte Vitalwertetabellen zugreifen*
get_Puls_Liste(vitalwerte(P,T,B,Z,M,EA)) = P;
get_Temp_Liste(vitalwerte(P,T,B,Z,M,EA)) = T;
get_BD_Liste(vitalwerte(P,T,B,Z,M,EA)) = B;
get_ZVD_Liste(vitalwerte(P,T,B,Z,M,EA)) = Z;
get_MAD_Liste(vitalwerte(P,T,B,Z,M,EA)) = M;
get_E-/A_Liste(vitalwerte(P,T,B,Z,M,EA)) = EA;

– *Vitalwerte hinzufügen (Messungen eintragen)*
ch_Vitalwerte(VW,v(Puls,Zp)) =
vitalwerte(append_Puls(get_Puls_Liste(VW), v(Puls,Zp)),
get_Temp_Liste(VW),
get_BD_Liste(VW),
get_ZVD_Liste(VW),
get_MAD_Liste(VW)),
get_E-/A_Liste(VW)),

ch_Vitalwerte(VW,v(Temp,Zp)) =
vitalwerte(get_Puls_Liste(VW),
append_Temp(get_Temp_Liste(VW), v(Temp,Zp)),
get_BD_Liste(VW),

```
              get_ZVD_Liste(VW),
              get_MAD_Liste(VW))
              get_E-/A_Liste(VW)),

    ch_Vitalwerte(VW,v(BD_syst,BD_diast,Zp)) =
    vitalwerte(get_Puls_Liste(VW),
              get_Temp_Liste(VW),
              append_BD(get_BD_Liste(VW), v(BD_syst,BD_diast,Zp)),
              get_ZVD_Liste(VW),
              get_MAD_Liste(VW))
              get_E-/A_Liste(VW)),

    ch_Vitalwerte(VW,v(ZVD, Zp)) =
    vitalwerte(get_Puls_Liste(VW),
              get_Temp_Liste(VW),
              get_BD_Liste(VW),
              append_ZVD(get_ZVD_Liste(VW), v(ZVD, Zp)),
              get_MAD_Liste(VW))
              get_E-/A_Liste(VW)),

    ch_Vitalwerte(VW,v(MAD,Zp)) =
    vitalwerte(get_Puls_Liste(VW),
              get_Temp_Liste(VW),
              get_BD_Liste(VW),
              get_ZVD_Liste(VW),
              append_MAD(get_MAD_Liste(VW), v(MAD,Zp) ),
              get_E-/A_Liste(VW)),

    ch_Vitalwerte(VW,v(Einf, Ausf, Zp)) =
    vitalwerte(get_Puls_Liste(VW),
              get_Temp_Liste(VW),
              get_BD_Liste(VW),
              get_ZVD_Liste(VW),
              get_MAD_Liste(VW)),
              append_E-/A(get_E-/A_Liste(VW), v(Einf, Ausf, Zp) ),
```

<u>VITALWERT</u> = <u>EIN-/AUSFUHR</u>+<u>PATID</u>+

sorts: Puls_1/min, Temperatur_°C, Blutdruck_syst_mmHG,
Blutdruck_diast_mmHG, ZVD_mmHG, MAD_mmHG, Vitalwert

opns:

– Vitalwerte sind polymorph spezifiziert:
– Vitalwerte haben nach dem Messen die Form (Wert, Meßzeitpunkt, PatId),
– da die Zuordnung zur Fieberkurve noch erfolgen muß

kein_Vitalwert: → Vitalwert
v: Puls_1/min × Zeitpunkt × PatId → Vitalwert
v: Temperatur_°C × Zeitpunkt × PatId → Vitalwert
v: Blutdruck_syst_mmHG × Blutdruck_diast_mmHG ×
    Zeitpunkt × PatId → Vitalwert
v: ZVD_mmHG × Zeitpunkt × PatId → Vitalwert
v: MAD_mmHG × Zeitpunkt × PatId → Vitalwert
v: Einfuhr × Ausfuhr × Zeitpunkt × PatId → Vitalwert

– *Vitalwerte haben für den Eintrag in die FK die Form (Wert, Meßzeitpunkt),*
– *da die Zuordnung mithilfe der PatId schon erfolgt ist*

v: Puls_1/min × Zeitpunkt → Vitalwert
v: Temperatur_°C × Zeitpunkt → Vitalwert
v: Blutdruck_syst_mmHG × Blutdruck_diast_mmHG ×
    Zeitpunkt → Vitalwert
v: ZVD_mmHG × Zeitpunkt → Vitalwert
v: MAD_mmHG × Zeitpunkt → Vitalwert
v: Einfuhr × Ausfuhr × Zeitpunkt → Vitalwert

EIN-/AUSFUHR =

Sämtliche Stoffe, die dem Patienten in relevanter Größenordnung zugeführt werden (Nahrung, Infusionen) bezeichnet man im medizinischen Kontext als Einfuhr. Dementsprechend werden alle Ausscheidungen (Urin, Stuhl, Erbrechen, sowie dem Körper mittels Drainage entzogene Stoffe) als Ausfuhr bezeichnet. Relevant sind diese Werte vor allem, wenn große Mengen eines Medikaments verabreicht werden und der untersuchten Ausfuhr entnommen werden kann, in welchem Verhältnis Absorption und Ausscheidung des Wirkstoffes stehen.

sorts: Einfuhr, Ausfuhr,
opns: einf_infusion_ml: → Wert
einf_blutplasma_ml: → Wert
einf_peros_ml: → Wert
ausf_drainagen_ml: → Wert
ausf_urin_ml: → Wert
ausf_stuhl_ml: → Wert
ausf_magensonden_ml: → Wert
einf_infusion_ml: → Wert
einf_infusion_ml: → Wert
einf_infusion_ml: → Wert
einf_infusion_ml: → Wert

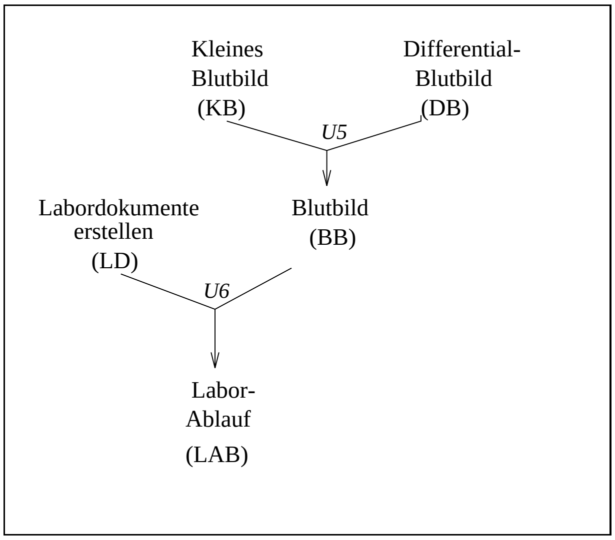
mk_Einfuhr: Wert^3 → Einfuhr

```
            mk_Ausfuhr: Wert^4 → Ausfuhr


INTENSIVSTATIONWERTE =
        actualize LISTE by INTENSIVSTATION
        using
            sortnames Intensivstation for D
                      Intensivstat_werte for Liste(Intensivstation )
            opnames list_Intens for list: Intensivstation → Intensivstat_werte
                      keine_Intens for nil: Intensivstationwerte
                      ch_Intensivstat_werte for radd: Intensivstat_werte ×
                      Intensivstation → Intensivstat_werte
        endact


INTENSIVSTATION = UNTERSUCHUNG+

                      – Untersuchungstabelle
        actualize LISTE by UNTERSUCHUNG
        using
            sortnames Untersuchung for D
                      Unter_Liste for Liste(Untersuchung )
            opnames list_Unter for list: Untersuchung → Unter_Liste
                      keine_Unter for nil: Unter_Liste
                      append_Unter for radd: Unter_Liste × Untersuchung → Unter_Liste
        endact +

        sorts:  Diagnose, Operation, Besonderes, Verabreichung, Untersuch_Tab,
                Unterschrift, Gewicht_kg, Intensivstation
        opns:   opdatum: → Datum
                diagnose: Text × Datum → Diagnose
                operation: Text × Datum → Operation
                besonderes: Text × Datum → Besonderes
                verabreichung: Text × Datum → Verabreichung
                unterschrift: → Unterschrift
                mk_Untersuch_Tab: Unter_Liste × Gewicht_kg × Unterschrift → Untersuch_Tab
                intensivstation: Diagnose × Operation × Besonderes ×
                      Verabreichung × Untersuch_Tab → Intensivstation


UNTERSUCHUNG =
        sorts:  Untersuchung
        opns:   po2,pco2,ph2,hco3,be,sat, ..., fib, f_ii, f_v: → Wert
                mk_Untersuchung: Wert^25 × Zeitpunkt → Untersuchung
```

3.2.3 Labor-Ablauf

Wir beschränken uns an dieser Stelle auf den Labor-Bereich der Blutuntersuchung (nicht Harnchemie, Serum-Chemie, ...), da der Ablauf in den anderen Labors analog ist. Verwendet wird das von der Station geschickte Blut des Patienten; die Blutentnahme wurde im Netz Angeordnete Behandlung (AB) modelliert.

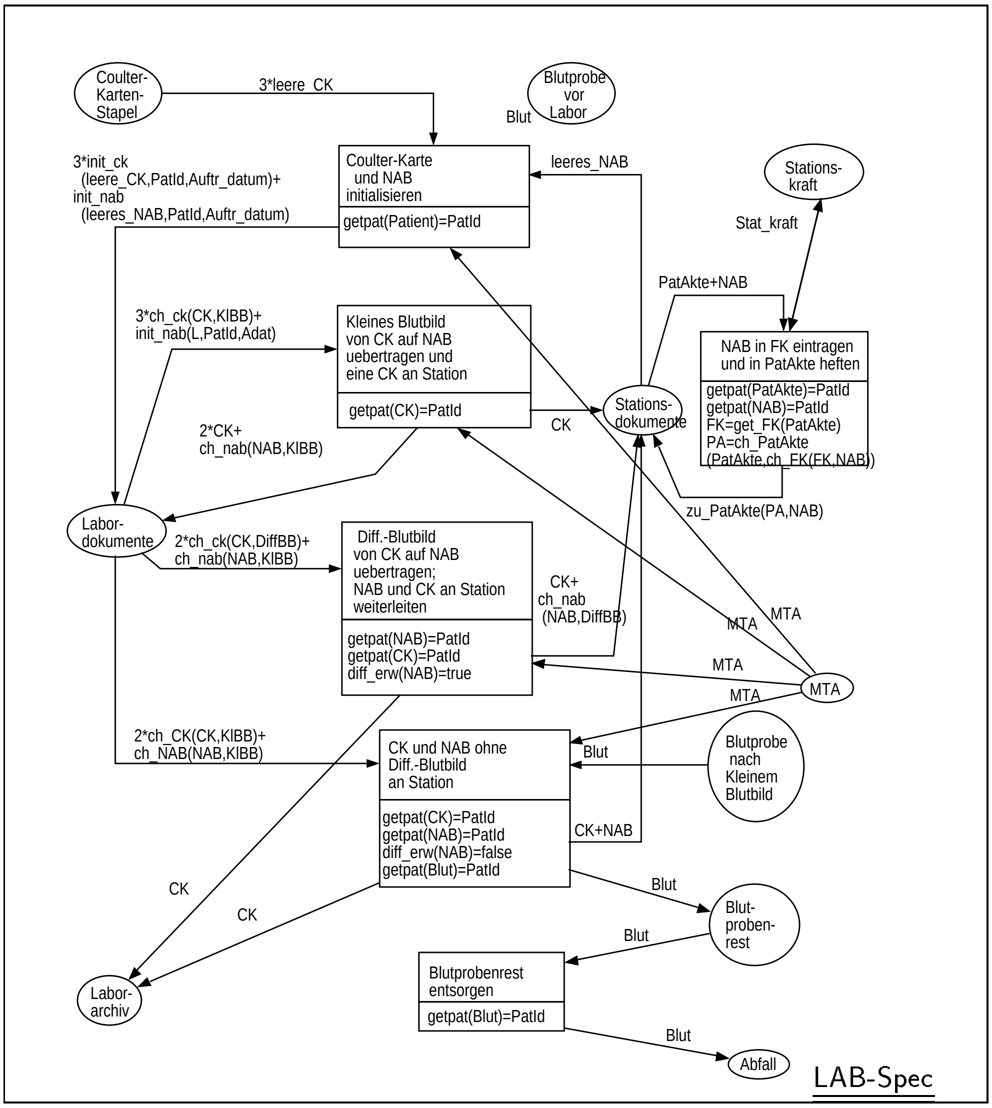
Folgende Netze gehören zum Bereich $Labor - Ablauf$:



Labor-Ablauf: Strukturierung der Netze

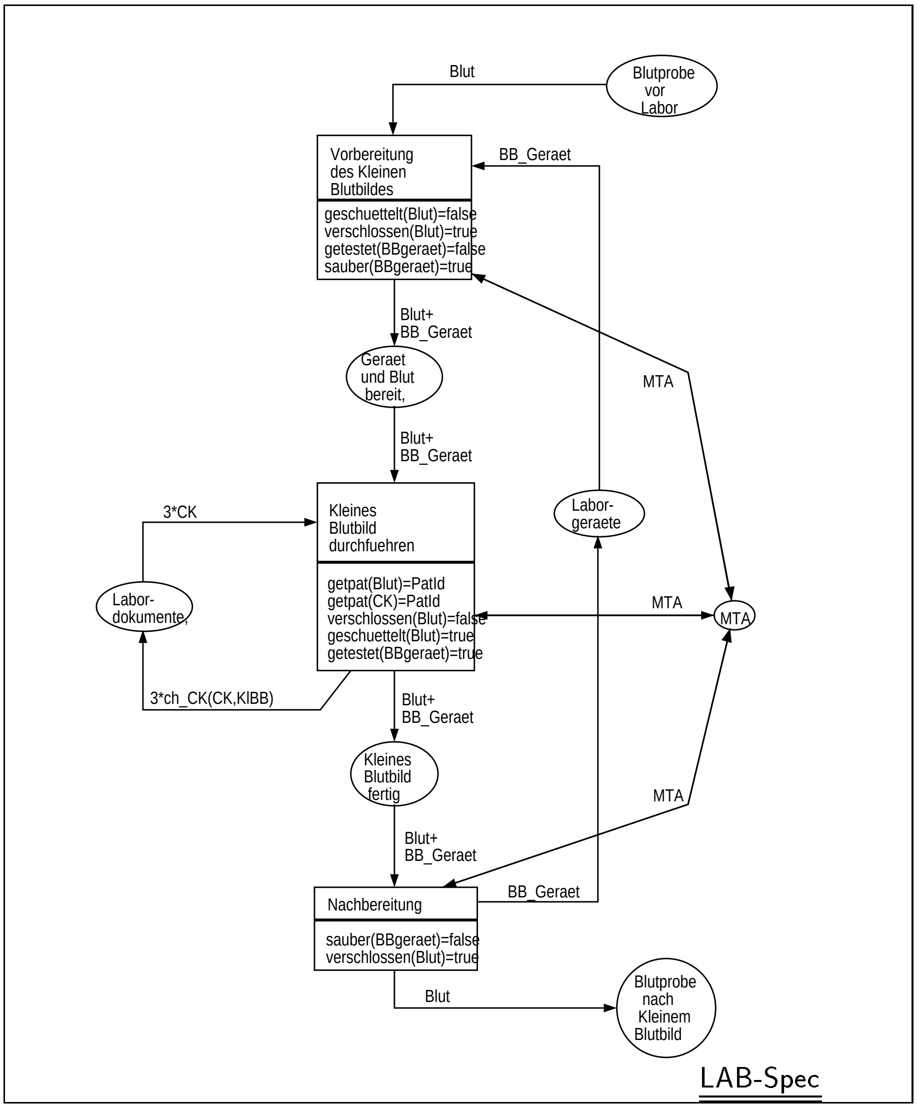
Wir beschränken uns an dieser Stelle auf den Labor-Bereich der Blutuntersuchung (nicht Harnchemie, Serum-Chemie, ...), da der Ablauf in den anderen Labors analog ist. Verwendet wird das von der Station geschickte Blut des Patienten; die Blutentnahme wurde im Netz Angeordnete Behandlung (AB) modelliert.

Bei der Blutuntersuchung im Labor wird ein Blutbild-Gerät (Coulter Counter) verwendet, das die ermittelten Daten automatisch auf eine spezielle Karte mit mehreren Durchschlägen ausdruckt, die Coulter-Karte. Von dort werden die Daten von Hand auf den Notfall-Labor- und Anforderungsbogen (NAB) übertragen. Im Netz (LD) werden diese Karte und der NAB mit der Patienten-Id und dem Auftragsdatum versehen. Außerdem werden die Daten nach abgeschlossener Laboruntersuchung von der Coulter-Karte auf den NAB übertragen und an die Station zurückgeschickt. Ein Konflikt zwischen den beiden Transitionen *Diff.-Blutbild von CK auf NAB übertragen; NAB und CK an Station weiterleiten* und *CK und NAB ohne Diff.-Blutbild an Station weiterleiten* ist beabsichtigt und wird über das Prädikat $diff\_erw(NAB)$ gesteuert: Ist die Bedingung $diff\_erw(NAB) = true$, so wird sichergestellt, daß das Differential-Blutbild nur dann auf den NAB übertragen wird, wenn es auf dem NAB auch angeordnet worden war. Ist dies nicht der Fall, d.h. gilt $diff\_erw(NAB) = false$, so schaltet die andere Transition, so daß der NAB ohne Diff.-Blutbild an die Station weitergeleitet wird. Dort werden die NAB-Werte in die Fieberkurve übertragen und der NAB selbst in die entsprechende Patientenakte geheftet.



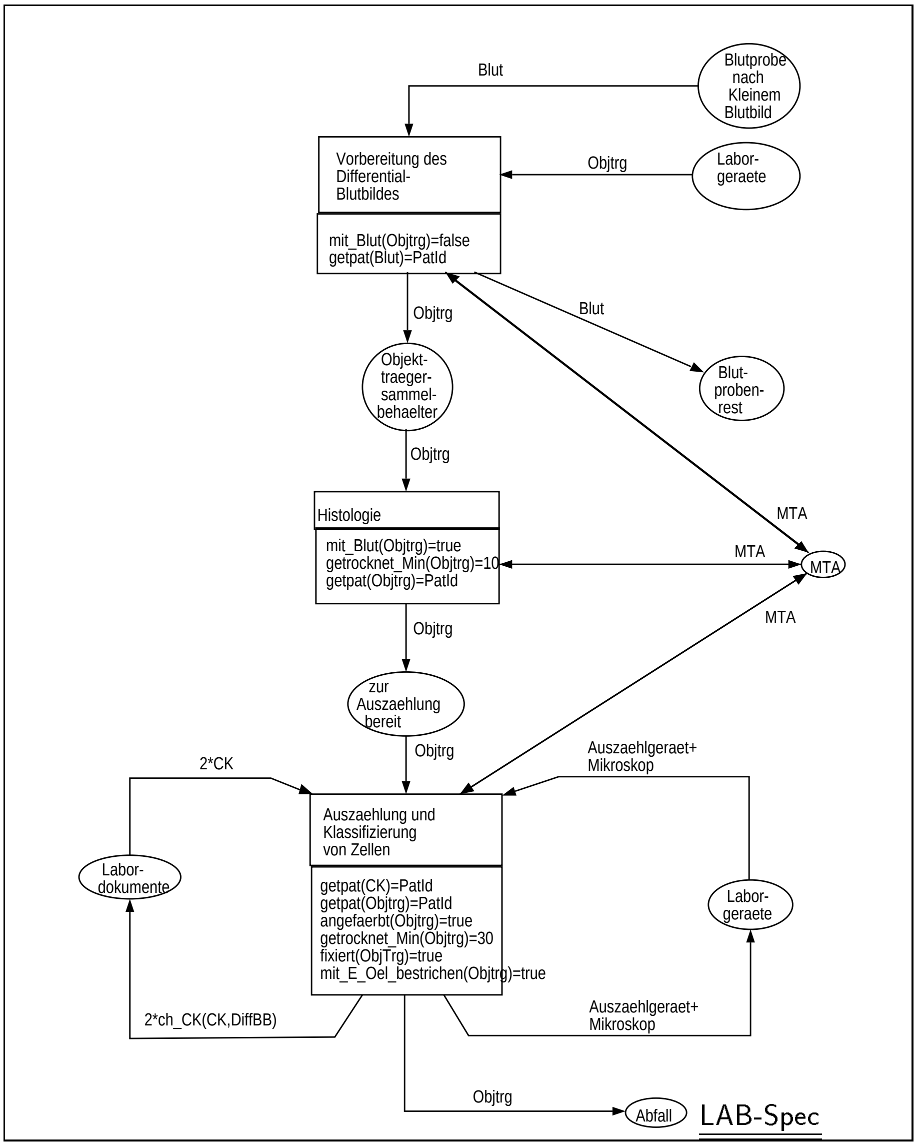
LD: Labordokumente erstellen

Im Netz (KB) wird der Umgang mit Blutprobe und Blutbildgerät vor und nach der Messung der Werte zum Kleinen Blutbild beschrieben. Die resultierenden Daten werden auf eine Coulter-Karte (CK) mit zwei Durchschlägen gedruckt.



KB: Kleines Blutbild

Der im Netz (DB) dargestellte Ablauf umfaßt die Zellenuntersuchung (Differential-Blutbild) anhand der Blutprobe des Patienten. Dabei wird der korrekte Umgang mit dem die Probe enthaltenden Objektträger als Bedingung für den ordnungsgemäßen Ablauf der Untersuchung modelliert. Die ermittelten Werte befinden sich nach Abschluß der Untersuchung auf der CK und einem Durchschlag.



DB: Differential-Blutbild

Wir erhalten nun das Netz Labor-Ablauf durch Anwendung der Operation *Union* auf die beiden Netze KB und DB, wobei das Schnittstellennetz U5 die Stellen *Blutprobe nach Kleinem Blutbild, MTA* und *Labordokumente* enthält. Das resultierende Netz BB wird wiederum über eine *Union* mit dem NetzLD zum Netz Labor-Ablauf vereinigt. Das Schnittstellennetz U6 enthält dabei die Stellen *Labordokumente, Blutprobe vor Labor, MTA, Blutprobenrest, Abfall* und *Blutprobe nach Kleinem Blutbild.*

Es folgt die Spezifikation LAB-Spec für den Labor-Bereich:

LAB-Spec: Spezifikation zum Netz Labor-Ablauf

LAB-Spec = DHZB-NAB $+_{BLUTBILD}$ COULTER-KARTE + BLUT +
LABORGERÄTE + PERSONAL

Wünscht der Arzt/die Ärztin eine Untersuchung des venösen Blutes des Patienten/der Patientin, so teilt er/sie dies auf dem Anordnungsbogen dem Pflegepersonal mit. Dieses entnimmt dem Patienten Blut, füllt einen Notfall-labor-Anforderungsbogen mit der Angabe den gewünschten Analysen aus und schickt alles gemeinsam zum Laborverteiler. Von dort geht es an die verschiedenen Abteilungen weiter (Hämatologie, Serum-Chemie, Harn-Chemie,...)
Wir haben uns bei der Spezifikation des Laborbereiches auf das Blutlabor beschränkt, da die anderen Labore im Prinzip ähnlich funktionieren und die Funktion des NAB im Blutlabor deutlich genug wird.
Das Labor trägt die gemessenen Werte in den DHZB-Notfallabor-Anforderungs- und Befundbogen (NAB) ein. Dieser wird nach Abschluß der Untersuchungen an die Station zurückgeschickt. Das Pflegepersonal überträgt die Analysewerte in die Fieberkurve.

DHZB-NAB = PATID + BLUTBILD +

Der DHZB-NAB ist das zentrale Dokument im Laborbereich.
Der Datentyp NAB wird hier, da nur aufs Blutlabor beschränkt, als Tupel (Kleines Blutbild, Differential-Blutbild) der gemessenen Blutbilddaten modelliert und entspricht damit genau dem Informationsgehalt der Coulter-Karte. Normalerweise kommen zum NAB noch Werte der Gerinnungsfähigkeit, Serum-Chemie und Elektrolytbestimmung hinzu.

sorts: NAB
opns: leeres_NAB: → NAB
diff_erw: NAB → Bool

–NAB initialisieren
nab: PatId × Datum × Kl_Blutbild × Diff_Blutbild → NAB
init_nab: NAB × PatId × Datum → NAB

–NAB identifizieren
getpat: NAB → PatId
get_Auftr_datum: NAB → Datum
get_kl_BB: NAB → Kl_Blutbild
get_diff_BB: NAB → Diff_Blutbild

–Blutbild auf NAB eintragen
ch_nab: NAB × Kl_Blutbild → NAB

ch_nab: NAB × Kl_Blutbild → NAB

eqns:

–NAB initialisieren

init_nab(N,Id,D)=nab(Id,D,kein_kl_BB, kein_Diff_BB);

–NAB identifizieren

getpat(init_nab(N,Id,D)) = Id;
getpat(ch_nab(N,BB)) = getpat(N);
get_Auftr_datum(nab(Id,D,Kl,Diff)) = D;
get_kl_BB(nab(Id,D,Kl,Diff)) = Kl;
get_diff_BB(nab(Id,D,Kl,Diff)) = Diff;

–Blutbild auf NAB eintragen

ch_nab(NAB,KBB) = nab(getpat(NAB),get_Auftr_datum(NAB),
KBB,get_diff_BB(NAB));
ch_nab(NAB,DBB) = nab(getpat(NAB),get_Auftr_datum(NAB),
get_kl_BB(NAB),DBB);

BLUTBILD =

sorts: Blutwert, Kl_Blutbild, Diff_Blutbild

opns: leukz, ery, hb, hk, mcv, mch, mchc, thromb, retic_z, stab, segm, lymph,
atpy_lym, eos, baso, mono, plansmaz, myelobl, promyeloz,
metayelo, anisocyt, poikilozyt: → Blutwert
kein_kl_BB: → Kl_Blutbild
kein_Diff_BB: → Diff_Blutbild
klBB: Blutwert^{22} → Kl_Blutbild
diffBB: Blutwert^{22} → Diff_Blutbild

COULTER-KARTE = BLUTBILD +

Die Coulter-Karte wird vom Blut-Auszählgerät (Coulter Counter) automatisch mit dem Kleinem Blutbild oder dem Differential-Blutbild bedruckt. Die Daten werden auf den NAB übertragen.

sorts: CK

opns: leere_CK: → CK

–CK initialisieren

ck: PatId × Datum × Kl_Blutbild × Diff_Blutbild → CK
init_ck: CK × PatId × Datum → CK

–CK identifizieren

getpat: CK → PatId
get_Auftr_datum: CK → Datum
get_kl_BB: CK → Kl_Blutbild
get_diff_BB: CK → Diff_Blutbild

–Blutbild auf CK eintragen
ch_ck: CK × Kl_Blutbild → CK
ch_ck: CK × Kl_Blutbild → CK

eqns:

–CK initialisieren
init_ck(N,Id,D)=ck(Id,D,kein_kl_BB, kein_Diff_BB);

–CK identifizieren
getpat(init_ck(N,Id,D)) = Id;
getpat(ch_ck(N,BB)) = getpat(N);
get_Auftr_datum(ck(Id,D,Kl,Diff)) = D;
get_kl_BB(ck(Id,D,Kl,Diff)) = Kl;
get_diff_BB(ck(Id,D,Kl,Diff)) = Diff;

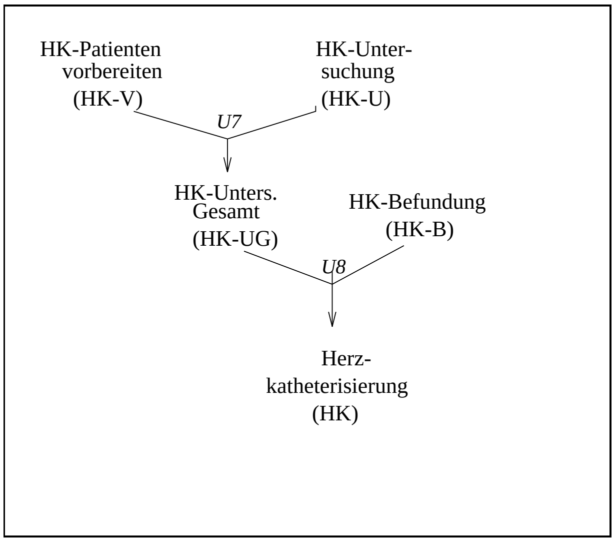
–Blutbild auf CK eintragen
ch_ck(CK,KBB) = ck(getpat(CK),get_Auftr_datum(CK),
KBB,get_diff_BB(CK));
ch_ck(CK,DBB) = ck(getpat(CK),get_Auftr_datum(CK),
get_kl_BB(CK),DBB);

BLUT = PATID +
sorts: Blut
opns: blut: PatId → Blut
getpat: Blut → PatId
geschüttelt, verschlossen: Blut → Bool
eqns: getpat(blut(Id)) = Id;

LABORGERÄTE =
sorts: BB_Gerät, Objekträger, Auszählgerät, Mikroskop
opns: getestet, sauber: BB_Gerät → Bool
mit_Blut, angefärbt, fixiert, mit_Emulsionsöl: Objektträger → Bool
getrocknet_Min: Objektträger → Nat

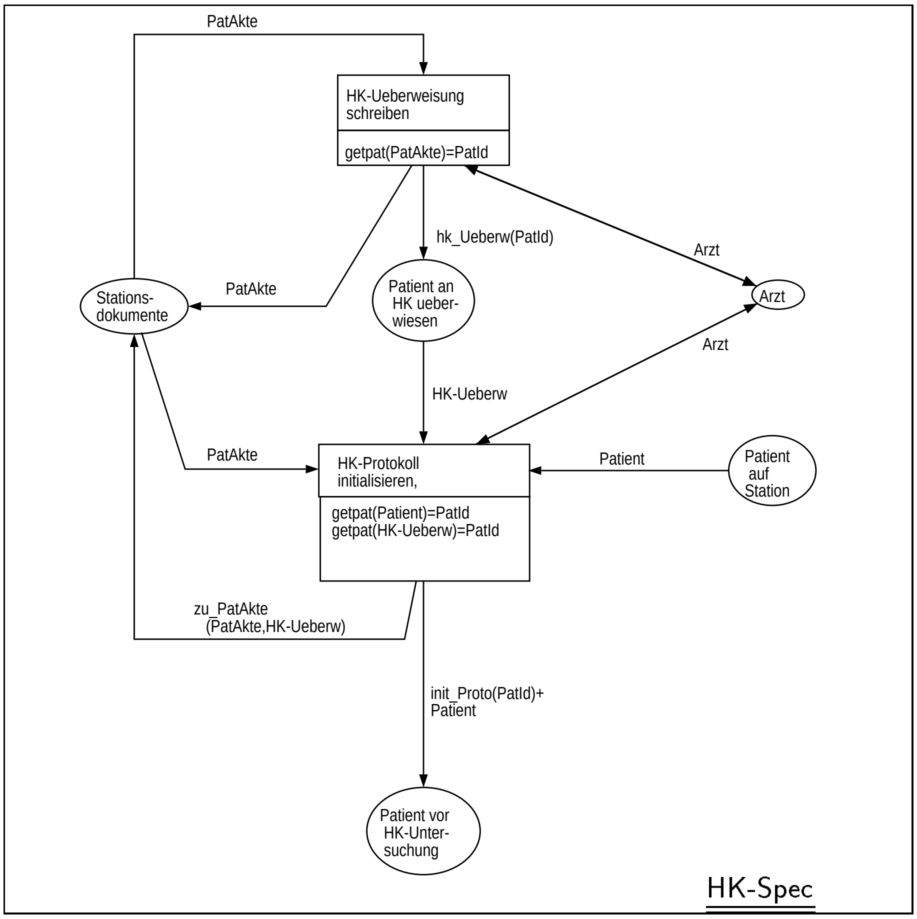
3.2.4 Herzkatheterisierung

Die Herzkatheterisierung umfaßt die Untersuchung des Patienten mit einem Herzkatheter. Wir legen folgende Strukturierung der Netze zum Thema Herzkatheterisierung zugrunde:



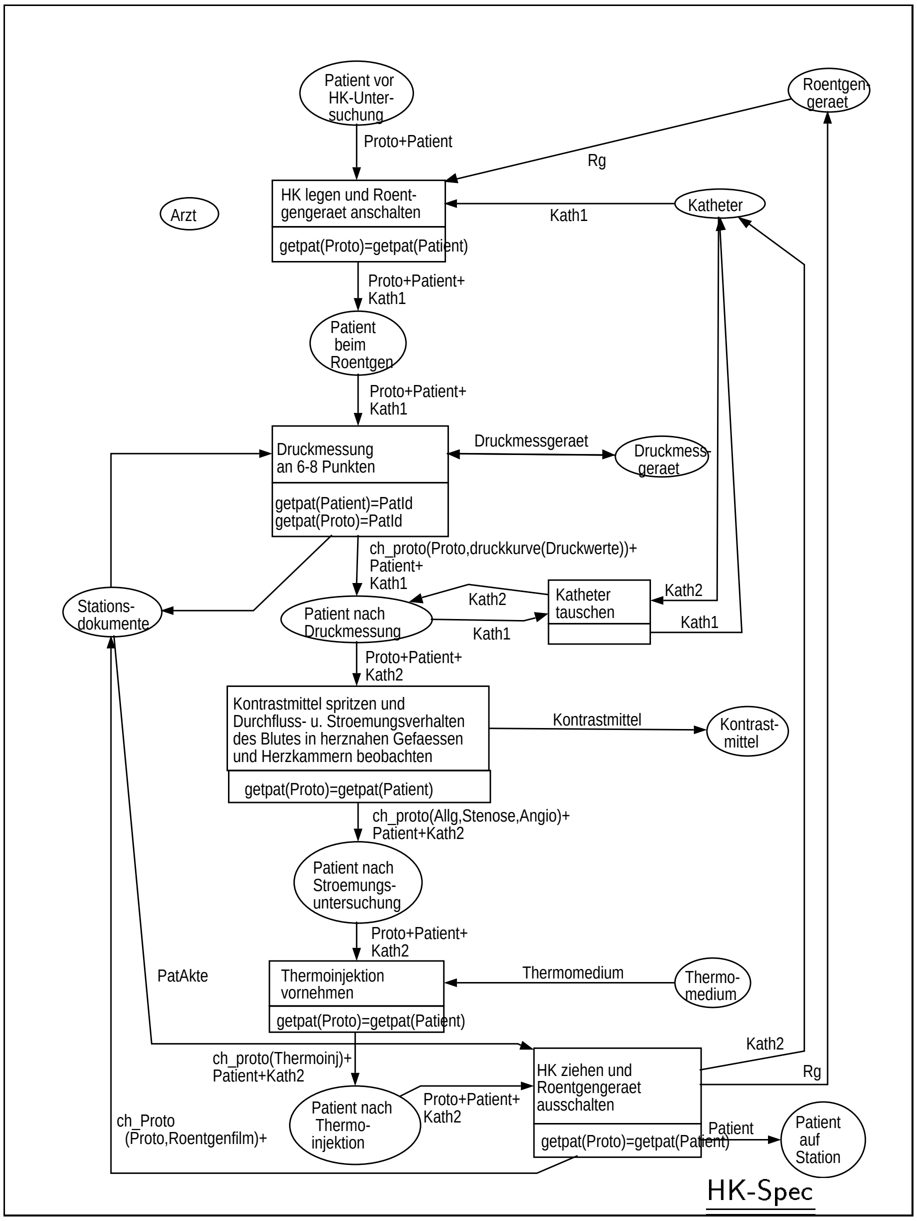
Herzkatheterisierung: Strukturierung der Netze

Soll ein Patient mit einem Herzkatheter untersucht/behandelt werden, so muß eine HK-Überweisung des behandelnden Arztes vorliegen. Ist dies der Fall, wird ein HK-Untersuchungsprotokoll für den Patienten angelegt, in das im Laufe der HK-Untersuchung alle Meßwerte aufgenommen werden. Netz (HK-V) zeigt die Vorbereitungen zur HK-Untersuchung.



