HERAKLIT-Fallstudie: Handelsbetrieb

Peter Fettke[1,2][0000−0002−0624−4431] und Wolfgang Reisig[3][0000−0002−7026−2810]

[1] German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
peter.fettke@dfki.de
[2] Saarland University, Saarbrücken, Germany
[3] Humboldt-Universität zu Berlin, Berlin, Germany
reisig@informatik.hu-berlin.de

Abstract. The modeling method HERAKLIT is presented on the basis of a business case study. The case study demonstrates how HERAKLIT can be used to systematically model, abstract, compose and analyze computer-integrated systems of business practice. Structural aspects, abstract descriptions of data and objects as well as behavioral models are integrated in a novel way.

Keywords: systems composition · data modelling · behaviour modelling · process modelling · composition calculus · algebraic specification · Petri nets · Systems Mining

Zusammenfassung. Anhand einer betrieblichen Fallstudie wird die Modellierungsmethode HERAKLIT vorgestellt. Die Fallstudie demonstriert, wie mit HERAKLIT rechner-integrierte Systeme der betrieblichen Praxis systematisch modelliert, abstrahiert, komponiert und analysiert werden können. Strukturelle Aspekte, abstrakte Beschreibungen von Daten und Gegenständen sowie Verhaltensmodelle werden auf neuartige Weise integriert.

Schlüsselwörter: Systemkomposition · Datenmodellierung · Verhaltensmodellierung · Prozessmodellierung · Composition Calculus · Algebraische Spezifikation · Petrinetz · Systems Mining

Vorbemerkung

Was ist HERAKLIT?

HERAKLIT ist eine Methode zur Modellierung rechnerintegrierter betrieblicher Systeme. Die Zielsetzung von HERAKLIT skizziert das HERAKLIT-*Memorandum* [4]. Den systematischen Aufbau von HERAKLIT, die Auswahl der einzelnen Konzepte und ihres Zusammenspiels motiviert das HERAKLIT-*Handbuch*,

zu zitieren als: FETTKE, P.; REISIG, W.: HERAKLIT-*Fallstudie: Handelsbetrieb*. 2020. – HERAKLIT–Arbeitspapier, v1, 8. November 2020, http://www.heraklit.org

das noch nicht vorliegt. Das HERAKLIT-*Tool* unterstützt den Modellierer beim Erstellen und Analysieren anschaulicher Modelle, insbesondere umfangreicher Modelle, die neben detaillierten Darstellungen auch Übersichten und Abstraktionen verwenden. Dieses Werkzeug wird erst später verfügbar sein.

Was bietet dieser Text?

Für Leser ohne Vorkenntnisse wird an einer umfangreichen Fallstudie erläutert, wie HERAKLIT-Modelle konstruiert werden. Wir modellieren die zentralen Geschäftsprozesse eines Handelsbetriebes und seiner Geschäftspartner, wir zeigen, wie sie komponiert und abstrahiert werden können, und wir beschreiben ihr Verhalten. Aus den Konstruktionen dieser Fallstudie kann der Leser intuitiv leicht die generellen Konzepte von HERAKLIT verstehen.

Wie ist der Text aufgebaut?

Die Fallstudie gliedert sich in acht Teile: Teil I erläutert zur Einführung den Inhalt der Fallstudie und des entwickelten Modells. Ein HERAKLIT-Modell ist grundsätzlich in einzelne Module gegliedert. Die Module der Fallstudie präsentiert Teil II. Teil III erläutert, wie Daten und Gegenstände in der Fallstudie modelliert werden. Die Beschreibung verteilter Abläufe erfolgt in Teil IV. Die Übertragung der Konzepte auf Schemata, um über beliebige Daten, Gegenstände und Abläufe zu reden, erfolgt in Teil V. Teil VI detailliert dann die einzelnen Module hinsichtlich ihrer Struktur und ihres Verhaltens. Das vollständige Modell im Überblick zeigt Teil VII. Teil VIII führt aus, wie weitere Sichten auf das Gesamtmodell beschrieben werden können.

I Inhalt der Fallstudie und des hier entwickelten Modells

1. Der Handelsbetrieb der Fallstudie und sein hier entwickeltes Modell

Ein Handelsbetrieb verkauft Artikel an seine Kunden über einen Webshop. Über andere Vertriebsformen wie Einzel- oder Großhandel oder Vertreter werden keine Artikel abgesetzt.

Der Handelsbetrieb hat drei Arten Geschäftspartner:

- seine *Kunden*: sie schicken ihm Bestellungen; der Handelsbetrieb informiert seine Kunden über Liefertermine etc.;
- den *Lieferanten*: bei ihm bestellt der Handelsbetrieb fehlende Ware nach; der Lieferant liefert bestellte Ware an den Handelsbetrieb;
- die *Speditionen*: sie erhalten vom Handelsbetrieb fertig gepackte Fracht und liefern sie an die Kunden aus.

