7 Beschreibungsmethoden

Für die Beschreibung von Kommunikationsprotokollen werden unterschiedliche Methoden genutzt. Speziell in den 80er Jahren sind eine Vielzahl von Ansätzen untersucht worden, aus denen sich einige grundlegende Methoden herauskristallisiert haben, von denen einige wiederum als semantische Modelle für die Entwicklung formaler Beschreibungstechiken genutzt wurden. Bevor wir auf letztere zu sprechen kommen, sollen die wichtigsten Beschreibungsmethoden in diesem Kapitel vorgestellt werden. Wir beginnen mit einer kurzen Zusammenfassung der Anforderungen an die Dienst- und Protokollspezifikation. Danach wird die Notwendigkeit formaler Beschreibungen begründet. Im dritten Abschnitt werden die prinzipiellen Herangehensweisen an die Protokollbeschreibung diskutiert. Anschließend werden die wichtigsten Beschreibungsmethoden vorgestellt.

7.1 Dienst- und Protokollspezifikation

Kommunikationsprotokolle werden zum größten Teil in Software realisiert. Diese Software wird als **Kommunikationssoftware** bezeichnet. Ihre Erstellung folgt prinzipiell den für die Softwareentwicklung typischen Abläufen (siehe Kapitel 9). Sie ist jedoch durch eine Reihe von Besonderheiten geprägt, die sich aus der Spezifik der Kommunikationsprotokolle ergeben. Diese Besonderheiten prägen auch die Beschreibung der Protokolle.

Die Beschreibung bzw. Spezifikation (beide Begriffe werden hier synonym gebraucht) der Kommunikationsprotokolle bildet die Grundlage des Protollentwicklungsprozesses. In ihr werden die Anforderungen an die zu realisierenden Dienste und Protokolle formuliert. Die Dienst- und die Protokollspezifikation sind die Referenz für die Implementierung des Protokolls sowie die zugeordneten Validationsschritte (Verifikation, Test u. a.). Sie bilden quasi die „Konstruktionszeichnungen" der Protokollentwickler, anhand derer alle später auftretenden strittigen Fragen zu entscheiden sind.

Die Anforderungen an die Beschreibung von Kommunikationsprotokollen sind prinzipiell die gleichen wie bei der Softwarespezifikation. Die Beschreibung soll *exakt*, *eindeutig*, *vollständig* und *abstrakt* sein [Boch 83], [Gotz 93]. Hinter dem Begriff *abstrakt* verbirgt sich die Forderung nach einer Implementierungsunabhängigkeit der Spezifikationen. Die Beschreibung muss wesentliche Anforderungen von unwesentlichen trennen. Als wesentlich werden dabei jene Anforderungen angesehen, die das äußere, sichtbare Verhalten der Kommunikationspartner bestimmen. Unwesentlich sind interne, implementierungsabhängige Abläufe. Dienst- und Protokollspezifikationen beschreiben somit nur die funktionellen Abläufe. Realisierungsbezogene Aspekte wie die Existenz mehrerer Verbindungen oder die Adressierung werden deshalb meistens nicht berücksichtigt. Aus dem gleichen Grund werden auch keine Aussagen über eine Einbindung des Protokolls in eine konkrete Ablaufumgebung (Betriebssystem, Proto-

kollstack) gemacht. Dies bleibt einer weiteren, implementierungsbezogenen Spezifikation (siehe Kapitel 9 und 12) vorbehalten, die die Anforderungen für die Umsetzung des Protokolls auf einem konkreten Zielsystem formuliert. Eine Protokollbeschreibung erlaubt verschiedene Implementierungen, die das spezifizierte, funktionelle Verhalten gewährleisten. Diese Implementierungen werden als *konform* zu der Spezifikation bezeichnet. Die Beschreibung abstrahiert von den implementierungsspezifischen Details, in denen sich die Implementierungen unterscheiden. Sie ist daher eine Abstraktion aller zulässigen Implementierungen.

Die Beschreibung von Kommunikationsprotokollen unterscheidet sich von der Softwarespezifikation vor allem dadurch, dass zwei Spezifikationen erforderlich sind: eine Spezifikation, die beschreibt, *was* bereitgestellt wird, und eine Spezifikation, die die Art und Weise bestimmt, *wie* es bereitgestellt wird. Die „Was"-Spezifikation ist die Dienstspezifikation. Sie entspricht in ihrem Wesen der herkömmlichen Softwarespezifikation. Die „Wie"-Spezifikation ist die Protokollspezifikation. Sie stellt eine abstrakte Implementierung des Dienstes dar.