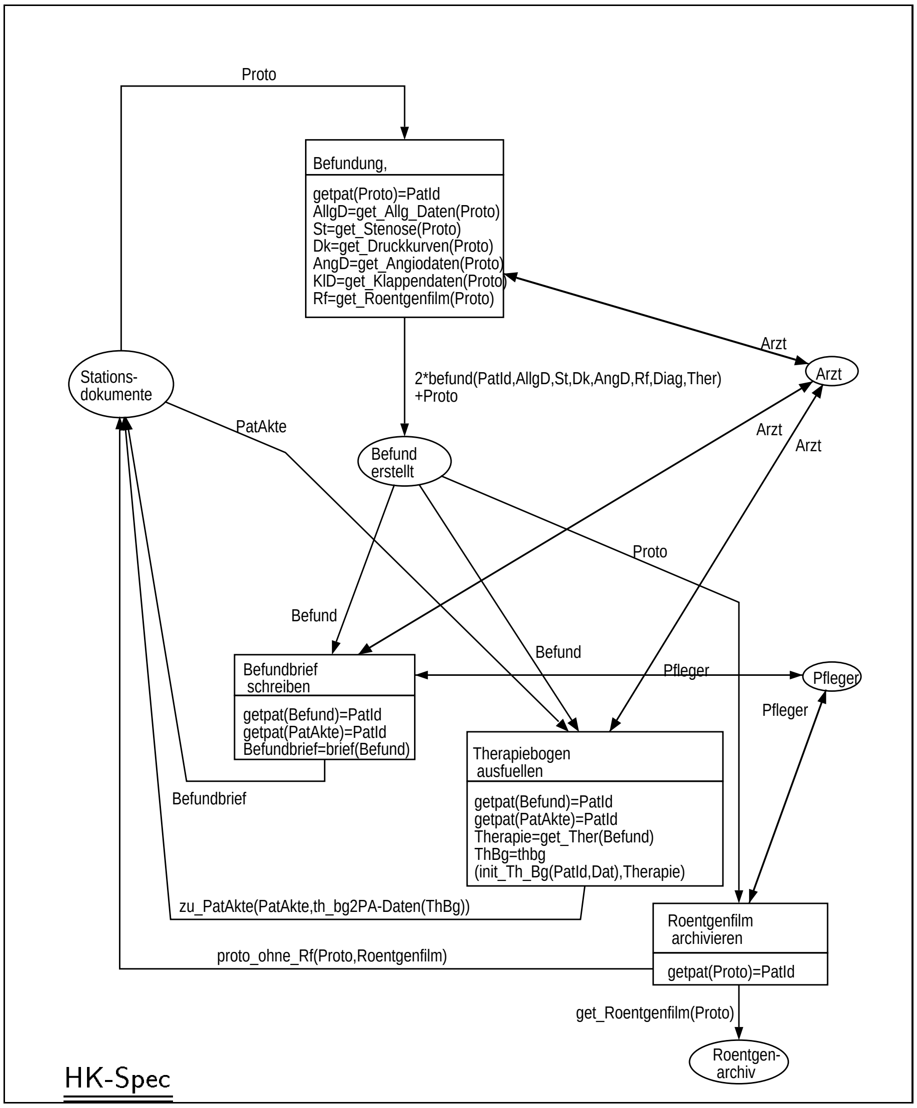
HK-V: HK-Patienten vorbereiten

Die meisten Herzkrankheiten sind Blutdurchflußstörungen im Herzen oder in der Herzumgebung. Die im Netz (HK-U) spezifizierte Herzkatheter-Untersuchung erlaubt eine direkte Untersuchung der Herzgefäße mit einem Katheter, an dessen Spitze Meßinstrumente befestigt sind. So werden z.B. Blutdruck und Strömungsgeschwindigkeit an bestimmten Stellen gemessen, um Gefäßverengungen feststellen zu können. Auch die Injektion eines Kontrastmittels bzw. eines Thermomediums zur Röntgenüberwachung eines bestimmten Blutstroms ist ein wichtiger Teil der Untersuchung. Sämtliche Meßwerte werden ins Untersuchungsprotokoll aufgenommen. Das Protokoll bildet zusammen mit den Röntgenbildern (Angiogrammen) die Grundlage der Diagnose.



HK-U: HK-Untersuchung

Bei der Ermittlung des Befundes (Netz HK-B) werden alle Daten des HK-Untersuchungsprotokolls zu Rate gezogen. Der ermittelte Befund wird in Form eines Befundbriefes an die Station geschickt. Außerdem wird aufgrund des Befundes ein Therapiebogen ausgefüllt und der Patientenakte des untersuchten Patienten hinzugefügt. Nach abgeschlossener Befundung wird der Röntgenfilm in einem speziellen Röntgenarchiv abgelegt.



HK-B: HK-Befundung

Wir erhalten nun das Netz Herzkatheterisierung durch Anwendung der Operation *Union* auf die beiden Netze HK-V und HK-U, wobei das Schnittstellennetz U7 die Stellen *Patient vor HK-Untersuchung, Patient auf Station, Arzt* und Stationsdokumente enthält. Das resultierende Netz HK-UG wird wiederum über eine *Union* mit dem NetzHK-B zum Netz Herzkatheterisierung vereinigt. Das Schnittstellennetz U6 enthält dabei die Stellen *Arzt* und *Stationsdokumente*.

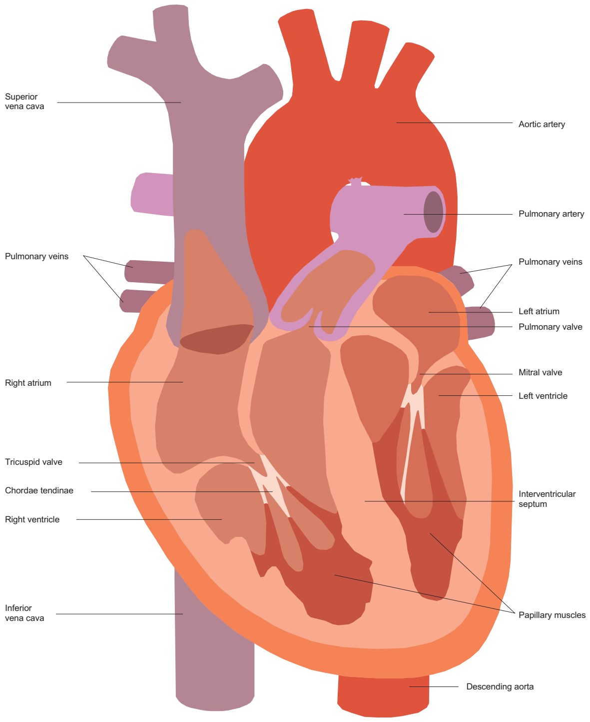


Abbildung 3.5: Schematische Herzdarstellung

Es folgt die dem Bereich Herzkatheterisierung zugrundeliegende Spezifikation HK-Spec. Zum besseren Verständnis der dort verwendeten Fachausdrücke für die verschiedenen Herzkammern, -klappen und -gefäße zeigt Abb. 3.5 eine schematische Darstellung des Herzens.

HK-Spec: Spezifikation zum Netz Herzkatheterisierung

$$\underline{\underline{\mathsf{HK\text{-}Spec}}} = \underline{\underline{\mathsf{PROTOKOLL}}} + \underline{\underline{\mathsf{HK\_BEFUNDUNG}}} + \underline{\underline{\mathsf{HK\text{-}GERÄTE}}} + \underline{\underline{\mathsf{HK\text{-}ÜBERWEISUNG}}} + \underline{\underline{\mathsf{PERSONAL}}}$$

Bei der Herzkatheter-Untersuchung erlangt der Arzt über den Herzkatheter, eine Leitung ausgehend von einer im Oberarm oder Oberschenkel befindlichen Arterie direkten Zugang zum Herzen. Um während des Eingriffs festzustellen, wo sich der Katheter zur Zeit jeweils befindet, wird der Patient ununterbrochen Röntgen-überwacht. Die häufigste Untersuchung ist das gezielte Spritzen eines Kontrastmittels, um Durchfluß- und Strömungsverhalten des Blutes in herznahen Gefäßen und den Herzkammern am Röntgenschirm zu beobachten. So ist z.B. feststellbar, wieviel Blut das Herz in einer Minute durchströmt und an welchen Stellen es eventuell behindert wird. Wichtig ist auch die Möglichkeit, verschiedene Druckmessungen direkt am Herzen vorzunehmen, wie z.B. der mittlere arterielle Druck (MAD), aber auch die Spann- und Widerstandskraft herznaher Gefäße. Auch zur Therapie kann der Herzkatheter eingesetzt werden: Durch Ballon-Dilatation (Aufblasen eines an der Spitze des Katheters angebrachten Ballons) lassen sich verengte

Gefäße erweitern. Alle anfallenden Daten der Untersuchung (auch der Röntgenfilm) werden in einem Protokoll gesammelt, das die Grundlage des vom Arzt erstellten Herzkatheterbefundes bildet.

PROTOKOLL = PATID + HK-DATEN + ALLG_DATEN +
STENOSE + DRUCKKURVE + ANGIO-DATEN +

Das Protokoll wird als Liste mit folgenden Elementen spezifiziert:
Allg.Daten: betreffen HK-Technik, Prätherapie, geröntgte Herzgebiete;
Stenose: Bereichsweiten verschiedener Herzgefäße, Kalkanreicherungen;
Angiodaten: Untersuchungen der Kammerwandbewegungen, Messungen des Herzminutenvolumens, Maße und Beschaffenheit der Herzklappen;
Röntgenfilm: (von der gesamten röntgenüberwachten Untersuchung);
Druckkurven: Druckwerte in verschiedenen Gefäßen und Herzkammern;

actualize LISTE by HK-DATEN
using
sortnames HK-Daten for D
Protokoll for Liste(HK-Daten)
opnames list_HK for list: HK-Daten → Protokoll
keine_HK_Daten for nodata: → HK-Daten
zu_Protokoll for radd: Protokoll × HK-Daten → Protokoll
head_HK for head: Protokoll → HK-Daten
search_HK for search: HK-Daten × Protokoll → HK-Daten
ladd_HK for ladd: HK-Daten × Protokoll → Protokoll
delete_HK for delete: Protokoll × HK-Daten → Protokoll
endact +

opns:
–Umwandlung verschiedener Typen in HK-Daten und zurück
patId2HK: PatId → HK-Daten
allg_Daten2HK: Allg_Daten → HK-Daten
stenose2HK: Stenose → HK-Daten
angiodaten2HK: Angiodaten→ HK-Daten
röntgenfilm2HK: Röntgenfilm→ HK-Daten
druckkurven2HK: Druckkurven→ HK-Daten

hk2PatId: HK-Daten → PatId
hk2Stenose HK-Daten → Stenose
hk2Angiodaten: Angiodaten → HK-Daten
hk2Röntgenfilm: Röntgenfilm → HK-Daten
hk2Druckkurven: Druckkurven → HK-Daten

– Protokoll identifizieren
getpat: Protokoll → PatId

– Protokoll nach bestimmten Einträgen durchsuchen
getpat: Protokoll → PatId
get_Stenose: Protokoll → Stenose
get_Angiodaten: Protokoll → Angiodaten
get_Röntgenfilm: Protokoll → Röntgenfilm
get_Druckkurven: Protokoll → Druckkurven

– Protokoll initialisieren
init_proto: PatId → Protokoll

– Dateneinträge ins Protokoll vornehmen
ch_proto: Protokoll × Druckkurven → Protokoll
ch_proto: Protokoll × Röntgenfilm → Protokoll
ch_proto: Protokoll × Allg_Daten × Stenose ×
Angiodaten → Protokoll

– Röntgenfilm aus dem Protokoll entnehmen
proto_ohne_Rf: Protokoll → Protokoll

eqns:
–Umwandlung verschiedener Typen in HK-Daten und zurück
patId2HK(hk2PatId(HK)) = HK;
allg_Daten2HK(hk2Allg_Daten(HK)) = HK;
stenose2HK(hk2Stenose(HK)) = HK;
angiodaten2HK(hk2Angiodaten(HK)) = HK;
röntgenfilm2HK(hk2Röntgenfilm(HK)) = HK;
druckkurven2HK(hk2Druckkurven(HK)) = HK;

hk2PatId(patId2HK(Id)) = Id;
hk2Allg_Daten (allg_Daten2HK(Allg)) = Allg;
hk2Stenose (stenose2HK(St)) = St;
hk2Angiodaten(angiodaten2HK(Ang)) = Ang;
hk2Röntgenfilm (röntgenfilm2HK(Rf)) = Rf;
hk2Druckkurven (druckkurven2HK(Dkv)) = Dkv;

– Protokoll identifizieren
getpat (init_Proto(Id)) = Id;
getpat (ch_proto(Pr, M)) = getpat (Pr);
getpat (ch_proto(Pr, Allg,Sten,Angio)) = getpat (Pr);
getpat(Proto) = hk2PatId(head_HK(Proto));

– Protokoll nach bestimmten Einträgen durchsuchen
get_Stenose(Pr) = hk2Stenose(search_HK(stenose2HK(Stenose));
get_Angiodaten(Pr) : hk2Angiodaten(search_HK(angiodaten2HK(Angiodaten));

get_Röntgenfilm(Pr) : hk2Röntgenfilm(search_HK((Röntgenfilm));
get_Druckkurven(Pr) : hk2Druckkurven(search_HK(druckkurven2HK
    (Druckkurven));

    – *Protokoll initialisieren*
init_proto(Id) = list_HK(patId2HK(Id));

    – *Dateneinträge ins Protokoll vornehmen*
ch_proto(Pr,Druckkv) = zu_ Protokoll(Pr,druckkurven2HK(Druckkv));
ch_proto(Pr,Röntgenf) = zu_ Protokoll(Pr,röntgenfilm2HK(Röntgenf));
ch_proto(Pr,Allg,Sten,Angio) =zu_Protokoll
    (zu_Protokoll(zu_Protokoll(Pr,angiodaten2HK(Angio)),
    stenose2HK(Sten)),allg_Daten2HK(Allg));

    – *Röntgenfilm aus dem Protokoll entnehmen*
proto_ohne_Rf(Pr,Rf) = delete_HK(Pr,röntgenfilm2HK(Rf));

HK-DATEN =
sorts: HK-Daten
opns: thermoinjektion: → HK-Daten

ALLG_DATEN = MEDIKATIONEN +
sorts: HK_Technik, Gefäßart, Gefäßgebiet, HK-Angiogebiete,
Pretherapie, HK-Gebiet, HK-Sondierungsgebiete, Allg_Daten
opns: arteria, vena: → Gefäßart
femoralis, brachialis : → Gefäßgebiet
    — *Femur = Oberschenkel, Brachium = Oberarm*

hk_Technik: Gefäßart × Gefäßgebiet → HK_Technik

pretherapie: Medikationen → Pretherapie

rechtes_Herz, linkes_Herz, rechte_Kammer,
    linke_Kammer: → HK-Gebiet
hk_sondierungsgebiete: List(HK-Gebiet) → HK-Sondierungsgebiete

a_linke_Kammer, a_rechte_Kammer,a_cor_dextra, a_cor_sinistra,
a_desc_ant, a_circumflexa: → Herzgebiet
    — *dextra = rechts, sinistra = links*
hk_angiogebiete: List(Herzgebiet) → HK-Angiogebiete

allg_daten:HK_Technik × Pretherapie ×
    HK-Sondierungsgebiete × HK-Angiogebiete → Allg_Daten

```
STENOSE =                     — Stenose = Verengung
        sorts: Kalk_Qual, Kalk_Daten, LKA, Lumen_LKA,
               LAD_Bereichsweite_%, LAD, CIRC_Bereichsweite_%, CIRC,
               RKA_Bereichsweite_%, RKA, Stenose
                      — LKA = Linke Koronar-Arterie, Hauptstamm
                      — LAD = Arteria Descendens Anterior
        opns:  gering, mäßig, viel: → Kalk_Qual
               kein_kalk: → Kalk_Daten
               kalk: Kalk_Qual → Kalk_Daten
               lumen_lka_%: → Lumen_LKA
                      — Lumen_LKA = Lichte Weite der LKA im Ursprung
               lka: Kalk_Daten × Lumen_LKA → LKA
               lu_prox_lad, lu_med_lad, lu_dist_lad, lu_d_ast1_lad,
                      lu_d_ast2_lad: → LAD_Bereichsweite_%
               lad: Kalk_Daten × List(LAD_Bereichsweite) × Text → LAD
                      — lu_*_lad = Lichte Weite in 5 Hauptkranzgefäßen
                      — _prox, _med, _dist = Entfernung der Gefäße vom Herzen:
                      — _prox = proximal (benachbart)
                      — _med = medial (mittel)
                      — _dist = distal (weit)
                      — _d_ast1, _d_ast2 = Werte für die beiden Diagonal-Äste
               lu_prox_circ, lu_dist_circ, lu_m_ast1_circ, lu_m_ast2_circ,
                      lu_desc_po_circ: → CIRC_Bereichsweite_%
               circ: Kalk_Daten × List(CIRC_Bereichsweite) × Text → CIRC
               lu_prox_rka, lu_med_rka, lu_dist_rka,
                      lu_desc_po_rka: → RKA_Bereichsweite_%
               rka: Kalk_Daten × List(RKA_Bereichsweite) × Text → RKA
                      — RKA = Rechte Koronar-Arterie

               stenose: LKA × LAD × CIRC × RKA → Stenose


DRUCKKURVE =
        actualize LISTE by DRUCK
        using
               sortnames Druck for D
                      Druckkurve for Liste(Druck)
               opnames list_DK for list: Druck → Druckkurve
                      keine_DK_Daten for nodata: → Druck
                      zu_Druckkurve for radd: Druckkurve × Druck → Druckkurve
                      head_DK for head: Druckkurve → Druck
                      search_DK for search: Druck × Druckkurve → Druck
                      ladd_DK for ladd: Druck × Druckkurve → Druckkurve
```

delete_DK for delete: Druckkurve × Druck → Druckkurve
endact +
opns:
druckkurve: Druck^8 → Druckkurve
eqns:
druckkurve(D1,D2,...,D8) = zu_Druckkurve
(... zu_Druckkurve(list_DK(D1),D2), ..., D8);

DRUCK =
sorts: Druckwert, D_Qualität, Druck_Tupel, Druck
opns: niedrig, normal, hoch: → D_Qualität
druckwert_mmHG: → Druckwert
druck_tupel: Druckwert × D_Qualität → Druck_Tupel

pw_a_x_v_y_m, ra_a_x_v_y_m, la_a_x_v_y_m :
Druck_tupel ^{5} → Druck
— *PW = Pulmonalkapillargebiet*
— *RA / LA = Rechter / Linker Vorhof*

pa_a_d_m, rv_a_d_ed, lv_s_d_ed, fa_s_d_m, ao_s_d_m:
Druck_tupel ^{3} → Druck
— *PA = Pulmonalarterie*
— *RV / LV= Rechte / Linke Kammer*
— *FA = Femoral Arterie (Oberschenkel)*
— *AO = Aorta*

ANGIO-DATEN = KLAPPENDATEN +
sorts: Volumenindex_ml/qm, Muskeldicke_cm, Auswurffraktion_%,
Qualitative_Beurteilung, Kammerwandseg_rechtsschräge_Sicht,
Kammerwandseg_linksschräge_Sicht, RAO, LAO,
Gefäßwiderstand_dyn*Scm^5, Herzindex_l*qm/min,
Herzminutenvol_l/min, Herzfrequenz_1/min, Schlagvolumenindex_ml/qm,
Arteriovenöse_Sauerstoffdifferenz_%, Shuntvolumen_ml/min,
Shunt_Daten, Technik, HMV, Angiodaten
opns: ed_volx, es_volx: → Volumenindex_ml/qm
es_md_rao, es_md_lao: → Muskeldicke_cm
lv_ausw_frkt: → Auswurffraktion_%

normal, hypokinese, akinese,
dyskinese: → Qualitative_Beurteilung
— *hypokinese = übermäßige Bewegung*
— *akinese = keine Bewegung*
— *dyskinese = uneinheitlich / falsche Bewegung*

antero_basal, antero_lateral, apical, diaphragmal,
    postero_basal: $\rightarrow$ Kammerwandseg_rechtsschräge_Sicht
r_beurteilung: Kammerwandseg_rechtsschräge_Sicht
    $\rightarrow$ Qualitative_Beurteilung
rao: Qualitative_Beurteilung^5 $\rightarrow$ RAO

superior_lateral, inferior_lateral, postero_lateral, apical_lateral,
    basal_septal: $\rightarrow$ Kammerwandseg_linksschräge_Sicht
l_beurteilung: Kammerwandseg_linksschräge_Sicht
    $\rightarrow$ Qualitative_Beurteilung
lao: Qualitative_Beurteilung^5 $\rightarrow$ LAO

tot_syst_wid, tot_pulm_wid, pulm_art_wid:
    $\rightarrow$ Gefäßwiderstand_dyn*Scm^5,
hmvi: $\rightarrow$ Herzindex_l*qm/min
syst_hmv, pulm_hmv: $\rightarrow$ Herzminutenvol_l/min
hf: $\rightarrow$ Herzfrequenz_1/min
svi: $\rightarrow$ Schlagvolumenindex_ml/qm
avdo2: $\rightarrow$ Arteriovenöse_Sauerstoffdifferenz_%

kein_shunt: $\rightarrow$ Shunt_Daten
shv: $\rightarrow$ Shuntvolumen_ml/min
shunt: Shuntvolumen_ml/min $\rightarrow$ Shunt_Daten

fick, dye, thermo, angio: $\rightarrow$ Technik

hmv: Technik $\times$ (Gefäßwiderstand_dyn*Scm^5)^3 $\times$
    Herzindex_l*qm/min $\times$ (Herzminutenvol_l/min)^2 $\times$
    Herzfrequenz_1/min $\times$ Schlagvolumenindex_ml/qm $\times$
    Arteriovenöse_Sauerstoffdifferenz_% $\times$ Shunt_Daten
    $\rightarrow$ HMV

angio_daten: (Volumenindex_ml/qm)^2 $\times$ (Muskeldicke_cm)^2 $\times$
    Auswurffraktion_% $\times$ RAO $\times$ LAO $\times$
    HMV $\times$ Klappendaten $\rightarrow$ Angiodaten

<u>KLAPPENDATEN</u> =

sorts: Gradient, Öff_flä_index_qcm/qm, Regur_vol_ml/min,
Verkalkung, Verdickte_Ränder, Beweglichkeit, Regurgitation,
Klappe, Prolaps, Bikuspidal, Papillarmuskel, Sehnenfäden,
Pericardkalk, Pericarderguß, Mitralklappe, Aortenklappe,
Pulmonalklappe, Trikuspidalklappe, Klappe_general

opns: nicht_eingeschr, mäßig_eingeschr,

stark_eingeschr: → Beweglichkeit
keine, leicht, mäßig, beträchtlich, stark: → Regurgitation
— *Regurgitation = Zurückführen des Blutes entgegen*
— *der Klappenrichtung*
verkalkt, nicht_verkalkt: → Verkalkung
verdickt, nicht_verdickt: → Verdickte_Ränder
klappe: Gradient × Öff_flä_index_qcm/qm × Regur_vol_ml/min ×
Verkalkung × Verdickte_Ränder × Beweglichkeit ×
Regurgitation → Klappe

vorderes_segel, hinteres_segel: → Prolaps
bi_ja, bi_nein: → Bikuspidal
pap_verdickt, pap_nicht_verdickt: → Papillarmuskel
verkürzt, normal: → Sehnenfäden
kein, gering, mäßig, viel: → Pericardkalk
p_erg, kein_p_erg: → Pericarderguß

mitral: Klappe × Prolaps → Mitralklappe
aorten: Klappe × Bikuspidal → Aortenklappe
pulmonal: Klappe → Pulmonalklappe
trikuspidal: Klappe × Prolaps → Trikuspidalklappe
klappe_general: Papillarmuskel × Sehnenfäden × Perikardkalk ×
Pericarderguß → Klappe_general

klappendaten: Mitralklappe × Aortenklappe × Pulmonalklappe ×
Trikuspidalklappe × Klappe_general → Klappendaten

HK-BEFUNDUNG = PROTOKOLL + DIAGNOSE+ THERAPIE + HK-GERÄTE +
sorts: Befundbrief , Befund
opns:
befund: PatId × Allg_Daten× Stenose × Druckkurven ×
Angiodaten × Röntgenfilm × Diagnose ×
Therapie → Befund
getpat: Befund → PatId
getpat: Befundbrief → PatId
befundbrief: Befund → Befundbrief
eqns: getpat(befund(Id,Allg,Sten,DK,Ang,R,Diag,Th)) = Id;
getpat(befundbrief(B)) = getpat(B)

DIAGNOSE =
sorts: Ersatzdiagnose, Dia_Urteil, Gefäßanzahl,
LV_Funktion, Khe, Trikus_Qual,
Klapp_Qual_Stenose, Klapp_Qual_Insuffizienz,

Mitral, Aorten, Trikuspidal, Klappen_Diagnose,
Kardiomyopathie, Diagnose

opns: mitralprolaps, mitralfehler,
aortenfehler, trikuspidalfehler, kardiomyopathie,
hypertonie_arteriell $\rightarrow$ Bool
eins, zwei, drei: $\rightarrow$ Ersatzdiagnose
dia_urteil: Bool^6, Ersatzdiagnose $\rightarrow$ Dia_Urteil

eins, zwei, drei: $\rightarrow$ Gefäsanzahl
eingeschränkt, normal: $\rightarrow$ LV_Funktion
papillarmuskeldysfunktion: $\rightarrow$ Bool
keine_Khe: $\rightarrow$ Khe
khe: Gefäsanzahl, LV_Funktion, Bool $\rightarrow$ Khe
— *KHE = Koronare Herzerkrankung*

organisch, relativ: $\rightarrow$ Trikus_Qual
leicht, mäßig, beträchtlich, schwer: $\rightarrow$ Klapp_Qual_Stenose
leicht, mäßig, beträchtlich, schwer: $\rightarrow$ Klapp_Qual_Insuffizienz
mitral: Klapp_Qual_Stenose $\times$ Klapp_Qual_Insuffizienz $\rightarrow$ Mitral
aorten: Klapp_Qual_Stenose $\times$ Klapp_Qual_Insuffizienz $\rightarrow$ Aorten
trikuspidal: Klapp_Qual_Stenose $\times$ Klapp_Qual_Insuffizienz
$\times$ Trikus_Qual $\rightarrow$ Trikuspidal
klappen_diag: Mitral Aorten Trikuspidal $\rightarrow$ Klappen_Diagnose

dilativ, hypertroph_obstruktiv, hypertroph_nicht_obstruktiv:
$\rightarrow$ Kardiomyopathie

zustand_nach_bypass, risikofaktoren, problematik: $\rightarrow$ Text

diagnose: Dia_Urteil $\times$ Khe $\times$ Klappen_Diagnose $\times$
Kardiomyopathie $\times$ Text^3 $\rightarrow$ Diagnose

HK-GERÄTE = PATID+

sorts: Röntgengerät, Druckmeßgerät, Kontrastmittel,
Thermomedium, Röntgenfilm, Katheter,

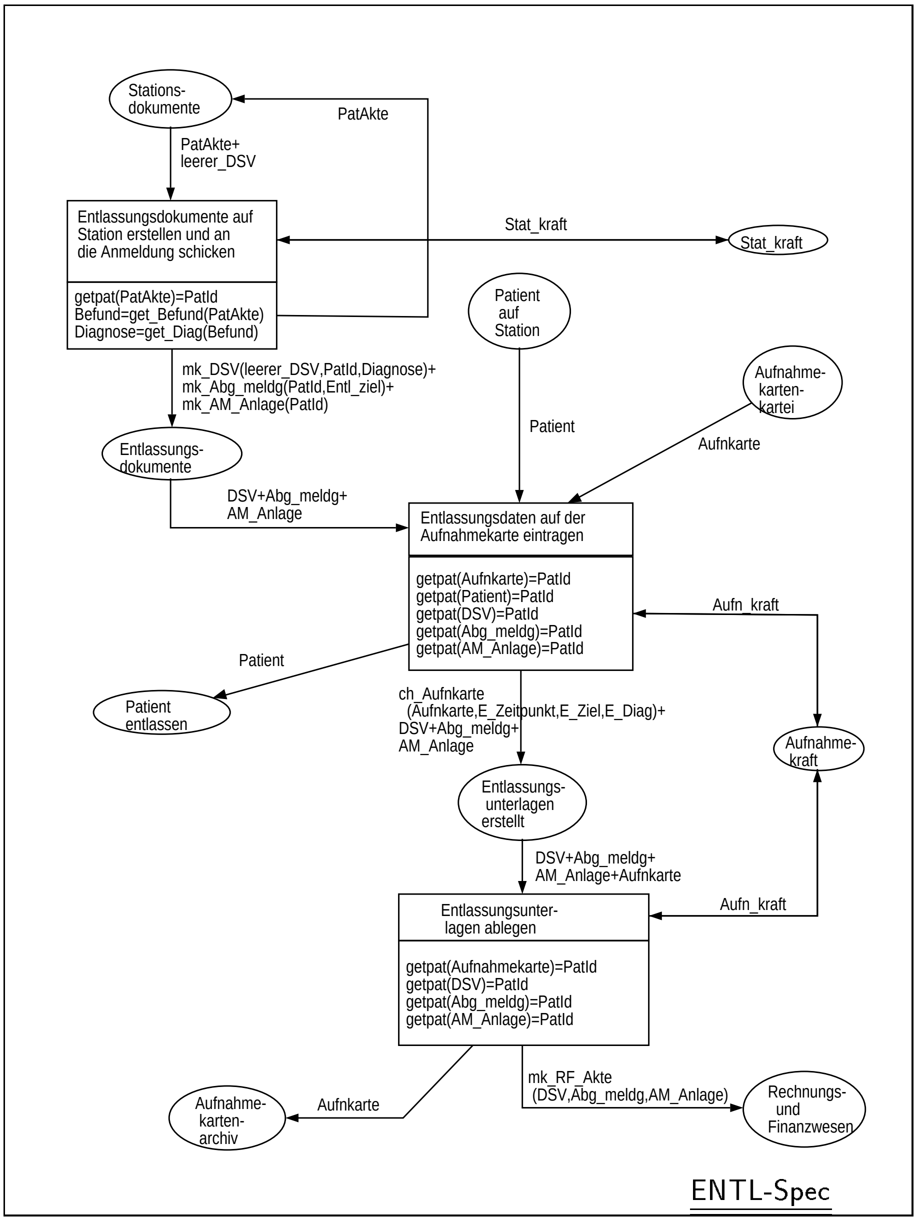
opns: röntgengerät: $\rightarrow$ Röntgengerät
druckmeßgerät: $\rightarrow$ Druckmeßgerät
kontrastmittel: $\rightarrow$ Kontrastmittel
thermomedium: $\rightarrow$ Thermomedium
röntgenfilm : PatId $\rightarrow$ Röntgenfilm
katheter: $\rightarrow$ Katheter
dkb: $\rightarrow$ Druckkurven-Bilder

HK-ÜBERWEISUNG = PATID+
sorts: HK-Überweisung,
opns: hk_Überweisung: PatId → HK-Überweisung
getpat:HK-Überweisung → PatId
eqns: getpat(hk_Überweisung(Id)) = Id;

– *Spezifikation PERSONAL siehe Spez. AUFN-Spec*

3.2.5 Entlassung

Die Entlassung von Patienten aus dem DHZB wird in Netz ENTL modelliert. Dabei wird die Diagnose auf den Diagnose-Statistik-Vordruck übertragen, der dann zusammen mit der Abgangsmeldung und einer Anlage an die Abteilung Rechnungs- und Finanzwesen geschickt wird. Die Aufnahmekarte, die bei der Anmeldung des Patienten erstellt wurde, wird aus der Aufnahmekartenkartei gesucht, um die Entlassungsdaten (Zeitpunkt, Ziel und Diagnose) ergänzt und im Aufnahmekartenarchiv abgelegt. Der Patient befindet sich nach der Entlassung auf der Stelle *Patient entlassen*:



ENTL: Entlassung

Dem Netz **Entlassung (ENTL)** liegt folgende Spezifikation zugrunde:

ENTL-Spec: Spezifikation zum Netz Entlassung

$$\underline{\underline{\text{ENTL-Spec}}} = \underline{\underline{\text{PATIENT}}} +_{\underline{\underline{PATID}}} \underline{\underline{\text{AUFNAHMEKARTE}}} +_{\underline{\underline{PATID}}}$$
$$\underline{\underline{\text{ENTLASSUNGSDOKUMENTE}}} + \underline{\underline{\text{PERSONAL}}}$$

Die bevorstehende Entlassung eines Patienten aus dem DHZB wird der Anmeldung von der zuletzt behandelnden Station über eine Abgangsmeldung mitgeteilt. Die Abgangsmeldung enthält Daten über Entlassungszeitpunkt und -ziel, sowie die Entlassungsdiagnose. Diese Daten werden dann auf die Aufnahmekarte übertragen, welche im Aufnahmekartenarchiv aufbewahrt wird. Die Entlassungsdokumente (Abgangsmeldung, ausgefüllter Diagnose-Statistik-Vordruck) werden der Abteilung Rechnungs- und Finanzwesen zugesandt.

– *Spezifikation AUFNAHMEKARTE siehe Spez. AUFN-Spec*

ENTLASSUNGSDOKUMENTE = PATID + DIAGNOSE +

– *Spezifikation DIAGNOSE siehe Spez. HK-Spec*

sorts: Abgangsmeldung, Abg_meldg_Anlage, Diagnosen-Statistik_Vordruck,
Rechn_Finanzwesen_Akte

opns: leerer_Diag-Stat_Vordruck: → Diagnosen-Statistik_Vordruck
mk_Diag-Stat_Vordruck: Diagnosen-Statistik_Vordruck × PatId ×
Diagnose → Diagnosen-Statistik_Vordruck
mk_Abgangsmeldung: PatId × Entlassungsziel → Abgangsmeldung
mk_Abg_meldg_Anlage: PatId → Abg_meldg_Anlage
mk_Rechn_Finanzwesen_Akte: Diagnosen-Statistik_Vordruck ×
Abgangsmeldung × Abg_meldg_Anlage
→ Rechn_Finanzwesen_Akte
getpat: Diagnosen-Statistik_Vordruck → PatId
getpat: Abgangsmeldung → PatId
getpat: Abg_meldg_Anlage → PatId
getpat: Rechn_Finanzwesen_Akte → PatId

eqns: getpat (mk_Diag-Stat_Vordruck(DV, Id, D))= Id;
getpat (mk_Abgangsmeldung(Id,Z))= Id;
getpat (mk_Abg_meldg_Anlage(Id))= Id;
getpat (mk_Rechn_Finanzwesen_Akte(DV,Am,AmA,) = getpat(DV)

– *Spezifikation PERSONAL siehe Spez. AUFN-Spec*

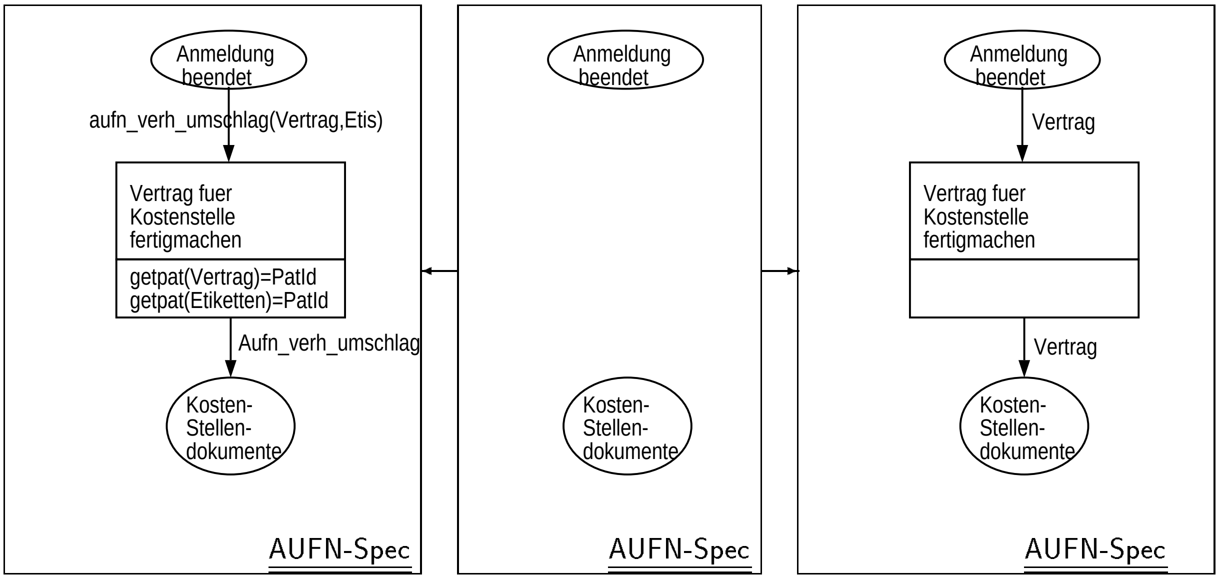
3.3 Transformation des IST-Systems in die IST-Essenz

Ziel dieses Kapitels ist es, mit Hilfe der regelbasierten Abstraktion/Verfeinerung die Essenz des IST-Systems abzuleiten. Wir prüfen das IST-System darauf, welche Daten und Arbeitsabläufe für die Funktion eines rechnergestützten Systems essentiell sind, eliminieren Technologieabhängigkeiten und zufällige Gegebenheiten, und reduzieren die Beschreibung auf ihren (aus Datensicht) wesentlichen Kern.

