Transformationstechniken bei Condition/Event-Netzen in der Sprachdefinition von Full-PEARL

Dipl.-Inform. Bernhard Karbe, München

0. Einleitung

Zur Erläuterung der Semantik der Programmiersprache PEARL, wie sie durch das Deutsche Institut für Normung in der DIN 66253 Teil 2 (Full PEARL) festgelegt ist, werden Petri-Netze in der Interpretation von Condition/Event-Netzen herangezogen. Dieser Beitrag ist als Lesehilfe gedacht, und soll die dabei verwendeten Petri-Netz-Transformationen erläutern.

Es wird vorausgesetzt, daß der Leser mit Petri-Netzen vertraut ist. Eine für Ingenieure geeignete Einführung in Petri-Netze stellt die Arbeit [5] dar.

1. Definitionen und Regeln

1.1. Stellen und Transitionen

Ein Petri-Netz besteht aus S-Knoten, T-Knoten und Pfeilen (Kanten). Jeder S-Knoten, im folgenden auch Stelle oder Platz genannt, wird als Kreis, jeder T-Knoten, auch Transition genannt, als Kästchen dargestellt. Jede Kante kann nur von einer Stelle (Kreis) zu einer Transition (Kästchen) oder von einer Transition zu einer Stelle gehen. Führt ein Pfeil von einem Knoten A zu einem Knoten B und ein anderer von B nach A, so wird als Abkürzung ein Pfeil mit zwei Köpfen (jeder an einem Ende), hier Doppelpfeil genannt, benutzt.

In Transitionsnetzen können Stellen eine variable Anzahl von sog. Markierungen (Tokens) besitzen. Die Transitionen beschreiben die möglichen elementaren Veränderungen der Verteilung der Markierungen in dem Netz. In einem Condition/Event-Netz kann jede Stelle höchstens eine Markierung besitzen. Markierungen werden graphisch durch Punkte in den Stellen repräsentiert. Jedes Netz besitzt eine initiale Markierung.

1.2. Fortschaltungsregeln

Im folgenden werden nur noch elementare Condition/Event-Netze und deren in dem DIN-Entwurf verwendete Abstraktionen betrachtet.

Führt ein Pfeil von einer Stelle zu einer Transition t, so wird die Stelle Eingabeplatz der Transition t genannt. Führt ein Pfeil von einer Transition t zu einer Stelle, heißt die Stelle Ausgabeplatz der Transition t.

Die Verteilung der Markierungen in einem Transitionsnetz ändert sich genau dann, wenn (mindestens) eine Transition "zündet", "schaltet" oder "feuert". Eine Transition t kann nur dann zünden, wenn alle Eingabeplätze von t markiert und alle Ausgabeplätze unmarkiert sind. Zünden bedeutet: Von jedem Eingabeplatz der Transition wird eine Markierung entfernt und in jedem Ausgabeplatz der Transition wird eine Markierung hinzugefügt. Transitionen zünden zufällig. Eine Aussage darüber, innerhalb welcher Zeit nach Eintreten der Voraussetzungen zum Zünden die Transition tatsächlich schaltet, kann nicht gemacht werden. Zwei Transitionen können genau dann gleichzeitig feuern, wenn sie unabhängig voneinander sind, d.h. keine gemeinsamen Eingabe- oder Ausgabeplätze besitzen.

Man kann also folgende zwei Situationen unterscheiden:
Eine Transition hat (prinzipiell) die Möglichkeit zu zünden, wenn alle Eingabeplätze von ihr markiert und alle Ausgabeplätze von ihr unmarkiert sind. Es sei darauf hingewiesen, daß sie zünden kann, aber nicht muß (siehe Bild 1 und Bild 2).
Zwei Transitionen befinden sich im Konflikt, wenn sie mindestens einen Eingabe- oder einen Ausgabeplatz gemeinsam haben und beide die Möglichkeit haben zu zünden. (Bild 3 und Bild 4). Zündet nämlich eine von beiden, so kann die andere nicht mehr zünden. Wie der Konflikt gelöst wird, wird nicht mehr durch das Transitionsnetz selbst beschrieben.

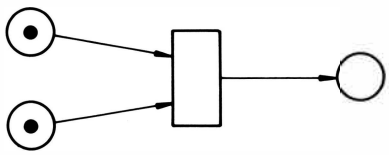


Bild 1: Transition kann zünden

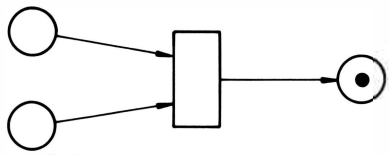


Bild 2: Transition hat gezündet

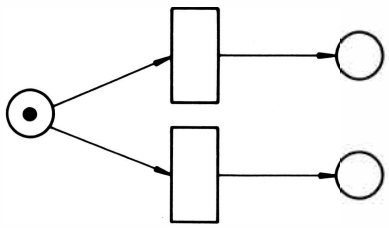


Bild 3: Transitionen haben Eingabeplatz gemeinsam

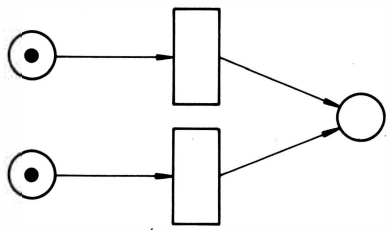
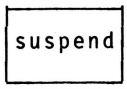


Bild 4: Transitionen haben Ausgabeplatz gemeinsam

Es gibt zahlreiche Beispiele für Konfliktsituationen in technischen Systemen und aus dem täglichen Leben. Man denke z.B. daran, daß der Rufknopf für einen einzelnen Aufzug in verschiedenen Stockwerken gleichzeitig gedrückt wird, oder daran, daß derselbe Teilnehmer in einem Telefonnetz von verschiedenen anderen Teilnehmern aus gleichzeitig angewählt wird. In diesen Fällen ist klar, daß nur genau eine der möglichen Folgesituationen eintreten kann. Häufig ist jedoch nicht vorhersagbar, welche dies ist. Im Telefonnetz hängt dies z.B. davon ab, welcher Ruf früher bei der Teilnehmerschaltung des angerufenen Teilnehmers ankommt, und dies ist letzlich zufällig. Eine korrekte Darstellung einer solchen Situation kann folglich nur darin bestehen, die Konfliktlösung offen zu lassen. Schließlich kann von außen nicht entschieden werden, welche Folgesituation eintritt. Anders ausgedrückt: Überall dort, wo in der Beschreibung der Sprache PEARL Konflikte in den Transitionsnetzen auftreten, ist es dem Implementierer der Sprache anheimgestellt, wie er den Konflikt löst. Er ist frei in der Art der Lösung. Andererseits darf ein Anwender von PEARL seine Programmsysteme nicht so entwerfen, daß sie nur dann bestimmungsgemäß funktionieren, wenn eine ganz bestimmte Art der Konfliktlösung unterstellt wird. Es wird hierdurch nicht nur die Übertragbarkeit der Programme gefährdet, sondern es ist auch durchaus denkbar, daß der Sprachimplementierer bei einer Revision des Übersetzers eine andere Konfliktlösung wählt.

2. Notationale Konventionen

Im beschreibenden Text wird auf Stellen mit den runden Klammern (...) und auf Transitionen mit den eckigen Klammern [...] Bezug genommen. In den Klammern steht der Knotenname, also z.B. [suspend] für



Die Knoten (P) und [T] sind elementare Knoten des Condition/Event-Netzes und lassen sich formal als Stellen der Kapazität 1 und Transitionen eines Condition/Event-Netzes interpretieren. Alle anderen Knoten sind Abstraktionen. Sie sind Platzhalter für einsetzbare Teilnetze (s. Abschnitt 3) oder Konnektoren (s. Abschnitt 4).

Prinzipiell lassen sich Netze mit solchen nicht elementaren Knoten auf Netze mit ausschließlich elementaren Knoten mittels der unten zu beschreibenden Transformationstechniken abbilden. Es sei jedoch davor gewarnt anzunehmen, man könne aus einem vorgegebenen Full-PEARL-Programm, der im DIN-Entwurf angegebenen Grammatik und insbesondere deren DYNAMICS-Klauseln auf mechanische Art ein rein elementares Condition/Event-Netz herleiten, das vollständig die Semantik des dazugehörenden Programms beschreibt. Eine Beschreibung sequentieller Abläufe mittels Petri-Netzen würde nämlich bewirken, daß in dem DIN-Entwurf wesentlich mehr Details vorkommen würden, als dies zur Spezifikation der Sprache PEARL notwendig und zweckmäßig wäre. Für den Sprach-Implementierer würden Dinge festgelegt, die nicht mehr Eigenschaften der Sprache definieren, sondern eine bestimmte Implementierung vorschreiben. Deshalb ist eine Spezifikation solcher Abläufe im DIN-Entwurf unterblieben [4]. Vielmehr werden durch die angegebenen Netze die nichtsequentiellen PEARL-Eigenschaften modelliert und damit auch die nichtsequentiellen Eigenschaften eines PEARL-Programms. Um jedoch den Zusammenhang der verschiedenen Eigenschaften bzw. den sie modellierenden Netzen beschreiben zu können, sind die unten erklärten Transformationsregeln ausgesprochen nützlich.

Die Inschrift der Knoten gibt deren Bedeutung bezüglich PEARL wider, u.U. gefolgt von einem Kommentar, dem % vorangestellt ist. Für die formale Interpretation als Petri-Netz sind die Inschriften bis auf die der Knoten (P), (P INITIAL) und [T] bedeutungslos. Verschiedene Knoten in einem Netz können durchaus dieselbe Inschrift besitzen. Knoten, die lediglich einen Kommentar enthalten, aber keinen Knotennamen, sind Platzhalter für einzusetzende Teilnetze, die (in dem DIN-Entwurf) nicht weiter ausgeführt werden und im wesentlichen Implementierungsabhängigkeiten beschreiben. Knoten, die Knotennamen enthalten, kennzeichnen Abstraktionen, die im DIN-Entwurf verfeinert werden. Entweder beschreiben sie Konnektoren, oder es gibt für sie einsetzbare Teilnetze. I.a. besitzen sie (im DIN-Entwurf) einen Kommentar, der eine Referenz auf eine Figur enthält, in der die Verfeinerung beschrieben ist.

Transitionsnetze besitzen eine initiale Markierung. Diese wird durch die Inschrift "INITIAL" in gewissen Knoten (P) und Vergröberungen, die (P) enthalten, beschrieben.

In den Beispielen werden oft Netze aus dem o.g. DIN-Entwurf verwendet. Dabei werden nicht in allen Fällen die dortigen Inschriften vollständig übernommen. Sie müssen aus Platzgründen abgekürzt werden. I.a. entfällt der Kommentar.

3. Einsetzung von Teilnetzen

Üblicherweise wird die Syntax einer Programmiersprache durch ein Produktionssystem beschrieben, durch das alle syntaktisch korrekten Programme hergeleitet werden können. Damit ist aber über die Semantik eines Programmes noch nichts ausgesagt. Eine entsprechende Aussage kann erreicht werden, wenn man mit der syntaktischen Ableitung die Herleitung eines Condition/Event-Netzes kombiniert.

3.1. Substitution von Knoten durch Teilnetze

Die Anwendung von syntaktischen Ableitungsregeln bedeutet eine Ersetzung einer nichtterminalen Zeichenreihe Z1 durch eine andere Zeichenreihe, die terminale und/oder nichtterminale Symbole enthält. Entsprechend bedeutet die Substitutionsregel für Condition/Event-Netze die Ersetzung eines Knotens K im Netz durch das zu K gehörende Teilnetz T_k. Gegebenenfalls, jedoch nicht in allen Fällen, kann die syntaktische Ableitung und die Transformation des Transitionsnetzes simultan bzw. kombiniert ablaufen, indem durch eine syntaktische Regel ein Teil-Transitionsnetz festgelegt ist.