Die **Dienstspezifikation** beschreibt wie der an der Dienstschnittstelle bereitgestellte Dienst durch den Dienstnutzer in Anspruch genommen werden kann (siehe Abbildung 7.1/1). Dazu gehören die Angabe der (Teil-)Dienste mit den zugehörigen Dienstprimitiven und Parametern sowie die Darstellung der Abhängigkeiten, die zwischen den Dienstprimitiven und deren Parametern an den Dienstzugangspunkten bestehen. Die Dienstspezifikation stellt ausschließlich die Wechselwirkungen an der Dienstschnittstelle dar. Das Verhalten des Dienstnutzers wird nicht betrachtet, weshalb er in Abbildung 7.1/1 nur gestrichelt angedeutet wird.

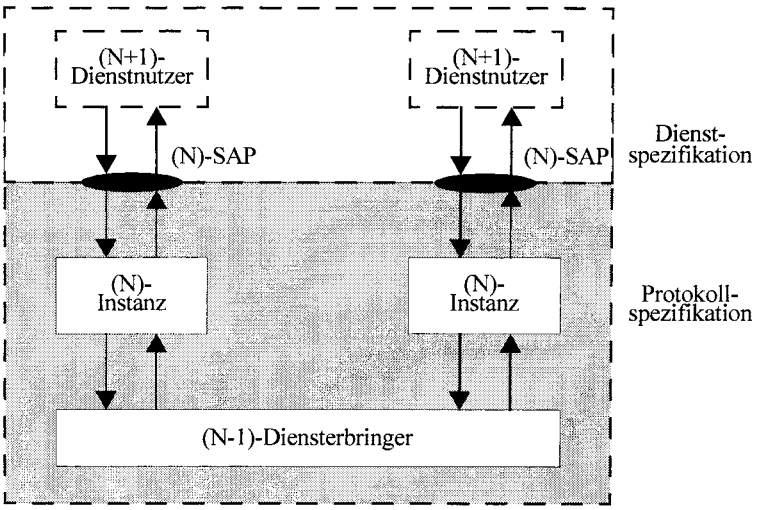


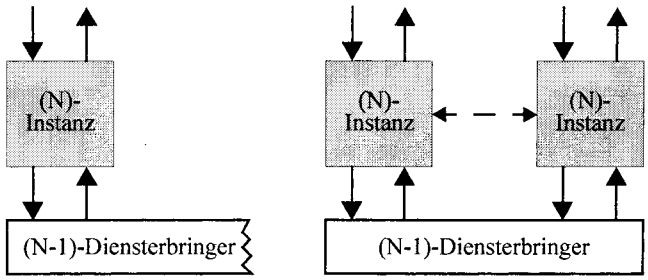
Abbildung 7.1/1: Beschreibungsebenen der Dienst- und Protokollspezifikation

Die **Protokollspezifikation** gibt vor, wie der spezifizierte Dienst durch den Diensterbringer bereitzustellen ist (siehe Abbildung 7.1/1). Sie beschreibt den Ablauf der

Kommunikation zwischen den diensterbringenden Instanzen und legt das Format der auszutauschenden Protokolldateneinheiten fest. In der Regel wird auch beschrieben wie die Kommunikation zwischen den (N)-Instanzen auf den (N-1)-Dienst abgebildet wird.

Die unterschiedliche Zielstellung beider Spezifikationen bestimmt auch ihren Nutzerkreis. Die Dienstspezifikation ist für den Dienstnutzer relevant. Sie zeigt ihm, wie er den bereitgestellten Dienst in Anspruch nehmen muss. Die Protokollspezifikation hingegen ist die Arbeitsgrundlage für Protokollentwickler, die sich mit der Validation, Bewertung und Implementierung des Protokolls beschäftigen. Sie ist für den Dienstnutzer kaum relevant.

Bei der Beschreibung von Kommunikationsprotokollen kann man zwei prinzipielle Darstellungsweisen unterscheiden, die wir hier als verhaltensorientierte und kommunikationsorientierte Darstellung bezeichnen. Die **verhaltensorientierte Darstellung** beschreibt das Protokoll durch das Verhalten der kommunizierenden Instanzen (siehe Abbildung 7.1/2a). Die Kommunikation zwischen den Instanzen wird nicht direkt dargestellt. Sie ergibt sich indirekt aus dem Verhalten der Instanzen. Die **kommunikationsorientierte Darstellung** dagegen beschreibt die Kommunikationsabläufe zwischen den Instanzen, d. h., beide Kommunikationspartner sind in die Beschreibung einbezogen (siehe Abbildung 7.1/2b).



a) verhaltensorientiert b) kommunikationsorientiert

Abbildung 7.1/2: Arten der Protokollbeschreibung

Die meisten formalen Beschreibungstechniken unterstützen die verhaltensorientierte Darstellung, weil dies durch die zugrundeliegenden semantischen Modelle (z. B. Zustandsautomaten, Prozessalgebren) unterstützt wird. Der Nachteil der verhaltensorientierten Beschreibung besteht darin, dass der Protokollablauf nur mit größerem Aufwand aus der Spezifikation abgeleitet werden kann. Die kommunikationsorientierte Darstellung, wie sie z. B. durch die Zeitablaufdiagramme (vgl. Abschnitt 1.3) realisiert wird, entspricht stärker dem Wesen der Protokolle. Sie ist jedoch schwieriger auszuführen, insbesondere wenn es um die Darstellung von Abhängigkeiten zwischen verschiedenen Protokollabläufen geht [Boch 87b]. Auch ist die Abbildung in das verwendete semantische Modell meistens komplizierter. Ein erster Ansatz, eine solche Darstellungsweise in einer Beschreibungstechnik umzusetzen, war PDL (*Protocol*