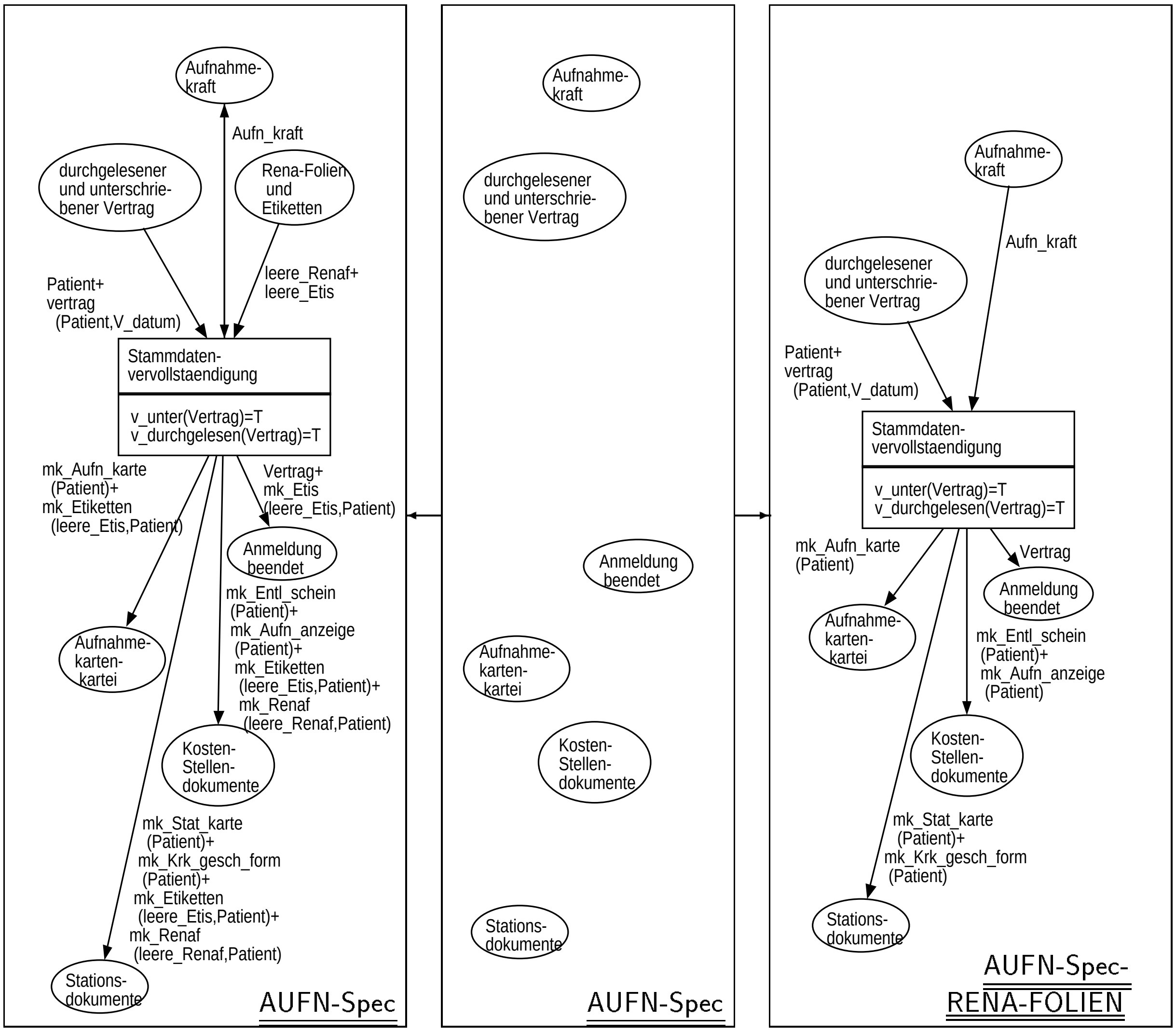
3.3.1 Aufnahme

Wir abstrahieren in den Regeln $r1$ bis $r5$ von folgenden Sachverhalten: von Rena-Folien, Etiketten, Aufnahmekraft, da sie innerhalb des zu spezifizierenden Systems keine Funktion mehr haben werden; von Einweisungsschein, Kostenübernahmeschein, Aufnahme-Verhandlungs-Umschlag, Vertrag durchlesen/unterschreiben, da sie außerhalb der Systemgrenzen liegen. Wir fassen folgende Arbeitsabläufe zusammen: Vertragsschreibung und Stammdatenvervollständigung wird zur Anmeldung, wobei die Patienten-Identifikation eine zentrale Rolle spielt. Die Anmeldung weist den Patienten in seine erste Station ein.

Die Regeln r_A1a und r_A1 löschen die Terme, die aus RENA-FOLIEN bzw. ETIKETTEN stammen, sowie die Stellen $Rena - Folien$ und $Etiketten$ aus dem Netz Aufnahme AUFN, da sie für keine Strukturierungsoperation mehr gebraucht werden:



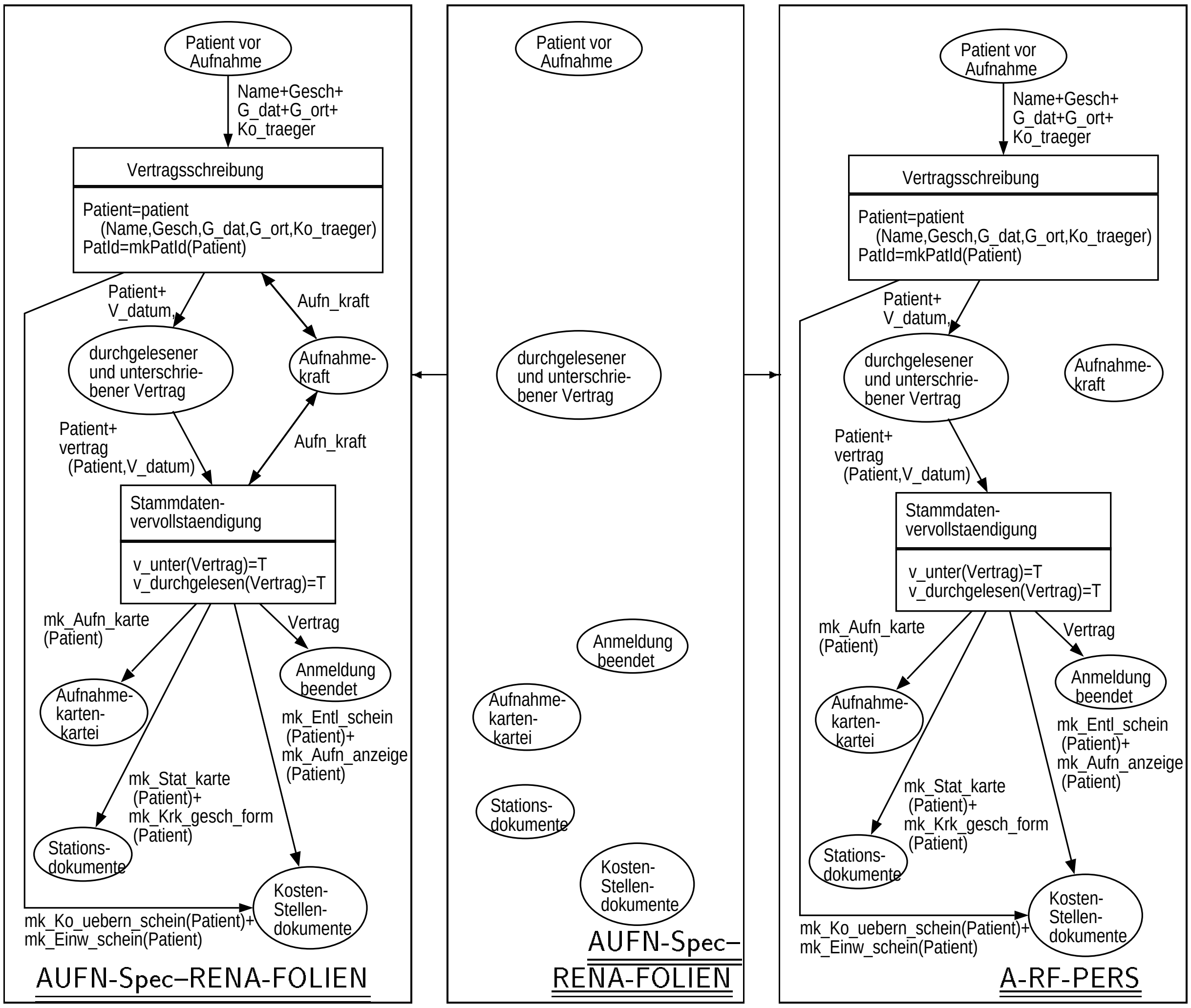
L_r_A1a K_r_A1a R_r_A1a



L_r_A1 K_r_A1 R_r_A1

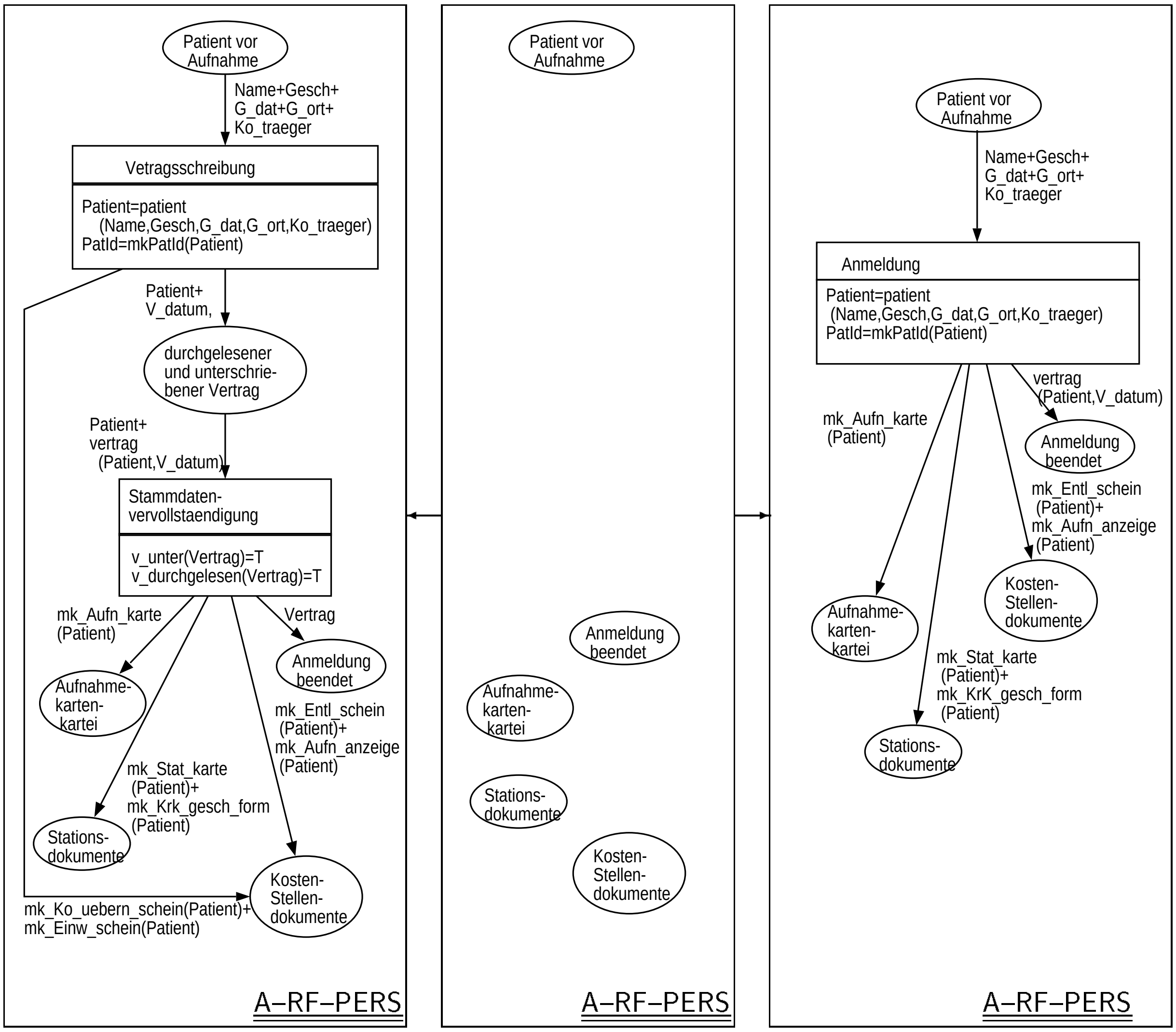
Die Regeln r_A1a und r_A1 wären parallel anwendbar, wenn keine der beiden Regeln Objekte löschte, die die andere Regel noch brauchte, oder Objekte benutzte, die von der anderen Regel erzeugt würden (formal als sequentielle und parallele Unabhängigkeit bezeichnet, siehe Seite 144). Es ließe sich dann eine Parallelregel $r_A1a + r_A1$ konstruieren, die dasselbe Netz ableitete, das bei sequentieller Anwendung der Regeln r_A1a und r_A1 in beliebiger Richtung abgeleitet würde. Auf den ersten Blick scheint das der Fall zu sein: Die Ansätze beider Regeln im Original-Netz AUFNAHME überlappen nur in der einen Stelle *Anmeldung beendet*, die sowohl im K-Netz K_r_A1a als auch im K-Netz K_r_A1 vorkommt. Dennoch sind beide Netze nicht sequentiell unabhängig, da wir im Fall von AHL-Netzen auch die zugrundeliegenden Spezifikationen und Algebren durch die Regeln (AHL-Netz-Morphismen) verändern. In diesem Fall löscht die zweite der angewendeten Regeln die Unterspezifikationen RENA-FOLIEN und ETIKETTEN aus der Original-Spezifikation AUFN-Spec. Von der ursprünglichen Spezifikation ETIKETTEN bleibt nur noch die Sorte *Etiketten* übrig, weil diese im Arbeitsbereich *Behandlung* noch gebraucht wird. Würde man nun diese Regel zuerst anwenden, so könnte die andere Regel keinen Ansatz in das resultierende Netz finden, da der Spezifikation des resultierenden Netzes bestimmte Operationen und Sorten der Spezifikation der anzuwendenden Regel bereits fehlen. Dennoch ist es möglich, eine einzige Regel $r_A1a + r_A1$ (auf die Netzstruktur beschränkt) anzugeben, die beide obigen Ersetzungen in einem Schritt ausführt, in dem dann auch die Reduzierung der Spezifikation stattfinden muß (*Concurrency*, siehe [PER95]).

Die zweite Regel r_A2 abstrahiert von der Aufnahmekraft, erhält aber die entsprechende Stelle, um die Kompatibiltät mit der Strukturierung zu gewährleisten. Aus der Spezifikation wird die Unterspezifikation PERSONAL gelöscht. Wir verwenden die Abkürzung A-RF-PERS für die neue Spezifikation:



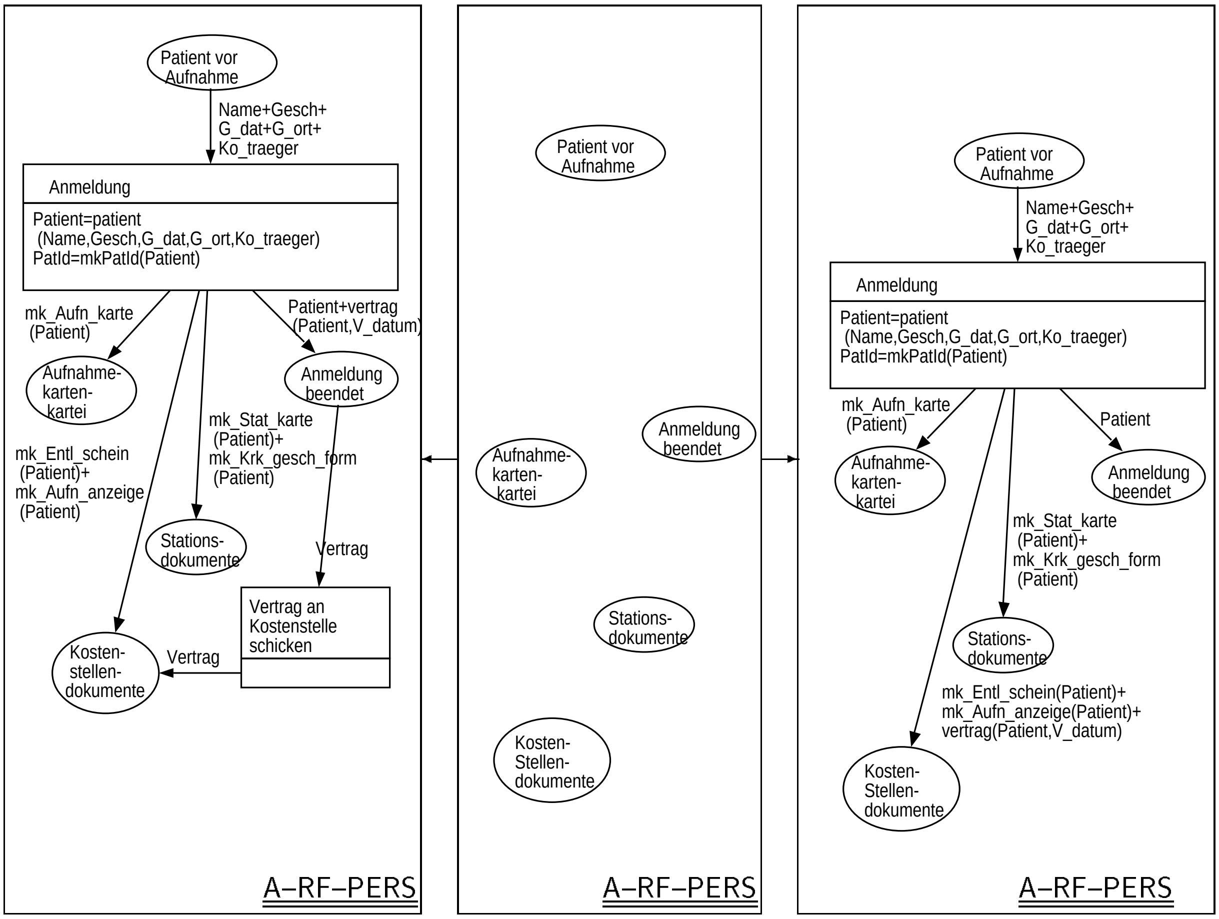
L_r_A2 K_r_A2 R_r_A2

In der Regel r_A3 werden die bisher getrennten Tätigkeiten *Vertragsschreibung* und *Stammdatenvervollständigung* zu einem Arbeitsablauf *Anmeldung* zusammengefaßt. Die Erstellung der Dokumente Kostenübernahmeschein und Einweisungsschein entfällt. Die Spezifikation wird durch r_A3 nicht verändert.



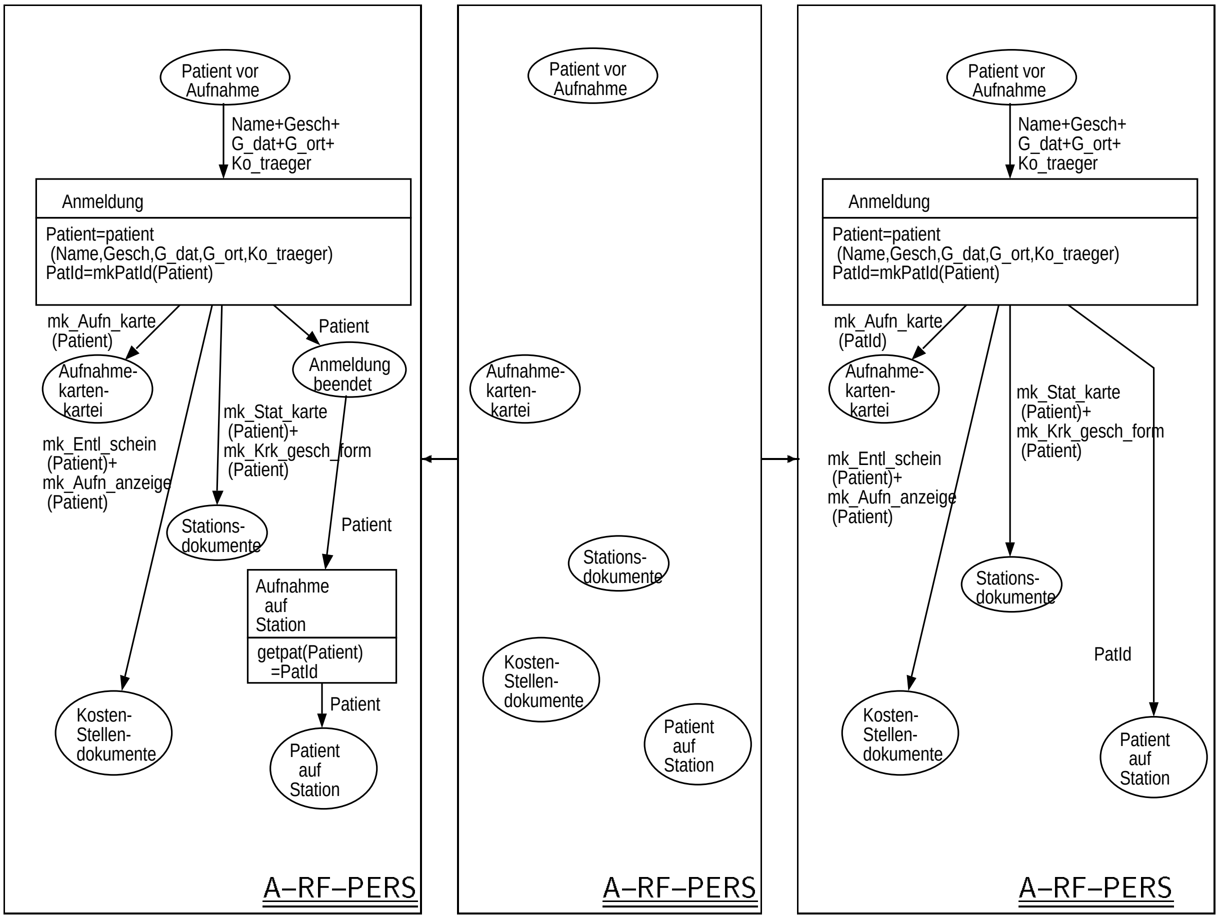
L_r_A3 K_r_A3 R_r_A3

Regel r_A4 integriert die bisher getrennt aufgeführte Tätigkeit *Vertrag für Kostenstelle fertigmachen* mit der Tätigkeit *Anmeldung*, die dann auch die Vertragsdaten an die Kostenstelle weiterreicht:



L_r_A4 K_r_A4 R_r_A4

Regel r_A5 schließlich integriert die bisher getrennt aufgeführte Tätigkeit *Aufnahme auf Station* mit der Tätigkeit *Anmeldung*, die dann auch die *PatId* des Patienten an seine erste Station weitergibt. Wir abstrahieren damit vom konkreten Patienten als Person, und arbeiten nur noch mit seiner *PatId* als Schlüssel.



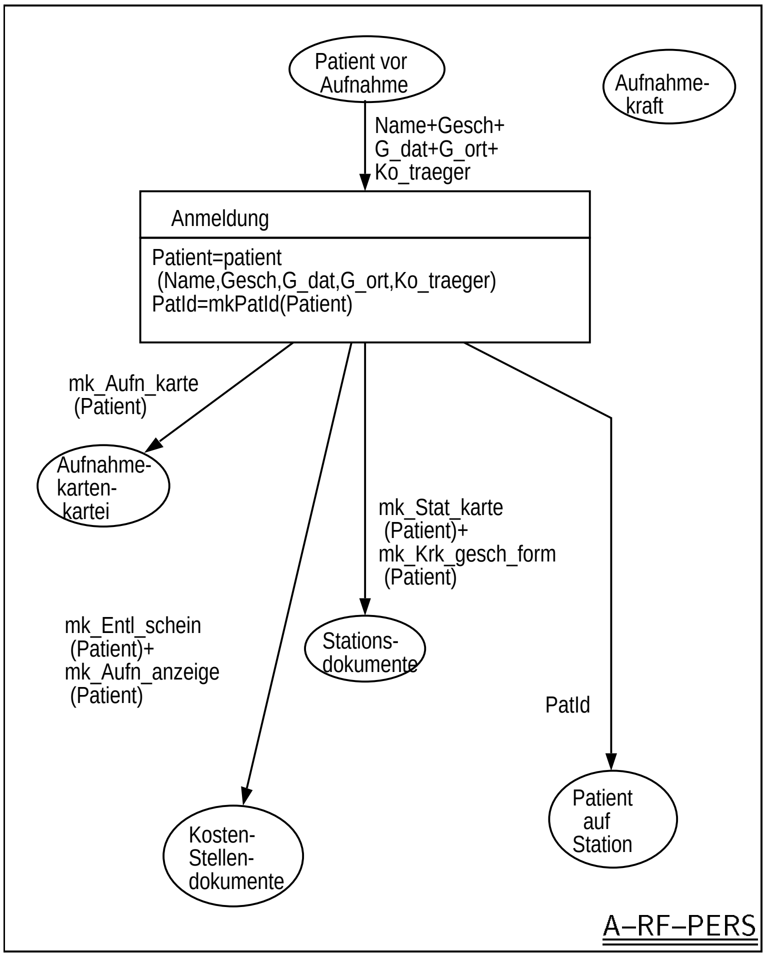
L_r_A5 K_r_A5 R_r_A5

Wenden wir nun die Regeln r1a, r1, ..., r5 sequentiell auf die jeweils vorher resultierenden Netze an, wobei wir mit dem Originalnetz AUFNAHME beginnen, so erhalten wir folgende Ableitungssequenz:

$$\boxed{\text{AUFNAHME}} \xrightarrow{r_A1a} \square \xrightarrow{r_A1} \square \xrightarrow{r_A2} \square \xrightarrow{r_A3} \square \xrightarrow{r_A4} \square \xrightarrow{r_A5} \boxed{\text{AUFNAHME\_IST-ESSENZ}}$$

Dabei lassen sich keine der definierten Regeln parallel anwenden, da stets ein Teil des Netzes oder der Spezifikation, den die eine Regel braucht, von der anderen gelöscht wird. Dies ist auch einsichtig, da hier auf einem relativ kleinen Netz operiert wird, das nur wenige Transitionen enthält, und schrittweise Transitionen oder Unterspezifikationen der bisherigen Spezifikation gelöscht werden. Wird aber eine Transition über eine Regel verändert (sei es, daß eine Eingangs- oder Ausgangskante bzw. die damit verbundene Stelle gelöscht wird, sei es, daß ihre Schaltbedingung oder auch nur eine Kantenbeschriftung geändert wird), so kann diese Transition nicht parallel durch eine zweite Regel verändert werden, damit das Gesamtnetz stets konsistent ersetzt wird.

Das resultierende Netz Aufnahme_IST-Essenz sieht folgendermaßen aus:



Aufnahme_IST-Essenz

Die Stelle *Aufnahmekraft* bleibt aus Gründen der Kompatibilität von horizontaler und vertikaler Strukturierung als isolierte Stelle erhalten. Wir werden sie und andere isolierte Stellen nach der kompletten Transformation aller Netze des IST-Systems in die IST-Essenz mit einer abschließenden Regel löschen.

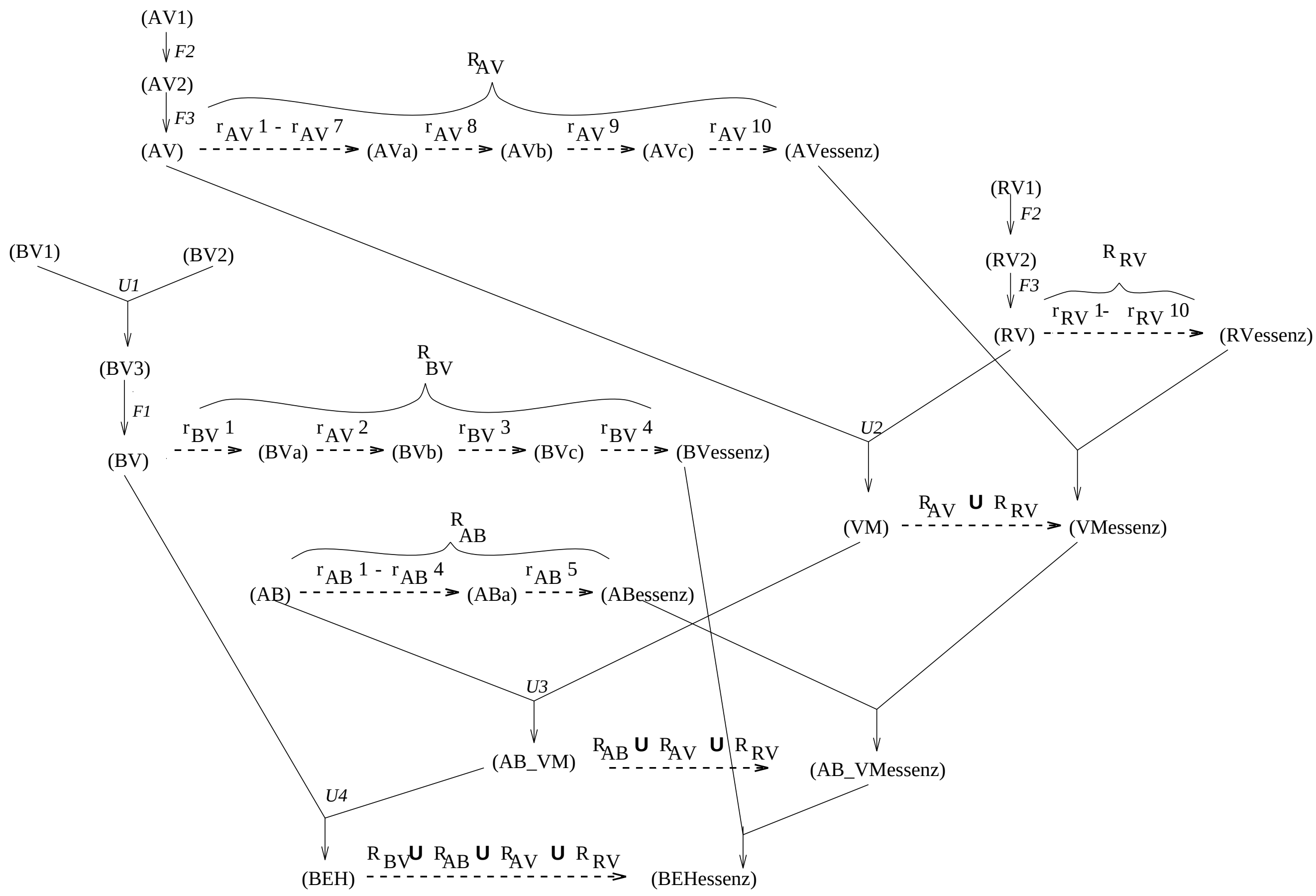


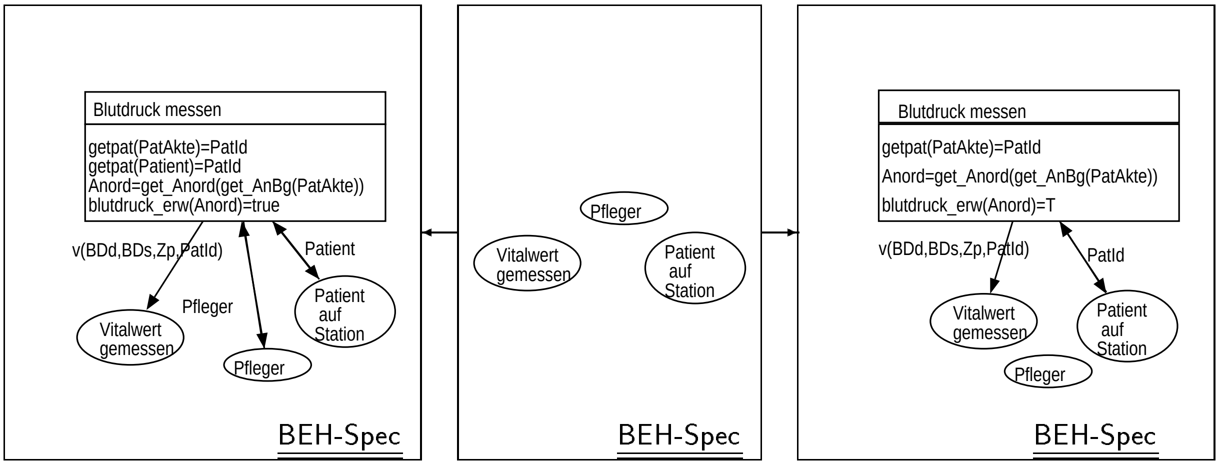
Abbildung 3.6: Behandlung: Strukturierung und Transformation der Netze

3.3.2 Behandlung

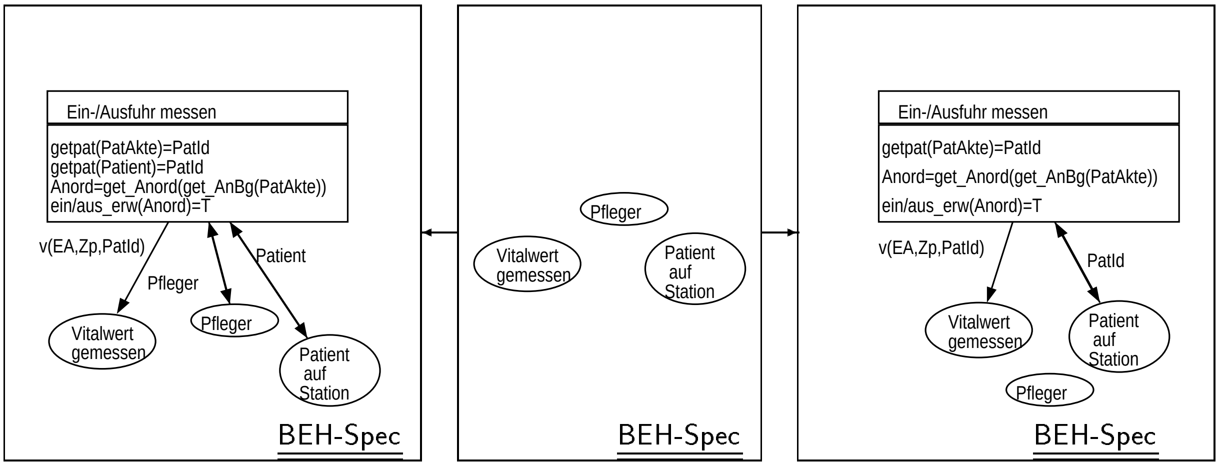
Die Übersicht in Abb. 3.6 zeigt die Transformation der Teilnetze aus dem Bereich *Behandlung*. Da die regelbasierten Transformationen kompatibel zur horizontalen Strukturierung der Teilnetze sind, ist das resultierende Netz Behandlung_IST-Essenz (BEHessenz) stets gleich, unabhängig davon, ob die Netze des IST-Systems erst über *Union* und *Fusion* zum Gesamtnetz BEH vereinigt werden und dann durch Regelanwendung dieses eine Netz in das Netz BEHessenz transformiert wird, oder ob die Regeln lokal auf die Teilnetze angewendet werden und dann die resultierenden transformierten Teilnetze über die horizontalen Strukturierungsoperationen zum Netz BEHessenz vereinigt werden. Einige Möglichkeiten (aber nicht alle), Strukturierungs- bzw. Transformationsschritte anzuordnen, sind in der Übersicht angegeben.

Wir geben im folgenden die dargestellten Regeln und resultierenden AHL-Netze explizit an.