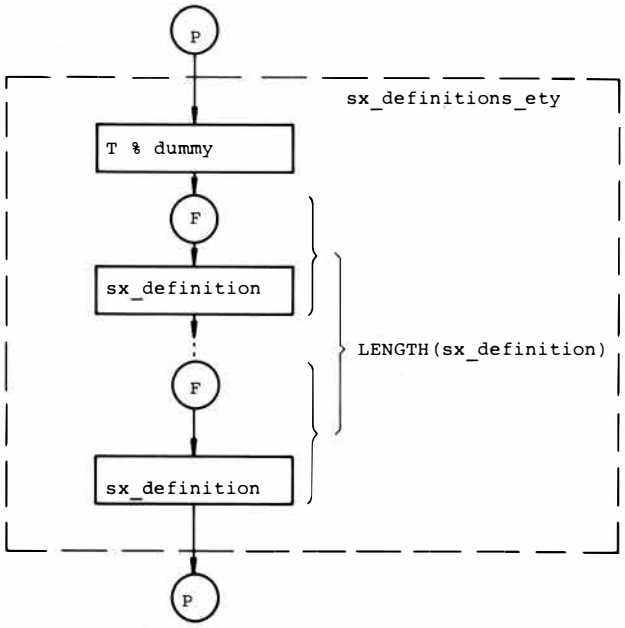


Bild 5: Petri-Netz zur Grammatik-Regel "sx_definitions_ety ::= (sx_definition ';')*" (Beispiel 3.1-0)

Beispiel 3.1-0

Betrachte die PEARL-Syntax-Regel in Abschnitt 13.2 "RULE sx_definitions_ety::= (sx_definition';')*" und das dazugehörige Petri-Netz in Figur #13 im DIN-Entwurf (Bild 5).

Wird bei einer syntaktischen, Ableitung diese Regel angewendet, so wird in dem (ja hoffentlich auch vorliegenden) Netz der Knoten [sx_definitions_ety] durch das Teilnetz in Bild 5 ersetzt. In dem Teil netz treten so viele Transitionen [sx_definition] auf, wie bei der Ersetzung des Nonterminals "sx__definitions__ety" der Term "sx_definition ';'" auftritt. Dies wiederum hängt von dem vorliegenden Programm ab. Das terminale Symbol Semikolon ";" wird hier im Petri-Netz durch (F) modelliert, das jedoch vor der Transition [sx__definition] steht, im Gegensatz zur Syntax, wo das Semikolon dem Nonterminal "sx_definition" folgt.

Einsetzbare Teilnetze werden graphisch derart dargestellt, daß der zu verfeinernde Knoten als gestrichelte Kontur (Kreis oder Rechteck) gezeichnet wird, die mit seinem Namen versehen ist. Darin befindet sich das (normal dargestellte) Teilnetz.

In dem Ausgangsnetz N, in dem der Knoten K substituiert werden soll, ist K durch Kanten a_i mit dem Restnetz von N verknüpft. Die a_i müssen nun durch die Kanten a'_j, mit denen das Teilnetz T_k mit dem Restnetz von N verknüpft werden soll, ersetzt werden. Die a'_j sind gerade die Kanten in der Darstellung von T_k, die die gestrichelte Kontur von K kreuzen.

Beispiel 3.1-1 [3]:

Sei das Netz in Bild 6 gegeben.

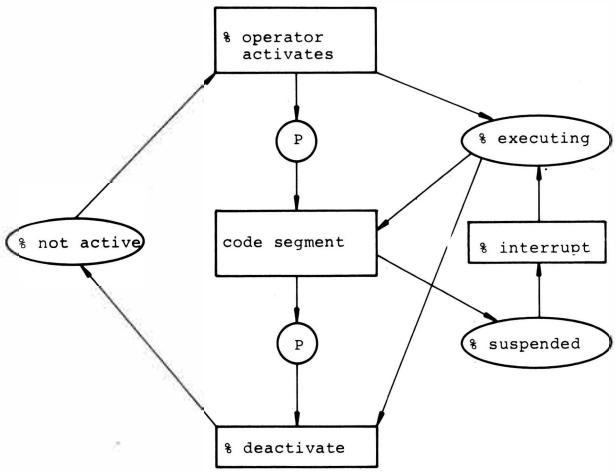


Bild 6: Ausgangsnetz, in dem der Knoten [code segment] ersetzt werden soll (Beispiel 3.1-1)

Es gebe in der Grammatik folgende Produktion:

<code segment> ::= <resume statement>
<code segment>|
<resume statement>

Aufgrund der syntaktischen Struktur eines vorliegenden Programmes soll die erste Alternative der Produktionsregel gewählt werden. Für diese gibt es das (einzusetzende) Teilnetz T_k aus Bild 7.

Dann hat das Gesamtnetz nach der Ersetzung des Knotens code segment durch das Teilnetz T_k die Struktur, wie sie in Bild 8 angegeben ist.

Bevor der Ersetzungsmechanismus weiter erläutert wird, sollen noch kurz zwei Begriffe eingeführt werden:
Jeder Knoten K', der mit dem Knoten K in einem vorgegebenen Netz durch eine Kante

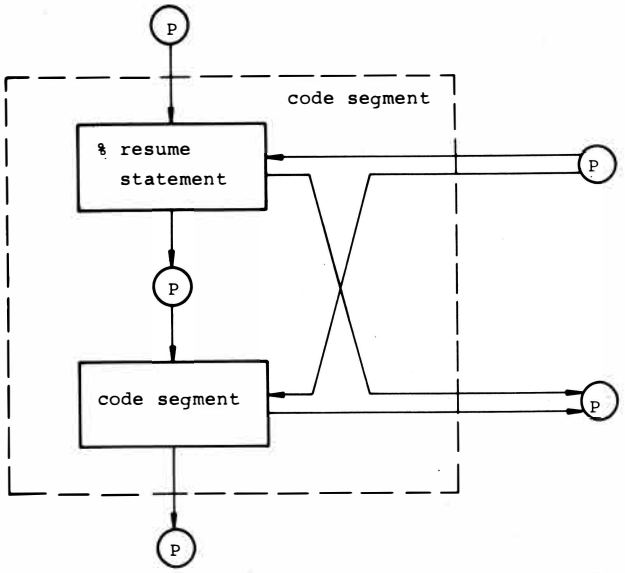


Bild 7: Einsetzbares Teilnetz für den Knoten [code segment] (Beispiel 3.1-1)

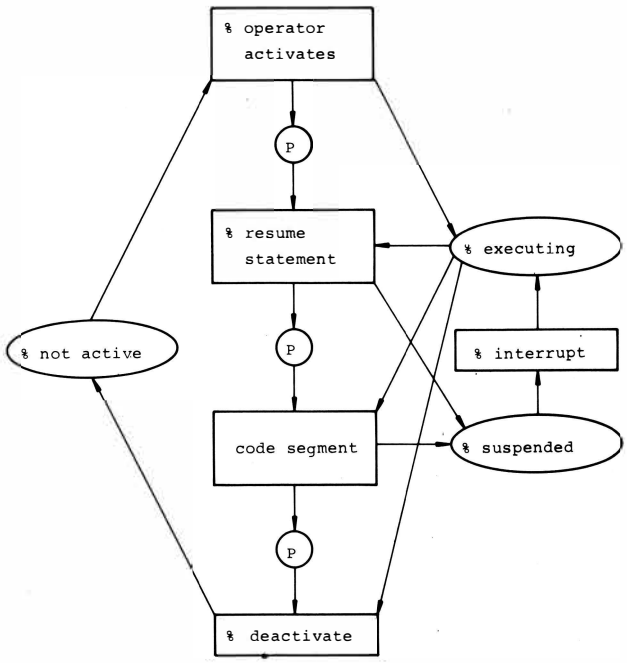


Bild 8: Ergebnisnetz nach einer Ersetzung des Knotens [code segment] durch sein Teilnetz (Beispiel 3.1-1)

verbunden ist, heißt Nachbarknoten von K. Die Nachbarknoten eines Teilnetzes T_k, durch das der Knoten K verfeinert wird, sind die Knoten, die außerhalb der K darstellenden gestrichelten Kontur liegen.
Jeder Knoten in dem Teilnetz T_k, der durch eine Kante mit mindestens einem der Nachbarknoten von T_k verbunden ist, heißt Randknoten von T_k.
Ein Teilnetz T_k, das eine Stelle (bzw. eine Transition) K verfeinert, muß also Stellen (bzw. Transitionen) als Randknoten und Transitionen (bzw. Stellen) als Nachbarknoten besitzen.

Nimmt man das Ausgangsnetz N, das den Knoten K enthält, und die Beschreibung des K verfeinernden Teilnetzes T_k her, so müssen weder die Anzahl der Nachbarknoten von K und von T_k, noch die Anzahl der Kanten übereinstimmen, über die K bzw. T_k mit einzelnen Nachbarknoten verbunden ist. Eine einzelne Kante a_i in N kann durchaus durch mehrere Kanten a'_j substituiert werden. Ein Nachbarknoten von T_k kann mit mehreren von K assoziiert werden. Gegebenenfalls muß ersterer genügend oft incl. der ihn mit T_k verbindenden Kanten dupliziert werden, bis eine 1:1 Abbildung möglich wird. Dann können die Nachbarknoten von K und T_k identifiziert werden. Umgekehrt gilt nicht, daß mehrere Kanten a_i zu einer Kante a'_j zusammenfallen können bzw. mehrere Nachbarknoten von T_k zu einem von K.

Es können demnach die Nachbarknoten von K mit denen von T_k und die Kanten a_i mit den a'_j nicht einfach mechanisch identifiziert werden. Diese Identifizierung soll in den folgenden fünf Punkten an Hand von Beispielen beschrieben werden:

1) Man betrachte die Darstellung des Teilnetzes T_k, in der K durch die gestrichelte äußerste Kontur gekennzeichnet ist. Wenn nun alle Kanten, die zu K führen, zu einem einzigen Randknoten K' in T_k führen und alle Kanten, die von K wegführen, von einem (gegebenenfalls mit K' identischen) Knoten K" von T_k wegführen, werden die Kanten gemäß ihrer Richtung identifiziert. Gemäß der Identifizierung von den Kanten werden die Nachbarknoten von K und T_k identifiziert. Die Namen der Nachbarknoten bei der Darstellung von T_k sind belanglos und können weggelassen werden.

Beispiel 3.1-2:

Betrachte die Figuren #25 und #02 aus dem DIN-Entwurf. Figur #02 aus Bild 10 soll in die Figur #25 aus Bild 9 eingesetzt werden. Es wird der Knoten (F) ersetzt.

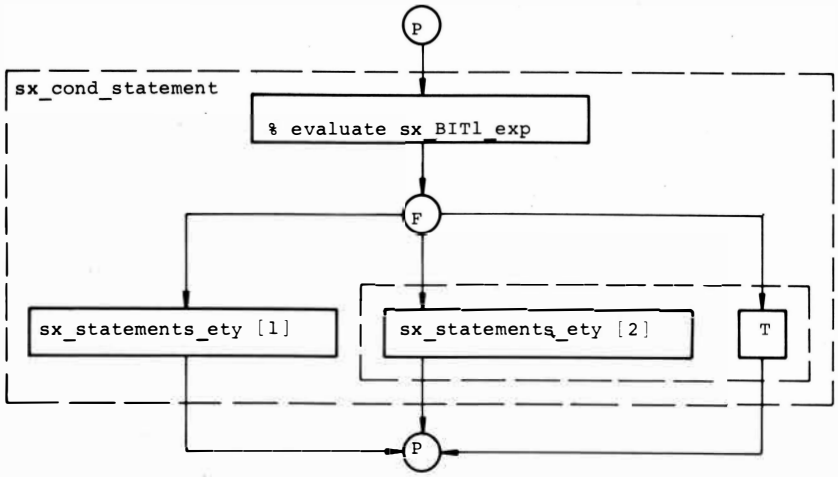


Bild 9: Figur #25 aus dem DIN-Entwurf

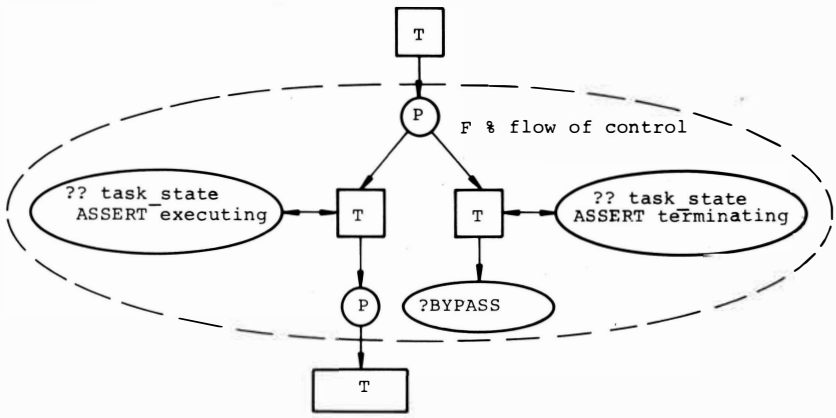


Bild 10: Figur #02 aus dem DIN-Entwurf

Die Einsetzung von #02 in #25 vollzieht sich in mehreren Schritten. Als erstes wird bei der Betrachtung von Bild 10 festgestellt, daß der hier betrachtete Fall vorliegt. In Bild 9 wird festgestellt, wieviele Kanten in den Knoten (F) hinein (hier eine) und wieviele herausführen (hier drei). Entsprechend wird in Bild 10 der untere Nachbarknoten [T] verdreifacht. Es gibt also drei Knoten $[T]_1$, $[T]_2$ und $[T]_3$. Zu jedem dieser Knoten führt eine Kante vom unteren Randknoten (P) in Bild 10. Die Identifizierung dieser Knoten mit den Knoten [sx_statements_ety [1]], [sx_statements_ety [2]] und [T] in Bild 9 ist beliebig. Der obere Nachbarknoten [T] in Bild 10 wird mit dem Knoten [% evaluate sx_BIT1_exp] in Bild 9 identifiziert. Da die Identifizierung der Nachbarknoten nun klar ist, kann man den Knoten (F) in Bild 9 streichen (incl. der zu ihm führenden Kanten) und durch das Teilnetz in Bild 10 ersetzen. Das erzeugte Netz ist in Bild 11 dargestellt.