Der Handelsbetrieb selbst ist dreigeteilt:

– Die *Auftragsverwaltung* nimmt Bestellungen der Kunden entgegen, lässt sich von der Bestandsverwaltung die Verfügbarkeit jedes bestellten Artikels bestätigen und erteilt dem Lager Lieferaufträge. Gegebenenfalls veranlasst die Auftragsverwaltung Teillieferungen.
– Die *Bestandsverwaltung* führt eine Liste über die Warenbestände im Lager. Wenn die Zahl verfügbarer Exemplare eines Artikels zu klein wird, bestellt die Bestandsverwaltung beim Lieferanten nach und lässt sich vom Lager den Empfang dieser Artikel quittieren.
– Das *Lager* verpackt gemäß der Lieferaufträge der Auftragsverwaltung bestellte Ware und übergibt das Frachtgut den Speditionen. Zudem empfängt es vom Lieferanten Ware, sortiert sie ins Lager ein und meldet der Bestandsverwaltung den Empfang der Ware.

Der Handelsbetrieb und seine Partner bilden zusammen ein System, das als Ganzes korrekt funktionieren muss. Beispielsweise sollen einem Kunden alle bestellten Artikel auch wirklich geliefert werden, gegebenenfalls in Teillieferungen. Umgekehrt sollen ihm nur Artikel geliefert werden, die er vorher bestellt hat. Innerhalb des Handelsbetriebs soll die Bestandsverwaltung nur dann das Lager mit einer Lieferung beauftragen, wenn die dafür nötigen Artikel im Lager verfügbar sind.

2. Das HERAKLIT-Modell für den Handelsbetrieb und seine Geschäftspartner

Wir konstruieren ein HERAKLIT-Modell für das gesamte System, also den geschilderten Handelsbetrieb und seine drei Geschäftspartner. Zunächst modellieren wir die grobe Struktur des gesamten Systems und der drei Abteilungen des Handelsbetriebes als HERAKLIT-*Module* im Kontext ihres prinzipiellen Zusammenwirkens. Es folgt eine detaillierte und systematische Darstellung der Daten und Gegenstände in einer abstrakten Fassung. Dabei bleibt die Darstellung auf schematischer Ebene: Wie genau Kunden, Artikel, Waren, Frachtgut et cetera aussehen, bleibt konkreten *Instanziierungen* überlassen. Schließlich wird das Verhalten der einzelnen Module in Form lokaler Schritte und mathematischer Operationen auf den beteiligten Daten und Gegenständen dargestellt.

Das Modell lässt eine Reihe von Entscheidungen offen. Insbesondere

– kann die Auftragsverwaltung im Lager verfügbare Artikel nach freier Wahl zu Teillieferungen bündeln;
– können im Lager verschiedene Waren für denselben bestellten Artikel gewählt werden;
– wird nicht festgelegt, welche der verfügbaren Speditionen eine Lieferung ausliefert.

Damit fasst das Modell genau die in Abschnitt 1 beschriebenen Aspekte des Handelsbetriebes.

Das Modell kombiniert Prinzipien, die aus *Algebraischen Spezifikationen*, *Petrinetzen* und dem *Composition Calculus* bekannt und bewährt sind. Sie leuchten

intuitiv ein, so dass für das Verständnis des graphischen Modells keine Vorkenntnisse erforderlich sind. Zugleich ist das Modell formal; so können die oben genannten Eigenschaften im Modell formal formuliert und bewiesen werden. Einzelheiten dazu folgen im HERAKLIT-Handbuch.

II Das Modulkonzept

3. HERAKLIT-Module

Ein reales System besteht im Allgemeinen aus Teilsystemen, die untereinander zusammenhängen. Auf dieser offensichtlichen Beobachtung basiert das zentrale Modellierungs-Konzept der HERAKLIT-*Module*: Ein HERAKLIT-Modul ist ein Modell, graphisch dargestellt als ein Rechteck, mit zwei wesentlichen Aspekten:

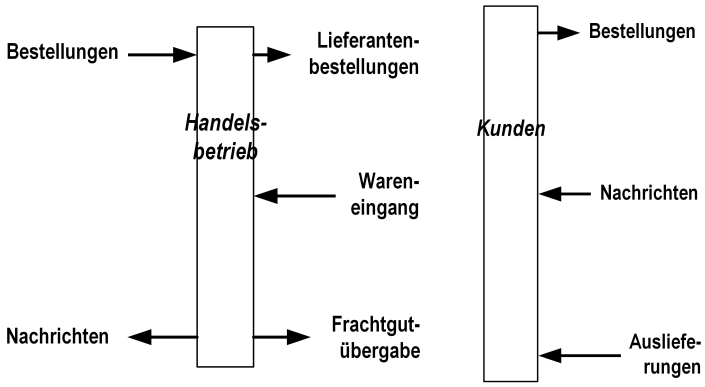
- seinem *Inneren*: das kann ganz beliebig ausgestaltet sein; es besteht gegebenenfalls nur aus dem Namen des Moduls, es zeigt seine Struktur oder sein dynamisches Verhalten;
- seiner *Oberfläche*: sie besteht aus einer Menge von *Gates*. Jedes Gate hat ein *Label*, also eine Beschriftung. Jedes Gate ist graphisch als kurze Linie repräsentiert, die am Rechteck der Komponente beginnt und am Label des Gates endet.