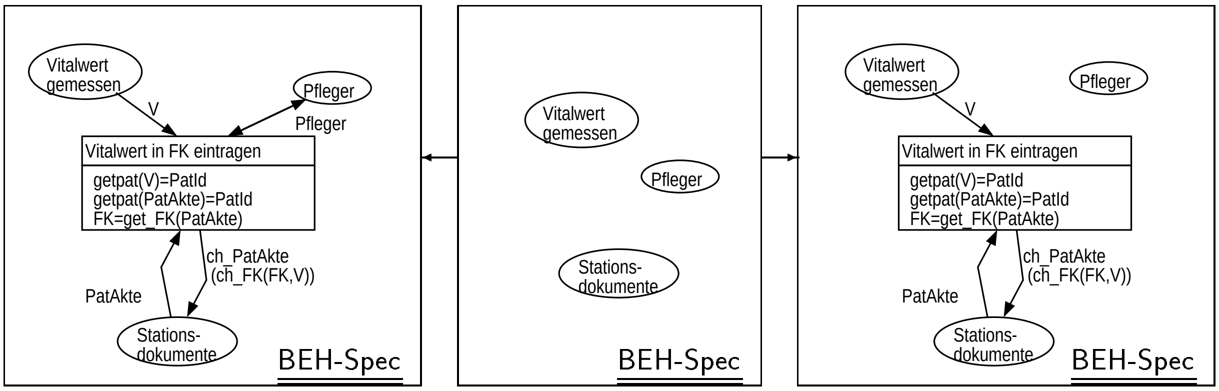
Die Regeln $r_{AV}1$ bis $r_{AV}7$ aus der Regelmenge R_{AV} abstrahieren von der Beteiligung des Pflegers an den Vitalmessungen und von Patienten als konkreten Personen:



L_$r_{AV}1$ K_$r_{AV}1$ R_$r_{AV}1$

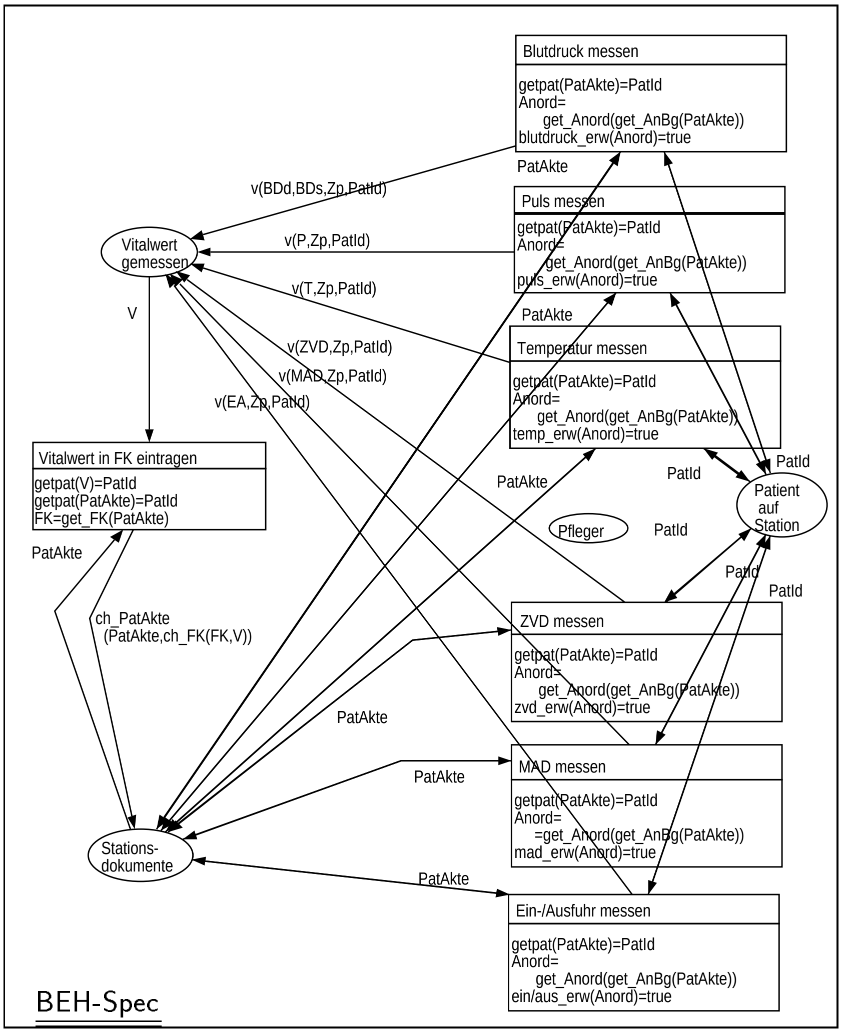


L_$r_{AV}6$ K_$r_{AV}6$ R_$r_{AV}6$



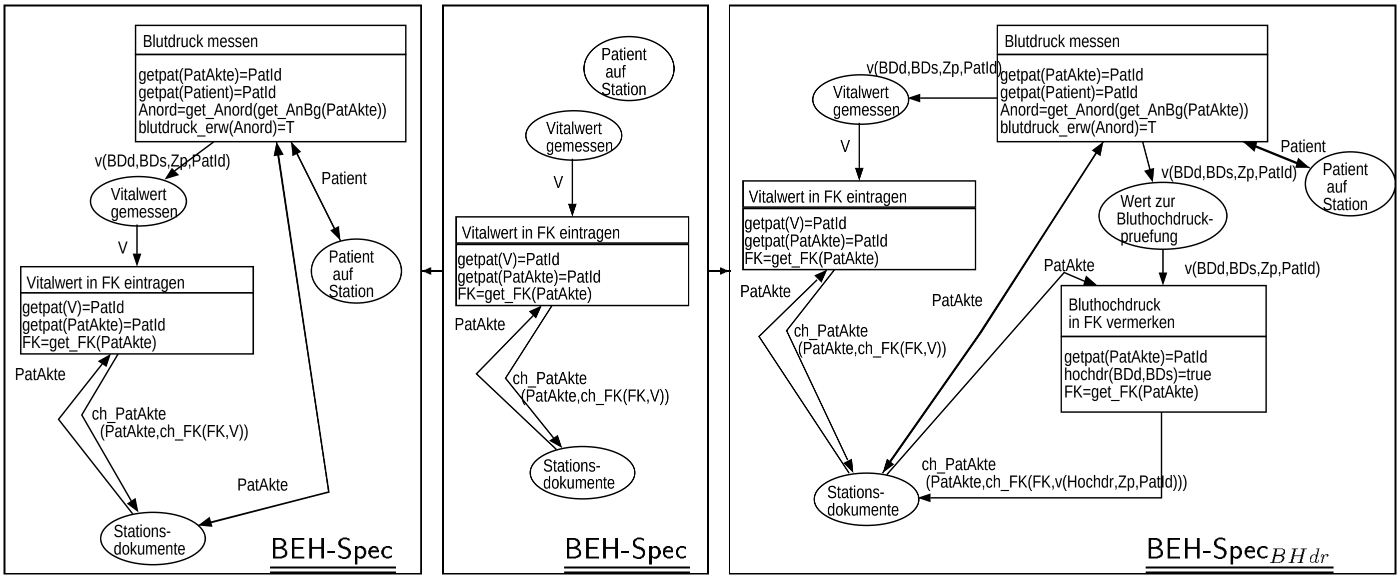
L_$r_{AV}7$ K_$r_{AV}7$ R_$r_{AV}7$

Angewendet in beliebiger Reihenfolge auf das Netz AV ergibt sich Netz AVa:



AVa: Abstraktion von Pfleger

Regel $r_{AV}8$ fügt dem Netz eine Überprüfungsfunktion hinzu, die einen Warnvermerk in die Fieberkurve einträgt, wenn es sich beim gemessenen Blutdruckwert um Bluthochdruck handelt:



L_$r_{AV}8$ K_$r_{AV}8$ R_$r_{AV}8$

Die Spezifikation BEH-Spec wird durch Anwendung von $r_{AV}8$ zu BEH-Spec$_{BHdr}$ transformiert. Folgende Teile der Spezifikation BEH-Spec werden durch die Regel $r_{AV}8$ (Bluthochdruckprüfung hinzufügen) verändert:

In der Teilspezifikation FIEBERKURVE kommt folgende Gleichung für den Eintrag des Bluthochdruckvermerks hinzu:

- *Werte in die Fieberkurve eintragen: Vitalwerte*
- *Hochdruckvermerk und Meßzeitpunkt eintragen*

ch_FK(FK,v(Hochdr,Zp,Id)) = fk (get_Ident(FK),
ch_Vitalw(get_Vitalwerte(FK),v(Hochdr,Zp)),
get_Medikationen (FK),
get_Perfusionen(FK),
get_Intensivstat_werte (FK),
get_NAB_Werte(FK))

Die Teilspezifikation VITALWERTE wird um eine Hochdruck-Liste erweitert:

- *Hochdr_Liste*

```
actualize LISTE by VITALWERT
using
      sortnames Vitalwert for D
              Hochdr_Liste for Liste(Vitalwert)
      opnames list_Hochdr for list: Vitalwert → Hochdr_Liste
              kein_Hochdrwert for nil: Hochdr_Liste
              append_Hochdr for radd: Hochdr_Liste × Vitalwert → Hochdr_Liste
endact +

opns:
              – Vitalwerte-Konstruktor, um Hochdr_Liste erweitert
      vitalwerte: Puls_Liste × Temp_Liste × BD_Liste ×
              ZVD_Liste × MAD_Liste × E-/A_Liste
              × Hochdr_Liste → Vitalwerte

      get_Hochdr_Liste: Vitalwerte → Hochdr_Liste

eqns:
              – auf bestimmte Vitalwertetabellen zugreifen
      get_Puls_Liste(vitalwerte(P,T,B,Z,M,EA,H)) = P;
      get_Temp_Liste(vitalwerte(P,T,B,Z,M,EA,H)) = T;
      get_BD_Liste(vitalwerte(P,T,B,Z,M,EA,H)) = B;
      get_ZVD_Liste(vitalwerte(P,T,B,Z,M,EA,H)) = Z;
      get_MAD_Liste(vitalwerte(P,T,B,Z,M,EA,H)) = M;
      get_E-/A_Liste(vitalwerte(P,T,B,Z,M,EA,H)) = EA;
      get_Hochdr_Liste(vitalwerte(P,T,B,Z,M,EA,H)) = H;

              – Vitalwerte hinzufügen (Messungen eintragen)
      ch_Vitalwerte(VW,v(Puls,Zp)) =
      vitalwerte(append_Puls(get_Puls_Liste(VW), v(Puls,Zp)),
              get_Temp_Liste(VW),
              get_BD_Liste(VW),
              get_ZVD_Liste(VW),
              get_MAD_Liste(VW)),
              get_E-/A_Liste(VW)),
              get_Hochdr_Liste(VW)),

      ch_Vitalwerte(VW,v(Temp,Zp)) =
      vitalwerte(get_Puls_Liste(VW),
              append_Temp(get_Temp_Liste(VW), v(Temp,Zp)),
              get_BD_Liste(VW),
              get_ZVD_Liste(VW),
              get_MAD_Liste(VW))
              get_E-/A_Liste(VW)),
              get_Hochdr_Liste(VW)),
```

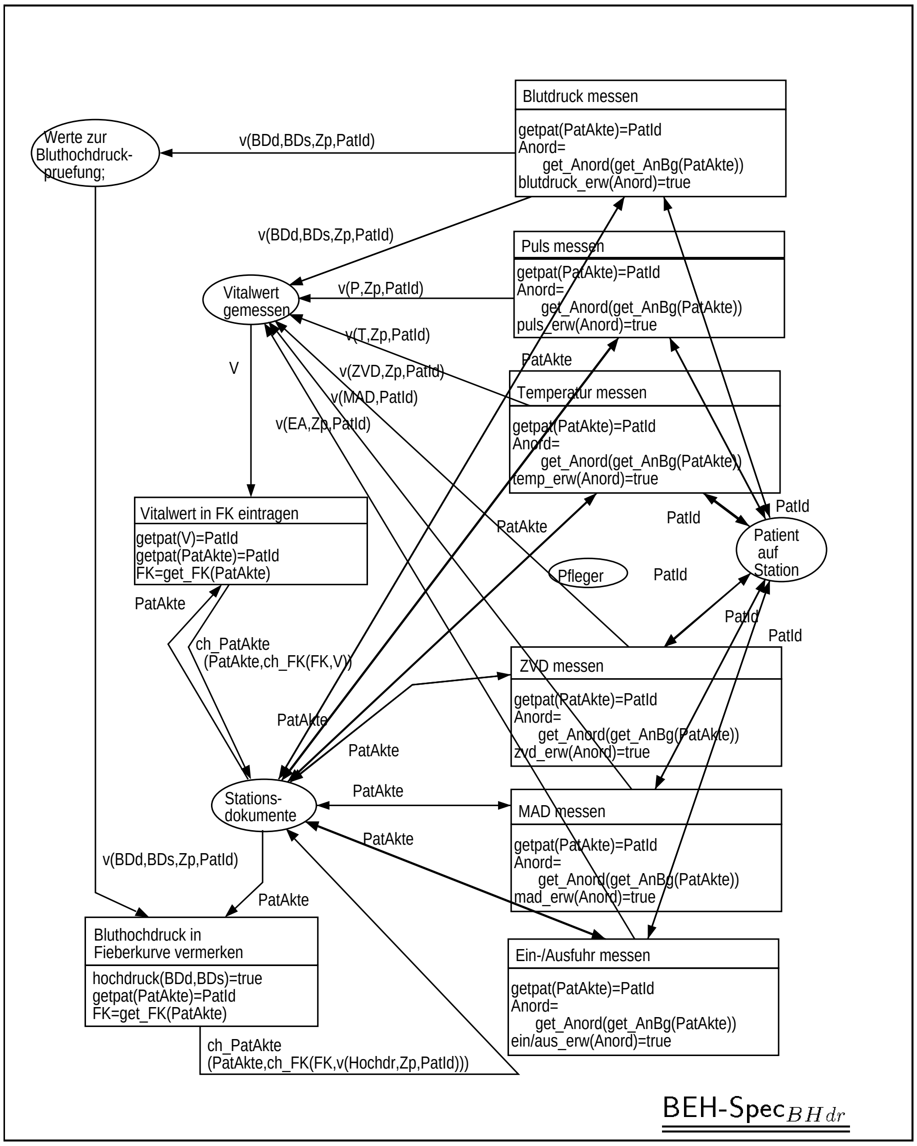
•
•
•

```
ch_Vitalwerte(VW,v(Hochdr,Zp)) =
vitalwerte(get_Puls_Liste(VW),
           get_Temp_Liste(VW),
           get_BD_Liste(VW),
           get_ZVD_Liste(VW),
           get_MAD_Liste(VW)),
           get_E-/A_Liste(VW)),
           append_Hochdr(get_Hochdr_Liste(VW), v(Hochdr,Zp) ),
```

Bei der Teilspezifikation VITALWERT kommen die Sorte Hochdr_vermerk, Konstanten und eine Op. zur Bluthochdruckprüfung sowie der Konstruktor für den Vitalwert Hochdruck hinzu:

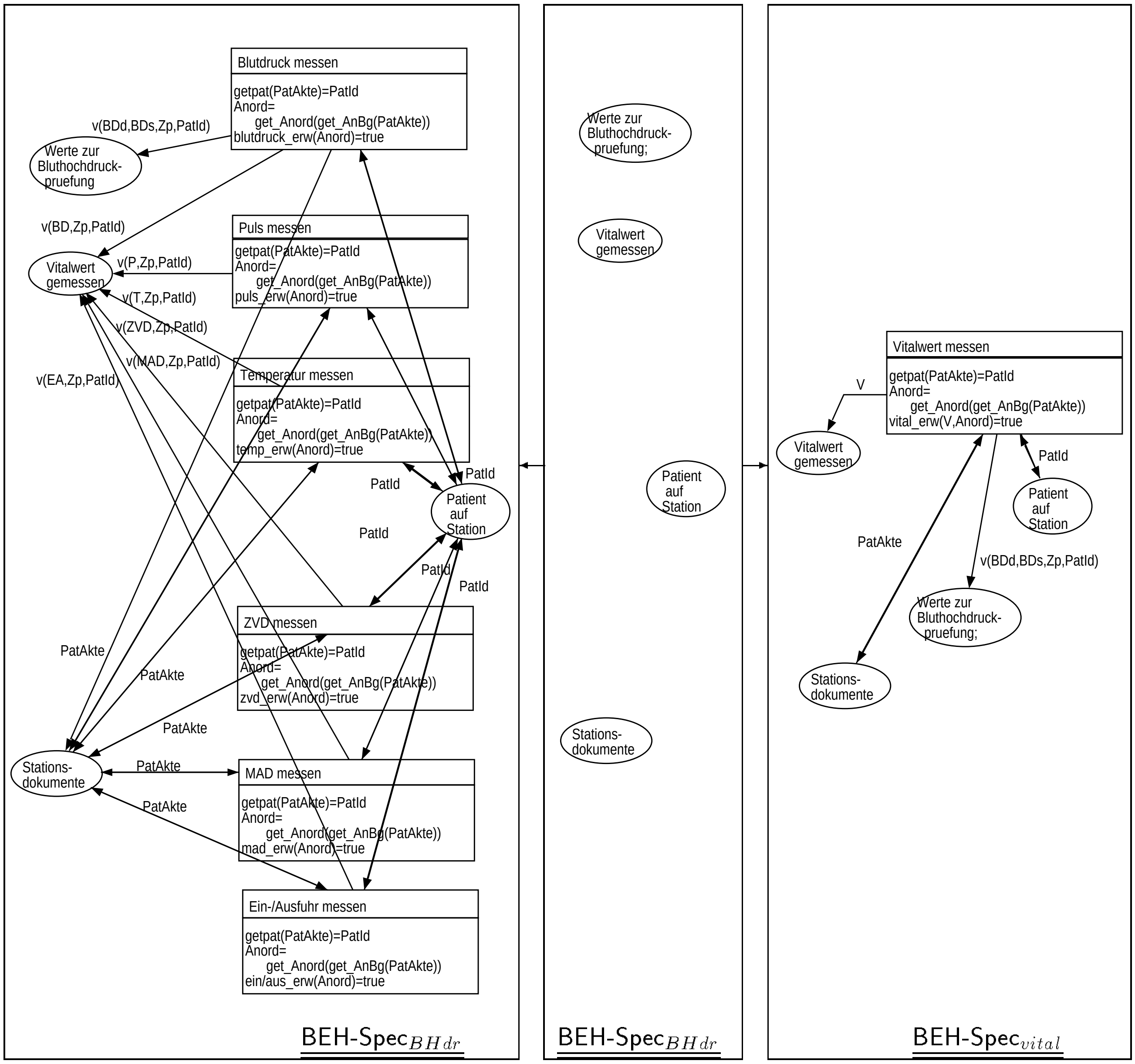
```
sorts: Hochdr_vermerk
opns:
       max_BDs: → Blutdruck_syst_mmHG
       max_BDd: → Blutdruck_diast_mmHG

       hochdr: Blutdruck_syst_mmHG × Blutdruck_diast_mmHG → Bool

       v: Hochdr_vermerk × Zeitpunkt → Vitalwert
       v: Hochdr_vermerk × Zeitpunkt × PatId → Vitalwert
eqns:
       hochdr(BDs,BDd) = or(BDs ≥ max_BDs, BDd ≥ max_BDd);
```

Wenden wir Regel $r_{AV}1$ bis $r_{AV}8$ auf das Netz AV an, so erhalten wir Netz AVb:



AVb: Bluthochdruckprüfung

Regel $r_{AV}9$ faßt alle einzelnen Vitalwertemessungen in einer einzigen Transition zusammen. Wir abstrahieren damit von der bisherigen Modellierung der Nebenläufigkeit der einzelnen Messungen.



L_$r_{AV}9$ K_$r_{AV}9$ R_$r_{AV}9$

Die Spezifikation BEH-Spec$_{BHdr}$ wird in $r_{AV}9$ zur Spezifikation BEH-Spec$_{vital}$, wobei folgende Teile der Spezifikation BEH-Spec$_{BHdr}$ durch die Regel $r_{AV}9$ (Vitalwertemessungen zusammenfassen) verändert werden:

Die Teilspezifikation ANORDN_BOGEN wird erweitert um die Teilspezifikation VITALWERT, denn es kommt eine Operation vital_erw hinzu, die prüft, ob der jeweils gemessene Vitalwert auch als Anordnung auf dem Anordnungsbogen steht:

ANORDN_BOGEN = PATID + VITALWERT +

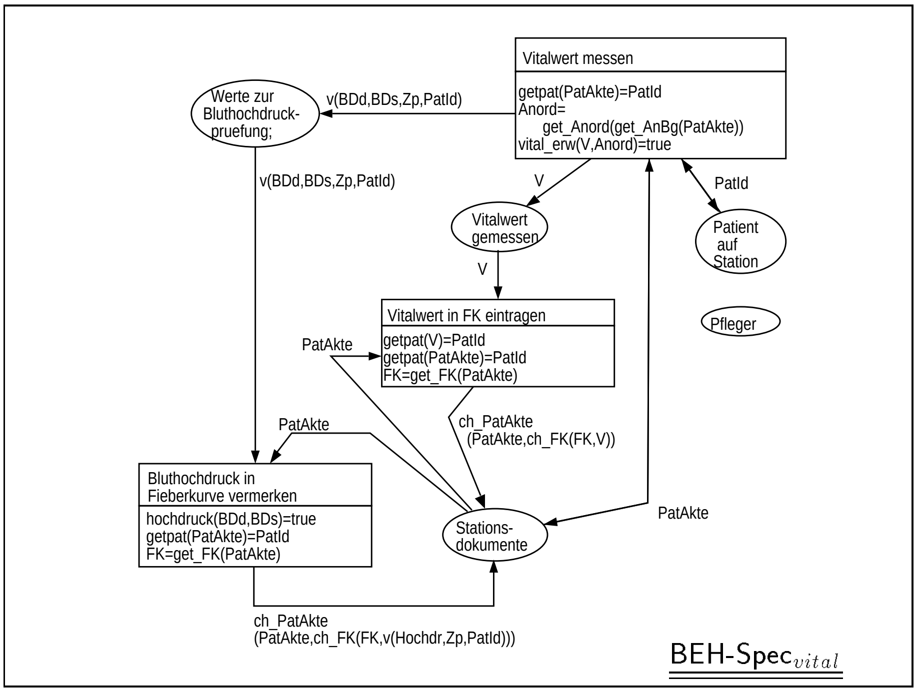
opns:

vital_erw: Vitalwert $\times$ Anordnungen $\rightarrow$ Bool

eqns:

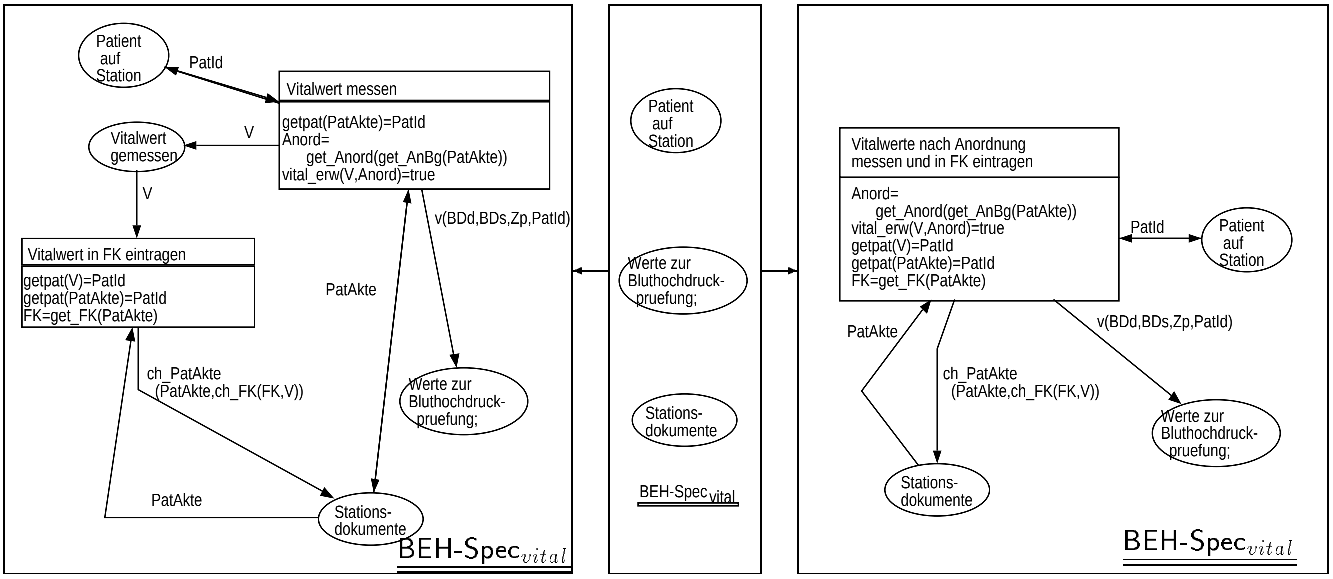
vital_erw(v(Puls,Zp,Id),An) = puls_erw(An);
vital_erw(v(Temp,Zp,Id),An) = temp_erw(An);
vital_erw(v(BDs,BDd,Zp,Id),An) = blutdruck_erw(An);
vital_erw(v(ZVD,Zp,Id),An) = zvd_erw(An);
vital_erw(v(MAD,Zp,Id),An) = mad_erw(An);
vital_erw(v(Einf,Ausf,Zp,Id),An) = ein/aus_erw(An);

Die Anwendung der Regel $r_{AV}9$ auf das Netz AVb liefert uns das Netz AVc:



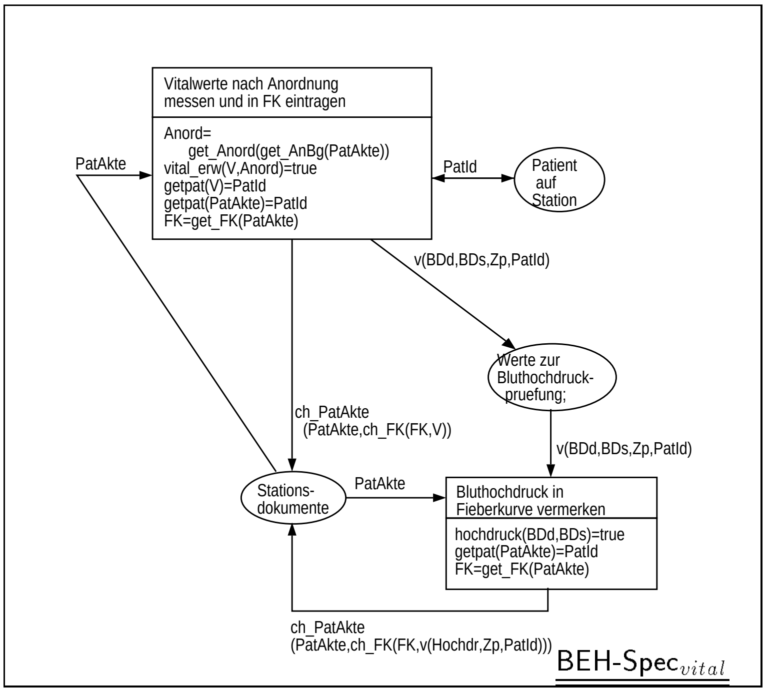
AVc: Zusammengefaßte Vitalwertemessungen

Die letzte Regel, die uns das Netz AVc in das Netz AVessenz transformiert, faßt die Vorgänge *Messen* und *Eintragen* in einer Transition zusammen:



L_$r_{AV}10$ K_$r_{AV}10$ R_$r_{AV}10$

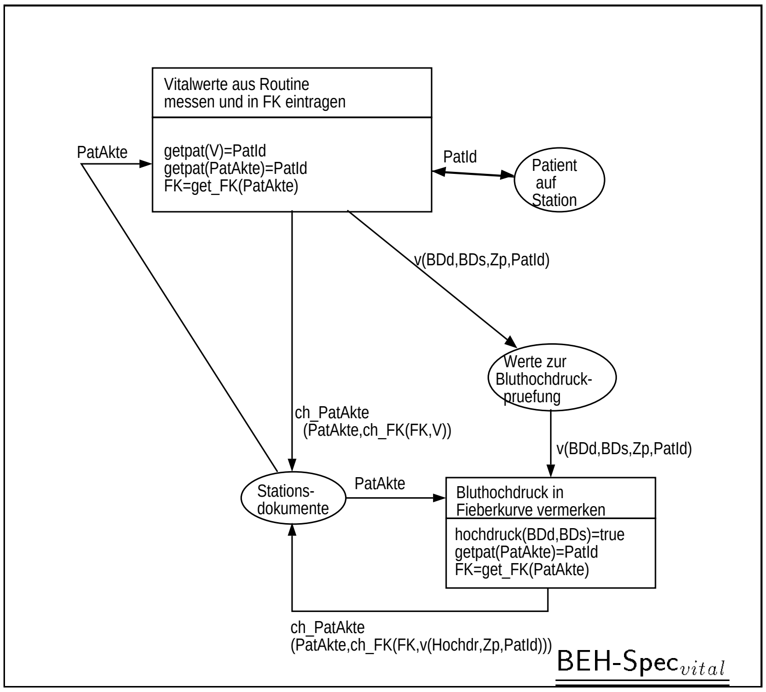
Als Ergebnis haben wir in Netz AVessenz eine Abstraktion von der tatsächlichen Angeordneten Vitalwertemessung. Dieses Teilnetz der gesamten IST-Essenz dient nun als Grundlage für regelbasierte Verfeinerungsschritte, die dann das Modell des angeforderten computergestützten Systems liefern (Funktionale Essenz).



AVessenz: Messen und Eintragen zusammengefaßt

Die Regeln aus der Regelmenge R_{RV} transformieren das Netz Routine-Vitalwertemessung (RV) in das Netz RVessenz. Da die transformierten Netze und die Transformationsregeln r_RV1 bis r_RV10 sehr den Netzen und Regeln aus dem Bereich *Angeordnete Vitalwertemessungen* ähneln, verzichten wir hier auf explizite Darstellung. Die Unterschiede bestehen lediglich in den Schaltbedingungen der Transitionen, da die Überprüfung der Anordnungen aus dem Anordnungsbogen entfällt.

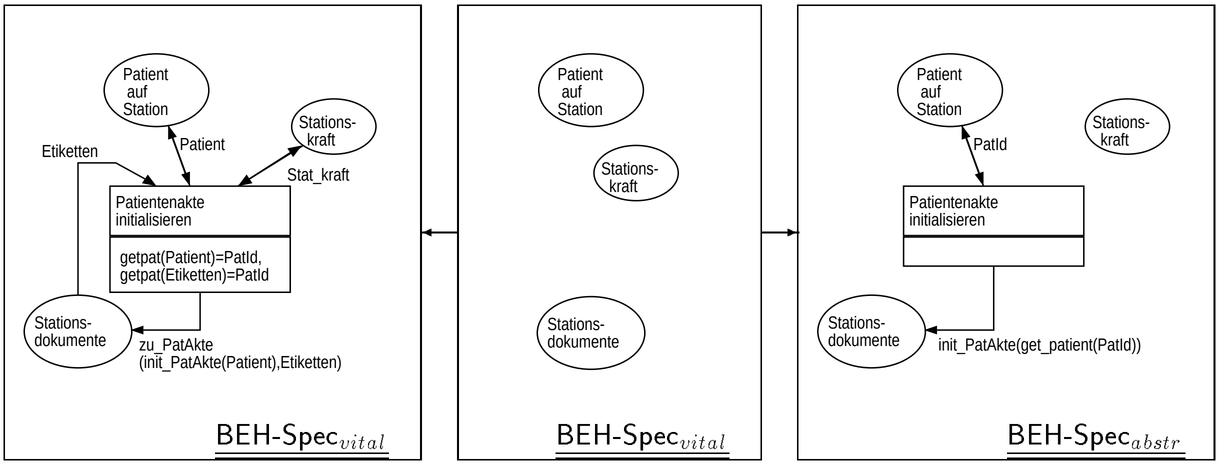
Wir beschränken uns daher darauf, das resultierende Netz RVessenz darzustellen, bei dem genau wie beim Netz AVessenz das Messen und Eintragen der Vitalwerte zusammengefaßt sind:



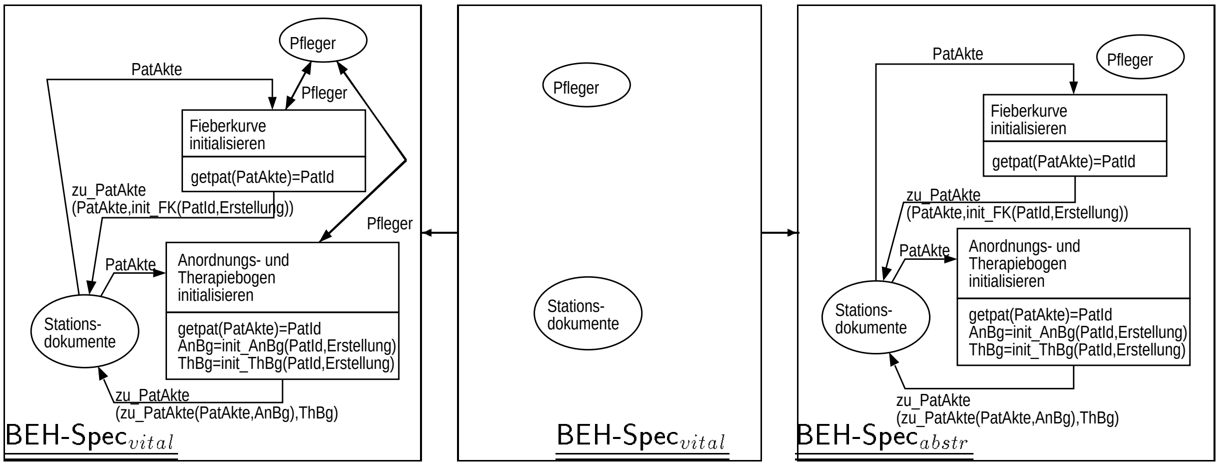
RVessenz: Routine-Vitalwertemessung, IST-Essenz

Kommen wir nun zur Regelmenge R_{BV}, deren Regeln das Netz Behandlungs-Vorbereitungen (BV) in das Netz BVessenz transformieren.

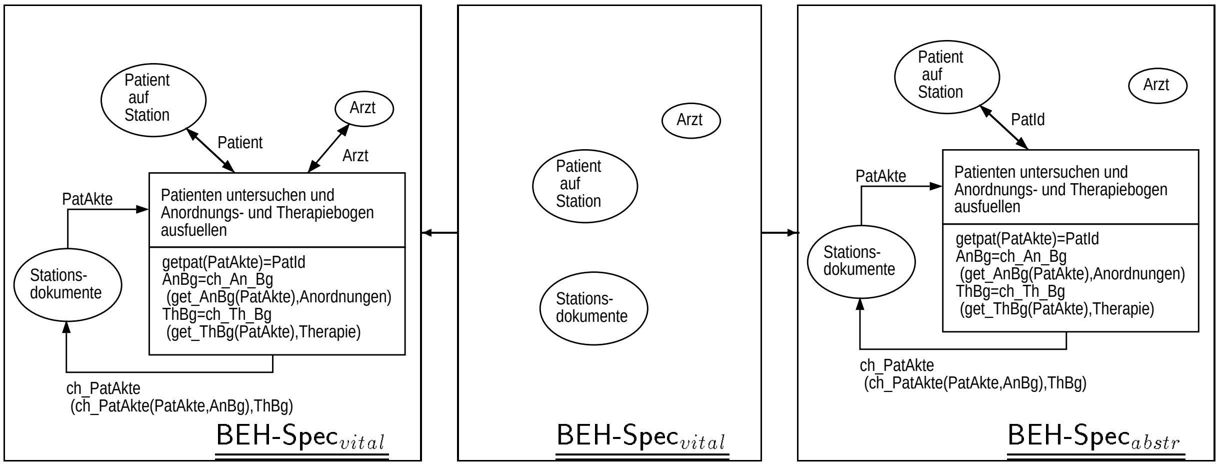
Die Regeln $r_{BV}1$ bis $r_{BV}3$ abstrahieren von den Etiketten, sowie vom an den Behandlungsvorbereitungen beteiligten Personal, wobei wiederum die jeweiligen Stellen *Stat_kraft*, *Pfleger* und *Arzt* erhalten werden, um die Kompatibilität mit den Strukturierungsoperationen sicherzustellen. Die Teilspezifikation ETIKETTEN, die nur noch die Sorte *Etiketten* enthielt, wird mit dieser Regel ebenfalls gelöscht.



L_$r_{BV}1$ K_$r_{BV}1$ R_$r_{BV}1$

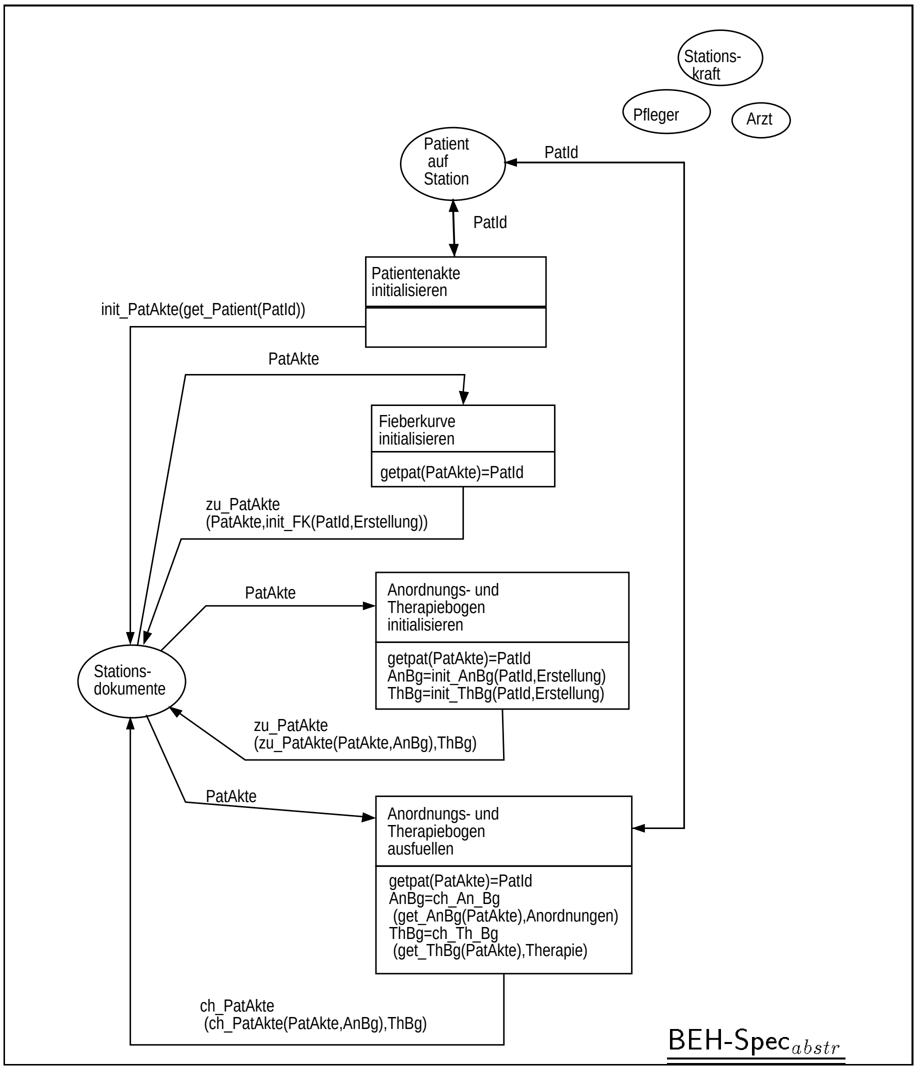


L_$r_{BV}2$ K_$r_{BV}2$ R_$r_{BV}2$



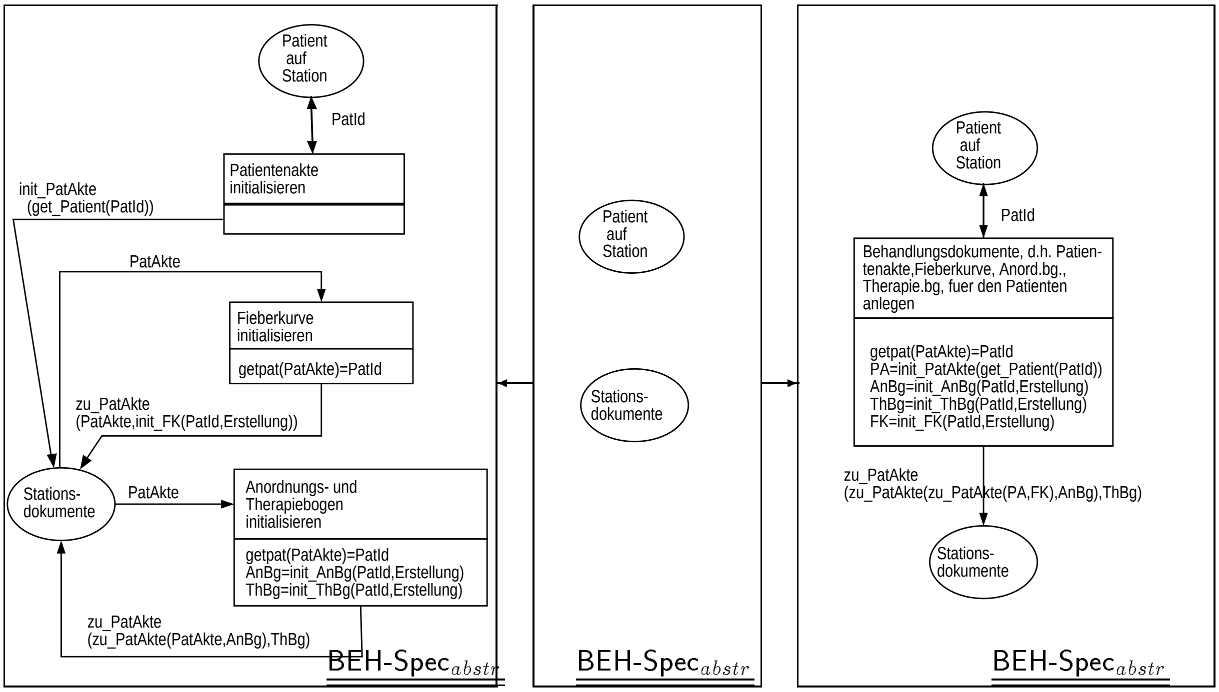
L_$r_{BV}3$ K_$r_{BV}3$ R_$r_{BV}3$

Die neue Spezifikation BEH-Spec$_{abstr}$ entspricht dabei der bisherigen BEH-Spec$_{vital}$ ohne die Teilspezifikationen PERSONAL und ETIKETTEN.
Wenden wir die Regeln in beliebiger Reihenfolge oder parallel auf das Netz BV an, so erhalten wir das Netz BVc:



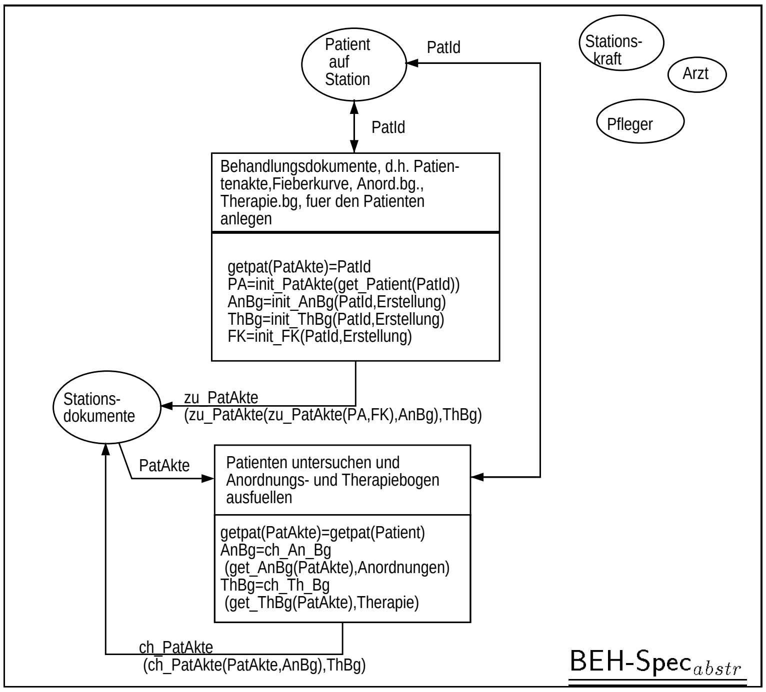
BVc: Abstraktion vom Personal

Die letzte Regel aus der Regelmenge R_{BV} faßt die drei Vorgänge *Patientenakte initialisieren, Fieberkurve initialisieren, Anordnungs- und Therapiebogen initialisieren* zu einem Vorgang *Behandlungsdokumente initialisieren* zusammen.