2) Ein ebenso einfacher Fall liegt vor, wenn alle Kanten, die zu T_k führen, von einem einzigen Nachbarknoten kommen und alle Kanten, die von T_k wegführen, zu einem einzigen Nachbarknoten hinführen.

Beispiel 3.1-3

Betrachte Figur #06 in Bild 13 und das linke obere Teilnetz in Figur #04 aus dem DIN-Entwurf in Bild 12. Der Knoten [ASSIGN true TO varname] in Bild 12 soll ersetzt werden.
Die Einfügung von Bild 13 in Bild 12 geschieht hier ohne die Übernahme der Konnektoren (Dies erfolgt in Beispiel 4-2). Diese Einsetzung benötigt etwas Fingerspitzengefühl. Klar ist, daß der untere Nachbarknoten (P) in Bild 13

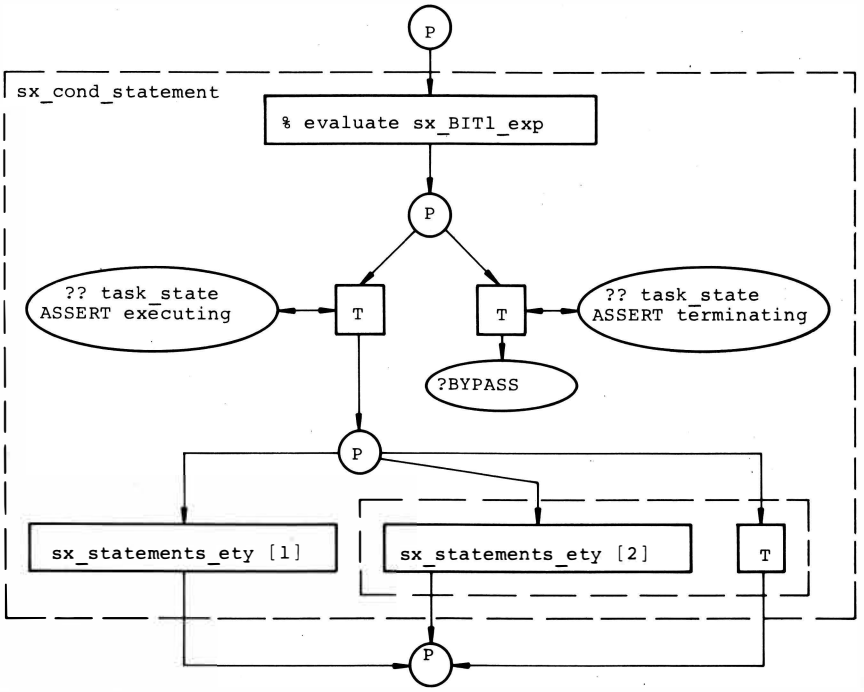


Bild 11: Figur #25 nach der Ersetzung des Knotens (F). durch Figur #02 aus dem DIN-Entwurf

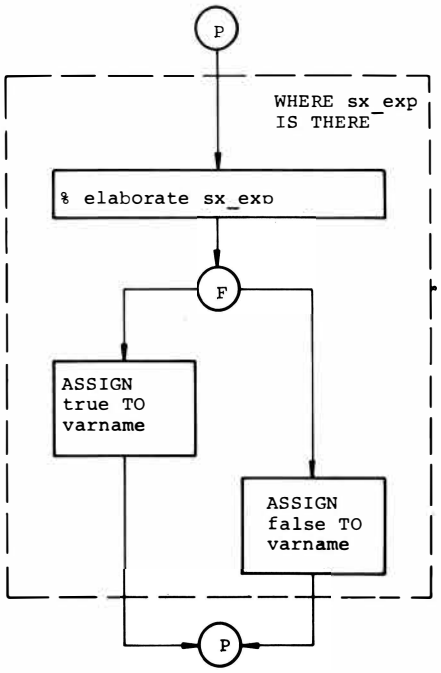


Bild 12: Linkes oberes Teilnetz aus Figur #04 aus dem DIN-Entwurf

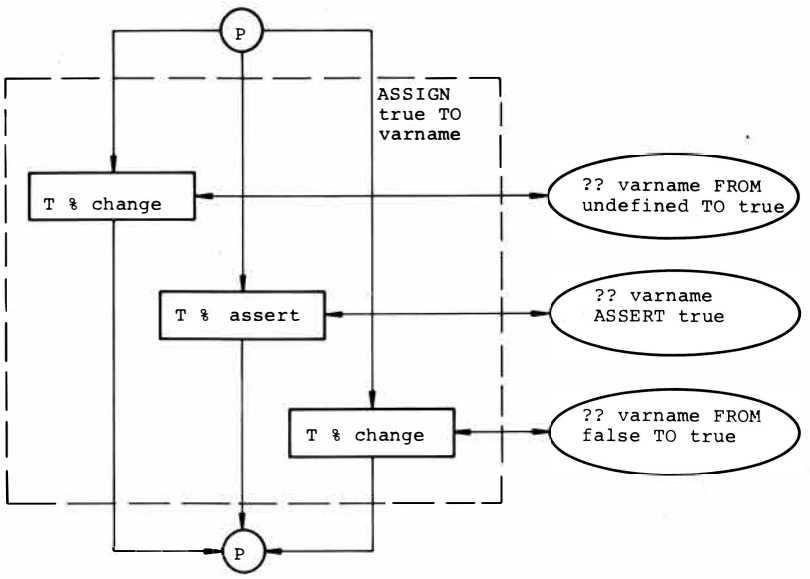


Bild 13: Figur #06 aus dem DIN-Entwurf

mit dem unteren Nachbarknoten (P) in Bild 12 identifiziert werden muß. Man erinnert sich nun, daß der Knoten (F) in Bild 12 als unteren Randknoten einen Knoten (P), siehe Bild 10, besitzt. Infolgedessen ist der obere Nachbarknoten (P) in Bild 13 mit dem Knoten (F) in Bild 12 zu identifizieren. Das Ergebnis der Einsetzung kann nun auf einfache Art erzeugt werden und ist in Bild 14 dargestellt.

3) Die Verteilung der Kanten kann auch durch die Unterscheidung von einfachen und Doppelpfeilen für T_k abgeleitet werden. Insbesondere können Doppelpfeile zu K durch mehrere Einfachpfeile bei T_k aufgelöst werden.

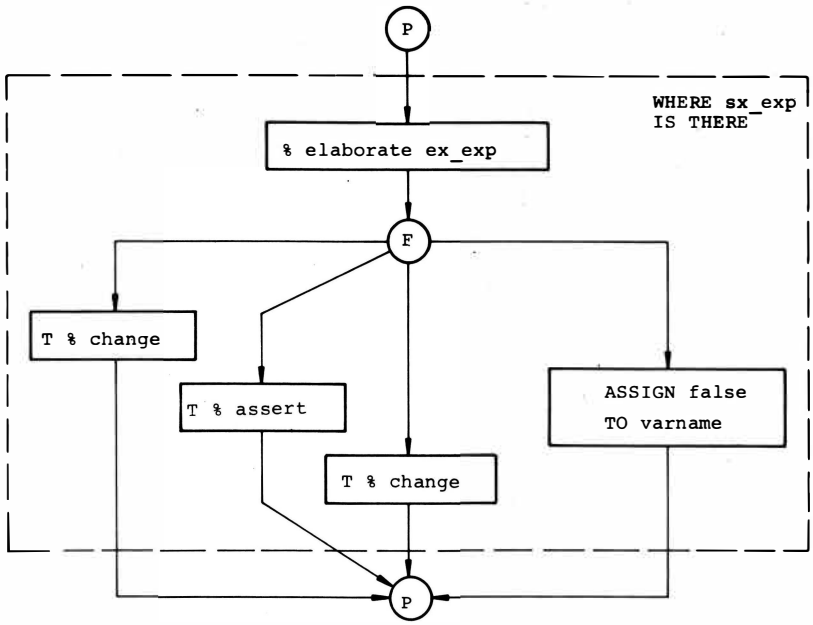


Bild 14: Einsetzung von Figur #06 in Figur #04, oberes linkes Teilnetz, ohne Übernahme der Konnektoren

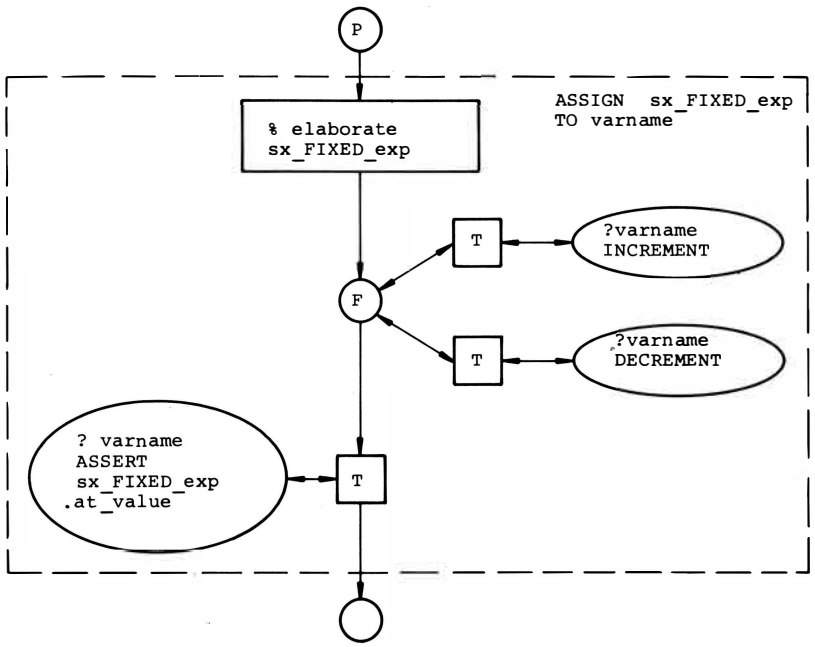


Bild 15: Figur #08, linkes Teilnetz, aus dem DIN-Entwurf

Beispiel 3.1-4

Ersetzung von Doppelpfeilen im Ausgangsnetz durch Einfachpfeile: Betrachte die Figuren #02 und #08, linkes Teilnetz, aus dem DIN-Entwurf in den Bildern 10 und 15. Geht man von Bild 15 aus, so müssen als erstes die beiden Doppelpfeile, die nach (F) führen, durch je zwei Pfeile entgegengesetzter Richtung ersetzt werden. Alle Pfeile, die zum Knoten (F) führen, müssen gemäß der in Beispiel 3.1-2 beschriebenen Regel zum oberen (P)-Randknoten des zu ersetzenden Knoten (F) in Bild 10 hinführen, und all die Kanten, die von (F) wegführen, müssen vom unteren (P)-Randknoten wegführen. Der Rest der Ersetzung ist recht einfach und braucht daher nicht noch einmal erklärt zu werden. Das Ergebnis der Ersetzung findet man in Bild 16.