Description Language) [Köni 86,90]. Kommunikationsorientierte Darstellungen erweisen sich jedoch für das Verständnis von Protokolllabläufen als sehr nützlich und hilfreich. Deshalb verwendet man heute kommunikationsorientierte Darstellungen häufig ergän-zend zu verhaltensorientierten Beschreibungen. Ein typisches Beispiel sind die *Message Sequence Charts* (MSC) [ITU-T 120], die wir im Abschnitt 8.2 vorstellen. Unsere in Teil I verwendete Modellsprache unterstützt ebenfalls eine kommunikationsorientierte Darstellung.

7.2 Notwendigkeit formaler Beschreibungen

Dienste und Protokolle werden in internationalen Standards mehrheitlich informal, beschrieben. Die Dienst- und Protokollabläufe werden verbal als Text formuliert, der durch Zustandsdiagramme, Zustandstabellen[1], Zeitablaufdiagramme und Tabellen (z. B. für die Parameter) ergänzt wird. Die Darstellung enthält dabei zumeist keine in sich geschlossene Beschreibung der Abläufe. Sie setzt sich vielmehr aus einer Beschreibung von Teilabläufen zusammen, die das Verhalten der Protokollinstanzen beim Eintreffen bestimmter Ereignisse beschreiben. Aus diesen Teilabläufen muss sich der Protokollimplementierer das Verhalten der Protokollinstanzen puzzleartig zusammensetzen.

Informale Beschreibungen erscheinen auf den ersten Blick leicht zugänglich und verständlich. Sie haben sich aber als nicht ausreichend für eine systematische Protokollentwicklung erwiesen. Hierfür gibt es vor allem zwei Gründe:

- Informale Beschreibungen sind mehrdeutig. Das führt zu unterschiedlichen Interpretationen der Beschreibungen durch die Protokollentwickler, was letztlich zu inkorrekten und inkompatiblen Implementierungen führen kann.
- Informale Beschreibungen bilden eine unzureichende Grundlage für die Entwicklung von Werkzeugen. Eine rechnerunterstützte Validierung einzelner Entwicklungsschritte ist auf dieser Basis nicht möglich.

Die Notwendigkeit formaler Beschreibungen für die Protokollentwicklung ist daher allgemein anerkannt und unbestritten. Unter einer **formalen Beschreibung** verstehen wir Beschreibungen unter Verwendung einer Beschreibungsmethode mit einer formalen Semantik, die eine eindeutige Interpretation der Beschreibung sichert (siehe Abbildung 7.2/1).

Die Anforderungen, die an formale Beschreibungen gestellt werden, entsprechen den oben erläuterten allgemeinen Anforderungen an Protokollbeschreibungen wie Exaktheit, Eindeutigkeit, Vollständigkeit, Abstraktion und Implementierungsunabhängigkeit. Weitere wünschenswerte Eigenschaften sind ein *modularer Aufbau*, um ihre Handhabung und Veränderbarkeit zu erleichtern, und *Verständlichkeit*, um das intuitive Verstehen zu unterstützen und Missinterpretationen vorzubeugen.

---

1. Zustandstabellen beschreiben die Zustände der Protokollinstanz im Sinne der Interpretation als erweiterter endlicher Zustandsautomat (siehe Abschnitt 7.3)

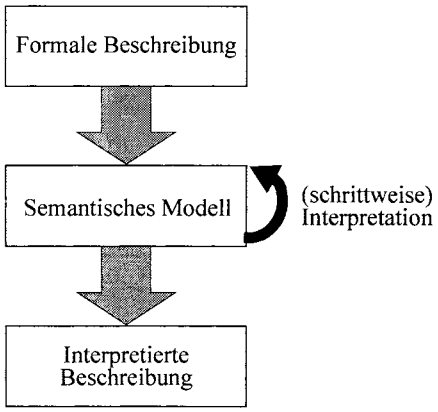


Abbildung 7.2/1: Interpretation formaler Beschreibungen

7.3 Herangehensweisen bei der formalen Beschreibung

Die Beschreibung von Kommunikationsdiensten und -protokollen setzt sich aus zwei Bestandteilen zusammen: der **Beschreibung der Kommunikationsabläufe** und der **Beschreibung der Datenformate** für die Dienstprimitive und die Protokolldateneinheiten. Dafür werden komplementäre Techniken verwendet. Die formalen Beschreibungsmethoden beziehen sich in erster Linie auf die funktionale Darstellung der Kommunikationsabläufe. Für die Beschreibung der Datenformate werden bekannte Techniken der Datenbeschreibung verwendet. Einige davon stellen wir im Kapitel 8 vor. Wir konzentrieren uns hier auf die Beschreibung der Kommunikationsabläufe.