L_$r_{BV}4$ K_$r_{BV}4$ R_$r_{BV}4$

Das resultierende Netz, die IST-Essenz des Netzes **Behandlungsvorbereitungen (BV)**, ist die Transformation des Netzes **BV4** mit der Regel $r_{BV}4$:

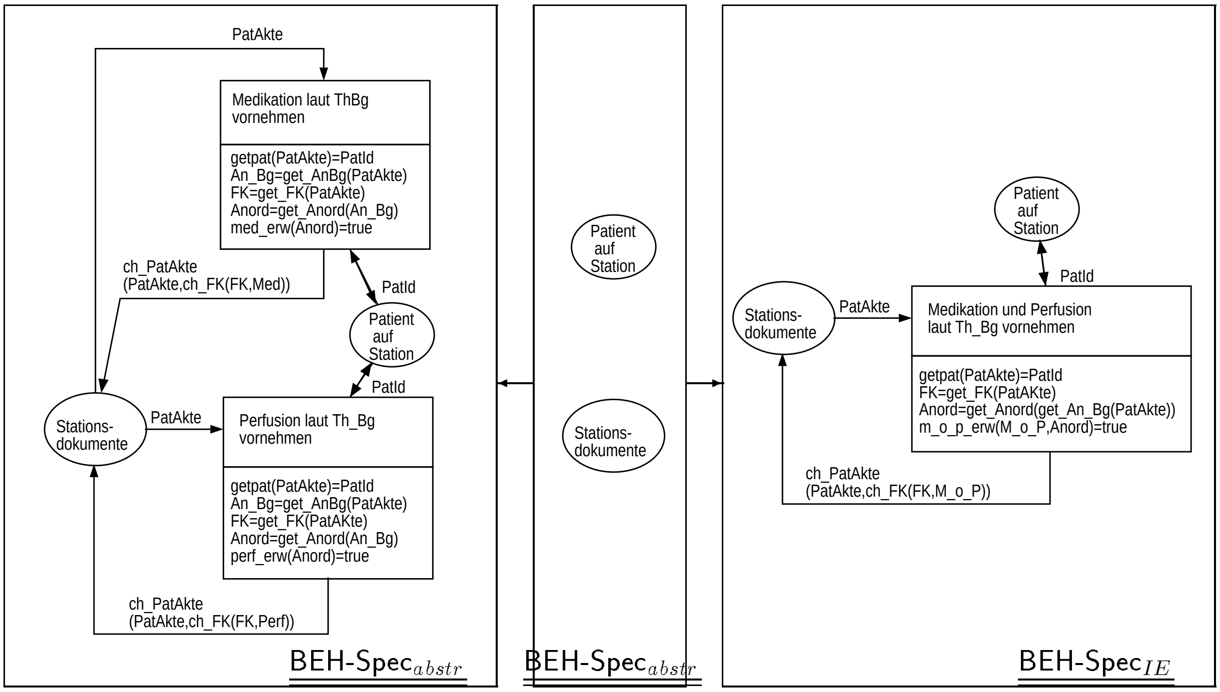


BVessenz: Behandlungsvorbereitung , IST-Essenz

Wir kommen nun zur letzten Regelmenge aus dem Arbeitsbereich *Behandlung*, R_{AB}, deren Regeln das Netz *Angeordnete Behandlung* in das Netz *ABessenz* transformieren.

Die ersten vier Regeln $r_{AB}1 - r_{AB}4$ abstrahieren von der Beteiligung eines Pflegers an der Angeordneten Behandlung, d.h. die Kanten zwischen der Stelle $Pfleger$ und den vier Transitionen im Netz AB werden gelöscht. Da wir analoge Regeln bereits bei anderen Netzen kennengelernt haben, verzichten wir hier auf die explizite Darstellung von $r_{AB}1 - r_{AB}4$ und des resultierenden Netzes ABa.

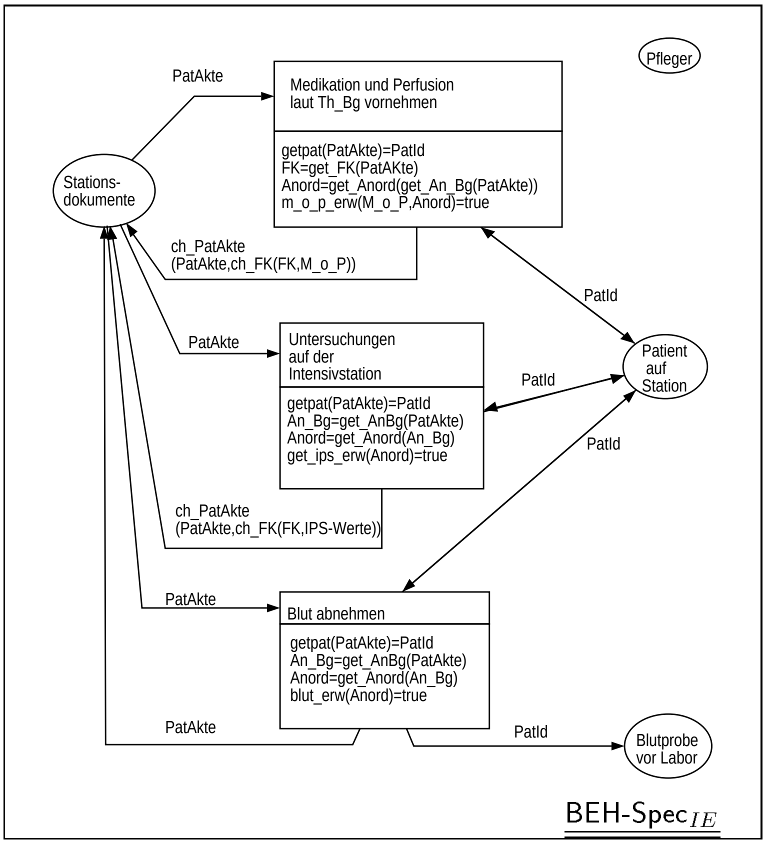
Die fünfte Regel $r_{AB}5$ faßt die Tätigkeiten *Medikation vornehmen* und *Perfusion vornehmen* in einer Transition zusammen, wobei die Teilspezifikation ANORDN_BOGEN der Spezifikation BEH-Spec$_{abstr}$ analog zu Regel $r_{AV}9$ um eine Operation $m\_o\_p\_erw$ erweitert werden muß, um zu prüfen, ob die vorgenommene Handlung auch angeordnet war.



L_$r_{AB}5$ K_$r_{AB}5$ R_$r_{AB}5$

Eine weitere (hier nicht dargestellte) Regel abstrahiert bei der Blutentnahme vom konkret abgenommenen Blut. Die Kantenbeschriftung $blut(PatId)$ wird geändert in $PatId$.Wir erhalten damit die Spezifikation BEH-Spec$_{IE}$ der IST-Essenz von Behandlung.

Das resultierende Netz, die IST-Essenz des Netzes Angeordnete Behandlung (AB), ist die Transformation des Netzes ABa mit der Regel $r_{AB}5$:



ABessenz: Angeordnete Behandlung, IST-Essenz

3.3.3 Labor-Ablauf

Da es den Rahmen dieser Arbeit sprengen würde, auch für die Bereiche *Labor – Ablauf*, *Herzkatheterisierung* und *Entlassung* wie bisher sämtliche Regeln explizit anzugeben, beschreiben wir für diese drei Bereiche lediglich die Auswirkungen der Regelmengen, die wir in Abb.3.7 darstellen, und geben die resultierenden Netze der IST-Essenz an.

Beginnen wir mit den Regeln aus der Regelmenge R_{KB}, die die IST-Essenz des Netzes Kleines Blutbild (KB) bestimmen:

- Regel $r_{KB}1$ abstrahiert von der im Labor arbeitenden Medizinischen Assistentin: Die ein- und aus-gehenden Kanten der Stelle MTA werden gelöscht. Die Stelle MTA selbst bleibt zunächst aus Gründen der Kompatibilität von Transformationsschritten und horizontalen Strukturierungsmechanismen erhalten, denn sie tritt auch im Schnittstellennetz U5 einer möglichen *Union* der Netze KB und DB auf.

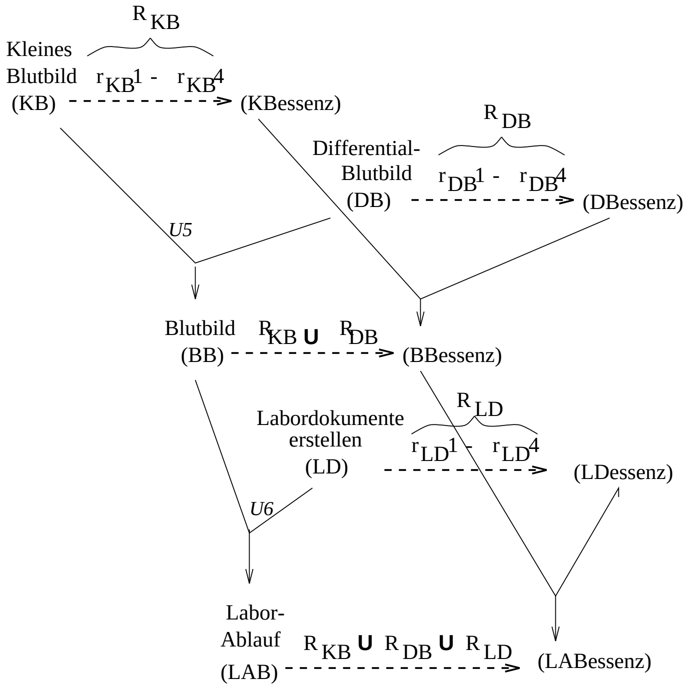
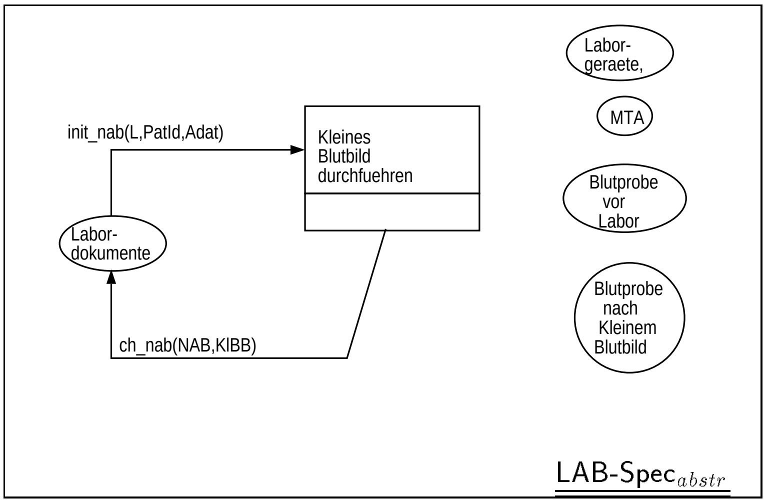


Abbildung 3.7: Labor-Ablauf: Strukturierung und Transformation der Netze

- Regel $r_{KB}2$ abstrahiert von den technischen Details der Blutbilddurchführung: Die beiden Transitionen *Vorbereitung des Kleinen Blutbildes* und *Nachbereitung* werden gelöscht, ebenso wie die Stellen *Gerät und Blut bereit* und *Kleines Blutbild fertig*. Die Stellen *Laborgeräte, Blutprobe vor Labor* und *Blutprobe nach Kleinem Blutbild* bleiben wiederum (isoliert) erhalten, da sie in den Schnittstellennetzen U5 bzw. U6 liegen.

- Regel $r_{KB}3$ abstrahiert von der vom Blutbildgerät ausgegebenen Coulter-Karte (CK). Da die Daten der Coulter-Karte direkt auf den Notfall-Labor- und Anforderungsbogen (NAB) übertragen werden, modellieren wir hier nur noch die einmalige Datenerfassung mittels NAB, wie sie ja auch mittels computergestützten Systems nur einmal erfaßt werden sollen. Statt drei Coulter-Karten mit den Werten des Kleinen Blutbildes zu bedrucken, werden diese Werte nun direkt auf den NAB geschrieben. Die Kantenbeschriftung der Kante von der Stelle *Labordokumente* zur Transition wird also durch den Term $init\_nab(L, PatId, Adat)$ ersetzt, und die Beschriftung der Kante mit umgekehrter Richtung durch $ch\_nab(NAB, KlBB)$.

Das resultierende Netz der Anwendung der Regeln $r_{KB}1$ bis $r_{KB}3$ auf KB ist die IST-Essenz von KB:

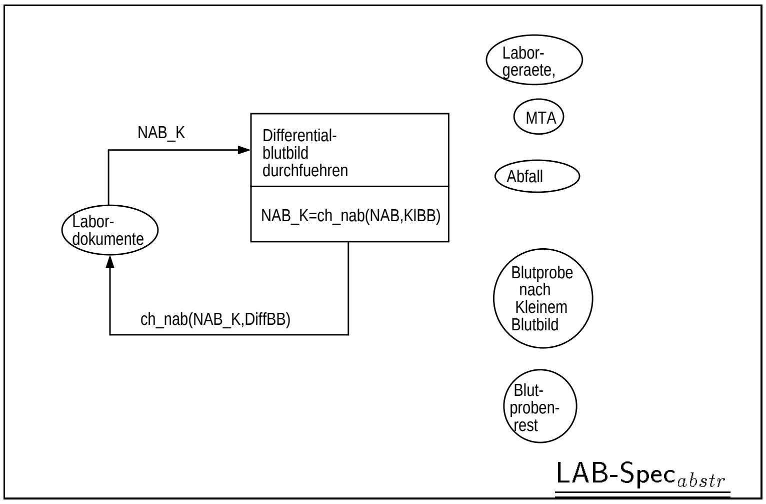


KBessenz: Kleines Blutbild, IST-Essenz

Kommen wir nun zu den Regeln aus der Regelmenge R_{DB}, die die IST-Essenz des Netzes Differential-Blutbild (DB) bestimmen:

- Regel $r_{DB}1$ abstrahiert ebenso wie $r_{KB}1$ von der MTA, die die Analyse durchführt.
- Analog abstrahiert Regel $r_{DB}2$ von den technischen Details der Analyse: Die Transition *Vorbereitung des Differential-Blutbildes* wird gelöscht, ebenso die Stelle *Objektträgersammelbehälter*. *Abfall*, *Laborgeräte* und *Blutprobenrest*, sowie *Blutprobe nach Kleinem Blutbild* bleiben als Objekte aus $U5$ bzw. $U6$ als isolierte Stellen erhalten. Die beiden Transitionen *Histologie* und *Auszählung und Klassifizierung von Zellen* werden zu einer Transition *Diferentialblutbild durchführen* zusammengefaßt.
- Regel $r_{DB}3$ abstrahiert wiederum von der Coulter-Karte (CK). Analog zu $r_{DB}3$ werden die Differential-Blutbild-Werte nur noch einmal, und zwar auf dem NAB, vermerkt.

Die sukzessive Anwendung der Regeln $r_{DB}1$ bis $r_{DB}3$ liefert das Netz DBessenz:

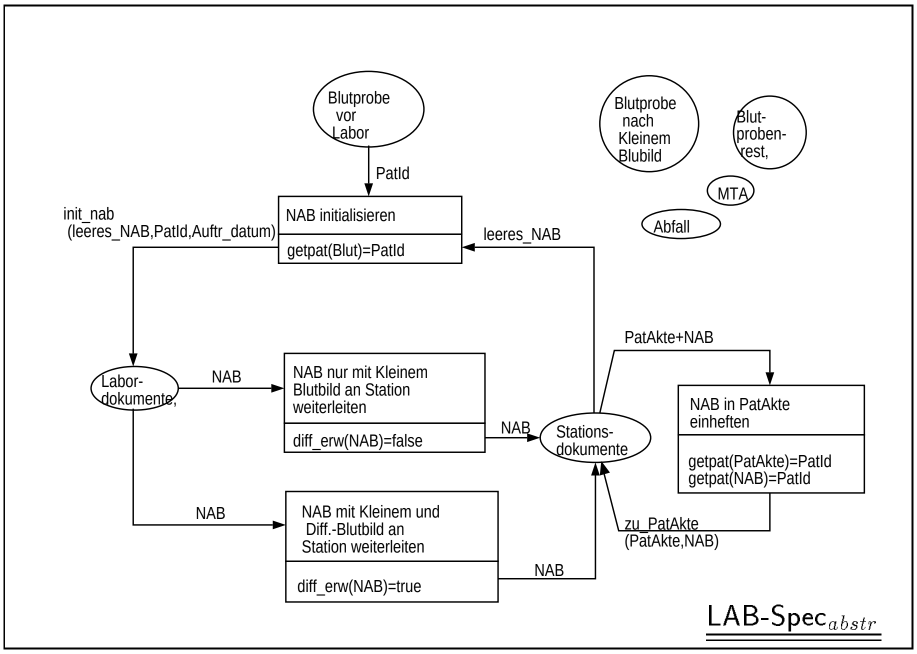


DBessenz: Differential-Blutbild, IST-Essenz

Es fällt auf, daß sich die Regeln der Regelmengen R_{KB} und R_{DB} sehr ähneln. Man kann in der Tat die Regelanzahl reduzieren, wenn man die Regeln direkt angepaßt für das Netz **Blutbild (BB)** formuliert, das aus der *Union* der Netze **KB** und **DB** hervorgeht. Allerdings verliert man damit die Möglichkeit, die Regeln auch auf die lokalen Teilnetze, noch vor der *Union*-Bildung, anzuwenden, was aber eine wichtige Eigenschaft der regelbasierten Verfeinerung ist.

Die Regeln aus der Regelmenge R_{LD} zur Bestimmung der IST-Essenz des Netzes **Labordokumente erstellen(LD)** haben neben der Abstraktion vom Personal (MTA, Stationskraft) den Sinn der Abstraktion von der Coulter-Karte. Deren Initialisierung, und somit auch die Stelle $Coulter - Karten - Stapel$, fallen weg, ebenso die Übertragungen der Blutbild-Daten von der Coulter-Karte auf den NAB, da die Eintragungen nunmehr nach der Messung direkt auf den NAB erfolgen. Bei allen Kanten, die mit Termen zur Sorte CK beschriftet sind, werden diese Terme gelöscht. Die Eintragung der NAB-Werte in das Dokument $Fieberkurve$ entfällt ebenfalls, da auch hier eine (aus Computersystem-Sicht) unnötige Kopie vorgenommen wird.

Es ergibt sich bei Anwendung der Regeln aus R_{LD} das Netz **(LDessenz)**, aus dem über *Union*-Bildung mit den Netzen **KBessenz** und **DBessenz** mit den Schnittstellennetzen **U5**, bestehend aus der Stellenmenge $\{MTA, Laborgeräte, Labordokumente,$ *Blutprobe nach Kleinem Blutbild*$\}$, und **U6**, bestehend aus der Stellenmenge {*Abfall, Blutprobe vor Labor, Blutprobe nach Kleinem Blutbild, Labordokumente, MTA, Blutprobenrest*} das Netz **LABessenz** hervorgeht, die IST-Essenz des gesamten Labor-Ablaufs.



LDessenz: Labordokumente erstellen, IST-Essenz

Die Spezifikation LAB-Spec$_{abstr}$ entspricht nun der Spezifikation DHZB-NAB. Die Teilspezifikationen COULTER-KARTE, BLUT, LABOR-GERÄTE und PERSONAL wurden gelöscht.

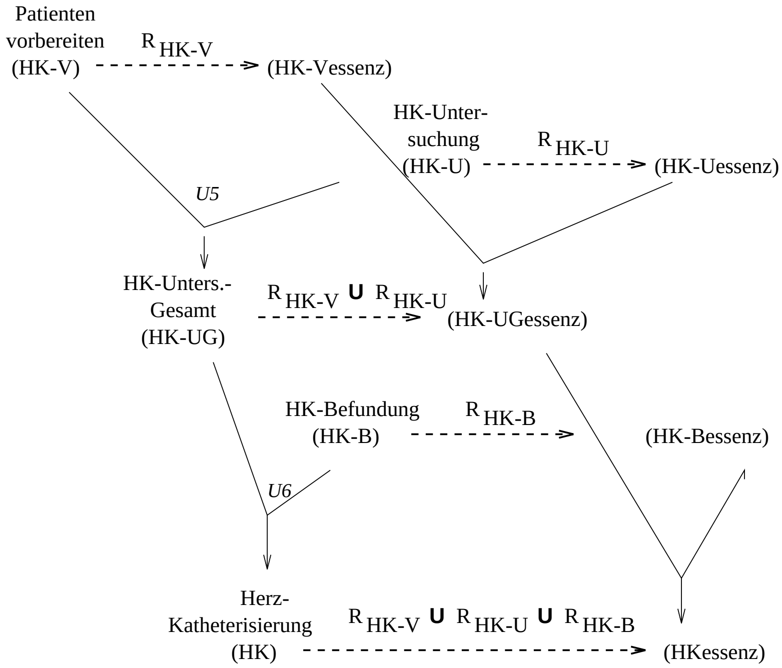


Abbildung 3.8: Herzkatheterisierung: Strukturierung und Transformation der Netze

3.3.4 Herzkatheterisierung

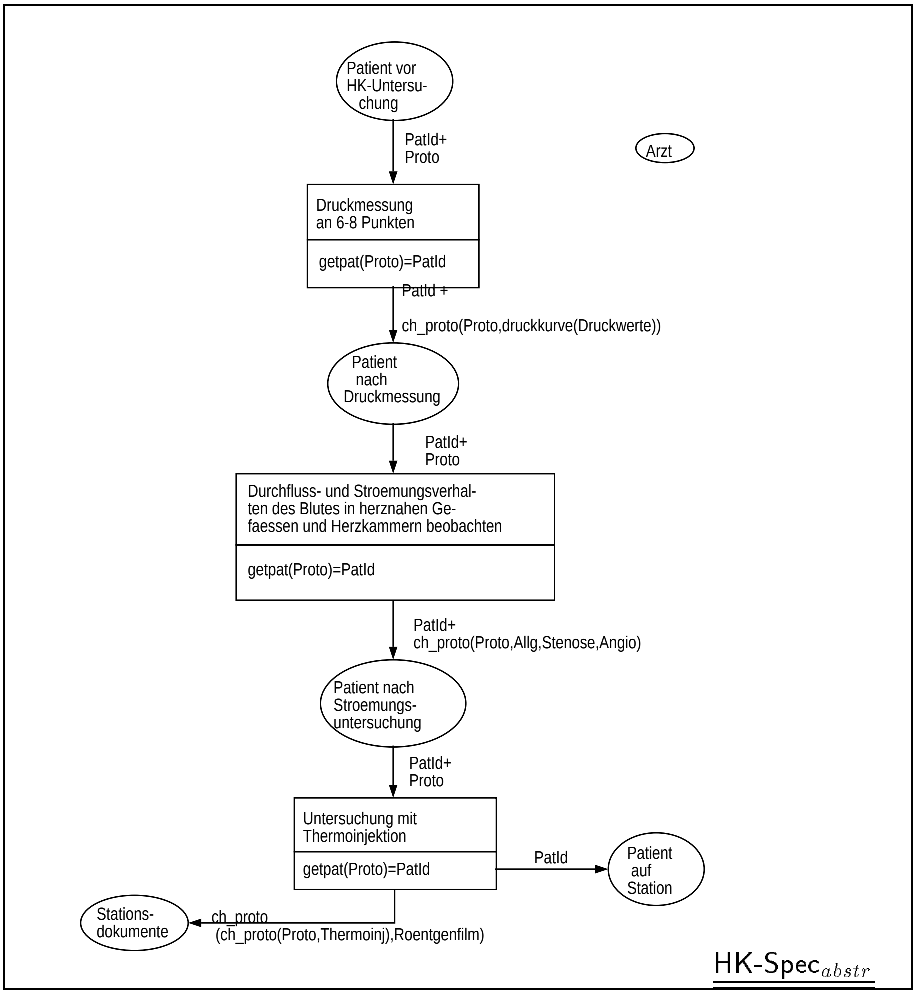
Wir beschreiben die Auswirkungen folgender Regelmengen bei der Transformation der Netze aus dem Bereich *Herzkatheterisierung* in die IST-Essenz:

Die Regeln der Regelmenge R_{HK-V} abstrahieren von den an den HK-Vorbereitungen beteiligten Arzt und Patienten, wobei die Stelle *Arzt* aus Kompatibilitätsgründen erhalten bleibt.

Das resultierende Netz der IST-Essenz bei Anwendung der Regeln aus R_{HK-V}, das Netz $(HK - Vessenz)$, entspricht dem Netz $(HK - V)$ ohne den mit *Arzt* beschrifteten Kanten. Statt mit dem Term *Patient* sind die betreffenden Kanten nun mit *PatId* beschriftet. Die Schaltbedingung $getpat(Patient) = PatId$ entfällt dann jeweils.

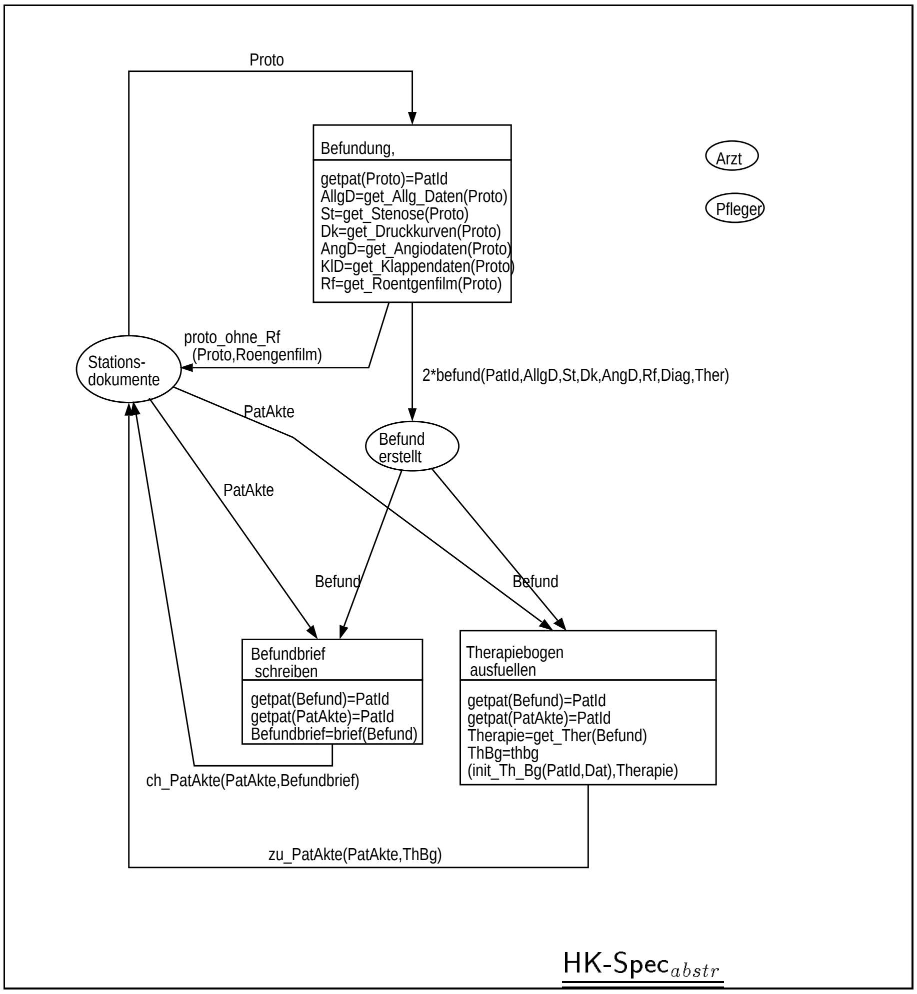
Die Regeln der Regelmenge R_{HK-U} abstrahieren einerseits vom beteiligten Personal, und andererseits von technischen Details wie den verwendeten Geräten (Röntgengerät, Katheter, Druckmeßgerät, Kontrastmittel und Thermomedium). Außerdem werden Tätigkeiten wie *HK legen und Röntgengerät anschalten* bzw. *HK ziehen und Röntgengerät ausschalten* gelöscht, da sie keine systemrelevanten Daten erzeugen, löschen oder verändern.

Das resultierende Netz der IST-Essenz bei Anwendung der Regeln aus R_{HK-U} ist das Netz $(HK - Uessenz)$:



HK-Uessenz: HK-Untersuchung, IST-Essenz

Mit den Regeln der Menge R_{HK-B} wird vom beteiligten Personal und der Archivierung der Röntgenfilme abstrahiert. Das resultierende Netz der IST-Essenz bei Anwendung der Regeln aus R_{HK-B}, das Netz $(HK - Bessenz)$, entspricht dem Netz $(HK - Befundung)$ ohne den mit $Arzt$ bzw. $Pfleger$ beschrifteten Kanten.



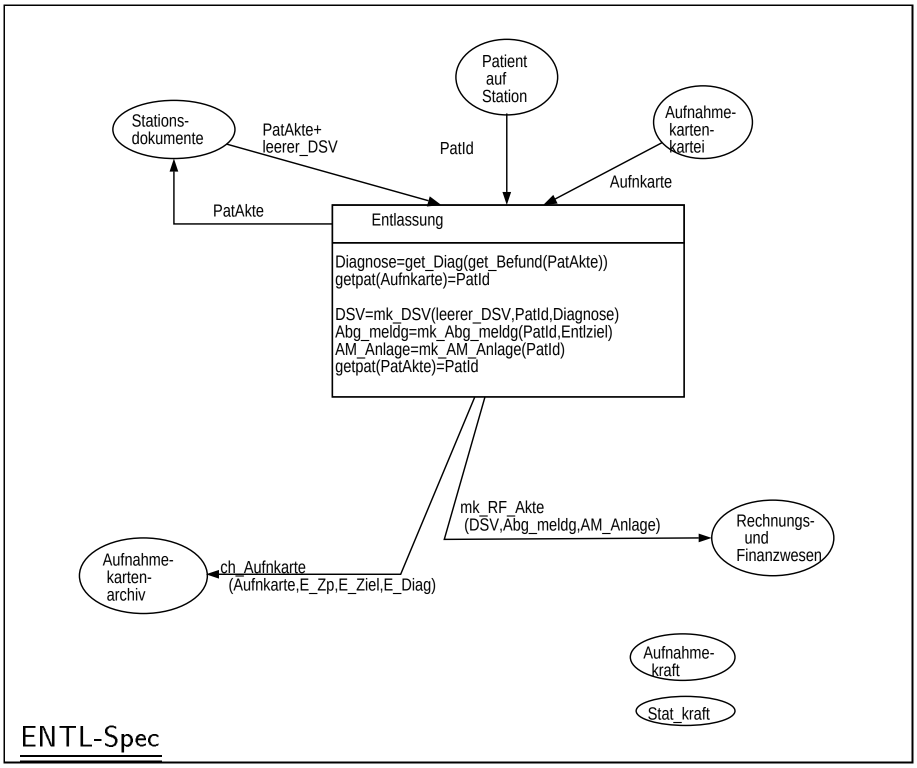
HK-Bessenz: HK-Befundung, IST-Essenz

Die Spezifikation HK-Spec$_{abstr}$ enthält nicht mehr die Teilspezifikationen HK-GERÄTE und PERSONAL.

3.3.5 Entlassung

Die Regeln, die auf das Netz Entlassung angewendet werden, (R_E), abstrahieren ebenfalls vom beteiligten Personal (Aufnahmekraft, Stationskraft). Außerdem werden die Tätigkeiten *Entlassungsdokumente auf Station erstellen und an die Anmeldung schicken, Entlassungsdaten auf der Aufnahmekarte eintragen* und *Entlassungsunterlagen ablegen* in einer einzigen Transition *Entlassung* integriert (analog zum Vorgehen bei der Entwicklung der IST-Essenz von Aufnahme, siehe Kap. 3.3.1).

Das resultierende Netz der IST-Essenz bei Anwendung der Regeln aus R_E ist das Netz ENTLessenz:



ENTLessenz: Entlassung, IST-Essenz

Analog zur Konstruktion des IST-Systems erhält man das Netz zur Beschreibung der IST-Essenz des Gesamtsystems, DHZB-Aufenthalt_IST-Essenz, durch *Union*-Bildung der jeweiligen IST-Essenz-Teilnetze gemäß dem Übersichtsdiagramm in Abb.3.3.

Allerdings hat das Netz noch sehr viele isolierte Stellen, die unsere bisherigen Regeln aus Kompatibilitätsgründen erhalten hatten. Als letzte Regel unserer Gesamttransformation zur IST-Essenz wenden wir deshalb auf DHZB-Aufenthalt_IST-Essenz die sogenannte „Garbage Collection"-Regel an. Diese Regel enthält in der linken Seite eine einzige Stelle beliebigen Namens. Das Kontextnetz und die rechte Regelseite sind leer, die Spezifikationen aller drei Netze ebenfalls. Aufgrund der *Gluing Condition* (siehe Kapitel 4.3, Satz 4.3.2) ist sichergestellt, daß diese Regel nur auf isolierte Stellen angewendet werden kann. Mit ihr löschen wir sämtliche isolierten Stellen aus dem Netz und erhalten dann das Netz DHZB-Aufenthalt_IST-Essenz ohne isolierte Stellen.

3.4 Transformation der IST-Essenz in die Funktionale Essenz

Die funktionale Essenz unseres Systems spiegelt, im Unterschied zur IST-Essenz, die Funktionen als Sicht auf die Datenbank wider. Es werden Operationen spezifiziert zum Lesen, zum Neueintragen und zum Ändern von Daten der Datenbank. Die Spezifikationen sind noch unabhängig von realen Hardware- und Software-Konfigurationen, d.h. wir arbeiten unter der Annahme „perfekter Technik“ (z.B. Speicher unbegrenzt, Zugriffszeit minimal).

Da die hauptsächlichen Unterschiede zwischen IST-Essenz und funktionaler Essenz Ergänzungen im Spezifikationsteil betreffen, und die Spezifikationen der funktionalen Essenz dann im wesentlichen den formalen SPECTRUM-Spezifikationen aus dem KORSO-Projekt [SNM+93] entsprechen, ist es im Rahmen dieser Arbeit nicht notwendig, den Übergang von der IST-Essenz zur funktionalen Essenz für alle bisher beschriebenen Netze und Spezifikationen durchzuführen. Wir werden uns daher in diesem Kapitel exemplarisch mit der Entwicklung der funktionalen Essenz aus der IST-Essenz für den Arbeitsbereich *Behandlung* (Kapitel 3.3.2) befassen.