4) Sollten diese Regeln noch nicht zu einer

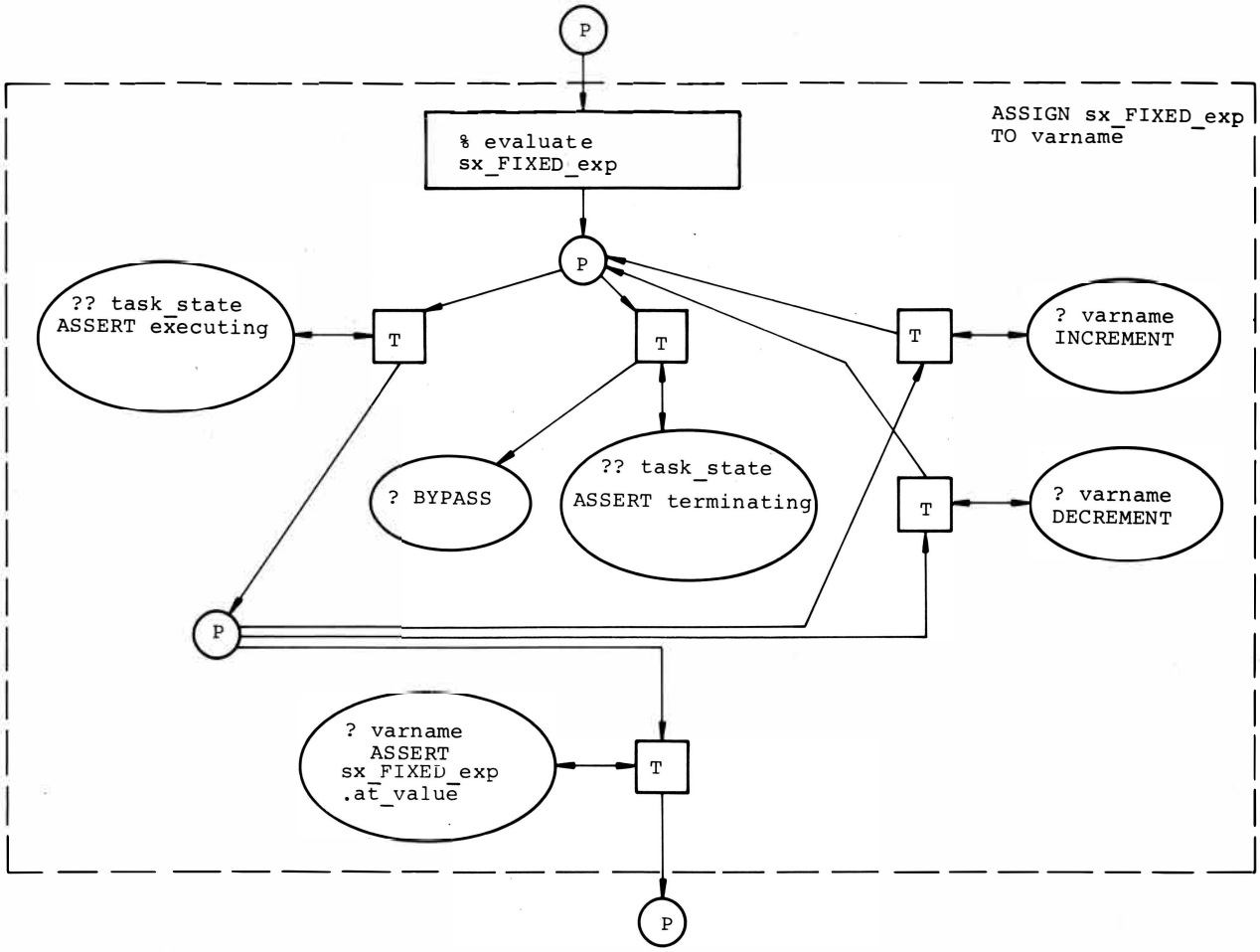


Bild 16: Ersetzung des Knoten (F) aus Bild 10 in Bild 15

eindeutigen Ersetzung bei unbenannten Nachbarknoten führen, ist die relative Lage der Kanten im Netz, in der sie K bzw. T_k berühren, ausschlaggebend. Neben dem obigen Beispiel 3.1-0 sei folgendes angegeben:

Beispiel 3.1-5

In Bild 17 findet man die Figur #27, unterer Teil, in der der Knoten (F) aus Bild 10 schon eingesetzt ist. Es geht jetzt darum, den Knoten [sx_loop_ scope] durch das Teilnetz in Bild 18 zu ersetzen. Das dort abgebildete Teilnetz hat zwei verschiedene Nach barknoten, zu denen Pfeile hinführen. Wie sind diese Knoten zu identifizieren? Entsprechend der Lage: Der linke (P)-Nachbarknoten in Bild 18 wird mit dem linken in Bild 17 identifiziert. Entsprechend wird mit dem rechten verfahren. Das Ergebnis der Ersetzung findet man in Bild 19.

5) Die Identifizierung der Nachbarknoten von K und T_k kann auch über die Knotennamen geschehen. Knoten mit denselben Inschriften werden identifiziert. Hiermit ist es bei komplizierteren Verknüpfungen insbesondere möglich, einen Nachbarknoten von T_k mit mehreren von K zu assoziieren.

Beispiel 3.1-6

Betrachtet man die Figuren #40 und #41 aus dem DIN-Entwurf unter der Annahme, daß die Bedingung "WHERE at_ task_op = sc_ACTIVATE" gilt, d.h. daß man den obersten Knoten (frame for a schedule ...) in Bild 20 und keinen der darunterliegenden ersetzen will, dann ist die Figur #41 aus dem DIN-Entwurf, wie in Bild 21 dargestellt, zu verwenden. Das dort dargestellte Teilnetz ersetzt den Knoten (frame for a schedule sc_ACTIVATE) in Bild 20. Die Verknüpfung des Teilnetzes in Bild

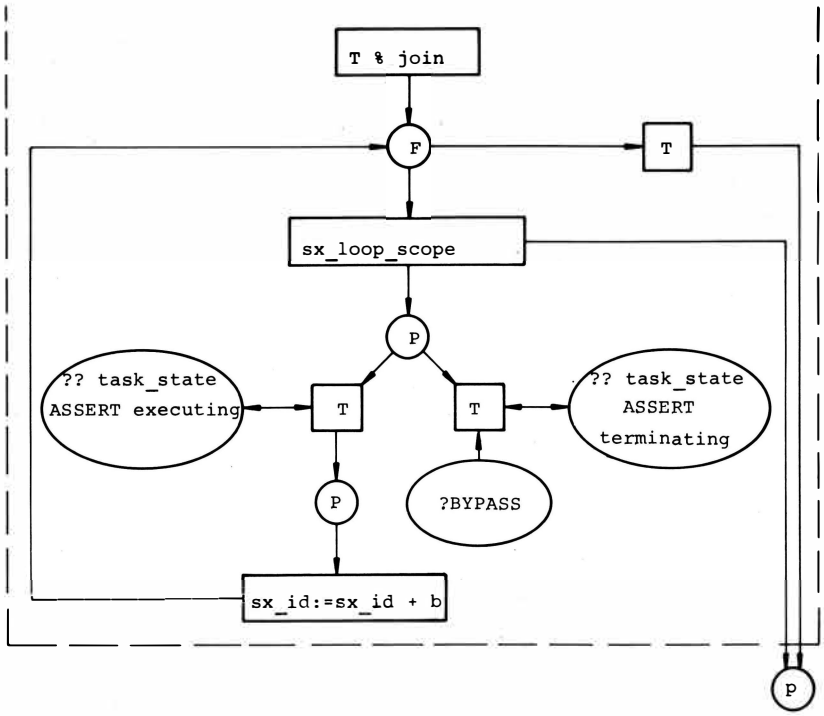


Bild 17: Figur #27, unteres Teilnetz, aus dem DIN-Entwurf

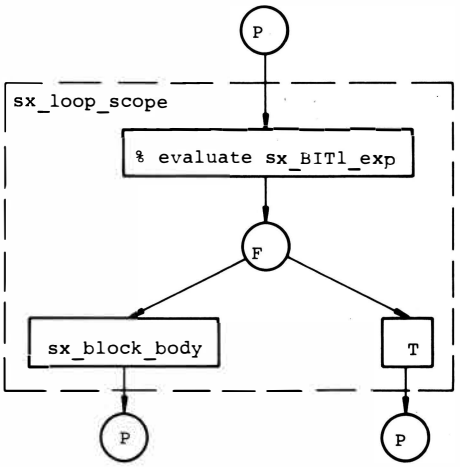


Bild 18: Figur #28 aus dem DIN-Entwurf

21 mit dem Knoten [? no activation pending] in Bild 20 muß vorgenommen werden. Die Doppelpfeile zu dem ersetzten Knoten werden z.T. durch mehrere Einfachpfeile ersetzt. Die weitere Identifizierung der Knoten ist so einfach, das das Ergebnis-Netz nicht noch einmal dargestellt werden soll.

3.2 Bedingte Ableitungen

Die Verfeinerung eines Netzes hängt generell vom einzelnen Programm, das zu erzeugen ist, also von der Wahl der Alternativen in den syntaktischen Produktionsregeln und von der möglichen Implementierung ab. Die Netze sind im DIN-Entwurf gewissen Ab-

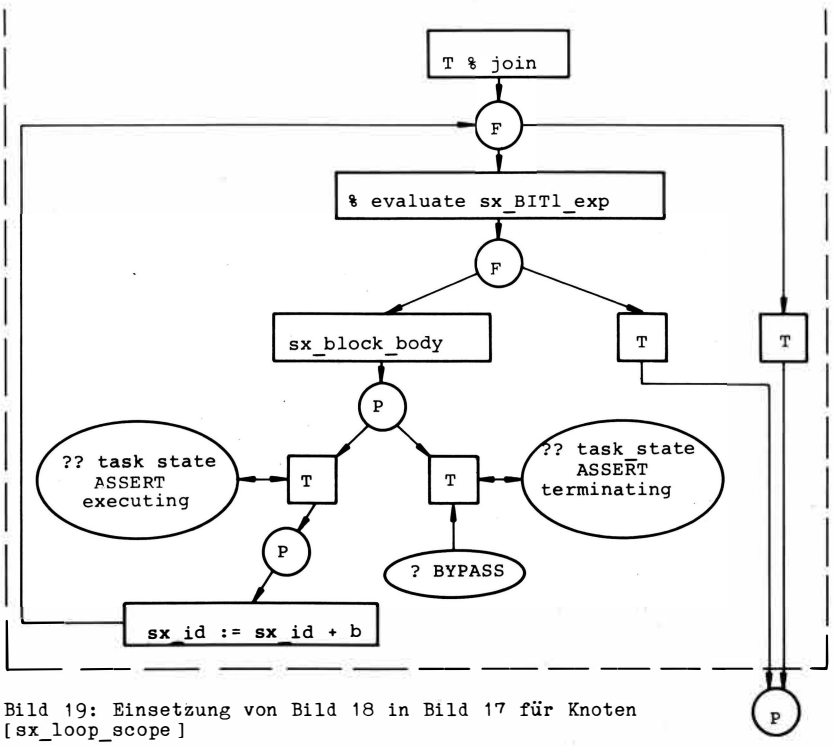


Bild 19: Einsetzung von Bild 18 in Bild 17 für Knoten [sx_loop_scope]

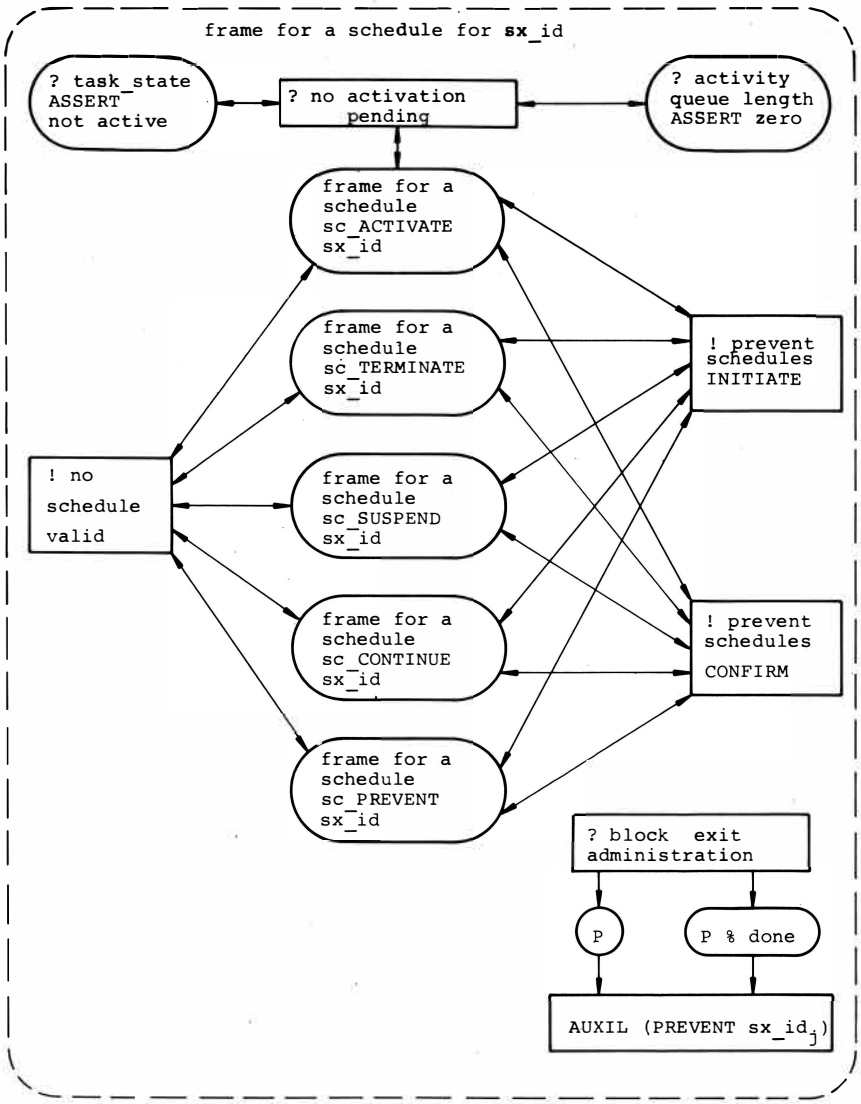


Bild 20: Figur #40 aus dem DIN-Entwurf

leitungen zugeordnet, deren Semantik sie beschreiben. Die vorgegebenen Netze bilden abstrakte Darstellungen der Konzepte und Implementationen gewisser syntaktischer Konstrukte.