Beschreibungsmethoden für Kommunikationsprotokolle werden in konstruktive und deskriptive Methoden unterschieden [Drob 86], [Gotz 92] (siehe Abbildung 7.3/1). **Konstruktive Methoden** beschreiben das Protokoll durch ein abstraktes Modell, dessen Ausführung bestimmt, wie sich die kommunizierenden Instanzen verhalten. Die Beschreibung stellt eine Quasi-Implementierung dar, d. h., das Protokoll wird durch ein konformes, abstrakteres Protokoll beschrieben. Die Spezifikationsaussagen ergeben sich durch die Ausführung des abstrakten Protokolls gemäß dem zugrundeliegenden semantischen Modell. Beispiele für konstruktive Beschreibungsmethoden sind endliche Zustandsautomaten und Labelled Transition Systems. Der Vorteil der konstruktiven Methoden besteht in der unmittelbaren Unterstützung des Entwurfs, seiner Validation sowie der späteren Implementierung des Protokolls. Komponenten des Beschreibungsmodells können als Bausteine innerhalb des Entwurfs wiederverwendet werden. Von der Spezifikation können ausführbare Prototypen abgeleitet werden, die zur Validation des Entwurfs genutzt werden können. Von Nachteil ist, dass spezifische Anforderungen an das Protokoll, wie Lebendigkeits- und Sicherheitseigenschaften (siehe Abschnitt 7.3.5), nicht explizit spezifiziert werden können. Sie müssen implizit im Rahmen des gewählten Beschreibungsmodells verifiziert werden.

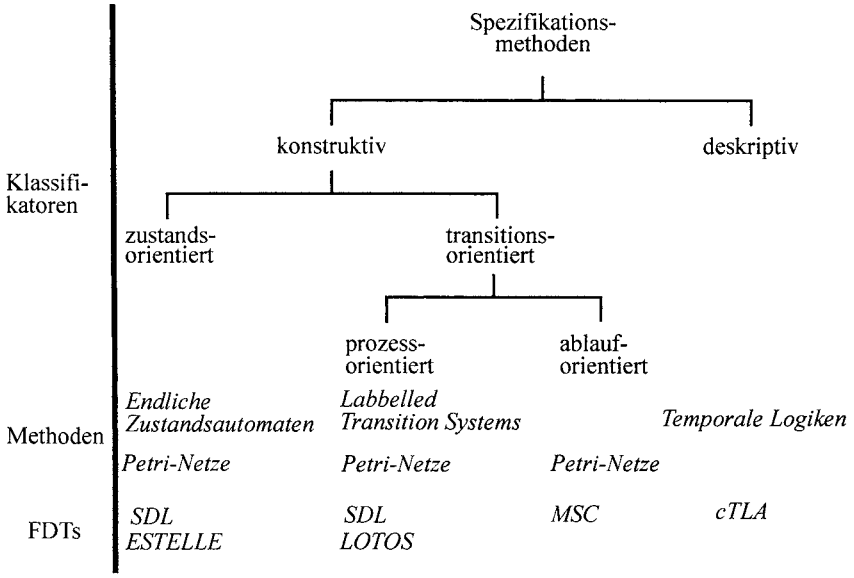


Abbildung 7.3/1: Klassifikation der Beschreibungsmethoden für Kommunikationsprotokolle

Die **deskriptiven Methoden** unterlegen der Spezifikation keine spezifische Interpretationsvorschrift, sondern formulieren die Eigenschaften, die das Protokoll erfüllen soll, durch logische Aussagen. Damit werden keine bestimmten Modelle spezifiziert sondern Klassen von Modellen, die diese Eigenschaften erfüllen. Vertreter deskriptiver Methoden sind Aussagenlogiken, Prädikatenlogiken, Modale Logiken und Temporale Logiken. Letztere stellen die wichtigste deskriptive Beschreibungsmethode des Protocol Engineering dar. Die Vorteile deskriptiver Methoden liegen darin, dass sie es gestatten, die Protokolleigenschaften unter völliger Abstraktion von einer Implementierung darzustellen und separat auf ihre Zweckmäßigkeit zu überprüfen. Sie unterstützen dabei vorteilhaft den Verifikationsprozess. Es ist allerdings i. Allg. nicht entscheidbar, ob eine Spezifikation das gewünschte Verhalten vollständig beschreibt. Deskriptive Methoden sind vor allem im frühen Entwurfsstadium sinnvoll, um Anforderungen an das Protokoll zu formulieren, ohne bereits das Protokollverhalten durch eine Quasi-Implementierung zu modellieren und somit Entwurfsentscheidungen zu treffen [Herr 98]. Ihr durchgehender Einsatz ist nur dann effizient, wenn der Übergang von der problemnahen zur realisierungsnahen Phase durch eine einheitliche Technik und entsprechende Entwicklungswerkzeuge unterstützt wird, was bislang nicht ausreichend gegeben ist. Deshalb und aufgrund des hohen Einarbeitungsaufwands werden deskriptive Techniken in der Praxis des Protocol Engineering weniger eingesetzt.