Zusätzlich zu den in der IST-Essenz spezifizierten Systemeigenschaften kommen in der funktionalen Essenz noch Sicherheitsüberprüfungen hinzu (siehe auch [Ren94]):
Wir unterscheiden

- *Zugriffskontrollmechanismen*, die sicherstellen, daß nur jeweils berechtigten Personen Zugang zu den Daten gewährt wird. Auch dürfen die gespeicherten Daten nicht beliebig übertragbar, ausdruckbar oder langfristig speicherbar sein; dies sind Anforderungen aus Gründen des Datenschutzes und werden erst in Schritt 4 der Anforderungsanalyse berücksichtigt (siehe Abb. 3.2 auf Seite 30);
- *Benutzungssicherheit*, die gewährleistet, daß eine Fehlbedienung des Systems nicht zum Verlust von Daten führt, daß die ausgeführten Funktionen im Falle von Fehleingaben oder Systemfehlern weitgehend umkehrbar sind, und daß die Reaktionszeit des Systems akzeptabel ist; dies sind Anforderungen aus Sicht der sicheren Benutzung, die wiederum erst Schritt 4 der Anforderungsanalyse berücksichtigt werden (siehe Abb. 3.2 auf Seite 30);
- *Korrektheitsprüfungen*, die z.B. testen, ob eingegebene Daten aus zulässigen Wertebereichen stammen, ob alle patientenbezogenen Daten jederzeit der richtigen Person zugeordnet werden können und ob bestimmte definierte Relationen zwischen den Daten bestehen (Integritätsbedingungen, z.B.: das Aufnahmedatum ist früher als das Entlassungsdatum, oder: es darf keine Anordnungen ohne Inhalt geben); diese Anforderungen aus Sicht der medizinischen Tätigkeiten müssen innerhalb der Spezifikation der funktionalen Essenz berücksichtigt werden.

Geändert werden auch die in der IST-Essenz verwendeten Datenstrukturen: Sie werden nicht mehr in Anlehnung an die verwendeten Papierdokumente spezifiziert. Es entfallen damit Mehrfacheintragungen derselben Daten in unterschiedliche Dokumente; dafür wird mit Schlüsseln gearbeitet, die die verschiedenen Datensätze miteinander in Beziehung setzen.

In den Regeln zur Transformation der IST-Essenz in die funktionale Essenz des Bereiches *Behandlung* werden konkret folgende Änderungen vorgenommen:

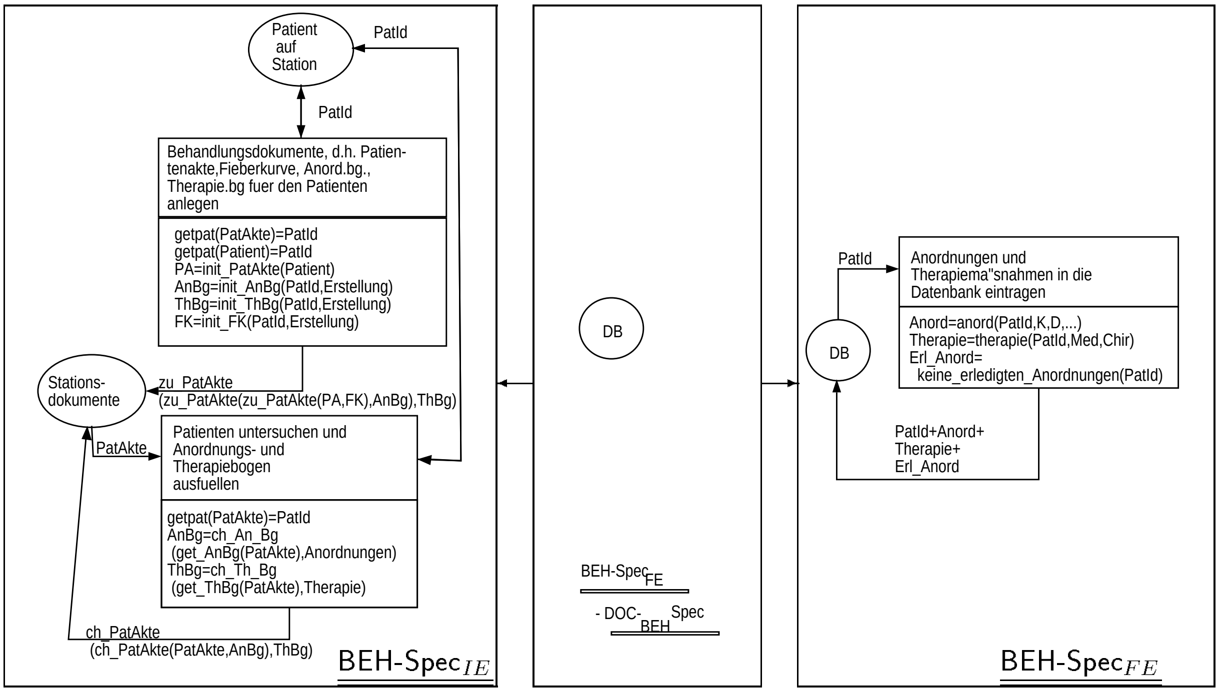
- die Datenbank wird als Sorte DB in der Spezifikation und als Stelle DB im Netz spezifiziert;
- die Patientenakte und die Fieberkurve existieren nicht mehr als Datentypen; sie sind eine Sicht auf mehrere Entities der Datenbank;
- die Tätigkeiten des Einheftens in die Patientenakte entfallen;
- die Tätigkeiten des Übertragens von schon einmal aufgenommenen Daten entfallen; in diesem Zusammenhang entfallen die Teilspezifikation $DHZB - NAB$ und $HK - \ddot{U}berweisung$, weil die Daten in anderen Arbeitsbereichen (*Labor – Ablauf* und $Herzkatheterisierung$) bereits in die Datenbank eingetragen werden;
- somit entfällt auch die Tätigkeit *Behandlungsdokumente anlegen*, da es keine Behandlungsdokumente mehr gibt;
- die Tätigkeit *Patienten untersuchen und Anordnungsbogen und Therapiebogen ausfüllen* wird geändert in *Anordnungen und Therapiemaßnahmen in DB eintragen*;
- die Stelle $Stationsdokumente$ wird umbenannt in DB, ebenso wie alle anderen „Dokumente"-Stellen im gesamten Netz der IST-Essenz;
- die eingetragenen Datensätze (Anordnungen, Therapie, Vitalwerte) werden um den Fremdschlüssel $PatId$ ergänzt;
- die Stelle *Blutprobe vor Labor* entfällt, da stattdessen ein Blutbildauftrag in die DB geschrieben wird, der im Labor gelesen wird.

Zusätzlich werden über die Regeln Integritätsbedingungen hinzugefügt. So darf es z.B. keine Anordnung oder Therapiemaßnahme ohne Inhalt geben und eine Anordnung, die bereits ausgeführt wurde (Status *erledigt*), darf nicht noch einmal ausgeführt werden.

Wir wenden die Regeln lokal auf die Teilnetze der IST-Essenz VMessenz ($Union$ von AVessenz, Seite 106 und RVessenz,Seite 106), auf BVessenz, Seite 109 und auf ABessenz, Seite 111 an, und erhalten durch mehrfache Anwendung der $Union$-Operation das Netz BEH_Funkt der funktionalen Essenz des Bereiches $Behandlung$. Die dazugehörige Spezifikation $\underline{\underline{\text{BEH-Spec}_{FE}}}$ enthält die Datensätze und Operationen zum Eintragen, Ändern und Löschen in der Datenbank.

Generell behalten die Regeln die bisherige Stelle $Stationsdokumente$ bei, da die Rolle dieser Stelle nach wie vor ist, alle patientenrelevanten Daten der Behandlung auf der Station aufzunehmen. Sie wird in den Regeln in DB (Datenbank) umbenannt, und die Terme, die sie dann enthält, lassen sich nicht mehr als Dokumente, sondern als Entities der Datenbank auffassen (siehe dazu auch [SNM+93]). Die Kompatibilität zur horizontalen Strukturierung wird durch die Stellenumbenennung nicht beeinflußt. Die Morphismen einer $Union$- oder $Fusion$-Operation nach Anwendung einer Regel unterscheiden sich von den Morphismen vor der Regelanwendung lediglich durch die Umbenennung der Stelle $Stationsdokumente$ in DB, sind also isomorph. Das Abziehen und Hinzufügen der $PatId$ aus der bzw. in die DB dient der Kontrolle, ob es aktuell einen Patienten in der DB gibt, zu dem der verwendete Fremdschlüssel in den neuen bzw. veränderten Einträgen paßt.

Die erste Regel $r_{FE}1$ löscht die Transition *Behandlungsdokumente anlegen* aus dem Netz (BVessenz), und überführt das bisherige Ausfüllen von Anordnungs- und Therapiebogen in das Eintragen von Anordnungen und Therapiemaßnahmen in die Datenbank. Dabei wird auch eingetragen, daß zu diesem Zeitpunkt noch keine der Anordnungen erledigt wurden. Die Anordnungen sollen vor iher Ausführung daraufhin geprüft werden, ob sie bereits erledigt wurden, und nach ihrer Ausführung als *erledigt* markiert werden.



L_$r_{FE}1$ K_$r_{FE}1$ R_$r_{FE}1$

Der Morphismus von K_$r_{FE}1$ nach L_$r_{FE}1$ bildet dabei die Stelle *DB* auf die Stelle *Stationsdokumente* ab, das heißt, es wird eine Umbenennung vorgenommen.
Aus der Spezifikation der IST-Essenz BEH-Spec$_{IE}$ werden die Spezifikationen der Datentypen *PatAkte*, *Anordnungsbogen*, *Therapiebogen* und *Fieberkurve* gelöscht. Die Spezifikation ohne die entsprechenden Sorten und Operationen heißt BEH-Spec$_{IE}$-DOC$_{BEH}$-Spec. Die Teilspezifikation Anordnungsbogen wird umbenannt in Anordnungen und enthält weiterhin die Sorte *Anordnungen*, sowie zusätzliche Sorten und Operationen zum Überprüfen, ob eine Anordnung bereits ausgeführt wurde. Führen wir solche Änderungen analog in allen Teilspezifikationen durch, so erhalten wir damit insgesamt die Spezifikation der funktionalen Essenz BEH-Spec$_{FE}$. Abb. 3.9 zeigt exemplarisch die Spezifikationsmorphismen, bezogen auf die bisherige Teilspezifikation ANORDN_BOGEN (siehe auch die Spezifikation des IST-Systems auf Seite 52 und die Erweiterung zur IST-Essenz auf Seite 104).

Wir geben hier die Teilspezifikation Anordnungen als Ausschnitt der Spezifikation BEH-Spec$_{FE}$ an:

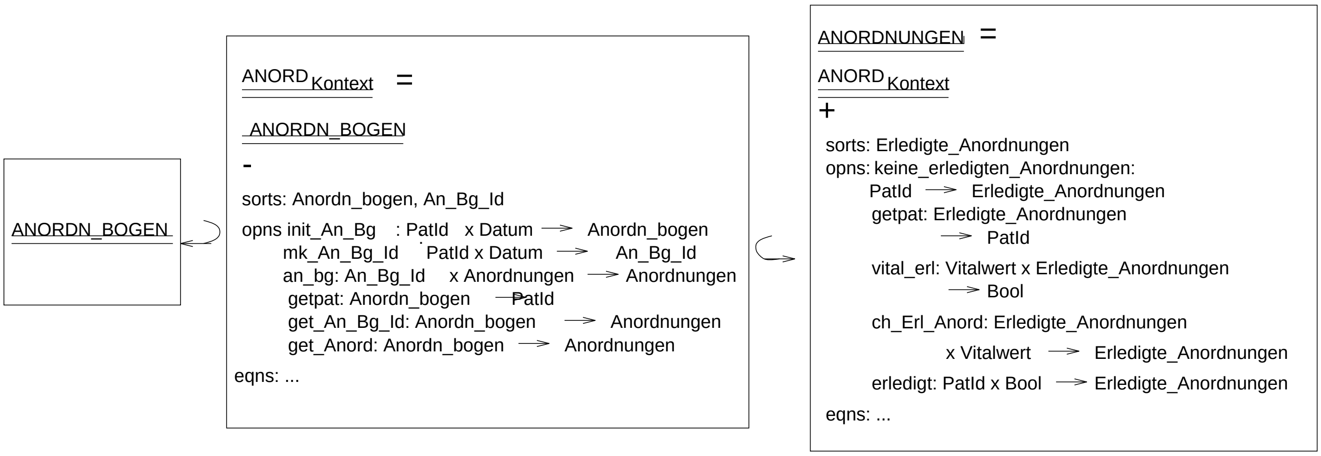


Abbildung 3.9: Beispiel eines Spezifikationsmorphismus von der Spezifikation der IST-Essenz zur Spezifikation der funktionalen Essenz

ANORDNUNGEN = PATID + VITALWERT

Die Anordnungen werden vom Arzt nach einer Untersuchung des Patienten in die Datenbank eingetragen
Sie betreffen Messungen bestimmter Werte
(Vitalwerte: Blutdruck, Puls, Temperatur, ...), aber auch Untersuchungen auf der Intensivstation, die Verabreichung von Medikamenten und Infusionen, sowie die Entnahme einer Blutprobe für eine Laboranalyse.

sorts: Anordnungen, Erledigte_Anordnungen
opns:
keine_Anordnungen: PatId → Anordnungen
keine_erledigten_Anordnungen: PatId → Erledigte_Anordnungen

– *Anordnungen identifizieren*
getpat: Anordnungen → PatId
getpat: Erledigte_Anordnungen → PatId

– *Vitalwerte-Anordnungen überprüfen*
vital_erw: Vitalwert × Anordnungen → Bool
vital_erl: Vitalwert × Erledigte_Anordnungen → Bool
– *eine Anordnung als erledigt markieren*
ch_Erl_Anord: Erledigte_Anordnungen × Vitalwert → Erledigte_Anordnungen

– *(Anordnungen: Tupel vom Arzt gewünschter Pflegemaßnahmen)*

– *12 Anordnungen gibt es maximal:*
– *1. Kleines Blutbild, 2. Diff.blutbild, 3. Blutabnahme,*
– *4. Blutdruckmessg, 5. MAD-Messg., 6. Puls-Messg.,*
– *7. Temp.-Messg, 8. ZVD-Messg., 9. Ein-/Ausfuhr,*
– *10.Intensivstat.messg., 11. Medikation, 12. Perfusion*
anord: PatId $\times$ Bool^{12} $\rightarrow$ Anordnungen
erledigt: PatId $\times$ Bool^{12} $\rightarrow$ Erledigte_Anordnungen

– *Anordnungen lesen*
klBB_erw, diffBB_erw, blut_erw, blutdruck_erw, mad_erw,
puls_erw, temp_erw, zvd_erw, ein/aus_erw, ips_erw,
med_erw, perf_erw: Anordnungen $\rightarrow$ Bool

– *Erledigte Anordnungen lesen*
klBB_erl, diffBB_erl, blut_erl, blutdruck_erl, mad_erl,
puls_erl, temp_erl, zvd_erl, ein/aus_erl, ips_erl,
med_erl, perf_erl: Anordnungen $\rightarrow$ Bool

eqns:

– *Anordnungen identifizieren*
getpat(keine_Anordnungen(Id)) = Id;
getpat(anord(Id, B1, .., B12)) = Id;
getpat(keine_erledigten_Anordnungen(Id)) = Id;
getpat(erledigt(Id, B1, .., B12)) = Id;

– *Vitalwerte-Anordnungen überprüfen*
vital_erw(v(Puls,Zp,Id),A) = puls_erw(AO);
vital_erw(v(Temp,Zp,Id),AO) = temp_erw(AO);
vital_erw(v(BDs,BDd,Zp,Id),AO) = blutdruck_erw(AO);
vital_erw(v(ZVD,Zp,Id),AO) = zvd_erw(AO);
vital_erw(v(MAD,Zp,Id),AO) = mad_erw(AO);
vital_erw(v(Einf,Ausf,Zp,Id),AO) = ein/aus_erw(AO);

vital_erl(v(Puls,Zp,Id),AO) = puls_erl(AO);
vital_erl(v(Temp,Zp,Id),AO) = temp_erl(AO);
vital_erl(v(BDs,BDd,Zp,Id),AO) = blutdruck_erl(AO);
vital_erl(v(ZVD,Zp,Id),AO) = zvd_erl(AO);
vital_erl(v(MAD,Zp,Id),AO) = mad_erl(AO);
vital_erl(v(Einf,Ausf,Zp,Id),AO) = ein/aus_erl(AO);

– *Erledigte Anordnungen als erledigt markieren*
ch_Erl_Anord(EA, v(Puls,Zp,Id) =
erledigt(getpat(EA),true, diffBB_erl(EA), blut_erl(EA),
blutdruck_erl(EA), mad_erl(EA),puls_erl(EA), temp_erl(EA), zvd_erl(EA),
ein/aus_erl(EA), ips_erl(EA),med_erl(EA) , perf_erl(EA))

ch_Erl_Anord(EA, v(Temp,Zp,Id) =
erledigt(getpat(EA),klBB_erl(EA), true, blut_erl(EA),
blutdruck_erl(EA), mad_erl(EA),puls_erl(EA),temp_erl(EA), zvd_erl(EA),
ein/aus_erl(EA), ips_erl(EA),med_erl(EA) , perf_erl(EA))

-
-
-

ch_Erl_Anord(EA, v(Temp,Zp,Id) =
erledigt(getpat(EA),klBB_erl(EA) ,diffBB_erl(EA), blut_erl(EA),
blutdruck_erl(EA), mad_erl(EA),puls_erl(EA),temp_erl(EA),zvd_erl(EA),
ein/aus_erl(EA), ips_erl(EA),true, perf_erl(EA))

ch_Erl_Anord(EA, v(Temp,Zp,Id) =
erledigt(getpat(EA),klBB_erl(EA) ,diffBB_erl(EA), blut_erl(EA),
blutdruck_erl(EA), mad_erl(EA),puls_erl(EA),temp_erl(EA),zvd_erl(EA),
ein/aus_erl(EA), ips_erl(EA),med_erl(EA),true)

– *Anordnungen lesen*

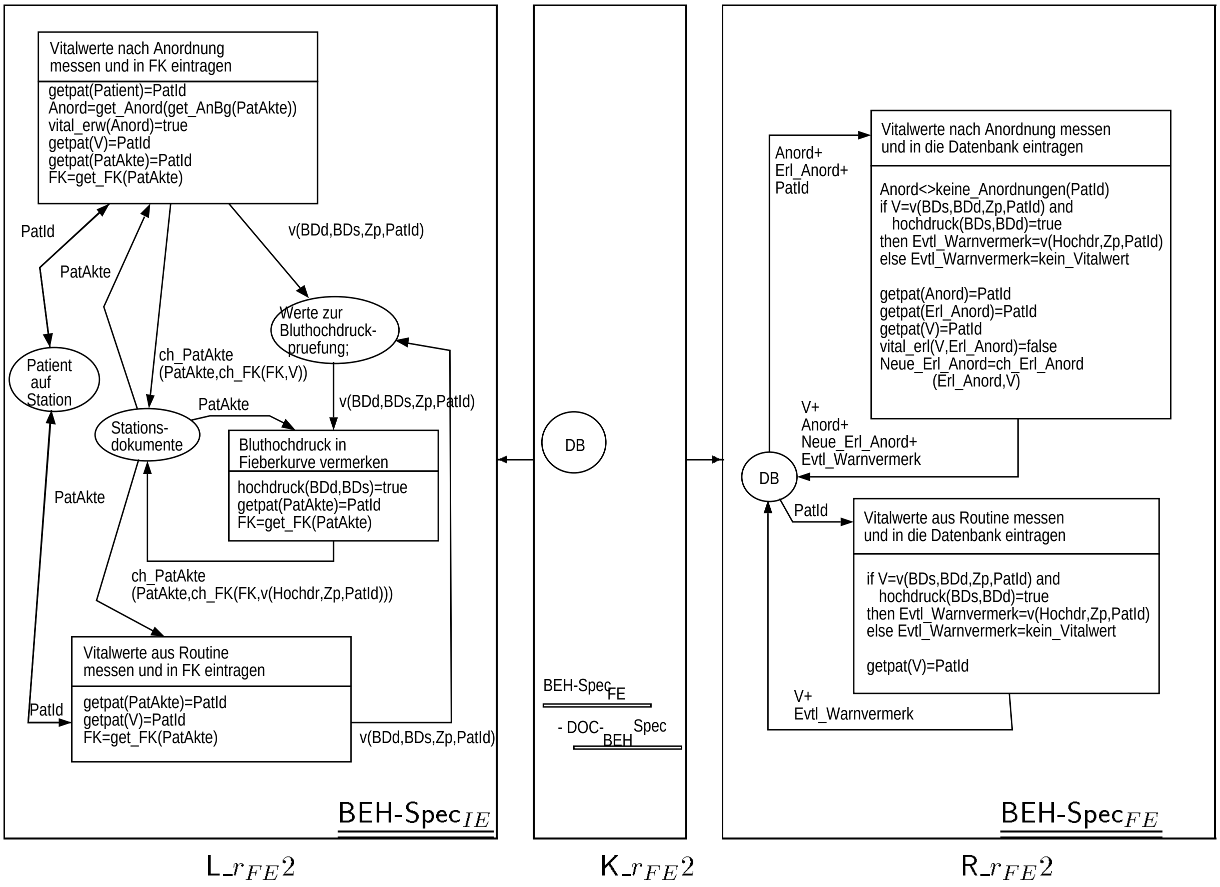
klBB_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = K;
diffBB_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = D;
blut_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = B;
blutdruck_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = BD;
mad_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = MAD;
puls_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = P;
temp_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = T;
zvd_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = ZVD;
ein/aus_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = EA;
ips_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = IPS;
med_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = MED;
perf_erw(anord(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = PER;

– *Erledigte Anordnungen lesen*

klBB_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = K;
diffBB_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = D;
blut_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = B;
blutdruck_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = BD;
mad_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = MAD;
puls_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = P;
temp_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = T;
zvd_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = ZVD;
ein/aus_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = EA;
ips_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER)) = IPS;

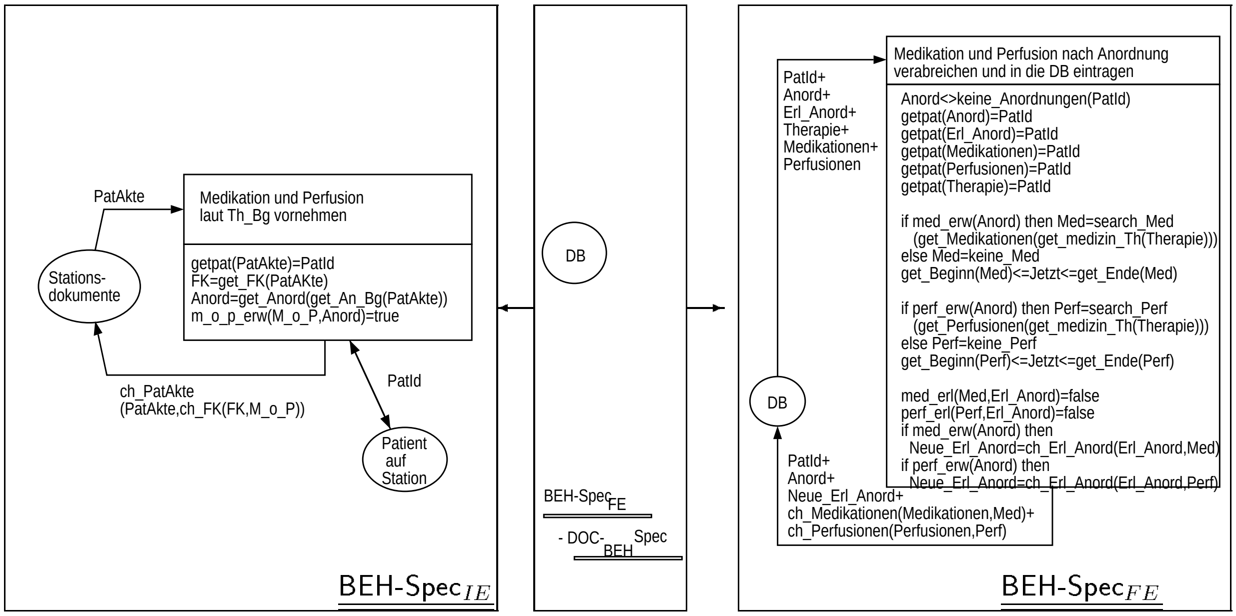
$$\mathsf{med\_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER))} = \mathsf{MED};$$
$$\mathsf{perf\_erl(erledigt(K,D,B,BD,MAD,P,T,ZVD,EA,IPS,MED,PER))} = \mathsf{PER};$$

In $r_{FE}2$ wird das Eintragen der gemessenen Vitalwerte in die Fieberkurve überführt in das Eintragen der Werte in die Datenbank. Dabei werden angeordnete Vitalwertemessungen nur durchgeführt, wenn sie noch nicht als *erledigt* markiert waren. Die Bluthochdruckprüfung wird in die beiden Transitionen *VW nach Anordnung messen und eintragen* und *VW nach Routine messen und eintragen* integriert.



Kommen wir nun zu den Regeln, die das Netz Angeordnete Behandlung, IST-Essenz (ABessenz), Seite 111, in die funktionale Essenz überführen. Wir geben drei Regeln an, die jeweils eine der drei Transitionen verändern. Alle drei Regeln können auch parallel ausgeführt werden.

In Regel $r_{FE}3$ spezifizieren wir das Eintragen von verabreichten Medikationen und Perfusionen in die Datenbank:

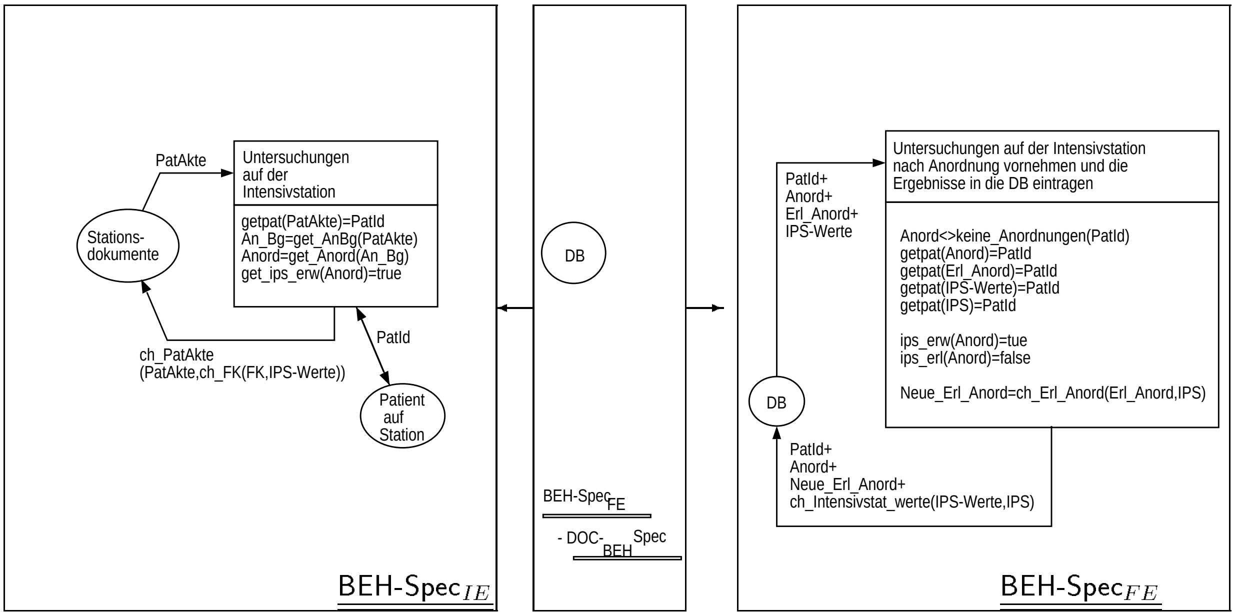


L_$r_{FE}3$ K_$r_{FE}3$ R_$r_{FE}3$

In der Spezifikation BEH$_{FE}$ müssen dann die Selektoren
get_Medikationen: Medizinische_Therapie $\rightarrow$ Medikationen und
get_Perfusionen: Medizinische_Therapie $\rightarrow$ Perfusionen
definiert sein. Die Operation *ch_Erl_Anord* wird auch auf die Anordnungen erweitert, die nicht die Vitalwertemessungen betreffen. Sie markieren die aktuell verabreichte Medikation oder Perfusion als erledigt:
ch_Erl_Anord: Erledigte_Anordnungen $\times$ Medikation $\rightarrow$ Erledigte_Anordungen
ch_Erl_Anord: Erledigte_Anordnungen $\times$ Perfusion $\rightarrow$ Erledigte_Anordungen

An dieser Stelle lassen sich in den Schaltbedingungen der Transition auch Integritätsbedingungen formulieren, die die verabreichten Medikamente betreffen: Sind z.B. die verschiedenen angeordneten Medikamente miteinander verträglich? Man kann sich dazu eine Medikamentendatenbank vorstellen, in der Eigenschaften der Medikamente und Relationen zwischen verschiedenen Medikamenten gespeichert sind.

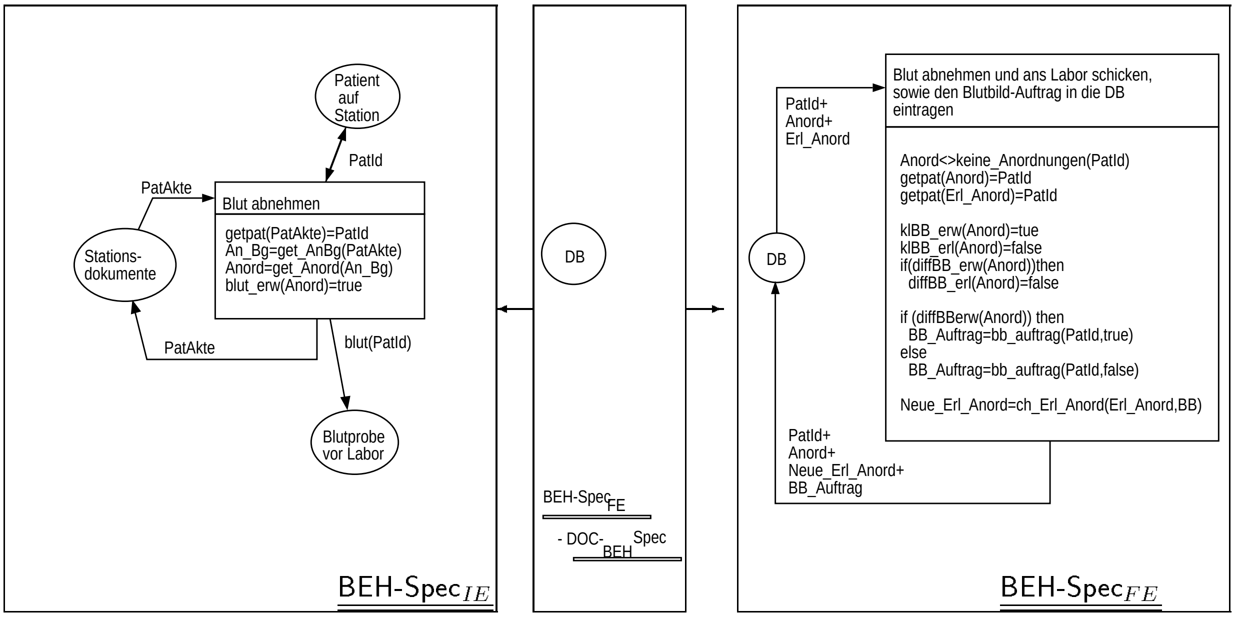
Regel $r_{FE}4$ modifiziert die Untersuchungen auf der Intensivstation dahingehend, daß die Untersuchungswerte nun direkt in die Datenbank eingetragen werden:



L_$r_{FE}4$ K_$r_{FE}4$ R_$r_{FE}4$

Im Netz der funktionalen Essenz werden damit eine Reihe von Meßergebnissen der Intensivstation in die entsprechenden Listen von IPS-Werten eingetragen.

Das aus Anwendung der folgenden Regel $r_{FE}5$ resultierende Netz gibt den Auftrag an das Labor, von der Blutprobe des Patienten das Kleine Blutbild bzw. Kleines und Differentialblutbild zu erstellen, in die Datenbank ein:

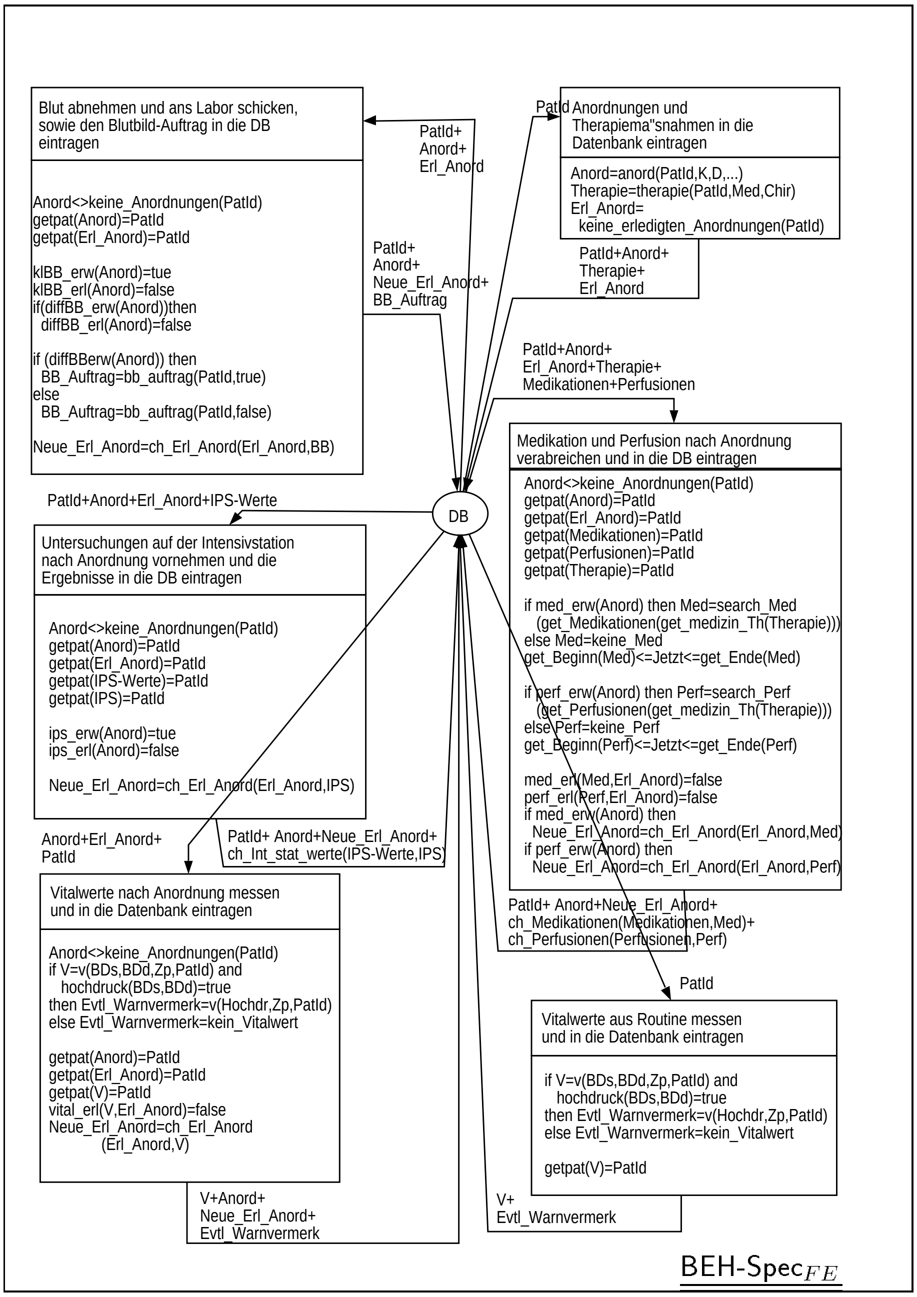


L_$r_{FE}5$ K_$r_{FE}5$ R_$r_{FE}5$

In der Spezifikation der funktionalen Essenz kommen dann die Sorte $BB\_Auftrag$ und die Operation $bb\_auftrag$ neu hinzu:

bb_auftrag: PatId $\times$ Bool $\rightarrow$ BB_Auftrag.

Das gesamte Netz **Behandlung: Funktionale Essenz** läßt sich durch wiederholte $Union$ über den Teilnetzen konstruieren:



Behandlung: Funktionale Essenz

3.5 Bewertung der Fallstudie und Vergleich mit dem KORSO-Projekt

In diesem Kapitel bewerten wir unsere Fallstudie HDMS-AHL im Vergleich mit den im KORSO-Projekt verwendeten Methoden zur Anforderungsanalyse von HDMS-A. Wir stellen fest, welche Verbesserungen wir mit AHL-Netzen und den Konzepten der regelbasierten Abstraktion/Verfeinerung und der horizontalen Strukturierung von Netzen erreicht haben und motivieren damit auch die im nächsten Kapitel folgende mathematische Formalisierung und Fundierung der verwendeten Konzepte von AHL-Netzen.

- **Integration von statischen und dynamischen Aspekten**
  Die Definition von AHL-Netzen (siehe Def. 4.1.1), bei denen die Netzstruktur das dynamische Verhalten des Systems beschreibt, während die statischen Aspekte durch abstrakte Datentypen (algebraische Spezifikationen) modelliert werden, führt zur Integration von statischen und dynamischen Aspekten des Systems innerhalb eines formalen Modells.

  Im KORSO-Projekt bestand die Modellierung der Arbeitsabläufe des IST-Systems aus Bedingungs-/Ereignis-Netzen. Die Spezifikation der Datentypen erfolgte davon unabhängig in der algebraischen Spezifikationssprache SPECTRUM. Dabei ließen sich Veränderungen von Daten in den Arbeitsabläufen nur informell darstellen. Für die späteren Entwicklungsschritte der Anforderungsanalyse wurden andere Formalismen verwendet.

- **Horizontale Strukturierung**
  Der Formalismus der AHL-Netze gestattet die horizontale Strukturierung des Systems, d.h. die Komposition von Teilnetzen zu einem Gesamtnetz. In unserer Fallstudie wurden die beiden horizontalen Strukturierungsmethoden *Union* (siehe Def. 4.3.6) und *Fusion* (siehe Def. 4.3.4) häufig verwendet, da ein System von so großer Komplexität ohne Strukturierung nicht konsistent und überschaubar modelliert werden kann. Die Möglichkeit, AHL-Netze formal horizontal zu strukturieren, basiert auf der Eigenschaft der Kovollständigkeit der Kategorie **AHLNET** der AHL-Netze (siehe Satz 4.2.3).