Die durch die attributierte Grammatik beschriebenen Kontextbedingungen sind in den Knoten bzw. Teilnetzen als WHERE-Konstrukte beschrieben. Der Gültigkeitsbereich einer Kontextbedingung wird durch eine gestrichelte Kontur, interpretierbar als inneres Teilnetz, gekennzeichnet. Ein Knoten oder Teilnetz wird genau dann erzeugt, wenn diese Kontextbedingungen zutreffen. Ansonsten wird der Knoten (bzw. das Teilnetz) inclusive aller ihn berührenden Kanten aus dem Netz entfernt. Die Bedingung kann die bekannten Boole'schen Operatoren (AND, OR, ...) und Prädikate aus der attributierten Grammatik der Sprache Full PEARL, z.B. "IS THERE", enthalten (s. auch Beispiel 3.1-3).

Beispiel 3.2-1

Man betrachte die Figur #04 aus dem DIN-Entwurf im Bild 22. Es sei angenommen, daß die Bedingung "UNLESS sx_exp IS THERE" nicht zutrifft. Dann entfällt das rechte gestrichelt umrandete Teilnetz in Bild 22, wie in Bild 23 dargestellt. Alle zu den [T]-Knoten hin- und von ihnen wegführenden Kanten entfallen.

Im allgemeinen können Daten und Deklarationen als Stellen und Operationen durch Transitionen dargestellt werden. Die Verknüpfung einer Deklaration (Stelle) einer Ope-

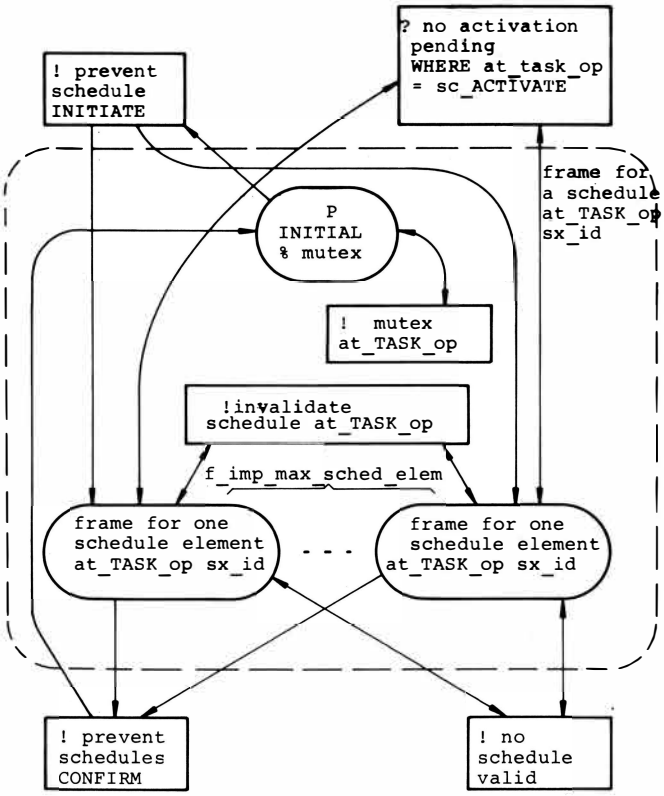


Bild 21: Figur #41 aus dem DIN-Entwurf

ration mit ihrem Aufruf (Transition) geschieht durch die im nächsten Abschnitt zu beschreibenden Konnektoren.

4. Konnektoren

Konnektoren beschreiben Kontextbeziehungen und semantische Eigenschaften eines Programms. Ihr Zweck ist in dem DIN-Entwurf dargestellt. Hier soll nur ihre Behandlung, d.h. insbesondere ihre Eliminierung, erörtert werden.

Konnektoren sind dadurch gekennzeichnet, daß ihrem Knotennamen ein Ausrufezeichen (!) oder Fragezeichen (? oder ??) voranstehen. Konnektoren mit Ausrufezeichen heißen Konnektordefinitionen; Konnektoren mit Fragezeichen heißen Konnektoranwendungen. Um eine u.U. entstehende Unsicherheit zu beseitigen, sei an dieser Stelle bemerkt, daß die Konnektoren mit einem vorangestellten Fragezeigen Kontextbeziehungen beschreiben, die schon zur Übersetzungszeit geklärt wer-

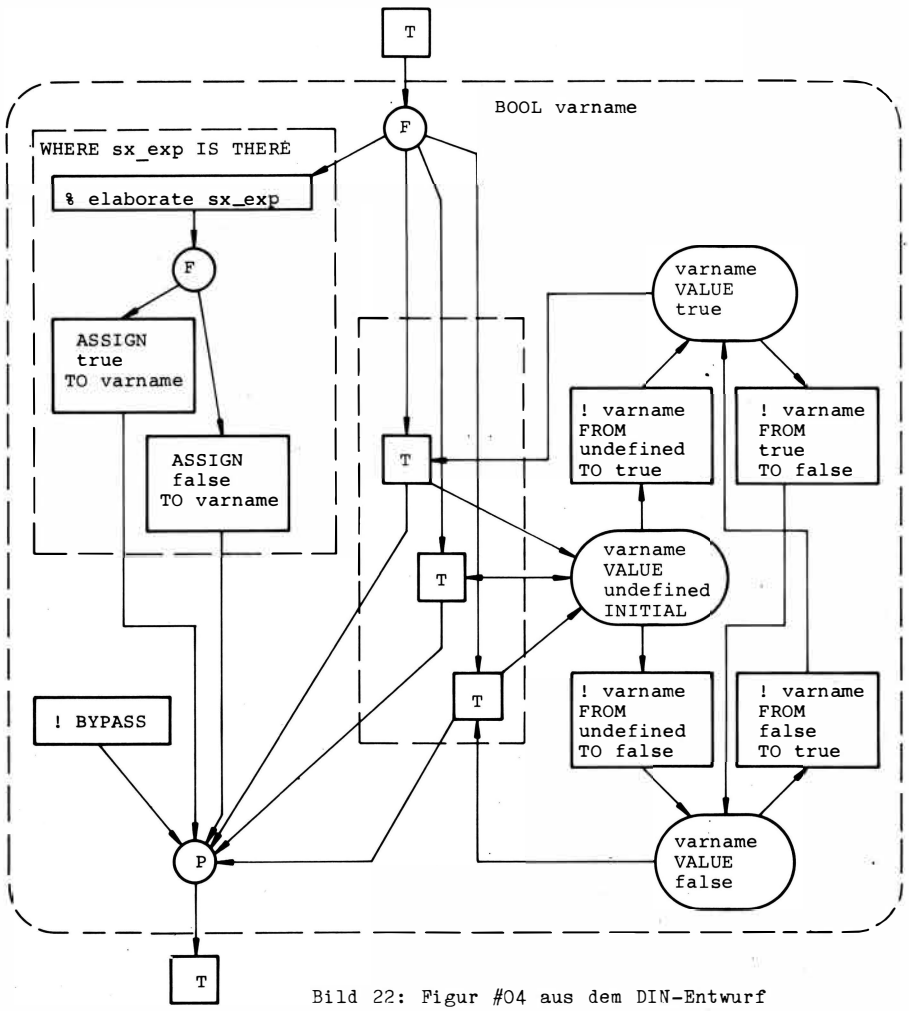


Bild 22: Figur #04 aus dem DIN-Entwurf

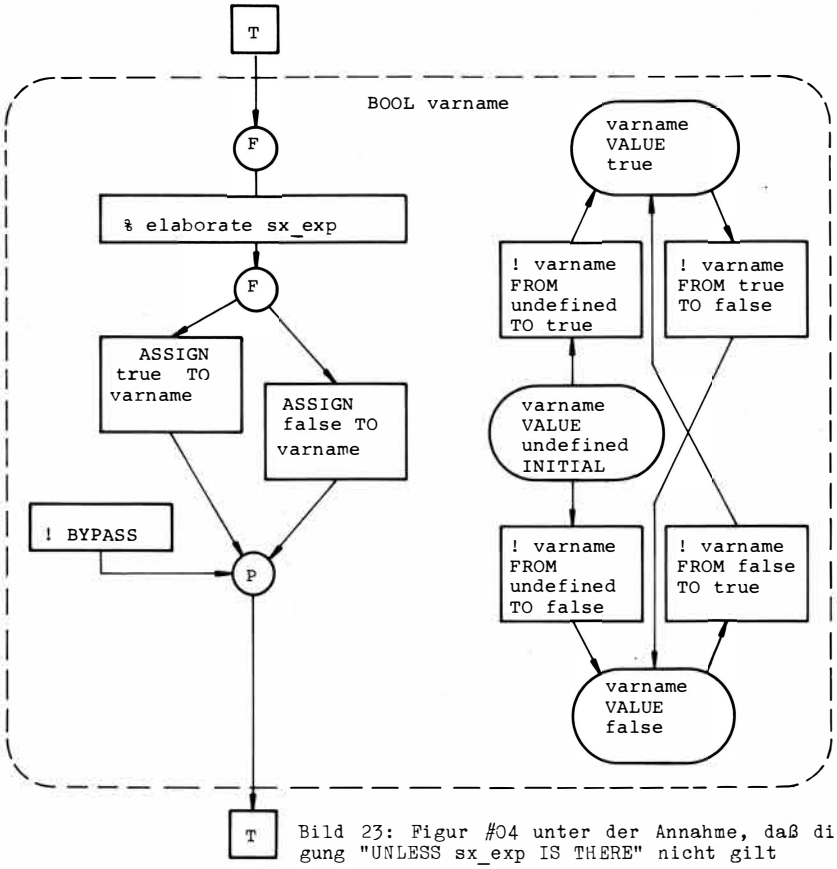


Bild 23: Figur #04 unter der Annahme, daß die Bedingung "UNLESS sx_exp IS THERE" nicht gilt

den können (statisches Binden), und die mit zwei vorangestellten Fragezeichen solche, die erst zur Laufzeit festliegen (dynamisches Binden). Das dynamische Binden wird in dem (statischen) Petri-Netz derart dargestellt, daß alle möglichen Abläufe modelliert werden. Die Tatsache, daß ein Konnektor ein oder zwei Fragezeichen besitzt, ist für die formalen Eliminationsregeln insofern relevant, als auf unterschiedliche Art von der Konnektoranwendung aus die dazugehörige Konnektordefinition zu suchen ist.

Will man ein Condition/Event-Netz mit ausschließlich elementaren Knoten herleiten, sind zuerst alle Ersetzungen von Knoten durch Teilnetze auszuführen. Ist man in diesem Produktions-Prozeß an dem Punkt angekommen, daß kein Knoten im Netz mehr ersetzt werden kann, müssen noch die Konnektoren eleminiert werden, um zu einem reinen Condition/Event-Netz gelangen zu können. Die bei der Substitiution von Knoten durch Teilnetze auftretenden Konnektoren dürfen also erst eliminiert werden, wenn das Netz restlos verfeinert ist. Es muß sichergestellt sein, daß nicht in einem späteren Verfeinerungsschritt Konnektoranwendungen auftreten, für die es keine Konnektordefinitionen mehr gibt, da diese schon eliminiert worden sind. Es ist also zweierlei zu beachten:

1. Die Substitutionsschritte müssen rekonstruierbar sein, insbesondere die eingesetzten Teilnetze.
2. Die Konnektoren dürfen erst eliminiert werden, wenn das Netz nicht weiter verfeinert werden kann.