Die konstruktiven Methoden werden weiterhin in zustandsorientierte und transitionsorientierte Beschreibungsmethoden unterschieden. *Zustandsorientierte Beschrei-*

bungsmethoden spezifizieren das Protokoll durch die Beschreibung der Zustände, die die Instanz bei der Protokollabarbeitung annehmen kann, und der Übergänge zwischen den Zuständen, den Transitionen. Zustandsorientierte Beschreibungsmethoden unterstützen eine verhaltensorientierte Beschreibung (vgl. Abschnitt 7.1). Sie sind einfacher in der Erstellung und werden auch deshalb bevorzugt angewendet. Die *transitionsorientierten Beschreibungsmethoden* betonen stärker die Darstellung der Abfolge der Zustandsübergänge. Die Zustände werden dabei nicht explizit dargestellt, sondern ergeben sich implizit aus den ausgeführten Transitionen. Die transitionsorientierten Methoden werden nochmals in prozess- und ablauforientierte Methoden unterschieden. Die *prozessorientierten Methoden* beschreiben die prozeduralen Abläufe in einer Instanz, während die ablauforientierten Methoden die Interaktion zwischen beiden (oder sogar mehreren) Instanzen darstellen. Das entspricht der kommunikationsorientierten Beschreibung (vgl. Abschnitt 7.1). Die transitionsorientierte Beschreibung ist anspruchsvoller in der Erstellung, da es i. Allg. wesentlich komplizierter ist, aus den informalen Protokollbeschreibungen die Abfolge der Zustandsübergänge abzuleiten [Boch 87b], [Köni 90]. Die Schwierigkeit besteht dabei insbesondere im Erkennen bestehender Nebenläufigkeiten, weniger im Erkennen der Zustände selber, die durch die Zustandstabellen meistens gegeben sind. Abbildung 7.3/1 enthält eine Zuordnung verschiedener Beschreibungsmethoden und -techniken zu den einzelnen Klassen. Einige Methoden können dabei unterschiedlich genutzt werden.

7.4 Methoden der formalen Beschreibung

Die Suche nach geeigneten Beschreibungsmethoden für Kommunikationsprotokolle hat in den 80er Jahren zu einem breiten Spektrum von Ansätzen geführt (siehe [PSTV]). Daraus haben sich die endlichen Zustandsautomaten, Petri-Netze, Prozessalgebren und temporale Logiken als die wichtigsten und gebräuchlichsten Beschreibungsmethoden herauskristallisiert. Sie werden nachfolgend vorgestellt. Neben diesen Methoden wurden auch andere Ansätze verfolgt wie die Verwendung von Grammatiken, Datenfluss-Sprachen und funktionalen Sprachen. Ebenfalls genutzt wurden höhere Programmiersprachen, die die algorithmische Umsetzung der Protokollabläufe sowie die Erzeugung von Prototypen unterstützen. Höhere Programmiersprachen besitzen in der Regel keine formal definierte Semantik. Außerdem sind Programmiersprachen in erster Linie Implementierungssprachen und keine Spezifikationssprachen, was zu implementierungsnahen Beschreibungen führt. Programmiersprachen werden deshalb nicht mehr zur Beschreibung von Protokollen eingesetzt. Viele Beschreibungstechniken nutzen jedoch programmiersprachliche Konzepte.

7.4.1 Endliche Zustandsautomaten

Endliche Zustandsautomaten (*finite state machines*, FSMs) beschreiben das Protokoll über das Verhalten der Protokollinstanzen. Sie sind ein einfaches und natürliches Modell für die Beschreibung von Protokollinstanzen, da sie das Warten der Instanzen auf

bestimmte Ereignisse (Eingaben), ihre Reaktion darauf (Ausgaben) und den Übergang in einen Folgezustand (Transition) unmittelbar darzustellen gestatten.

Ein **endlicher Zustandsautomat** ist ein Quintupel $<S,I,O,T,s_0>$ mit

- S - endliche, nicht leere Menge von *Zuständen*,
- I - endliche, nicht leere Menge von *Eingaben*,
- O - endliche, nichtleere Menge von *Ausgaben*,
- $T \subseteq S \times (I \cup \{\tau\}) \times O \times S$ - eine *Zustandsüberführungsfunktion* und
- $s_0 \in S$ - *Initialzustand* des Automaten.