  Im KORSO-Projekt wurden die verwendeten Bedingungs-/Ereignis-Netze informell hierarchisch strukturiert dargestellt. Für die algebraischen Spezifikationen wurden die für diesen Formalismus üblichen Strukturierungstechniken verwendet.

- **Regelbasierte Entwicklung**
  Die Entwicklungsschritte der Anforderungsanalyse, d.h. die Schritte vom IST-System über die IST-Essenz zur funktionalen Essenz des Systems, werden in unserer Fallstudie durch regelbasierte Verfeinerungs- bzw. Abstraktionsschritte modelliert. Mit den Regeln werden die Änderungen des Modells innerhalb eines kleinen, überschaubaren Kontexts ausgedrückt. Die Anwendung der Regeln (Ableitungen) betten diese Änderungen dann in einen größeren Kontext ein. Ein großer Vorteil der Verwendung regelbasierter Verfeinerung ist die Beibehaltung eines einheitlichen Formalismus. Es ändert sich lediglich die Sichtweise auf das Modell von einer benutzerorientierten Spezifikation bei der IST-Analyse bis hin zur zielsprachennahen Design-Spezifikation. In unserer Fallstudie werden beim Schritt von

der IST-Essenz zur funktionalen Essenz die Netzstrukturen immer einfacher, dafür aber die Schaltbedingungen der Transitionen, sowie die Spezifikationen der dort verwendeten Operationen immer komplexer. Dies entspricht genau unserer Absicht: Wir entfernen uns auf dem Weg zur Anforderungsanalyse immer mehr vom detailgetreuen IST-System, bei dem die Netze sehr ausführlich (benutzergerecht), aber ohne Zusammenhang zu dem beabsichtigten Computersystem, spezifiziert waren. Die Netze der funktionalen Essenz des Systems sind schon rein syntaktisch weit näher an einer Implementierung, da viele Operationen hier als algebraische Spezifikation spezifiziert sind, und nicht mehr (wie noch bei der IST-Essenz) als Teil der Netzstruktur. So werden z.B. Verzweigungen in der funktionalen Essenz über if-then-else-Abfragen in der Schaltbedingung modelliert, während sie bis hin zur IST-Essenz über zwei nicht unabhängig aktivierbare Transitionen modelliert werden.

Der nächste Schritt auf dem Weg zur Implementierung würde dann die Netzstruktur vollkommen auflösen, und nur noch die algebraische Spezifikation aller Operationen erzeugen. Diese Entwurfsspezifikation läßt sich dann in eine ausführbare, bzw. zielsprachenangepaßte (in unserem Fall: datenbanksprachenangepaßte) Spezifikation transformieren, siehe Abb. 3.2.

Im KORSO-Projekt wurden für die verschiedenen Entwicklungsphasen unterschiedliche Spezifikationsformalismen verwendet: Bedingungs-/Ereignis- Netze und SPECTRUM-Spezifikationen für das IST-System, semiformale Datenflußdiagramme und Entity-Relationship-Modelle für die funktionale Essenz. Die Übergänge zwischen den verschiedenen Formalismen konnten nur informell angegeben werden. Außerdem mußten die semiformalen E-/R- und Datenflußdiagramme dann noch in eine formale SPECTRUM-Spezifikation übersetzt werden, um die weitere Entwicklung formal vollziehen zu können.

- **Kompatibilität von horizontaler und vertikaler Strukturierung**
  Um die Techniken der regelbasierten Abstraktion/Verfeinerung und der horizontalen Strukturierung wirklich für die Modellierung komplexer Systeme anwendbar zu machen, ist es erforderlich, die Kompatibilität zwischen horizontaler und vertikaler Strukturierung sicherzustellen. Das bedeutet, daß wir das selbe Ergebnis erhalten, unabhängig davon ob wir erst Teilnetze über Regeln verfeinern und die resultierenden Netze über Strukturierungsoperationen komponieren, oder ob wir erst die Teilnetze komponieren und dann auf das resultierende Netz die gleichen Verfeinerungsregeln anwenden. Die Voraussetzung zur Kompatibilität bei AHL-Netzen ist formal gegeben mit Satz 4.3.3 auf Seite 144 und den darauf aufbauenden Sätzen 4.3.5 auf Seite 145 und 4.3.7 auf Seite 146.

- **Vermeidung von Seiteneffekten bei der Entwicklung**
  Um die Konsistenz der Entwicklung sicherzustellen, reicht die Eigenschaft der Kompatibilität allerdings nicht aus. Wir brauchen zusätzlich eine Bedingung, daß wir über Regeln nicht Teile eines Systems löschen oder verändern können, die von anderen, nicht gelöschten Teilen noch gebraucht werden. Diese Bedingung ist die *Gluing Condition*, formal beschrieben in Satz 4.3.2. Die *Gluing Condition* gewährleistet also, daß wir nur Ableitungen ohne Seiteneffekte erhalten.

Kapitel 4

Formale Definition von Algebraischen High-Level-Netzen

In diesem Kapitel wollen wir die bisher informell eingeführten Eigenschaften von AHL-Netzen formalisieren. Wir geben zunächst die Definition von AHL-Netzen an und formalisieren ihr Schaltverhalten. Dann definieren wir die Konstruktion eines zu einem AHL-Netz verhaltensäquivalenten Stellen/Transitionen-Netzes, was für die Definition der Semantik eines AHL-Netzes von Bedeutung ist. Wir geben die Kategorie der AHL-Netze und ihre Eigenschaften an. Dann führen wir die formale Basis für regelbasierte Verfeinerung in Anlehnung an Hgh-Level-Replacement-Systeme ([PER95]) ein und definieren horizontale Strukurierung als Konstruktion von Kolimiten in **AHLNET**. Abschließend zeigen wir die Kompatibilität dieser Konstruktionen mit Ableitungen.

Wir veranschaulichen die formalen Begriffe jeweils durch Beispiele aus unserer Fallstudie.

4.1 Definition und Schaltverhalten

Definition 4.1.1 (AHL-Netz) *Ein Algebraisches High-Level-Netz (AHL-Netz) N = (SPEC, P, T, pre, post, A, cond) besteht aus:*

- *SPEC = (S,OP,E), einer algebraischen Spezifikation,*
- *P, einer Menge von Stellen,*
- *T, einer Menge von Transitionen,*
- $pre, post : T \longrightarrow (T_{OP}(X) \times P)^{\otimes}$, *zwei Funktionen, die jeder Transition ein Element der freien Abelschen Gruppe über dem kartesischen Produkt aus der Menge der Terme mit Variablen* $T_{OP}(X)$ *und der Menge der Stellen P zuweist. Die Funktionen ordnen also einer Transition ihren Vor- bzw. Nachbereich zu, d.h. die mit Termen der Termalgebra* $T_{OP}(X)$ *beschrifteten Kanten in die bzw. aus der Transition.*
- *A, einer SPEC-Algebra,*

- *cond*: $T \longrightarrow \mathcal{P}_{fin}(EQNS(SIG))$, *einer Funktion, die jeder Transition* $t \in T$ *eine endliche Menge* $cond(t)$ *von Gleichungen über der Signatur* $SIG = (S, OP)$ *von SPEC und der Variablenmenge* X *zuweist. Die Menge* $cond(t)$ *heißt auch Schaltbedingungen der Transition* t.

Bemerkung :
Einfachheitshalber nehmen wir an, daß die Variablenmenge X, die in der Termalgebra $T_{OP}(X)$ verwendet wird, für alle AHL-Netze gleich ist. Wir erhalten X durch Indizierung einer festen Menge X_{fix} von Variablen mit Sorten aus S der Signatur $SIG = (S, OP)$, also

$$X = \{x_s | x \in X_{fix}, s \in S\} \cong X_{fix} \times S.$$

Außerdem wird vorausgesetzt, daß Variablen $Var(t)$ einer Transition t gleichen Namens auch dieselbe Sorte haben. $Var(t)$ enthält die Variablen, die im Vor- und/oder im Nachbereich $pre(t)$ und $post(t)$ auftreten, sowie die Variablen in der Schaltbedingung $cond(t)$:

$$x_{s1}, x_{s2} \in Var(t) \Longrightarrow s1 = s2$$

Diese Bedingung hält man bei der Entwicklung eines realistischen Modells mit AHL-Netzen schon intuitiv ein.

Wir stellen als Beispiel ein Teilnetz des IST-Systems unserer Fallstudie, das Netz (BV2) (Pflegemaßnahmen anordnen), in Tupelschreibweise dar:

BV2 = (*BEH-Spec*, P_{BV2}, T_{BV2}, *pre*, *post*, A_{BEH}, *cond*) ist gegeben durch:

$\underline{\underline{BEH-Spec}}$	=	PAT_AKTE + PERSONAL, siehe Kapitel 3.2.2;
P_{BV2}	=	{Patient auf Station, Arzt, Stationsdokumente};
T_{BV2}	=	{Patient untersuchen und Anordnungs- und Therapiebogen ausfüllen (abgekürzt P_An_Th)};
pre	=	{P_An_Th $\mapsto$ (Patient, Patient auf Station) $\oplus$ (Arzt, *Arzt*) $\oplus$ (PatAkte, *Stationsdokumente*) };
post	=	{P_An_Th $\mapsto$ (Patient, Patient auf Station) $\oplus$ (Arzt, *Arzt*) $\oplus$ (ch_PatAkte(ch_PatAkte(PatAkte,AnBg),ThBg)), *Stationsdokumente*) };
A_{BEH}	=	die Termalgebra $T_{SPEC}(M)$ zur Spezifikation BEH-Spec über den Mengen $(M_s)_{s \in S}$, wobei $M_{Name} = \{name_1, ..., name_n\}$, $M_{Arzt} = \{arzt1, ..., arztn\}$, $M_{Kostenträger} = \{kostenträger\}$, $M_{PatId} = \{patid1, ..., patidn\}$ und $M_s = \emptyset$ für alle anderen $s \in S$
cond	=	{P_An_Th $\mapsto$ {getpat(PatAkte) = PatId $\wedge$ AnBg = ch_An_Bg(get_AnBg(PatAkte),Anordnungen) $\wedge$ ThBg = ch_Th_Bg(get_ThBg(PatAkte), Therapie) };

Bemerkung :
Die Algebra A_{BEH} wurde so konstruiert, daß zu jeder Sorte mindestens ein Grundterm existiert. Dies wäre bei der Wahl der Termalgebra T_{SPEC} nicht der Fall gewesen, da in der Spezifikation auch Sorten (z.B. *Arzt* oder *Name*) existieren, zu denen es keine Terme gibt. Variablen zu diesen Sorten können nur mit Daten einer „realistischen“ Algebra, die auch z.B. die konkreten Arzt-

und Patientennamen liefert, belegt werden. Für solche elementaren Sorten liefert unsere Algebra (vereinfachte) konkrete Daten, alle anderen Sorten werden, wie bei der reinen Termalgebra, aus den entsprechenden Grundtermen zusammengesetzt.

Definieren wir nun formal das Schaltverhalten eines AHL-Netzes. Dazu benötigen wir zunächst den Begriff *Markierung* eines Netzes, der beschreibt, welche Token zu einem bestimmten Zeitpunkt auf welchen Stellen liegen. Eine Markierung m ist also eine Menge von Tupeln (a, p), wobei a Datenelement der Algebra A ist, und $p \in P$ Stelle von N:

Definition 4.1.2 (Markierung) *Sei ein AHL-Netz $N = (SPEC, P, T, pre, post, A, cond)$ gegeben. Dann ist eine Markierung m von N gegeben als Element des Stellenvektors*

$$PV = (A \times P)^{\otimes}.$$

A ist in diesem Fall die disjunkte Vereinigung aller Trägermengen der Algebra A, d.h.

$$A = \biguplus_{s \in S} A_s$$

Als nächstes untersuchen wir, mit welchen Belegungen die Variablen $Var(t)$ einer Transition t konsistent belegbar sind:

Definition 4.1.3 (Konsistente Transitionsbelegungen) *Sei N AHL-Netz, und A definiert wie in Def. 4.1.2. Die Menge CT der konsistenten Transitionsbelegungen ist*

$$\begin{aligned} CT \quad = \quad & \{(t, asg_A) | t \in T, asg_A : Var(t) \to A, \\ & \textit{so daß die Datenelmente in } A \textit{ unter der Belegung } asg_A \\ & \textit{die jeweiligen Gleichungen } cond(t) \textit{ erfüllen } \} \end{aligned}$$

Mithilfe der konsistenten Transitionsbelegungen können wir nun Funktionen definieren, die uns die Datenelemente liefern, die beim Schalten einer Transition aus ihrem Vorbereich abgezogen, bzw. ihrem Nachbereich hinzugefügt werden:

Definition 4.1.4 (A-induzierte Funktionen) *Seien N, A, PV und CT definiert wie oben. Dann sind die A-induzierten Funktionen $pre_A, post_A : CT \to PV$ von N definiert für alle $(t, asg_A) \in CT$ durch*

$$pre_A(t, asg_A) = ASG_A(pre(t))$$

und

$$post_A(t, asg_A) = ASG_A(post(t)).$$

Dabei ist $ASG_A = (T_{OP}(Var(t)) \times P)^{\otimes} \to (A \times P)^{\otimes} = PV$ definiert durch $ASG_A(term, p) = (asg_A(term), p)$ für alle $p \in P$ und $term \in T_{OP}(Var(t))$.

Damit kommen wir zur Hauptdefinition, die den Übergang von einer Markierung durch Schalten einer Transition zur Folgemarkierung eines AHL-Netzes formalisiert:

Definition 4.1.5 (Folgemarkierung) *Seien eine Markierung $m \in PV$ (Def. 4.1.2) und eine konsistente Transitionsbelegung $(t, asg_A) \in CT$ (Def. 4.1.3) eines AHL-Netzes N gegeben. Dann ist die Folgemarkierung $m' \in PV$ von m definiert durch*

$$m' = m \ominus pre_A(t, asg_A) \oplus post_A(t, asg_A).$$

Zur Veranschaulichung betrachten wir das Schaltverhalten der Transition in unserem Beispielnetz BV2 auf Seite 43:

Wählen wir zunächst eine Markierung des Netzes. Eine mögliche Markierung mit Daten der Algebra A_{BEH} ist:

m = (patient(name1,weiblich,29.1.95,kostenträger,patid1), *Patient auf Station*)
$\oplus$ (arzt1, *Arzt*)
$\oplus$ (zu_PatAkte(zu_PatAkte
(init_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),
init_AnBg(patid1,30.1.95)),
init_ThBg(patid1,30.1.95)), *Stationsdokumente*)

Ein mögliche Variablenbelegung $asg_A : Var(P\_An\_Th) \to A$ ist:

$asg_A(Arzt)$	=	arzt
$asg_A(Patient)$	=	patient(name,weiblich,29.1.95,kostenträger,patid1)
$asg_A(PatAkte)$	=	zu_PatAkte(zu_PatAkte (init_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)), init_AnBg(patid1,30.1.95)), init_ThBg(patid1,30.1.95))
$asg_A(Patid)$	=	patid1
$asg_A(Anordnungen)$	=	anord(true,false,...,true)
$asg_A(Therapie)$	=	therapie(chirg_Th(true,false,...,kein_Klappenersatz), med_Th(true,keine_Medikation,keine_Perfusion,text))

Prüfen wir nun, ob die gewählte Variablenbelegung auch eine konsistente Transitionsbelegung von P_An_Th ist; sind die Gleichungen aus $cond(P\_An\_Th)$ unter der Belegung $asg_A(Var(P\_An\_Th))$ erfüllt?

$$
\begin{aligned}
getpat(asg_A(PatAkte)) &= \text{getpat(zu\_PatAkte(zu\_PatAkte} \\
&\quad \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
&\quad \text{init\_AnBg(patid1,30.1.95)),} \\
&\quad \text{init\_ThBg(patid1,30.1.95))} \\
&= \text{getpat(get\_Patient(zu\_PatAkte(...))} \\
&= \text{getpat(pa-Daten2Patient(head\_PA(zu\_PatAkte(...))} \\
&= \text{getpat(patient(name,weiblich,29.1.95,kostenträger,patid1)} \\
&= \text{patid1} \\
&= asg_A(\text{PatId}) \\
AnBg &= \text{ch\_AnBg(get\_AnBg(}asg_A\text{(PatAkte)),}asg_A\text{(Anordnungen))} \\
&= \text{ch\_AnBg(get\_AnBg(zu\_PatAkte(zu\_PatAkte} \\
&\quad \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
&\quad \text{init\_AnBg(patid1,30.1.95)),} \\
&\quad \text{init\_ThBg(patid1,30.1.95))),} \\
&\quad anord(true, false, ..., true)) \\
ThBg &= \text{ch\_ThBg(get\_ThBg(}asg_A\text{(PatAkte)),}asg_A\text{(Anordnungen))} \\
&= \mathit{ch\_ThBg(get\_ThBg(zu\_PatAkte(zu\_PatAkte} \\
&\quad \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
&\quad \text{init\_AnBg(patid1,30.1.95)),init\_ThBg(patid1,30.1.95))),} \\
&\quad \text{therapie(chirg\_Th(true,false,...,kein\_Klappenersatz),} \\
&\quad \text{med\_Th(true,keine\_Medikation,keine\_Perfusion,text))}
\end{aligned}
$$

Bei (P_An_Th, asg_A) handelt es sich offensichtlich um eine konsistente Transitionsbelegung, da die erste Gleichung erfüllt ist, und die Belegung asg_A aus den anderen beiden Gleichungen gültige Belegungen der Variablen $AnBg$ (zur Sorte $Anordnungsbogen$) und $ThBg$ (zur Sorte $Therapiebogen$) erzeugt.

Berechnen wir als nächstes die A-induzierten Funktionen $pre_A(P\_An\_Th, asg_A)$ und $post_A(P\_An\_Th, asg_A)$ für die Transition P_An_Th aufgrund der Menge konsistenter Transitionsbelegungen:

$$
\begin{aligned}
pre_A(P\_An\_Th, asg_A) &= ASG_A(pre(P\_An\_Th)) \\
&= ASG_A((\text{Patient}, \textit{Patient auf Station}) \oplus (\text{arzt}, \textit{Arzt}) \oplus \\
&\quad (\text{PatAkte}, \textit{Stationsdokumente})) \\
&= ASG_A(\text{Patient}, \textit{Patient auf Station}) \oplus \\
&\quad ASG_A(\text{Arzt}, \textit{Arzt}) \oplus \\
&\quad ASG_A(\text{PatAkte}, \textit{Stationsdokumente}) \\
&= (asg_A(\text{Patient}), \textit{Patient auf Station}) \oplus \\
&\quad (asg_A(\text{Arzt}), \text{Arzt}) \oplus \\
&\quad (asg_A(\text{PatAkte}), \textit{Stationsdokumente})) \\
&= (\text{patient(name,weiblich,29.1.95,kostenträger,patid1)}, \textit{Patient auf Station}) \oplus \\
&\quad (\text{arzt1,Arzt}) \oplus \\
&\quad (\text{zu\_PatAkte(zu\_PatAkte} \\
&\quad \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
&\quad \text{init\_AnBg(patid1,30.1.95)), init\_ThBg(patid1,30.1.95)),} \\
&\quad \textit{Stationsdokumente})
\end{aligned}
$$

$$
\begin{aligned}
post_A(P\_An\_Th, asg_A) &= ASG_A(post(P\_An\_Th)) \\
&= ASG_A((\text{Patient}, \textit{Patient auf Station}) \oplus (\text{arzt}, \textit{Arzt}) \oplus \\
&\quad (\text{ch\_PatAkte(ch\_PatAkte(PatAkte,AnBg),ThBg))}, \textit{Stationsdokumente}) \\
&= ASG_A(\text{Patient}, \textit{Patient auf Station}) \oplus \\
&\quad ASG_A(\text{Arzt}, \textit{Arzt}) \oplus \\
&\quad ASG_A(\text{ch\_PatAkte(ch\_PatAkte(PatAkte,AnBg),ThBg)}, \textit{Stationsdokumente}) \\
&= (asg_A(\text{Patient}), \textit{Patient auf Station}) \oplus \\
&\quad (asg_A(\text{Arzt}), \textit{Arzt}) \oplus \\
&\quad (\text{ch\_PatAkte(ch\_PatAkte(}asg_A\text{(PatAkte),}asg_A\text{(AnBg)),}asg_A\text{(ThBg)),} \\
&\quad \textit{Stationsdokumente}) \\
&= (\text{patient(name,weiblich,29.1.95,kostenträger,patid1)}, \textit{Patient auf Station}) \oplus \\
&\quad (\text{arzt1}, \textit{Arzt}) \oplus \\
&\quad (\text{ch\_PatAkte (ch\_PatAkte (zu\_PatAkte (zu\_PatAkte} \\
&\quad \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
&\quad \text{init\_AnBg(patid1,30.1.95)), init\_ThBg(patid1,30.1.95)),} \\
&\quad \text{ch\_AnBg (get\_AnBgn(zu\_PatAkte (zu\_PatAkte} \\
&\quad \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
&\quad \text{init\_AnBg(patid1,30.1.95)), init\_ThBg(patid1,30.1.95)),} \\
&\quad \text{anord(true,false,...,true))),} \\
&\quad \text{ch\_ThBg(get\_ThBg(zu\_PatAkte(zu\_PatAkte} \\
&\quad \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
&\quad \text{init\_AnBg(patid1,30.1.95)), init\_ThBg(patid1,30.1.95)),} \\
&\quad \text{therapie(chirg\_Th(true,false,...,kein\_Klappenersatz),} \\
&\quad \text{med\_Th(true,keine\_Medikation,keine\_Perfusion,text))),} \\
&\quad \textit{Stationsdokumente})
\end{aligned}
$$

Aus diesen, durch die Belegung sehr schnell komplex gewordenen Termen ergibt sich nun unsere Folgemarkierung durch $m' = m \ominus pre_A(P\_An\_Th, asg_A) \oplus post_A(P\_An\_Th, asg_A)$:

$$
\begin{aligned}
m' \quad = \quad & (\text{patient(name,weiblich,29.1.95,kostenträger,patid1)}, \textit{Patient auf Station}) \oplus \\
& (\text{arzt1}, \textit{Arzt}) \oplus \\
& (\text{ch\_PatAkte (ch\_PatAkte} \\
& \text{(zu\_PatAkte (zu\_PatAkte} \\
& \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
& \text{init\_AnBg(patid1,30.1.95)),} \\
& \text{init\_ThBg(patid1,30.1.95)),} \\
& \text{ch\_AnBg (get\_AnBgn(zu\_PatAkte (zu\_PatAkte} \\
& \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
& \text{init\_AnBg(patid1,30.1.95)),} \\
& \text{init\_ThBg(patid1,30.1.95)),} \\
& \text{anord(true,false,...,true))),} \\
& \text{ch\_ThBg(get\_ThBg(zu\_PatAkte(zu\_PatAkte} \\
& \text{(init\_PatAkte(patient(name,weiblich,29.1.95,kostenträger,patid1)),} \\
& \text{init\_AnBg(patid1,30.1.95)),} \\
& \text{init\_ThBg(patid1,30.1.95)),} \\
& \text{therapie(chirg\_Th(true,false,...,kein\_Klappenersatz),} \\
& \text{med\_Th(true,keine\_Medikation,keine\_Perfusion,text))),} \\
& \textit{Stationsdokumente})
\end{aligned}
$$

Trotz der Komplexität der Terme ist ersichtlich, daß die Patientenakte auf der Stelle *Stationsdokumente* nach dem Schalten der Transition *P_An_Th* nun nicht mehr die initialen Werte für *AnBg* und *ThBg* enthält, sondern die ausgefüllten Bögen, auf denen konkrete Anordnungen und Therapiemaßnahmen eingetragen wurden.

Bemerkung:
Das Schaltverhalten von AHL-Netzen kann so erweitert werden, daß sich Markierungsgraphen definieren lassen, deren Knoten mit Markierungen und deren Kanten mit konsistenten Transitionsbelegungen beschriftet sind.

Zu jedem AHL-Netz N existiert bezüglich des Schaltverhaltens der Transitionen ein äquivalentes abelsches Stellen/Transitionen-Netz (P/T-G-Netz, Def. siehe [EGH92],[EPR94]), auch *Flattening*(N) genannt. Es können dann Analyseuntersuchungen (z.B. Invariantenanalyse) für kleinere AHL-Netze N an deren Äquivalent *Flattening*(N) mit bekannten Analysemethoden für P/T-Netze vorgenommen werden. Auch die Semantik eines AHL-Netzes N wird auf dieser Basis als Markierungsgraph des P/T-G-Netzes *Flattening*(N) definiert. Dabei wird allerdings von den dynamischen Aspekten wie Nebenläufigkeit, Synchronisation und Verteiltheit abstrahiert. Außerdem ist die *Flattening*-Konstruktion nicht kompatibel mit der Strukturierung der Netze, d.h. man kann nicht mit demselben Ergebnis einerseits auf zwei AHL-Netze die *Flattening*-Konstruktion anwenden und dann die resultierenden P/T-G-Netze vereinigen oder andererseits die Vereinigung beider AHL-Netze über *Union* bilden und die *Flattening*-Konstruktion auf das resultierende AHL-Netz anwenden.

Es gibt bereits Ansätze, die Semantik von AHL-Netzen unter Einbeziehung der dynamischen Aspekte direkt, ohne den Umweg der *Flattening*-Konstruktion zu definieren [Pad95], auf die wir allerdings in dieser Arbeit nicht eingehen.

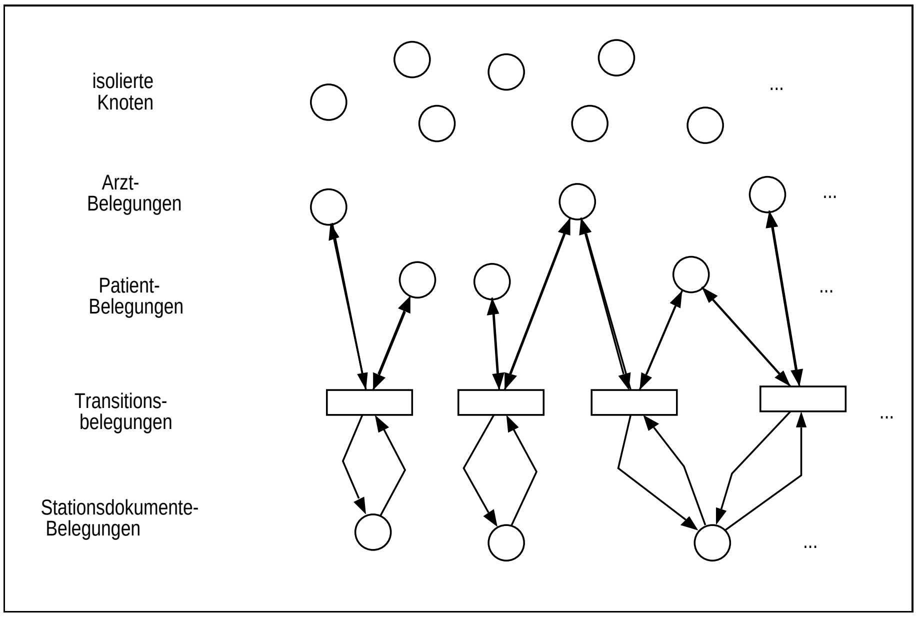
Satz 4.1.6 (Flattening von AHL-Netzen) *Sei N=(SPEC, P, T, pre, post, A, cond) AHL-Netz (Def. 4.1.1). Dann ist das zu N äquivalente P/T-G-Netz* $FLAT(N) = (P_F, T_F, pre_F, post_F)$ *mit* $P_F = (A \times P), T_F = CT, pre_F = pre_A$ *und* $post_F = post_A$*.*

Beweisskizze : N und $FLAT(N)$ haben dieselben Stellenvektoren $PV = (A \times P)^{\otimes} = P_F^{\otimes}$ wegen der Konstruktion von $FLAT(N)$. Da die Transitionen T_F den konsistenten Transitionsbelegungen CT entsprechen und die Vor- und Nachbereichsfunktionen $pre_F = pre_A$ und $post_F = post_A$ gleich definiert wurden, sind die Folgemarkierungen, bestimmt durch $m' = m \ominus pre_A(t, ass_A) \oplus post_A(t, ass_A)$ offensichtlich auch gleich.

Wir veranschaulichen die *Flattening*-Konstruktion anhand unseres Beispielnetzes (BV2): Die Menge der Transitionen in $FLAT(N)$ ist die Menge der konsistenten Transitionsbelegungen von P_An_Th. Sie ist in unserem Fall sehr groß, da es insbesondere für die Variablen *PatAkte*, *Anordnungen* und *Therapie* sehr viele gültige Belegungen gibt, von denen diverse Kombinationen die Gleichungsmenge $cond(P\_An\_Th)$ erfüllen und somit in der Menge CT enthalten sind. Jedes Element von CT entspricht einer Transition von $FLAT(N)$.

Die Menge der Stellen in $FLAT(N)$ entspricht dem Kreuzprodukt $(A \times P)$. Das Netz $FLAT(N)$ enthält demnach sehr viele isolierte Stellen, die nicht über Kanten mit Transitionen verbunden sind. Jede der auftretenden Transitionen hat, wie im AHL-Netz N, genau drei Stellen im Vorbereich und drei Stellen im Nachbereich, die aber für verschiedene Transitionen auch gleich sein können, je nach der Variablenbelegung, über die sie konstruiert wurden.

Das P/T-G-Netz $FLAT(N)$ kann dann graphisch etwa folgendermaßen angedeutet werden:



Flattening von Netz BV2

Ein Beispiel für das vollständige Flattening eines AHL-Netzes, dessen Spezifikation auf sehr kleinen Trägermengen beruht, und daher nur wenige Variablenbelegungen zuläßt, findet sich in [EPR94].

Die Flattening-Konstruktion kann zu einem Funktor von der Kategorie **AHLNET** der AHL-Netze (siehe Kap. 4.2) in die Kategorie der P/T-G-Netze erweitert werden. In [Hum89] [DH90] wird ein ähnlicher Funktor konstruiert von **AHLNET** in die Kategorie der Coloured Petrinetze [Jen92]. Der Funktor definiert die Semantik Algebraischer High-Level-Netze und ist kovollständig, d.h. die Semantik läst sich allgemein über Pushoutkonstruktion aus den Semantiken von Teilnetzen komponieren.

4.2 Die Kategorie AHLNET

In diesem Kapitel definieren wir die Kategorie **AHLNET** der AHL-Netze. Die Eigenschaften dieser Kategorie, insbesondere ihre Kovollständigkeit, bilden eine wichtige Voraussetzung für die Komposition von Netzen (siehe Kapitel 4.3) und für die Kompatibilität von vertikaler und horizontaler Strukturierung (siehe Kapitel 4.3).

Im Fall von AHL-Netzen haben wir es mit Morphismen zwischen Algebren zu unterschiedlichen Spezifikationen zu tun, wir können also nicht einfach auf die bekannten SIG-Homomorphismen zurückgreifen. Aus diesem Grund führen wir zunächst den Begriff *generalisierter Homomorphismus* ein. Dies ist ein Spezialfall eines *generalisierten Morphismus*, wie er im Rahmen von Spezifikationslogiken [EBO91] und indizierten Kategorien [TBG91] verwendet wird.

Definition 4.2.1 (Generalisierter Homomorphismus) *Gegeben sei die Kategorie* **SPEC** *der Spezifikationen mit* $SPEC_i = (S_i, OP_i, E_i)$ *und Algebren* $A_i \in Cat(SPEC_i)$ *mit* $i \in \{1,2\}$. *Dann besteht ein generalisierter Homomorphismus* $g : A_1 \to A_2$ *aus einem Paar* (f_{SPEC}, f_A), *wobei* $f_{SPEC} : SPEC_1 \to SPEC_2$ *Spezifikationsmorphismus ist und* $f_A : A_1 \to V_{f_{SPEC}}(A_2)$ *Homomorphismus mit dem Vergißfunktor* $V_{f_{SPEC}} : Cat(SPEC_2) \to Cat(SPEC_1)$.
Die Komposition zweier generalisierten Homomorphismen $g = (f_{SPEC}, f_A) : A_1 \to A_2$ *und* $g' = (f'_{SPEC}, f'_A) : A_1 \to A_2$ *ist definiert durch* $g' \circ g = (f_{SPEC} \circ f'_{SPEC}, V_{f_{SPEC}}(f'_A) \circ f_A) : A_1 \to A_3$.

Abb. 4.1 veranschaulicht die Konstruktion generalisierter Homomorphismen und ihre Komposition. Die gestrichelten Pfeile sind dabei Funktoren zwischen Kategorien (gepunktete Bereiche). Durchgezogene Pfeile kennzeichnen Morphismen innerhalb einer Kategorie bzw. die generalisierten Homomorphismen g, g' und $g \circ g'$.

Wir können nun die Katerie **GALG** angeben als die Kategorie mit Algebren als Objekten und generalisierten Homomorphismen als Morphismen. Die Kategorie ist vollständig und kovollständig (siehe [TBG91], [Hum89] und [EBO91]). Im Hinblick auf die Erfüllung der Eigenschaften sogenannter HLR-Systeme betrachten wir als Grundlage der Kategorie **AHLNET** die eingeschränkte Kategorie $\mathbf{GALG}_{ISO}$, in der alle Homomorphismen auf Algebren Isomorphismen sind, und die ebenfalls vollständig und kovollständig ist.

Definieren wir nun die Kategorie Algebraischer High-Level-Netze :

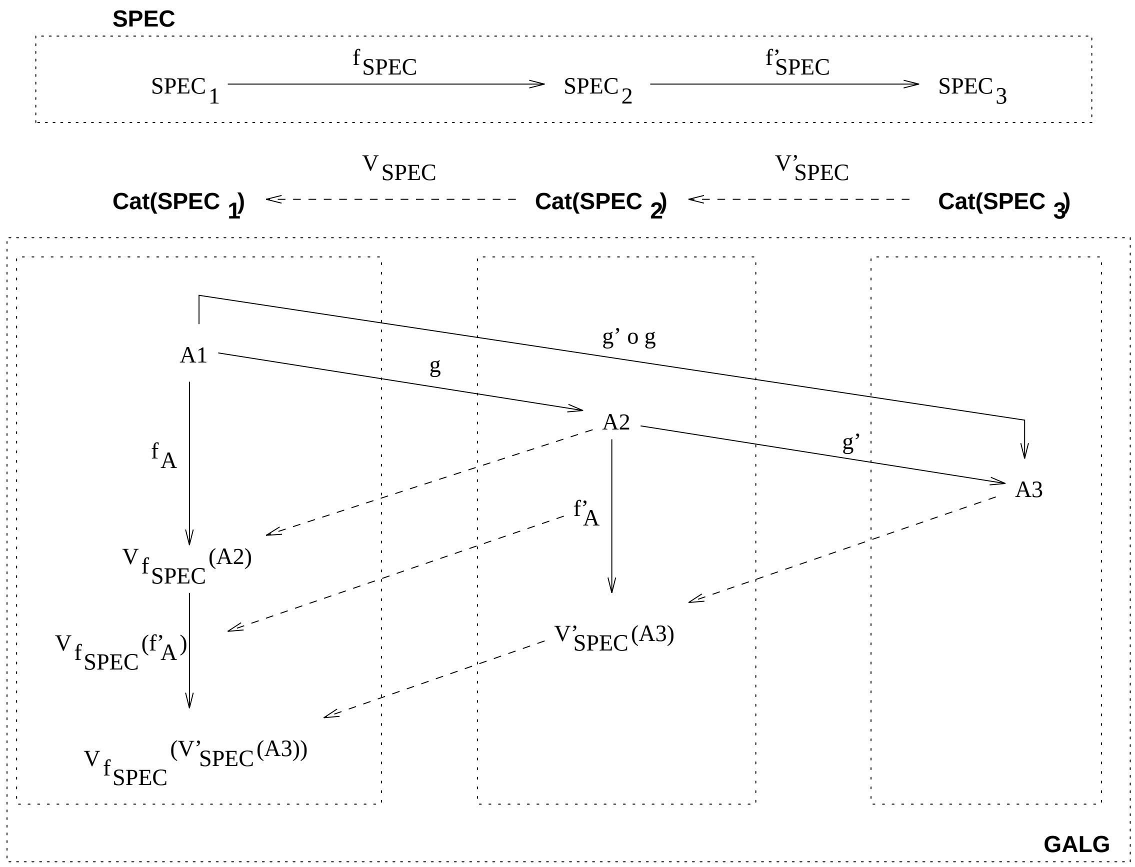


Abbildung 4.1: Konstruktion und Komposition generalisierter Homomorphismen

$$\begin{tikzcd}
\mathcal{P}_{fin}(EQNS(SIG1)) \arrow[d, "\mathcal{P}_{fin}(f^{\#}_{SIG})"'] \arrow[dr, phantom, "(=)"] & T1 \arrow[l, "cond1"'] \arrow[r, shift left, "pre1"] \arrow[r, shift right, "post1"'] \arrow[d, "f_T"] \arrow[dr, phantom, "(=)"] & (T_{OP1}(X1) \times P1)^{\otimes} \arrow[d, "(f^{\#}_{SIG} \times f_P)^{\otimes}"] \\
\mathcal{P}_{fin}(EQNS(SIG2)) & T2 \arrow[l, "cond2"'] \arrow[r, shift left, "pre2"] \arrow[r, shift right, "post2"'] & (T_{OP2}(X1) \times P2)^{\otimes}
\end{tikzcd}$$

Abbildung 4.2: Kategorie **AHLNET**

Definition 4.2.2 (Kategorie AHLNET)
Die Kategorie AHLNET besteht aus Algebraischen High-Level-Netzen N = (SPEC, P, T, pre, post, A, cond) als Objekten (s. Def. 4.1.1). Ein AHL-Netz-Morphismus $f : N1 \to N2$ *ist ein Quadrupel* $f = (f_{SPEC}, f_P, f_T, f_A)$ *mit*

- $f_{SPEC} : SPEC1 \to SPEC2$ *Spezifikationsmorphismus mit* $f^{\#}_{SPEC}(E1) \subseteq E2$, *wobei* $f^{\#}$ *die Erweiterung von f auf Terme und Gleichungen ist;*
- $f_T : T1 \to T2$ *und* $f_P : P1 \to P2$ *Funktionen;*
- $(f_{SPEC}, f_A) : A1 \to A2$ *generalisierter Homomorphismus (siehe Def. 4.2.1) und* $f_A : A1 \to V_{f_{SPEC}}$ *Isomorphismus in Cat(SPEC1),*

so daß das Diagramm in Abb. 4.2 komponentenweise kommutiert.