Die Eliminierung von Konnektoren wird in drei Schritten durchgeführt.

1) Für jede Konnektoranwendung finde eine einzelne, eindeutige Konnektordefinition. Die Inschriften, insbesondere die Knotennamen, der Knoten müssen identisch sein bis auf die Konnektorkennzeichnungen (!, ? oder ??). Der Knoten der Konnektordefinition hat dabei eine andere Kontur als der bzw. die Knoten der Konnektoranwendung. Entweder sind alle Anwendungen S-Knoten und die Definition ein T-Knoten oder umgekehrt.

2) Ersetze jede Kante zu einer Konnektoranwendung durch eine Kante zu jedem der Knoten, zu dem es von der Konnektordefinition aus eine Kante gibt. Tue entsprechend dasselbe mit den Kanten, die von einer Konnektoranwendung wegführen. Dabei müssen Doppelpfeile gegebenenfalls in je einen Hin- und einen Rückpfeil aufgespalten werden.

3) Entferne alle Konnektoranwendungen, deren Pfeile im Schritt 2 ersetzt worden sind und die jetzt isolierte Knoten darstellen. Entferne dann alle Konnektordefinitionen, auch wenn es zu ihnen keine Konnektoranwendungen gegeben hat, incl. aller zu oder von ihnen wegführenden Kanten.

Der einfachste Fall ist die Verknüpfung zweier Netze über Konnektoren. Dieser soll als erster erläutert werden.

Beispiel 4-1

Man betrachte die Figur #03 aus dem DIN-Entwurf mit ihren beiden Teilnetzen, wie sie in Bild 24 dargestellt sind. Das obere Teilnetz besitzt zwei verschiedene Konnektoranwendungen, die zwei gleichbenannten Konnektordefinitionen des unteren Teilnetzes gegenüberstehen. Die Doppelpfeile zu den Konnektoranwendungen spalte

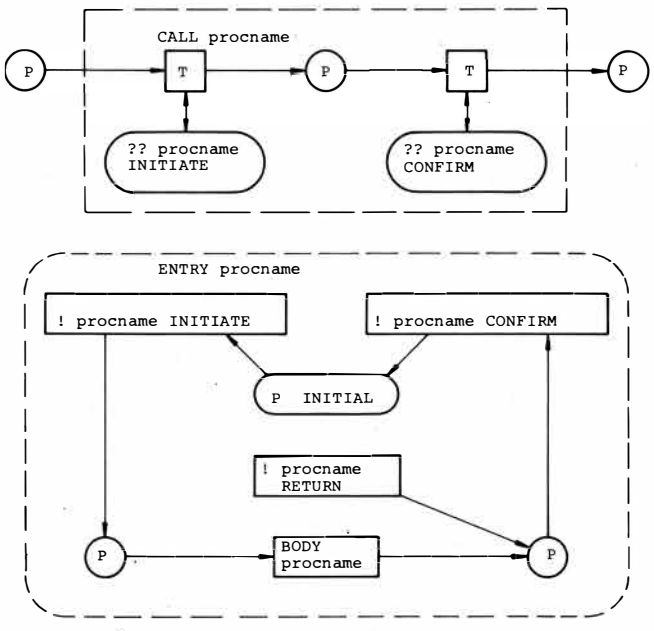


Bild 24: Figur #03 aus dem DIN-Entwurf

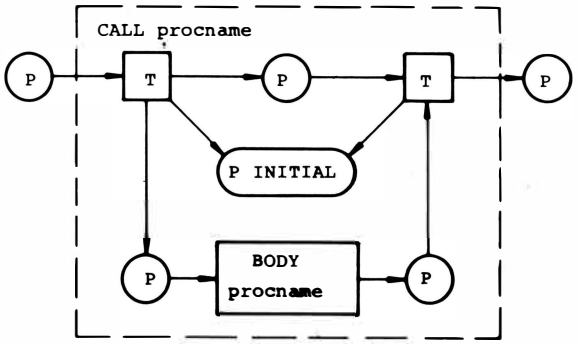


Bild 25: Verknüpfung der beiden Teilnetze aus Bild 24 durch Eliminierung der Konnektoren

in je zwei Einfachpfeile entgegengesetzter Orientierung auf. Identifiziere die Konnektoranwendungen mit ihren Konnektor definitionen und ersetze die Kanten. Die Konnektordefinition [!procname RETURN] findet keine -anwendung und wird gestrichen incl. der von ihr ausgehenden Kante. Das Ergebnis der Konnektoreliminierung findet man in Bild 25.

Dieses Beispiel kennzeichnet jedoch nur einen Spezialfall davon, daß dynamisches Binden kennzeichnende Konnektoren eliminiert werden sollen (also solche, deren Konnektoranwendungen durch "??" gekennzeichnet sind). Die wesentlichen (praktischen) Schwierigkeiten bei der Eliminierung von Konnektoren liegen bei der Suche zusammengehöriger Konnektoranwendungen einerseits und der dazugehörigen Konnektordefinition andererseits. Und bezüglich dieser Suche unterscheiden sich Konnektoranwendungen, die statisches Binden (also solche mit "?"), und die, die dynamisches Binden kennzeichnen, erheblich. Konnektordefinitionen gleichen Namens können mehrfach auftreten in einem Netz (z.B. [!BYPASS] in Beispiel 4-3) und die -anwendungen müssen in korrekter Weise den -definitionen zugeordnet werden. Deshalb besitzen Suche und Zuordnung von Konnektordefinitionen eine besondere Bedeutung. Beschäftigen wir uns als erstes mit Konnektoren, die dynamisches Binden kennzeichnen.

Betrachtet man die Verfeinerung eines Netzes N, so werden für einige Knoten K_i in N Teilnetze T_{ki} eingesetzt. In diesen Teilnetzen T_{ki} werden wiederum gewisse Knoten

K_{ij} durch Teilnetze T_{kij} ersetzt usw. Die Verfeinerungsstruktur eines Netzes läßt sich also als Baumstruktur auffassen, wobei das Ausgangsnetz N, das verfeinert wird, die Wurzel des Baumes ist und die Knoten K_i, die durch Teilnetze T_{ki} ersetzt werden, Unterbäume beschreiben. T_{ki} ist in dem Unterbaum i der Wurzelknoten, der wiederum Unterbäume (i,j) hat für all die Knoten K_{ij}, die durch Teilnetze T_{kij} ersetzt werden, usw. Die Wurzel N bilde die Ebene 0, die Knoten K_i die Ebene 1, die Knoten K_{ij} die Ebene 2 usw. Bei der Suche der Konnektordefinition zu einer gegebenen Konnektoranwendung K, die durch "??" gekennzeichnet ist, gehe, wie folgt, vor: Angenommen K tritt auf der Ebene n in dem Verfeinerungsbaum auf, und zwar in dem Teilnetz zum Knoten $K_{i1,i2,...,i(n-1),in}$. Prüfe nun, ob in demselben Teilnetz eine passende Konnektordefinition liegt oder in einem Teilnetz, das bezüglich der Verfeinerungsstruktur in dem Baum unterhalb des Knotens $K_{i1,i2,...,in}$ liegt; suche also alle Teilnetze des Unterbaums (i1,i2,...,in) nach einer passenden Konnektordefinition ab. Führt diese Suche nicht zum Erfolg, gehe bzgl. der Verfeinerungsstruktur einen Knoten in Richtung Wurzel zurück, also zum Knoten $K_{i1,i2,...,i(n-1)}$; untersuche das dazugehörende Teilnetz sowie alle Teilnetze des Baums (i1,i2,...,i(n-1)) in Richtung zu den Blättern. Die Suche erfolgt beim Absteigen zu den Blättern Ebene für Ebene.

Findet man auch in diesem Unterbaum keine passende Konnektordefinition, so gehe noch einen Knoten zurück im Gesamtverfeinerungsbaum des Netzes N, also zum Knoten $K_{i1,i2,...,i(n-2)}$ und untersuche dessen Unterbaum (i1,i2,...,i(n-2)). Wiederhole diesen Prozeß, bis eine erste passende Konnektordefinition im Unterbaum (i1,i2,...,ik) gefunden ist. Nimm dann die Konnektordefinition K', die als erste passende auftritt und die in dem Unterbaum (i1,i2,...,ik) die kleinste Ebenennummer besitzt. Zeichnet man den Verfeinerungsbaum von oben nach unten, d.h. derart, daß die Wurzel oben liegt und die Blätter des Baums unten, so nimm die Konnektordefinition, die am weitesten oben in dem betrachteten Unterbaum liegt.

Beispiel 4-2

Als erstes soll die Figur #06 (siehe Bild 13) aus dem DIN-Entwurf in das Netz aus Beispiel 3.2-1 in Bild 23 eingesetzt werden. Im Bild 26 ist das Ergebnis der Einsetzung abgebildet. Damit das Netz auf normalem Papierformat noch darstellbar ist, soll die Eliminierung der Konnektoren [!varname FROM undef/true/false TO true/false] vorgezogen werden. Es wird deshalb der Einfachheit halber angenommen, daß weitere Verfeinerungen keine weiteren Konnektoranwendungen für die Konnektordefinitionen im rechten Teilnetz produzieren. Die Konnektordefinition [!BYPASS] darf nicht gestrichen werden, da sie die Konnektoranwendungen aus den (F)-Knoten zu befriedigen hat. Die Konnektoranwendungen (?? varname ASSERT true/false) bleiben vorerst stehen. Sie werden ja sowieso nicht gestrichen nach dem oben besprochenen Eliminierungsverfahren. Die Doppelpfeile, die zu den zu eliminierenden Konnektoranwendungen führen, müssen durch zwei Einfachpfeile entgegengesetzter Orientierung ersetzt werden. Führt man dann die oben genannten drei Eliminierungsschritte aus, erhält man das Netz in Bild 27. Bei den Konnektoranwendungen (?? varname FROM undefined/true/false TO true/false) lagen die dazugehörigen Konnektordefinitionen in demselben Teilnetz. Bei der Suche der Konnektordefinitionen [! varname ASSERT true/false] müssen die Verfeinerungen der Knoten (varname VALUE true/false) untersucht werden. Eines der beiden (im wesentlichen gleichen) Teilnetze ist in Bild 28 dargestellt (siehe Figur #05 im DIN-Entwurf). Dort treten die gesuchten Konnektordefinitionen auf. Wir setzen also diese Teilnetze in das Bild 27 ein und eliminieren gleich die Konnektoren. Das Ergebnis ist in Bild 29 dargestellt.

Es bleibt noch der Fall zu besprechen, daß Konnektoren, die statisches Binden kennzeichnen, eliminiert werden sollen, also solche, bei denen die Konnektoranwendungen durch "?" gekennzeichnet sind. Die Frage ist wieder, wie findet man zu einer gegebe-