Eine Transition $t \in T$ ist definiert durch das Quadrupel $<s,i,o,s'>$, wobei $s \in S$ den aktuellen Zustand, $i \in I$ eine Eingabe, $o \in O$ die zugehörige Ausgabe und $s' \in S$ den Folgezustand bezeichnen. $\tau \notin I$ bezeichnet eine leere Eingabe. Damit werden spontane Transitionen modelliert, um interne Ereignisse zu beschreiben (vgl. Abschnitt 1.2).

Endliche Automaten werden u. a. durch Zustandstabellen, Zustandsübergangsdiagramme oder Zustandsübergangsmatrizen dargestellt. Wir nutzen hier Zustandsübergangsdiagramme (siehe Abbildung 7.4/1). Sie repräsentieren die Zustände durch markierte Kreise und die Zustandsübergänge bzw. Transitionen durch gerichtete Kanten. An den Kanten werden in der Form *i/o* die zugeordneten Eingabe-/Ausgabeereignisse $i \in I$ und $o \in O$ angegeben. Das Zustandsübergangsdiagramm in Abbildung 7.4/1 zeigt den endlichen Zustandsautomaten für die Empfänger-Instanz des XDT-Protokolls. Der dargestellte Automat weicht etwas von der oben gegebenen Definition ab. Er lässt mehrere Ausgaben auf eine Eingabe zu. Es handelt sich hierbei um eine häufig verwendete Vereinfachung, um die Darstellung der Automaten nicht zu komplex werden zu lassen. Des Weiteren ist eine leere Ausgabe λ zugelassen, d. h., der Automat gibt auf eine Eingabe eine definierte endliche Zeit keine Ausgabe. Die Ausgabemenge O' des so dargestellten Zustandsautomaten ist durch $O' = \wp(O) \cup \{\lambda\}$ definiert, wobei $\wp(O)$ die Potenzmenge von O bezeichnet. Die Zustandsüberführungsfunktion T' modifiziert sich entsprechend zu $T' \subseteq S \times (I \cup \{\tau\}) \times O' \times S$.

Für die Beschreibung eines Protokolls sind meistens mehrere Automaten erforderlich, um das Verhalten der Senderinstanz, der Empfängerinstanz und des (N-1)-Diensterbringers darzustellen. Symmetrische Protokolle benötigen nur einen Automaten für die Beschreibung der Sender- und Empfängerinstanz.

Bei komplexeren Protokollen werden die Protokollautomaten häufig weiter in Teilautomaten zerlegt. Für die Kommunikation zwischen den Automaten gibt es zwei Möglichkeiten: die asynchrone und die synchrone Kopplung [Boch 78], [Holz 91], die verschiedenes Verhalten modellieren. Bei der *asynchronen Kopplung* werden die Interaktionen (Ereignisse, Nachrichten) über endliche oder unendliche FIFO-Warteschlangen ausgetauscht. Der empfangende Automat entnimmt das jeweils erste Ereignis und führt die dem Ereignis zugeordnete Transition aus. Die Ausführung der Transitionen in den kommunizierenden Automaten ist nicht miteinander gekoppelt. Bei der *synchronen Kopplung* ist die Ausführung bestimmter Transitionen der kommunizierenden Automaten an ein synchronisierendes Ereignis gebunden. Das Ereignis ist Ausgabeereignis in dem einen Automaten und Eingabeereignis in dem anderen. Die Ausführung der Transitionen erfolgt simultan. Alle nichtgekoppelten Transitio-

nen werden nebenläufig ausgeführt. Die asynchrone Kopplung erlaubt eine natürliche Modellierung der Protokollabläufe. Sie entspricht dem üblichen Paradigma der Protokoll-implementierung (siehe Abschnitt 12.3). Die synchrone Kopplung ist abstrakter. Sie unterstützt eine überschaubarere Modellierung der Interaktionen, was die Verifikation begünstigt.