Folgende wichtige kategorielle Eigenschaft von **AHLNET** erlaubt es einerseits, beliebige Kolimiten zu konstruieren und somit, AHL-Netze horizontal zu strukturieren, und sind andererseits die wesentliche Grundlage für das Konzept der regelbasierten Abstraktion/Verfeinerung sind:

Satz 4.2.3 (Kategorielle Eigenschaften von AHLNET)

1. **AHLNET** *ist kovollständig;*
2. *AHL-Netz-Morphismen bleiben unter Pushouts erhalten;*
3. **AHLNET** *hat Pullbacks.*

Beweisskizze:

1. **AHLNET** ist kovollständig, da alle Kolimiten separat in den zugrundeliegenden Komponenten-Kategorien **SPEC, SET** und $\mathbf{GALG}_{ISO}$ konstruiert werden können (zur Kovollständigkeit der Algebra-Komponente siehe [TBG91] und [EBO91]).

2. AHL-Netz-Morphismen bleiben unter Pushouts erhalten, weil dieselbe Eigenschaft in den Komponenten-Kategorien **SPEC, SET** und $\mathbf{GALG}_{ISO}$ gilt.

3. Pullbacks werden komponentenweise konstruiert, unter Ausnutzung der Tatsache, daß die Funktoren, die zur Konstruktion von $(T_{OP}(X) \times P)^{\otimes}$ verwendet werden, Pullbacks für injektive Morphismen erhalten.

(zum Beweis siehe auch [PER95].

4.3 Strukturierungsmechanismen für AHL-Netze

Wir formalisieren in diesem Kapitel die vertikale Strukturierungstechnik der Transformation von AHL-Netzen (Regelbasierte Abstraktion/Verfeinerung) und die horizontale Strukturierung von AHL-Netzen über die Operationen $Union$ und $Fusion$. Wir prüfen, welche Bedingungen für eine Unabhängigkeit von Ableitungen und $Union$ bzw. $Fusion$ gelten, und zeigen darauf aufbauend die Kompatibilität zwischen horizontalen und vertikalen Strukturierungskonzepten.

Regelbasierte Verfeinerung von AHL-Netzen

Wie bereits in Kapitel 2 eingeführt, entspricht die Anwendung einer Transformationsregel einer Doppelpushout-Konstruktion. Wir formalisieren die dort informell eingeführten Begriffe in folgender Definition:

Definition 4.3.1 (AHL-Netz-Transformationssystem) *1. Eine Regel $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ besteht aus einem Paar von AHL-Netzen L und R, genannt linke und rechte Regelseite, und einem Netz K, genannt Klebenetz oder Schnittstelle, und zwei injektiven AHL-Netz-Morphismen $K \stackrel{l}{\rightarrow} L$ und $K \stackrel{r}{\rightarrow} R$.*

2. *Gegeben seien eine Regel p und ein Netz C (context net), zusammen mit einem Morphismus $K \rightarrow C$, dann ist eine direkte Ableitung $G \stackrel{p}{\Longrightarrow} H$ von einem Netz G in ein Netz H über p gegeben durch die beiden Pushoutdiagramme* **(1)** *and* **(2)** *in*

$$\begin{tikzcd} L \arrow[d, "g1"'] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "g2"'] \arrow[dl, phantom, "(1)"] \arrow[dr, phantom, "(2)"] & R \arrow[d, "g3"] \\ G & C \arrow[l, "c1"] \arrow[r, "c2"'] & H \end{tikzcd}$$

3. *Eine Ableitungssequenz $G \stackrel{*}{\Longrightarrow} H$ (kurz Transformation oder Ableitung) von G nach H bedeutet entweder, daß G zu H isomorph ist, oder daß es eine Folge von $n \geq 1$ direkten Ableitungen gibt, so daß*

$$G = G0 \stackrel{p1}{\Longrightarrow} G1 \stackrel{p2}{\Longrightarrow} \ldots \stackrel{pn}{\Longrightarrow} Gn = H$$

4. *Ein AHl-Netz-Transformationssystem $H = (P, AN_0)$ besteht aus einer Menge P von Regeln und einem Startnetz AN_0 in* **AHLNET**.

Um eine Regel auf ein AHL-Netz G anwenden zu können, muß eine Bedingung, die sogenannte *Gluing Condition*, erfüllt sein, die sicherstellt, daß einerseits keine Stellen/Kantenbeschriftungen ohne die mit ihnen verbundenen Kanten aus dem Netz gelöscht werden, und andererseits keine Sorten aus der Spezifikation gelöscht werden, zu denen noch Daten in der Algebra oder Terme in Schaltbedingungen existieren. Die *Gluing Condition* gewährleistet also, daß das Resultat einer Regelanwendung ein syntaktisch korrektes AHL-Netz ist.

Satz 4.3.2 (Anwendbarkeit von Regeln) *Sei eine Regel $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ gegeben, ein AHL-Netz G und ein AHL-Netz-Morphismus $L \stackrel{g1}{\rightarrow} G$ (Vorkommen von L in G. Dann ist die Regel p auf G anwendbar über $L \stackrel{g1}{\rightarrow} G$, wenn die beiden folgenden Bedingungen erfüllt sind:*

1. *Es gibt ein AHL-Netz C (auch Pushout-Komplement genannt), zusammen mit zwei AHL-Netz-Morphismen $K \stackrel{g2}{\rightarrow} C$ und $C \stackrel{c1}{\rightarrow} G$, sodaß das Diagramm* **(1)** *in Def. 4.3.1 ein Pushout ist.*

2. *Es gib ein Pushout-Objekt H zusammen mit Morphismen $R \stackrel{g3}{\rightarrow} H$ und $C \stackrel{c2}{\rightarrow} H$, sodaß das Diagramm* **(2)** *in Def. 4.3.1 ein Pushout ist*

Wenn beide Bedingungen erfüllt sind, dann kann eine direkte Ableitung $G \stackrel{p}{\Longrightarrow} H$ konstruiert werden.
Die *Gluing Condition* für AHL-Netze kann auch konkret (konstruktiv) definiert werden. Dabei werden die sogenannten *dangling points* angegeben, also die Mengen von Stellen/Kantenbeschriftungen , die mit nicht zu löschenden Kanten verbunden sind, und die Menge von Gleichungen, die auf Schaltbedingungen von nicht zu löschenden Transitionen abgebildet werden. Die *Gluing Condition* ist dann erfüllt, wenn die Objekte aus diesen Mengen über die Regel nicht gelöscht werden. Die formale Definition der *dangling points* für AHL-Netze findet man in [PER95].

Beispiel:
Wenn wir versuchen, mit unserer *Garbage-Collection*-Regel (siehe S. 118) Stellen zu löschen, die nicht isoliert im Netz auftreten, so ist die erste Bedingung von Satz 4.3.2 nicht erfüllt, denn es gibt kein AHL-Netz C, so daß Diagramm **(1)** Pushout wird, wie Abb.4.3 zeigt.

Kommen wir nun zu einigen wichtigen Ergebnissen, die die Kommutativität und Parallelität von Regeln und Ableitungen betreffen. In dieser Arbeit können wir diese Ergebnisse nur kurz beschreiben. Für eine ausführliche formale Version im Kontext von Graphgrammatiken und High-Level-Replacement-Systemen verweisen wir auf [Ehr79], [EHKP91], und [PER95].

Es gibt drei wichtige Sätze, die Parallelität betreffen: Die lokalen Church-Rosser-Eigenschaften I und II und das Parallelismustheorem. Die Sätze basieren auf **paralleler und sequentieller Unabhängigkeit** von Ableitungen: In der einen Ableitung werden also keine Objekte gelöscht, die die andere noch braucht. Intuitiv bedeutet das, daß die Vorkommen

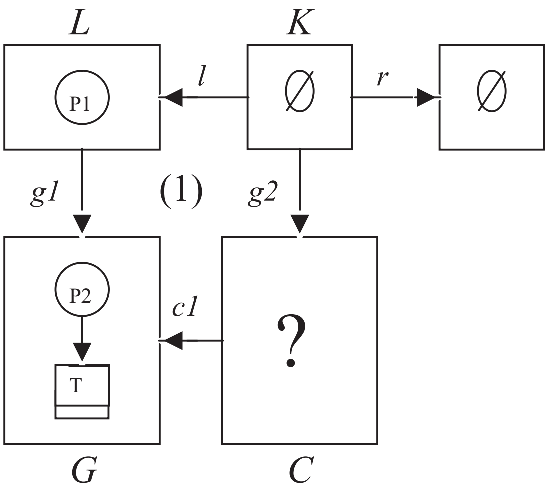


Abbildung 4.3: Beispiel: nicht erfüllte Gluing Condition

beider Regeln entweder disjunkt sind oder sich nur in Klebeobjekten des Schnittstellennetzes überlappen.

Satz 4.3.3 (Ergebnisse für AHL-Netz-Transformationssysteme)

- *Die* **Lokale Church-Rosser-Eigenschaft I** *sagt aus, daß zwei parallel unabhängige Ableitungen* $G \overset{p}{\Longrightarrow} H$ *und* $G \overset{p'}{\Longrightarrow} H'$ *in beliebiger Reihenfolge mit dem gleichen Ergebnis ausgeführt werden können. Das bedeutet, es gibt ein AHL-Netz* X *und zwei sequentiell unabhängige Ableitungen* $G \overset{p}{\Longrightarrow} H \overset{p'}{\Longrightarrow} X$ *und* $G \overset{p'}{\Longrightarrow} H' \overset{p}{\Longrightarrow} X$ *.*

- *Die* **Lokale Church-Rosser-Eigenschaft II** *sagt aus, daß zwei sequentiell unabhängige Ableitungen in beliebiger Reihenfolge ausgeführt werden können. Das bedeutet, wenn es eine sequentiell unabhängige Ableitung* $G \overset{p}{\Longrightarrow} H \overset{p'}{\Longrightarrow} X$ *gibt, dann gibt es auch eine weitere sequentiell unabhängige Ableitung* $G \overset{p'}{\Longrightarrow} H' \overset{p}{\Longrightarrow} X$*, so daß die Regeln, angewendet in umgekehrter Reihenfolge, dasselbe Ergebnis liefern.*

- *Eine* **Parallelregel** *zweier unabhängiger Regeln* p *und* p' *ist gegeben durch komponentenweise disjunkte Vereinigung beider Regeln, es wird also das Koprodukt* $p + p'$ *von* p *und* p'*, konstruiert. Eine Ableitung über die Parallelregel* $p + p'$ *heißt parallele Ableitung.*

- *Das* **Parallelismustheorem** *besagt, daß es zwei inverse Operationen ANALYSE und SYNTHESE gibt, die parallele Ableitungen in sequentiell unabhängige Ableitungen überführen und umgekehrt:*

  1. *ANALYSE*
     Eine parallele Ableitung $G \overset{p+p'}{\Longrightarrow} X$ *kann in zwei sequentiell unabhängige Ableitungen* $G \overset{p}{\Longrightarrow} H \overset{p'}{\Longrightarrow} X$ *und* $G \overset{p'}{\Longrightarrow} H' \overset{p}{\Longrightarrow} X$ *zerlegt (analysiert) werden, die dasselbe Netz* X *ergeben, wie die parallele Ableitung.*

2. *SYNTHESE*
   Jede sequentiell unabhängige Abeleitung $G \stackrel{p}{\Longrightarrow} H \stackrel{p'}{\Longrightarrow} X$ *kann zu einer parallelen Ableitung* $G \stackrel{p+p'}{\Longrightarrow} X$ *zusammengesetzt werden, die das gleiche Netz* X *ergibt.*
3. *Bijektive Korrespondenz:*
   Die Operationen ANALYSE und SYNTHESE sind invers zueinander.

Horizontale Strukturierung von AHL-Netzen

Wie bereits in Kapitel 2 beschrieben, gibt es zwei wichtige Konzepte zur horizontalen Strukturierung von AHL-Netzen: die Identifikation von ausgezeichneten Teilnetzen innerhalb eines Netzes, genannt *Fusion*, und die Komposition von kleineren Netzen über einem gemeinsamen Teilnetz zu einem größeren Netz, genannt *Union*. Wir definieren die beiden Konzepte als kategorielle Konstruktionen in der Kategorie **AHLNET** und zeigen die Kompatibilität mit der vertikalen Strukturierung (regelbasierten Verfeinerung) von AHL-Netzen.

Definition 4.3.4 (Fusion) *Die Fusion zweier AHL-Netz-Morphismen* $f1, f2 : F \to N$ *ist der Koegalisator* $(c : N \to N', N')$ *von* $f1$ *und* $f2$ *und wird mit* $N \circ\!\!\longrightarrow N'$ *über* $(F, f1, f2, c)$ *bezeichnet, kurz* $N \circ\!\!\xrightarrow{F} N'$.

Bemerkung:

1. Die Vorkommen eines Teilnetzes $f1(F)$ und $f2(F)$ werden innerhalb des Netzes N verklebt.
2. Mehrere Vorkommen von Teilnetzen können über mehrfache Anwendung der *Fusion*-Operation verklebt werden, siehe dazu die Beispiele in Kapitel 2, Seiten 16 und 17.

Wenn wir die Beziehung zwischen Ableitungen und *Fusion* untersuchen, so wird deutlich, daß eine Ableitung und eine *Fusion*-Operation, angewendet auf dasselbe Netz, nur dann kompatibel sein können, wenn die Ableitung keine Objekte löscht, die im Teilnetz F der *Fusion* liegen. Wir bezeichnen diese Voraussetzung der Kompatibilität als **parallele Unabhängigkeit von Fusion und Ableitung** (formal siehe [PER95]).

Satz 4.3.5 (Kompatibilität von Fusion und Ableitung)
Gegeben seien eine Fusion $N \circ\!\!\xrightarrow{F} N'$ *und eine dazu parallel unabhängige Ableitung* $N \stackrel{p}{\Longrightarrow} H$. *Dann gibt es ein AHL-Netz* H'*, daß sowohl über die Fusion* $H \circ\!\!\xrightarrow{F} H'$ *als auch über die Ableitung* $N \stackrel{p}{\Longrightarrow} H'$ *konstruiert werden kann:*

$$N \stackrel{p}{\Longrightarrow} H \circ\!\!\xrightarrow{F} H' = N \circ\!\!\xrightarrow{F} N' \stackrel{p}{\Longrightarrow} H'$$

(Beweis von Satz 4.3.5 siehe [PER95])

Beispiel aus der Fallstudie:
Die Anwendung der mehrfachen *Fusion*-Operationen auf den Seiten 16 und 17 auf das Netz AV1: Einzelne Angeordnete Vitalwertemessungen (Seite 46) über das gemeinsame Teilnetz B7b (Seite

17) ergibt das Netz AV: **Angeordnete Vitalwertemessungen - Eintrag zusammengefaßt** auf Seite 46. Die anschließende Anwendung der Regel $r_{AV}8$ (Seite 99) auf das Netz AV leitet das Netz AVb′ ab, das im wesentlichen dem Netz AVb auf Seite 102 entspricht, in dem aber die Kanten von und zur Stelle $Pfleger$ noch vorhanden sind:

$$\mathsf{AV1} \overset{\mathsf{B7b}}{\circ\!\!\longrightarrow} \mathsf{AV} \overset{r_{AV}8}{\Longrightarrow} \mathsf{AVb}'$$

Gehen wir umgekehrt vor, so wenden wir zunächst die Regel $r_{AV}8$ auf Netz AV1 an und leiten damit Netz AV1′ ab, das immer noch aus sechs einzelnen Teilnetzen besteht. Nur das Teilnetz mit der Transition *Blutdruck messen* wurde nun um die Überprüfung des Blutdurcks auf Hochdruckwerte und gegebenenfalls um den Eintrag eines Warnvermerks in die Fieberkurve erweitert. Da die Ableitung keine Objekte löscht, also parallel unabhängig von der $Fusion$ ist, läßt sich nun die $Fusion$ auf AV1′ anwenden und liefert als Ergebnis wiederum das Netz AVb′ :

$$\mathsf{AV1} \overset{r_{AV}8}{\Longrightarrow} \mathsf{AV1}' \overset{\mathsf{B7b}}{\circ\!\!\longrightarrow} \mathsf{AVb}'$$

Definition 4.3.6 (Union)
Die Union zweier AHL-Netze (N1, N2) über einem Schnittstellennetz I ist gegeben durch die Konstruktion des Pushouts **(1)** *und wird mit* $(N1, N2) \rightarrowtail N$ *über* $(I, i1, i2)$ *bezeichnet, kurz* $(N1, N2) \overset{I}{\rightarrowtail} N$*:*

$$\begin{tikzcd} I \arrow[r, "i1"] \arrow[d, "i2"'] \arrow[dr, phantom, "\mathbf{(1)}"] & N1 \arrow[d] \\ N2 \arrow[r] & N \end{tikzcd}$$

Bemerkung:
Die $Union$ von beliebig vielen Objekten erfolgt über die mehrfache Anwendung der Operation $Union$ (siehe dazu das Beispiel in Kapitel 2, Seite 19).Dies kann kategoriell als die Erweiterung von Pushouts auf multiple Pushouts beschrieben werden (siehe [AHS90]).

Analog zur $Fusion$ bezeichnen wir eine Ableitung und eine $Union$-Operation, angewendet auf dasselbe Netz, als **parallel unabhängig**, wenn die Ableitung keine Objekte löscht, die im Schnittstellennetz I der $Union$ liegen.

Satz 4.3.7 (Kompatibilität von Union und Ableitung)
Gegeben seien eine Union $(N1, N2) \overset{I}{\rightarrowtail} N$ *und zwei davon parallel unabhängige Ableitungen*

$N1 \overset{p1}{\Longrightarrow} H1$ und $N2 \overset{p2}{\Longrightarrow} H2$. Dann gibt es ein AHL-Netz H, daß sowohl über die Union $(H1, H2) \overset{I}{\rightarrowtail} H$ als auch über die parallele Ableitung $N \overset{(p1+p2)}{\Longrightarrow} H$ konstruiert werden kann:

$$(N1, N2) \overset{(p1,p2)}{\Longrightarrow} (H1, H2) \overset{I}{\rightarrowtail} H = (N1, N2) \overset{I}{\rightarrowtail} N \overset{(p1+p2)}{\Longrightarrow} H$$

(Beweis von Satz 4.3.7 siehe [PER95])

Beispiel aus der Fallstudie:
In unserer Fallstudie haben wir oft die Operation *Union* zur horizontalen Strukturierung der Netze verwendet. Um die parallele Unabhängigkeit zu den Ableitungen der regelbasierten Verfeinerung zu gewährleisten, haben die Regeln die Stellen, die in den Schnittstellennetzen von *Union*-Anwendungen lagen, als isolierte Stellen erhalten. So blieb die *Union*-Operation stets auch nach Anwendung der Regeln auf die resultierenden Netze anwendbar. Die isolierten Stellen wurden nach Anwendung aller Transformationsregeln durch die *Garbage-Collection*-Regel gelöscht (siehe Beispiel zu Satz 4.3.2).

Kapitel 5

Zusammenfassung und Ausblick

Wir haben anhand der Fallstudie HDMS-AHL Algebraische High-Level-Netze zur Durchführung der Anforderungsanalyse eines medizinischen Informationssystems eingesetzt. Dabei wurde zunächst formal das IST-System spezifiziert, wofür die informelle IST-Analyse des KORSO-Projekts aus *Ein Fallbeispiel für KORSO: IST-Analyse von HDMS-A* ([CKL93a]) als Grundlage diente. Wir unterschieden fünf Phasen des Krankenhausaufenthaltes eines Patienten: die *Aufnahme*, die *Behandlung* des Patienten auf der Pflegestation, der *Labor*-Ablauf bei der Blutbildanalyse, die *Herzkatheterisierung* und die *Entlassung*.

Zur Spezifikation jeder dieser Phasen gehören AHL-Netze, die die Arbeitsabläufe beschreiben, und eine algebraische Spezifikation, in der die verwendeten Dokumente und Datentypen definiert werden. Zur Strukturierung der Spezifikationen verwendeten wir die horizontalen Strukturierungstechniken *Union* und *Fusion*, die auf der kategoriellen Konstruktion von Kolimiten in der Kategorie der AHL-Netze beruhen. Horizontale Strukturierungsmechanismen, sonst bekannt für algebraische Spezifikationen, werden somit im Zuge der Integration statischer und dynamischer Aspekte auch auf die Netzstruktur (den dynamischen Teil des Modells) anwendbar.

Als nächster Schritt wurde die IST-Essenz, also die für das zu spezifizierende System essentiellen Bestandteile des IST-Systems, formal abgeleitet. Das dazu verwendete Prinzip der regelbasierten Abstraktion/Verfeinerung bietet die Möglichkeit, innerhalb desselben Formalismus auf eine überschaubare und konsistente Weise die Schritte vom IST-System zur funktionalen Essenz zu definieren. Dabei finden wichtige theoretische Ergebnisse über die Anwendbarkeit von Regeln, über die Konstruktion paralleler Verfeinerungsschritte und über die Kompatibilität von horizontaler und vertikaler Strukturierungsmethoden in unserer Fallstudie ihre Anwendung.

Den Schritt von der IST-Essenz zur funktionalen Essenz demonstrierten wir durch die Anwendung von regelbasierter Verfeinerungen auf die Netze des Bereichs *Behandlung*. Wir erhielten eine weniger komplexe, abstraktere Netzstruktur und eine algebraische Spezifikation, in der von den spezifischen Dokumenten des IST-Systems abstrahiert wurde.

Offensichtlich weist die Verwendung von AHL-Netzen gegenüber den im KORSO-Projekt verwendeten Methoden deutliche Vorteile auf, nämlich die Integration von statischen und dynamischen Aspekten, eine integrierte horizontale Strukturierung und eine dazu kompatible regelbasierte Entwicklungsmethode (siehe Kapitel 3.5).

Der deutlich größere Umfang unserer Fallstudie im Vergleich zu bisherigen Anwendungsbeispielen für AHL-Netze verdeutlicht aber auch, welche Eigenschaften der Modellierungsmethode mit AHL-Netzen eine Weiterentwicklung der formalen Konzepte im Hinblick auf ihre praktische Anwendbarkeit wünschenswert machen:

- Die Technik der regelbasierten Abstraktion/Verfeinerung, die wir zur Ableitung der funktionalen Essenz aus der IST-Essenz des Systems verwenden, stellt eine syntaktische Formalisierung der Entwicklungsschritte dar. Auf semantischer Ebene bliebe zu prüfen, ob eine Ableitung die Relation von Markierung zu Folgemarkierung auf den nicht gelöschten Stellen entsprechend abbildet **(semantische Kompatibilität)**. In diesem Zusammenhang kann man sich auch die Formulierung von Bedingungen vorstellen, deren Einhaltung dann bei den einzelnen Ableitungen geprüft werden kann.

- Um die Kompatibilität von horizontaler und vertikaler Strukturierung zu gewährleisten, mußten wir bei der Aufstellung von Transformationsregeln stets im Auge haben, welche Teilnetze innerhalb von Schnittstellen- oder *Fusion*-Netzen liegen, die bei der horizontalen Strukturierung der Netze angewendet werden. Dieses Verfahren war bei der Vielzahl der Netze und der Komplexität ihrer Strukturierung sehr mühsam. Ein **Modulkonzept**, das zu jedem Netz Informationen über die nicht zu löschenden Objekte liefert, wäre in diesem Fall sehr hilfreich gewesen.

- Für eine Weiterführung der Fallstudie im Hinblick auf Vervollständigung der Anforderungs-Spezifikation und auf die **Design-Spezifikation** bietet die vorliegende Arbeit eine gute Grundlage. Bei der formalen Design-Spezifikation mit AHL-Netzen sind insbesondere Methoden der in dieser Arbeit nicht behandelten Invarianten-Analyse von Interesse.

- Es ist zu erwarten, daß bei einer Fortführung der Fallstudie die Vielzahl und Komplexität der Netze und algebraischen Spezifikationen noch zunimmt. Eine **Werkzeugunterstützung** bei der Erstellung und Analyse von AHL-Netzen, sowie bei der Anwendung von Transformationsregeln ist ein wichtiges Ziel bei der praxisorientierten Weiterentwicklung der AHL-Netz-Konzepte.

Literaturverzeichnis

[AHS90] J. Adamek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories.* Series in Pure and Applied Mathematics. John Wiley and Sons, 1990.

[BFG+93] M. Broy, C. Facchi, G. Grosu, R. Hettler, and Hußmann, H., et al. The Requirement and Design Specification Language SPECTRUM - An Informal Introduction, Version 1.0, Part II. Technical Report TUM-I9312, Institut für Informatik, Technische Universität München, 1993.

[CEW93] I. Claßen, H. Ehrig, and D. Wolz. *Algbraic Specification Techniques and Tools for Software Development - The ACT Approach.* AMAST Series in Computing Vol. 1. World Scientific, 1993.

[CHL95] F. Cornelius, H. Hußmann, and M. Löwe. The KORSO Case Study for Software Engineering with Formal Methods: A Medical Information System. In M. Broy and S. Jähnichen, editors, *KORSO: Methods, Languages, and Tools for the Construction of Correct Software*, pages 417–445. Springer LNCS 1009, 1995. also appeared as technical report 94-5, TU Berlin.

[CKL93a] F. Cornelius, M. Klar, and M. Löwe. Ein Fallbeispiel für KORSO: Ist–Analyse HDMS-A. Technical Report 93-28, Technische Universität Berlin, 1993.

[CKL93b] Felix Cornelius, Marcus Klar, and Michael Löwe. Ein Fallbeispiel für KORSO: Ist–Analyse HDMS-A. Technical Report 93-28, Technical University of Berlin, 1993.

[Con93a] S. Conrad. Einbindung eines bestehenden Datenbanksystems in einen formalen Software–Entwicklungsprozeß — ein Beitrag zur HDMS-A–Fallstudie. In H.-D. Ehrich, editor, *Beiträge zu* KORSO- *und* TROLL *light–Fallstudien.* TU Braunschweig, Informatik-Bericht 93-11, 1993.

[Con93b] S. Conrad. Spezifikation eines vereinfachten Datenbanksystems — ein Beitrag zur HDMS-A–Fallstudie. In H.-D. Ehrich, editor, *Beiträge zu KORSO– und TROLL light–Fallstudien*, pages 15–26. Technische Universität Braunschweig,, Informatik–Bericht 93–11, 1993.

[Dam93] W. Damm. KORSO-Applikationen im HDMS-A-Teilprojekt, Vorhabensbeschreibung. Technical report, Universität Oldenburg, 1993.

[DH90] C. Dimitrovici and U. Hummert. Composition of Algebraic High-Level Nets. In *Proc. of the 7th ADT-Workshop*, Wusterhausen/Dosse, 1990. LNCS 534(1991).

[DHP91] C. Dimitrovici, U. Hummert, and L. Petrucci. Composition and net properties of algebraic high-level nets. In *Advances of Petri Nets*. Springer Verlag Berlin LNCS 524, 1991.

[EBO91] H. Ehrig, M. Baldamus, and F. Orejas. New concepts for amalgamation and extension in the framework of specification logics. Technical Report 91-05, TU Berlin, 1991.

[EGH92] H. Ehrig, M. Große-Rhode, and A. Heise. Specification techniques for concurrent and distributed systems. Technical Report 92/5, Technical University of Berlin, jan. 1992. Invited paper for 2nd Maghr. Conference on Software Engineering and Artificial Intelligence, Tunis,1992.

[EHKP91] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. From graph grammars to High Level Replacement Systems. In *Lecture Notes in Computer Science*, pages 269–291. Springer Verlag, 1991.

[Ehr79] H. Ehrig. Introduction to the algebraic theory of graph grammars. In V. Claus, H. Ehrig, and G. Rozenberg, editors, *1st Graph Grammar Workshop, Lecture Notes in Computer Science 73*, pages 1–69. Springer Verlag, 1979.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1985.

[EPR94] H. Ehrig, J. Padberg, and L. Ribeiro. Algebraic high-level nets: Petri nets revisited. In *Recent Trends in Data Type Specification*, pages 188–206. Springer, LNCS 785, 1994.

[Erm94] C. Ermel. Kategorielle Darstellung von Stellen/Transitions-Netzen. Studienarbeit, TU Berlin, 1994.

[FHMO91] E. Fleck, H. Hansen, B. Mahr, and H. Oswald. Systementwicklung für die Integration und Kommunikation von Patientendaten und -dokumenten. Forschungsbericht 02-91, PMI am DHZB, 1991.

[Fuc94] T. Fuchß. Translating E/R-diagrams into Consistent Database Specifications. Technical Report 2/94, Universität Karlsruhe, January 1994.

[Gen91] H.J. Genrich. Predicate/Transition Nets. In *High-Level Petri Nets: Theory and Application*, pages 3–43. Springer, 1991.

[GH93] M. Grosse and H. Hufschmidt. SOLL–Spezifikation aus Sicht der Sicherheit. Technical Report A/07/93, Universität des Saarlandes, Saarbrücken, December 1993.

[GL81] H.J. Genrich and K. Lautenbach. System modelling with high-level Petri nets. *Theoretical Computer Science*, 13:109–136, 1981.

[Het93] R. Hettler. Übersetzung von E/R-Modellen nach SPECTRUM. Technical Report TUM-I9333, Technische Universität München, 1993.

[Hum89] U. Hummert. *Algebraische High-Level Netze.* PhD thesis, Technische Universität Berlin, 1989.

[Huß93] H. Hußmann. Zur formalen Beschreibung der funktionalen Anforderungen an ein Informationssystem. Technical Report TUM-I9332, Technische Universität München, 1993.

[Jen81] K. Jensen. Coloured petri nets and the invariant method. *Theoretical Computer Science*, 14:317–336, 1981.

[Jen92] Kurt Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 1. Springer, 1992.

[KBS93] B. Krieg-Brückner and Hui Shi. Entwurf der Benutzerschnittstelle für die Herzkatheter- Untersuchung. Technical report, Universität Bremen, Juni 1993.

[LCFW92] M. Löwe, F. Cornelius, J. Faulhaber, and R. Wessäly. Ein Fallbeispiel für KORSO: Ein Vorschlag. Technical Report 92-45, Technische Universität Berlin, 1992.

[MM90] J. Meseguer and U. Montanari. Petri nets are monoids. *Information and Computation*, 88(2):105–155, 1990.

[MZ94] M. Mehlich and W. Zhang. Specifying Interactive Components for Configurating Graphical User Interfaces. Technical Report 9401, Ludwig-Maximilians-Universität München, 1994.

[Pad95] J. Padberg. Abstract datatype semantics for algebraic high-level nets using dynamic abstract datatypes. In G. Hommel, editor, *Proc. Int. Workshop on Quality of Communication Based Systems*, pages 1 – 18. Kluwer Academic Publishing, 1995.

[Pad96] J. Padberg. *Abstract Petri Nets: A Uniform Approach and Rule-Based Refinement.* PhD thesis, Technical University Berlin, 1996.

[PER95] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic high-level net transformation systems. *Mathematical Structures in Computer Science*, 5:217–256, 1995.

[Pet62] C.A. Petri. *Kommunikation mit Automaten.* PhD thesis, Schriften des Institutes für Instrumentelle Mathematik, Bonn, 1962.

[PREC93] J. Padberg, L. Ribeiro, H. Ehrig, and F. Cornelius. Formal requirement analysis using algebraic high-level nets and transformations: a KorSo case study improving parts of HDMS-A. Technical Report 93/34, TU Berlin, 1993.

[PW94] P. Pepper and Wirsing, M., et al. KORSO – A Methodology for the Development of Correct Software. In M. Broy and S. Jähnichen, editors, KORSO , *Correct Software by Formal Methods, LNCS.* Springer, 1994.

[Rei85] Wolfgang Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, 1985.

[Ren94] K. Renzel. Formale Beschreibung von Sicherheitsaspekten für das Fallbeispiel HDMS-A. Technical Report 9402, Ludwig-Maximilians-Universität Munich, January 1994.

[REP93] L. Ribeiro, H. Ehrig, and J. Padberg. Formal development of concurrent systems using algebraic high-level nets and transformations. In *Proc. VII Simpósio Brasileiro de Engenharia de Software*, pages 1–16, Rio de Janeiro, oct. 1993. Tech-report no. 93-13, TU Berlin.

[Shi94] H. Shi. Benutzerschnittstelle und -Interaktion für die HK-Untersuchung. Technical Report at the Universität Bremen, to appear, February 1994.

[SNM+93] O. Slotosch, F. Nickl, S. Merz, H. Hußmann, and R. Hettler. Die funktionale Essenz von HDMS-A. Technical Report TUM-I9335, Technische Universität München, 1993.

[Ste93] K. Stenzel. A Verified Access Control Model. Technical Report 26/93, Fakultät für Informatik, Universität Karlsruhe, Germany, December 1993.

[TBG91] A. Tarlecki, R.M. Burstall, and J.A. Goguen. Some fundamental algebraic tools for the semantics of computation. Part III: Indexed categories. *TCS*, 91:239–264, 1991. Also appeared as technical report ECS-LFCS-89-90, Univ. of Edinburgh, 1989.

[Vau86] J. Vautherin. Parallel specification with coloured Petri nets and algebraic data types. In *Proc. of the 7th European Workshop on Application and Theory of Petri nets*, pages 5–23, Oxford, England, jul. 1986.

Anhang A

Standard-Spezifikationen

LISTE[D] =

parameterisierter Datentyp, der in der eigentlichen Spezifikation in den Typen PatAkte, Protokoll, Vitalwerte, Medikationen... aktualisiert wird.

sorts: Liste(D)
opns: nil : $\rightarrow$ Liste(D)
nodata:$\rightarrow$ D

– list(A): generiere Liste mit einem Element A.
list: D $\rightarrow$ Liste(D)

– ladd(A,X): füge links Element A der Liste X hinzu
ladd: D $\times$ Liste(D) $\rightarrow$ Liste(D)

– concat(X,Y): hänge Listen X und Y zu einer zusammen .
concat: Liste(D) $\times$ Liste(D) $\rightarrow$ Liste(D)

– radd(X,A): füge rechts Element A der Liste X hinzu
radd: Liste(D) $\times$ D $\rightarrow$ Liste(D)

– head(X): liefert erstes Element der Liste X.
head: Liste(D) $\rightarrow$ D

– tail(X): liefert Liste X ohne erstes Element.
tail: Liste(D) $\rightarrow$ Liste(D)
– isempty(X): Abfrage: Liste X leer?.
isempty: Liste(D) $\rightarrow$ Bool

– search(A,X): liefert passendes Listenelement aus X, das mit A matcht.

search:D × Liste(D) → D

– *len(X): liefert die Länge der Liste X.*
len:Liste(D) → Nat

– *length_until(A,X): liefert die Länge der Liste X bis zum ersten Auftreten von A*
length_until: D × Liste(D) → Nat

– *take(X,N): liefert die Teilliste mit den ersten N Elementen der Liste X.*
take: Liste(D) × Nat → Liste(D)

– *drop(X,N): liefert die Teilliste ohne die ersten N Elemente der Liste X.*
drop:Liste(D) × Nat → Liste(D)

– *nthdel(X,N): liefert die Teilliste ohne das N-te Element der Liste X.*
nthdel:Liste(D) × Nat → Liste(D)

– *delete(X,A): liefert die Teilliste von X ohne das mit A matchende Listenelement.*
delete:Liste(D) × D → Liste(D)

– *change(X,A,B): ersetzt ein mit A matchendes Listenelement in X durch B.*
change: Liste(D) × D × D → Liste(D)

eqns: list(A) = ladd(A,nil);

concat(nil,X) = X;
concat(ladd(A,X),Y) = ladd(A, concat(X,Y));

radd(X,A) = concat(X, list(A));

head(nil) = nodata;
head(ladd(A,X)) = A;

tail(nil) = nil;
tail(ladd(A,X)) = X;

isempty(nil) = T;
isempty(ladd(A,X)) = F;

search(A,nil) = nodata;
search(A,ladd(B,X)) = if (A=B) then B else search(A,X);

len(nil) = 0;
len(ladd(A,X)) = s(len(X));

```
length_until(A,nil) = 0;
length_until(A,ladd(B,X)) = if (A=B) then s(0) else s(length_until(A,X));

take(X,N) = if((X = concat(Y,Z) ) and (len(Y) = N)) then Y else nil;

drop(X,N) = if((X = concat(Y,Z) ) and (len(Y) = N)) then Z else nil;

nthdel(X,N) = concat(take(X,p(N)), drop(X,N);
delete(X,A) = nthdel(X,length_until(A,X));
change(X,A,B) = radd(delete(X,A),B);
```

Die Standard-Spezifikationen NAT und BOOL werden hier nur angedeutet und sind so für unsere Zwecke ausreichend. Sie sind um viele bekannte Funktionen erweiterbar (ebensowie auch LIST[D]); siehe dazu [CEW93].

NAT =

```
sorts: Nat
opns:  0 → Nat
       s, p: Nat → Nat
eqns:  p(0) = 0;
       p(s(N)) = N;
```

BOOL =

```
sorts: Bool
opns:  true, false → Bool
       non:Bool → Bool
       and, or : Bool × Bool → Bool
eqns:  non(true) = false;
       non(false) = true;
       non(true) = false;
       and(true,B) = B;
       and(false,B) = false;
       or(true,B) = true;
       or(false,B) = B;
       and(A,B) = and(B,A);
       or(A,B) = or(B,A);
```