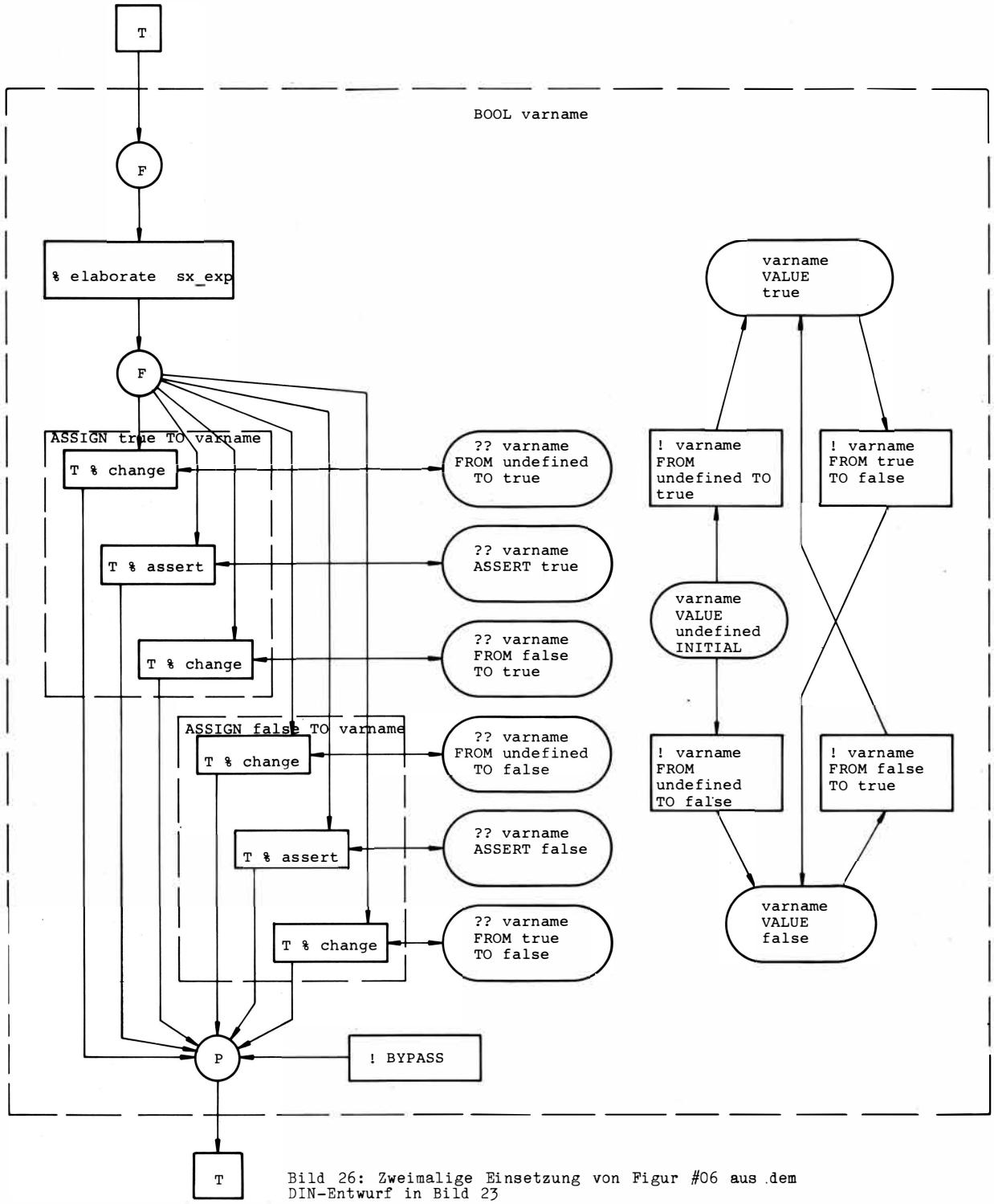


Bild 26: Zweimalige Einsetzung von Figur #06 aus dem DIN-Entwurf in Bild 23

nen Konnektoranwendung K die dazugehörige -definition. Sei wieder ein Netz N mit seinem Verfeinerungsbaum, wie oben besprochen, gegeben. Eine Konnektoranwendung ist dann mit der in der Verfeinerungsstruktur in Richtung des Baumknotens nächstliegenden Konnektordefinition zu identifizieren. Anders ausgedrückt: Wenn eine Konnektoranwendung K auf der Ebene n in dem Baum in dem Teilnetz zum Knoten $K_{i1,i2,\ldots,in}$ auftritt, dann sucht sie ihre Konnektordefinition erst in ihrem eigenen Teilnetz, dann in dem zum Knoten $K_{i1,i2,\ldots,i(n-1)}$, dann in dem zu $K_{i1,i2,\ldots,i(n-2)}$ usw., bis sie die erste passende Konnektordefinition K' gefunden hat, und bricht ab mit der Suche. Diese Konnektordefinition K' wird mit K assoziiert.

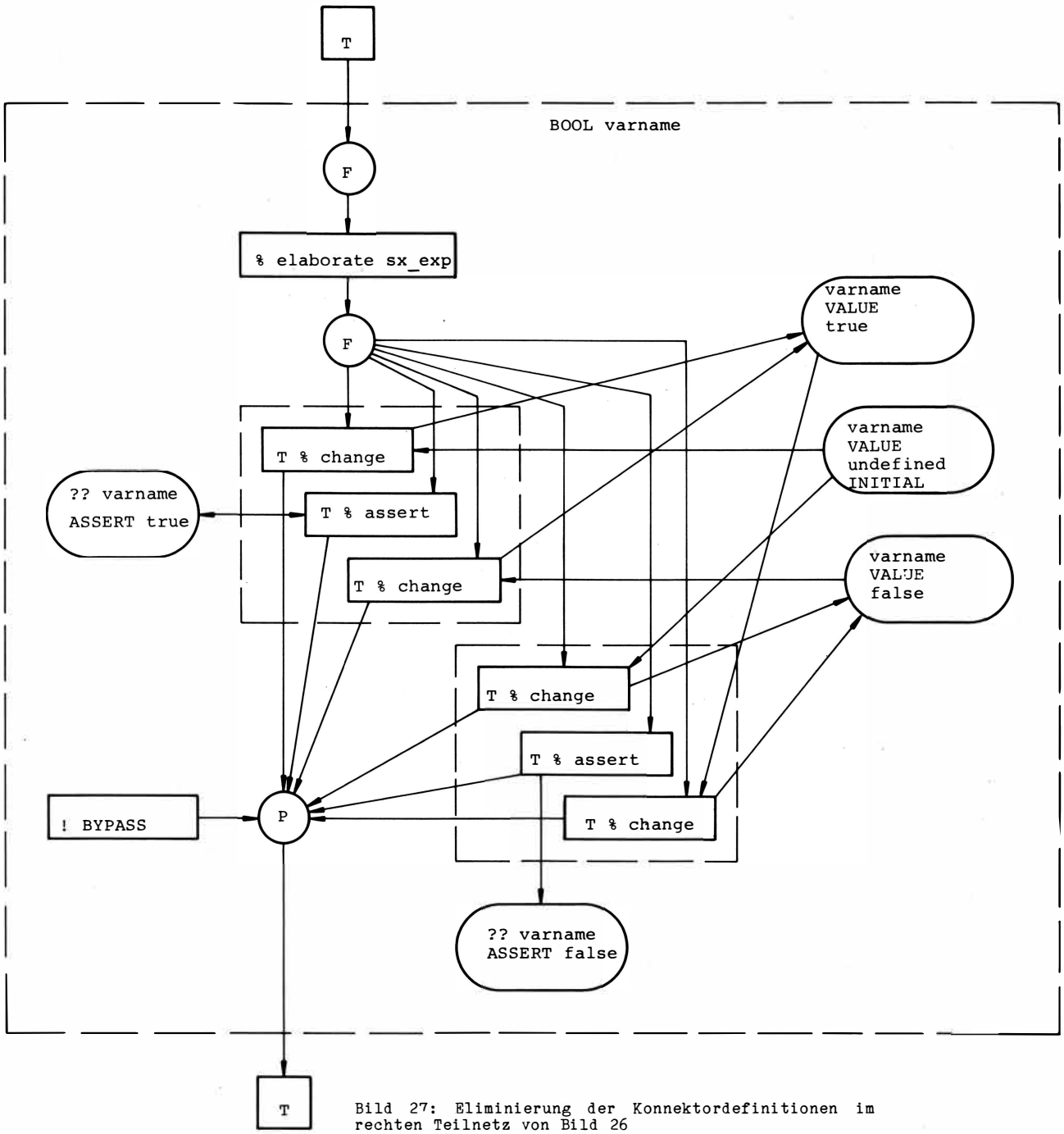


Bild 27: Eliminierung der Konnektordefinitionen im rechten Teilnetz von Bild 26

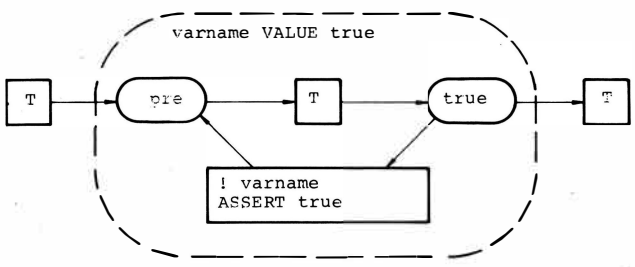


Bild 28: Figur #05 aus dem DIN-Entwurf für "valname = true"

Beispiel 4-3

Das Auftreten mehrerer Konnektordefinitionen sei an der Figur #38 aus dem DIN-Entwurf in Bild 30 demonstriert, in die die Figuren #11 in Bild 31 und #02 in Bild 10 eingesetzt werden sollen. Wieder sollen die Einschränkungen über die Verfeinerungen aus dem vorigen Beispiel gel-

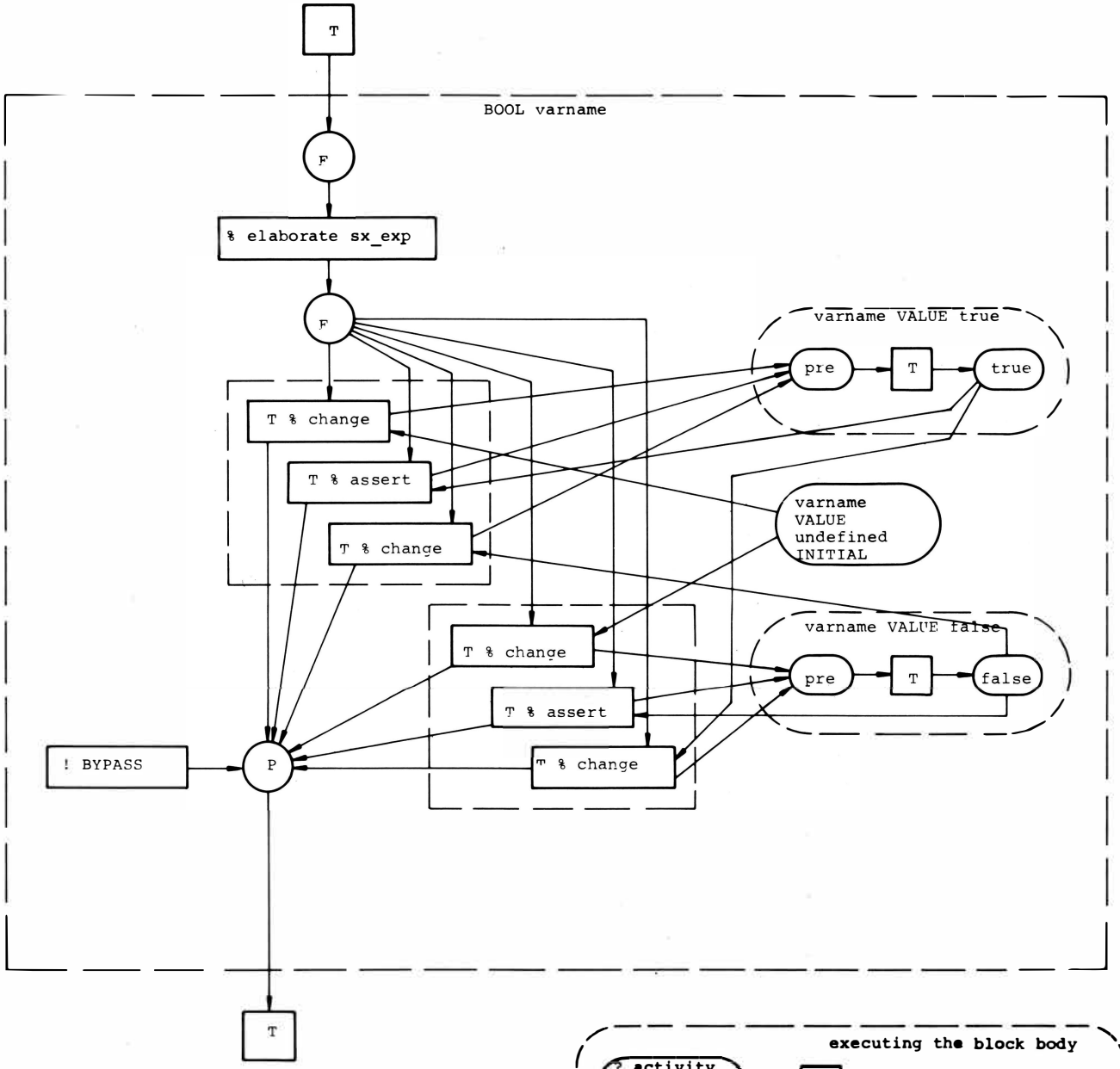


Bild 29: Bild 27 nach Einsetzung des Bildes 28 und Eliminierung der Konnektoren

ten. Es sei angenommen, daß der Knoten (task_state) keine Konnektoren (?BYPASS) mehr erzeugt. Nach der Einsetzung hat das Netz die Struktur wie in Bild 32, wenn im Teilnetz [sx_block_body] die Knoten (F) auch wiederum durch das Teilnetz in Bild 10 ersetzt worden sind. Die Konnektoranwendungen (?BYPASS) im Teilnetz [sx_block_body] werden mit der Konnektordefinition [!BYPASS] in diesem Teilnetz identifiziert. Würde man das Netz noch weiter verfeinern, müßten die Knoten (?BYPASS), die durch die Verfeinerung der Knoten [sx_