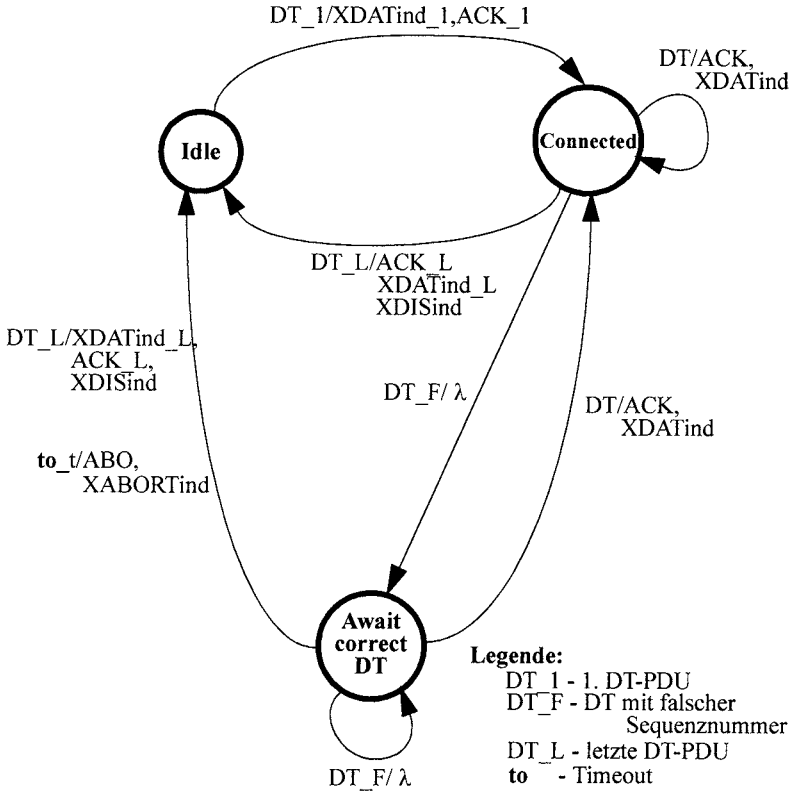


Abbildung 7.4/1: Vereinfachte FSM-Darstellung der XDT-Empfänger-Instanz

Endliche Zustandsautomaten beschreiben nur die funktionalen Protokollabläufe, d. h. den Kontrollfluss. Veränderungen in den Datenstrukturen, den Datenfluss, können sie nicht darstellen. Soll z. B. die Verwendung von Sequenznummern (modulo eines Maximalwerts) für unser Beispielprotokoll modelliert werden, so müssen dafür unterschiedliche Zustände und Ereignisse eingeführt werden, um jede Änderung der Sequenznummer zu beschreiben. Das lässt den Automaten schnell sehr komplex werden, weshalb wir in Abbildung 7.4/1 vereinfachend auf die Darstellung der Sequenznummer verzichtet haben. Die Darstellung der XDT-Sender-Instanz durch einen endlichen Zustandsautomaten ist aus diesem Grund wenig praktikabel.

Endliche Zustandsautomaten werden deshalb für die Protokollbeschreibung nur beschränkt genutzt. Die Beschreibungen werden schnell zu komplex. Sie sind schwer zu

überschauben und zu nutzen. FSM-Darstellungen dienen meistens nur der Veranschaulichung wesentlicher Protokollabläufe. Endliche Zustandsautomaten werden weiterhin für die Ableitung von Testfällen genutzt. Die wichtigsten Ableitungsverfahren gehen von FSM-Darstellungen der Protokollinstanzen aus (siehe Abschnitt 13.3).

7.4.2 Erweiterte endliche Zustandsautomaten

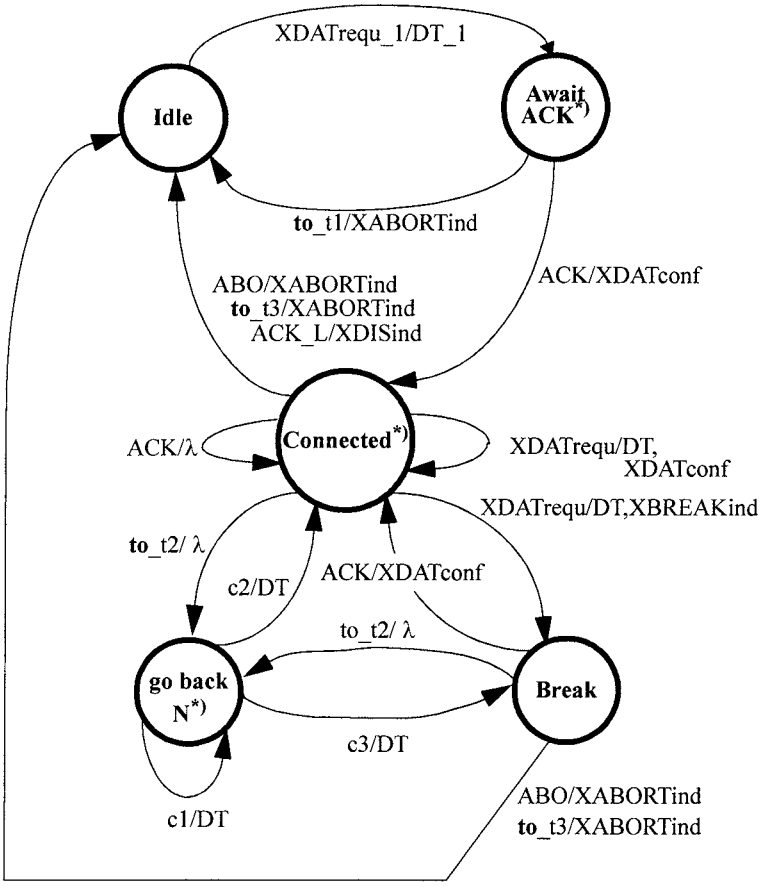
Die Probleme endlicher Zustandsautomaten in der Protokollbeschreibung können zum Teil durch die Einführung von Variablen umgangen werden. Variable können genutzt werden, um Kontextinformationen, z. B. Sequenznummern, abzuspeichern. Derart modifizierte Zustandsautomaten werden als erweiterte endliche Zustandsautomaten (*extended finite state machines*, EFSMs) bezeichnet.

Ein **erweiterter endlicher Zustandsautomat** ist ein Tupel $<S,C,I,O,T,s_0,c_0>$ mit

- S - endliche, nicht leere Menge von *Zuständen*,
- $C = domain(v_1) \times \ldots \times domain(v_n)$ - nicht leere Menge von *Kontexten* mit $v_i \in V$, wobei V eine endliche, nicht leere Menge von Variablen und $domain(v_i)$ eine nicht leere, abzählbare Menge von Werten, der Wertebereich von v_i, ist,
- I - endliche, nicht leere Menge von *Eingaben*,
- O - endliche, nicht leere Menge von *Ausgaben*,
- $T \subseteq S \times C \times (I \cup \{\tau\}) \times O \times S \times C$ - eine *Zustandsüberführungsfunktion*,
- $s_0 \in S$ - Initialzustand und
- $c_0 \in C$ - Anfangskontext des Automaten.

Ein Kontext C ist durch eine Wertebelegung der Variablen gegeben. Eine Transition $t \in T$ ist durch das Tupel $<s,c,i,o,s',c'>$ definiert, wobei $s \in S$ den aktuellen Zustand, $c \in C$ den Kontext vor Ausführung der Transition, $i \in I$ eine Eingabe, $o \in O$ die zugehörige Ausgabe, $s' \in S$ den Folgezustand und $c' \in C$ den Kontext nach Ausführung der Transition bezeichnen. Auch hier wird die leere Eingabe τ verwendet, um spontane Transitionen zu modellieren. Die Zustände $s \in S$ werden auch **Hauptzustände** genannt. Entsprechend werden die durch $s \in S$ und $c \in C$ bestimmten Zustände als **Nebenzustände** bezeichnet.

Mit Hilfe eines erweiterten endlichen Zustandsautomaten können wir nun auch die Sendeinstanz unseres XDT-Protokolls beschreiben, da die notwendigen Kontextinformationen über ausstehende Bestätigungen oder zu wiederholende Übertragungen der *DT*-PDUs jetzt in Variablen gespeichert werden können. Abbildung 7.4/2 zeigt den Automaten. Für die Darstellung des Automaten gelten ebenfalls die bereits im vorigen Abschnitt begründeten Vereinfachungen. Es sind wiederum mehrere Ausgaben einschließlich der leeren Ausgabe zugelassen. Die Ausgabemenge ist definiert durch $O' = \wp(O) \cup \{\lambda\}$ mit $\wp(O)$ als Potenzmenge von O und die Zustandsübergangsfunktion entsprechend als $T \subseteq S \times C \times (I \cup \{\tau\}) \times O' \times S \times C$. Für die Zustände *Await ACK*, *Connected* und *go back N* sind die Variablen angegeben, die in diesen Zuständen relevant sind. Es sind dies die Variablen, deren Werte sich bei der Ausführung einer Transition ändern können. Beispielsweise wird im Zustand *Connected* bei Eintreffen einer korrekten Bestätigung *ACK* der Wert der Variablen *N* erhöht.



Legende:

to	-	Timeout
DT_1	-	1. DT_PDU
ACK_L	-	letzte ACK
c1	-	interne Bedingung für das Senden einer DT-PDU während *go back N*
c2, c3	-	interne Bedingungen "Ende *go back N*"

***)Relevante Variablen der Kontexte**
Await ACK: sequ
Connected: sequ, last, N, eom
go back N: i, N, sequ

Abbildung 7.4/2: Vereinfachte EFSM-Darstellung der XDT-Sender-Instanz

Die erweiterten endlichen Zustandsautomaten sind die am häufigsten verwendete Beschreibungsmethode für Kommunikationsprotokolle. Die semantischen Modelle der formalen Beschreibungstechniken SDL und Estelle wie auch unserer Modellsprache basieren darauf. Zu den Vorteilen der erweiterten endlichen Zustandsautomaten gehören neben der natürlichen Modellierung des Verhaltens der Protokollinstanzen auch die Unterstützung einer relativ geradlinigen Überführung in eine Implementierung

(siehe Abschnitt 12.2). Allerdings verführt dies häufig zu sehr implementierungsnahen Spezifikationen. Die Darstellung komplexer Protokolle bleibt trotzdem problematisch. Die formalen Beschreibungstechniken bieten hierfür verschiedene Techniken der Zerlegung in Teilautomaten an.