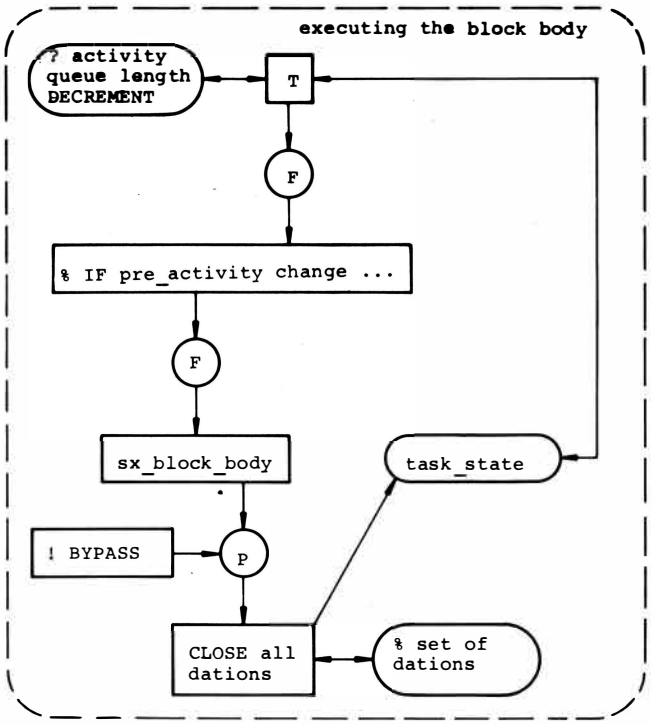


Bild 30: Figur #38 aus dem DIN-Entwurf

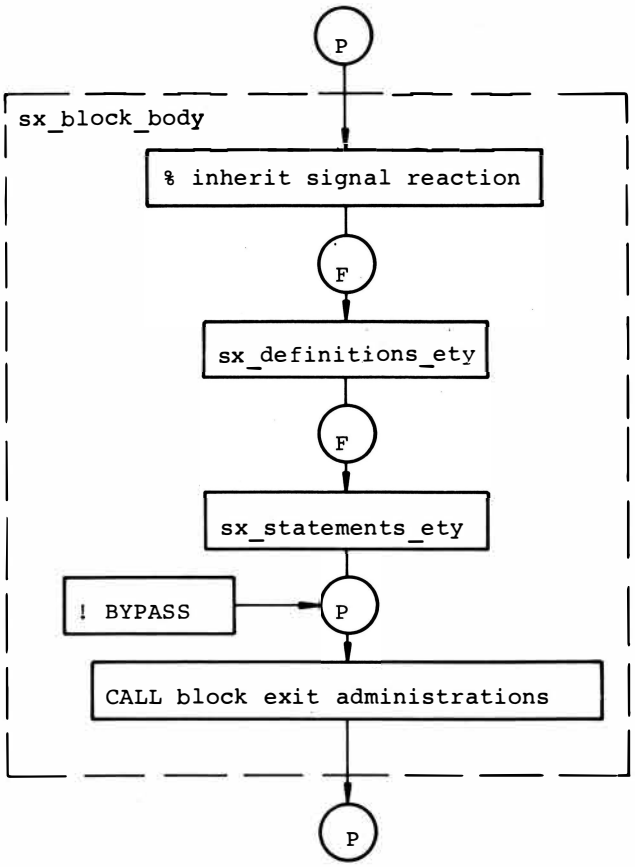


Bild 31: Figur #11 aus dem DIN-Entwurf

definitions_ety] und [sx_statements_ety] produziert werden, mit dem [!BYPASS]-Knoten im Teilnetz [sx_block_body] identifiziert werden. Die beiden übrig bleibenden Konnektoranwendungen (?BYPASS) ganz oben werden mit der Konnektordefinition [!BYPASS] ganz unten in Bild 32 assoziiert. Diese Konnektoren liegen in einem äußeren Verfeinerungsblock bezüglich des Teilnetzes [sx_block_body]. Das Ergebnis der Konnektoreliminierung findet man in Bild 33.

Zum Abschluß möchte ich noch Prof. Dr. H. Rzehak, Hochschule der Bundeswehr, und Dr. E. Wegner, GMD, für die Durchsicht des Manuskriptes und die vielen Anregungen danken.

Literatur

[1] DIN 66 253 Teil 2 (Full PEARL), DIN-Entwurf, Nov. 1980, Beuth Verlag

[2] E. Wegner: Transforming nets along the syntactic production of programs; First European Workshop on the Application and Theory of Petri-Nets, Strasbourg, Sept. 1980, (revised paper)

[3] E. Wegner: Semantics of a Language for Describing Systems and Processes; IST Report 36 revised, manuscript Jan. 1978

[4] E. Wegner: private communication

[5] K. Zuse: Petri-Netze aus der Sicht des Ingenieurs; Vieweg + Sohn Verlag, Braunschweig, 1980

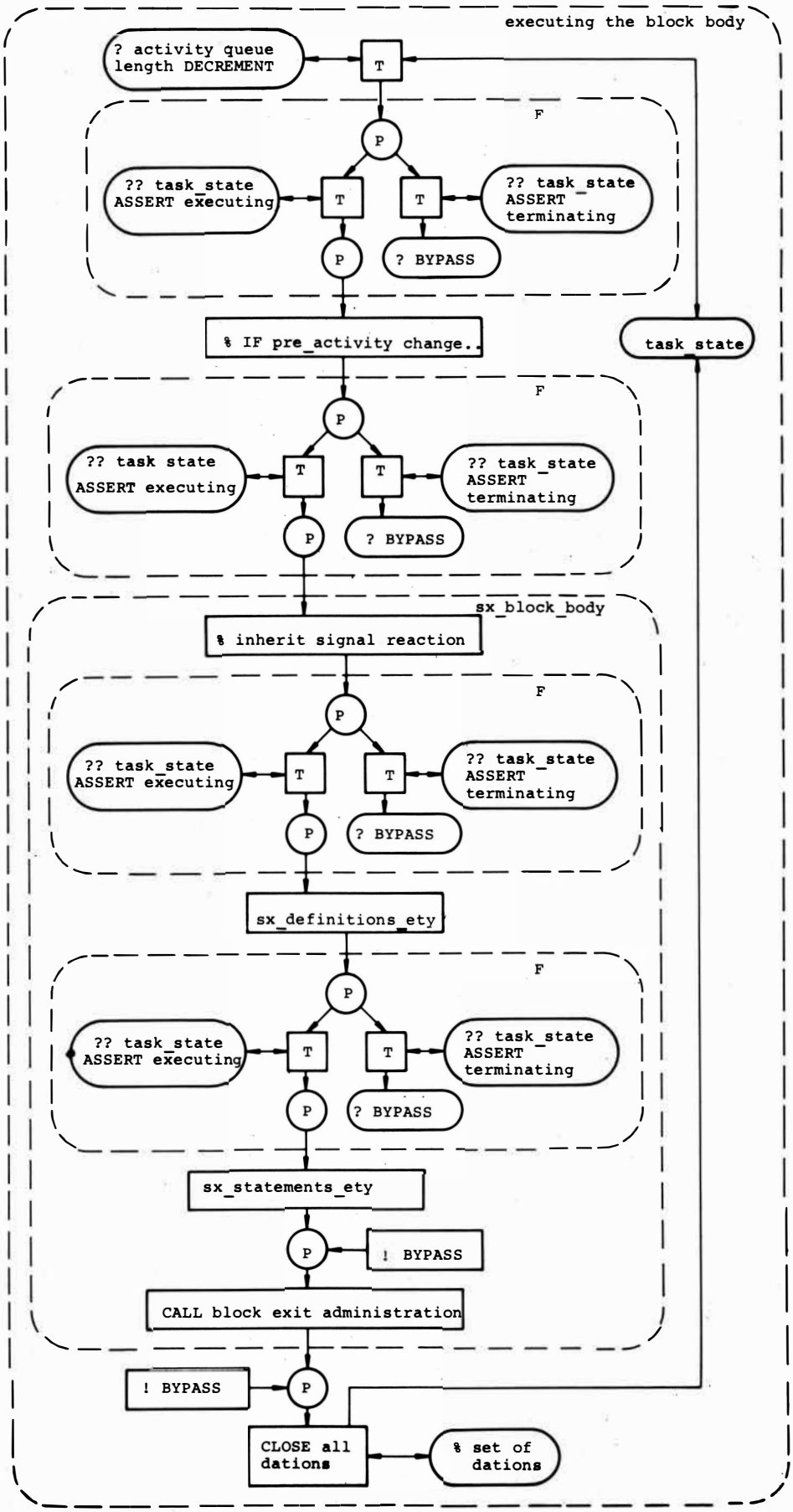


Bild 32: Bild 31 nach Einsetzung der Teilnetze

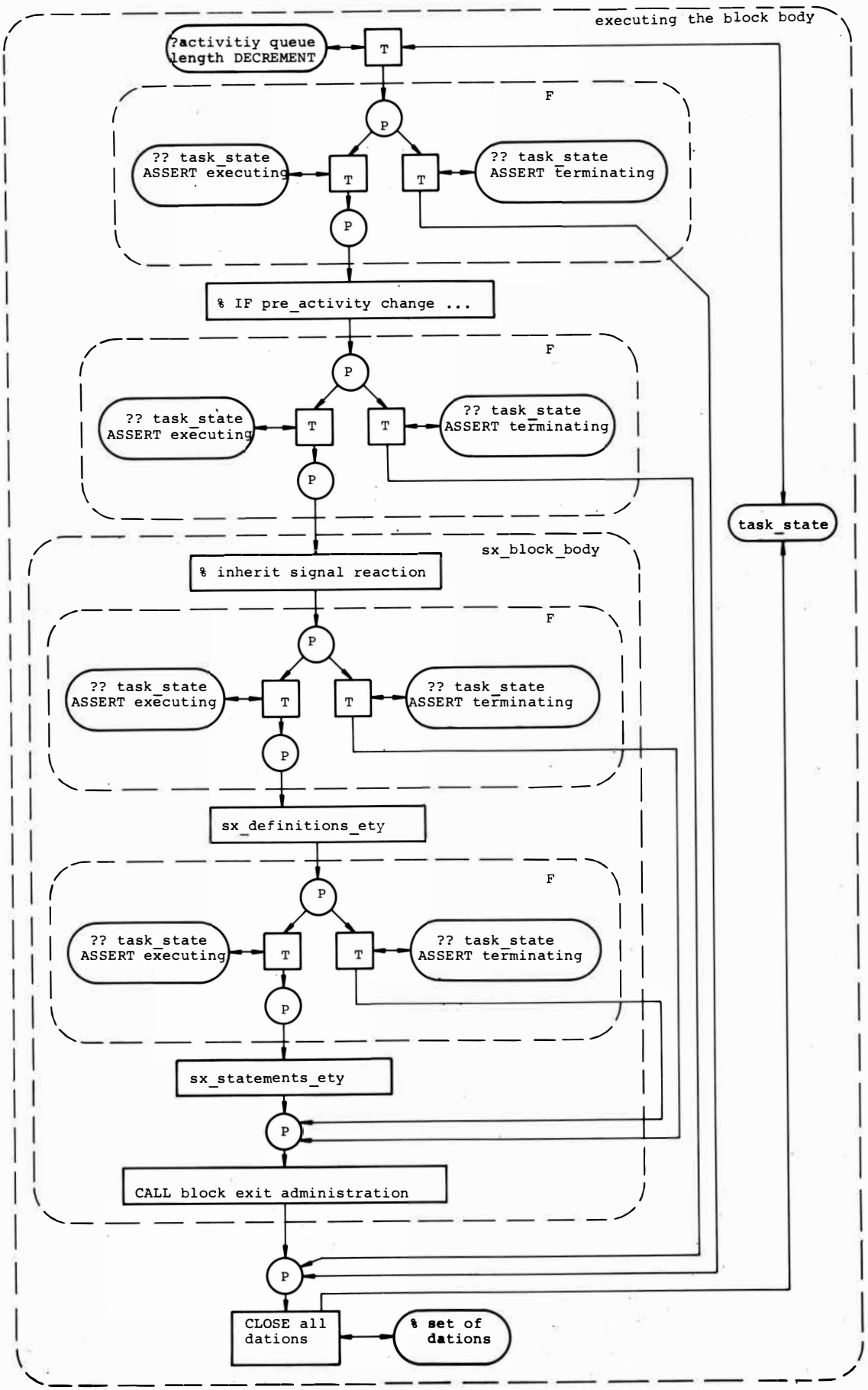


Bild 33: Bild 32 nach Eliminierung der Konnektoren (?BYPASS) und [!BYPASS]