Wie das im Einzelnen geht, zeigt das Modul in Abbildung 1a am Beispiel des Handelsbetriebes: Das Innere des Handelsbetriebes wird im Inneren des Rechtecks dargestellt; wir abstrahieren hier von inneren Einzelheiten vollständig und schreiben nur den Namen des Moduls hin. Die Oberfläche besteht aus fünf Gates mit den Labeln *Bestellungen* (von Kunden), *Nachrichten* (an Kunden), *Lieferantenbestellungen* (an den Lieferanten), *Wareneingang* (vom Lieferanten) und *Frachtgutübergabe* (an Speditionen).

Analog dazu zeigen die Module der Abbildungen 1b, 1c und 1d die drei Arten Geschäftspartner: *Kunden*, *Lieferant* und *Speditionen* mit ihren jeweiligen Gates. Abbildung 2 zeigt die drei Abteilungen des Handelsbetriebs, wiederum als HERAKLIT-Module.

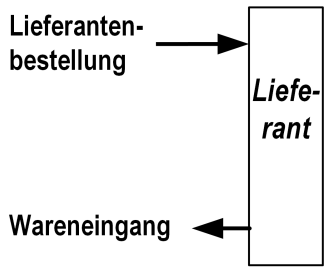
Alle HERAKLIT-Module in den Abbildung 1 und Abbildung 2 sind *abstrakte* Module: Ihr Inneres enthält nur den Namen des Moduls. Typischerweise zeigt das Innere eines HERAKLIT-Moduls jedoch Einzelheiten seiner Struktur oder seines Verhaltens. Die Struktur eines Moduls ist oft eine *Komposition* von Teilmodulen.

Jedes Gate der Module in den Abbildungen 1 und 2 ist – intuitiv formuliert – ein *Eingang* oder ein *Ausgang*, durch den Gegenstände, Dokumente, Informationen et cetera fließen. Die Richtung dieses Flusses hilft beim Verständnis des Verhaltens des Moduls; wir deuten sie mit einer Pfeilspitze an. Solche Pfeilspitzen – und andere Beschriftungen – gehören nicht zum formalen Rahmen von HERAKLIT.

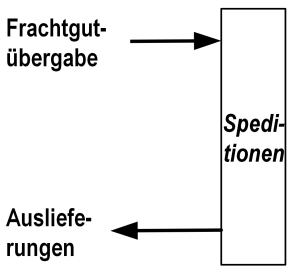


(a) der Handelsbetrieb

(b) die Kunden



(c) der Lieferant



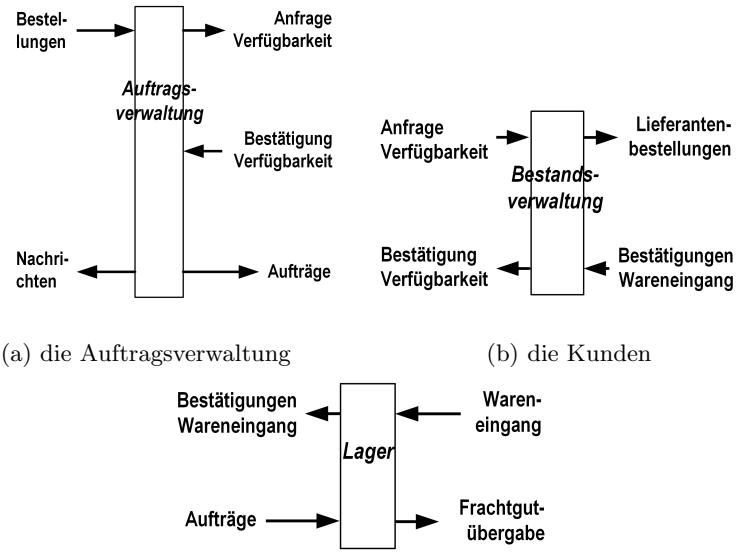
(d) die Speditionen

Abb. 1: abstrakte HERAKLIT-Module: Der Handelsbetrieb und seine drei Sorten Partner

4. Die Komposition von HERAKLIT-Modulen

Teilsysteme eines großen realen Systems werden vernünftigerweise so gebildet, dass sie nach einem festen Verfahren komponiert werden können und dass ihre Komposition wieder das originale System ergibt. Dieses Konzept verfolgt HERAKLIT auch bei der Komposition von HERAKLIT-Modulen. Technisch ist die Komposition $L \bullet M$ zweier HERAKLIT-Module L und M definiert als Verschmelzung gleich gelabelter Gates von L und M. Das verschmolzene Element verbleibt dann im Inneren von $L \bullet M$. Die anderen Gates von L und von M werden zu Gates von $L \bullet M$.

Abbildung 3a zeigt ein Beispiel: Die *Auftragsverwaltung* und die *Bestandsverwaltung* aus den Abbildung 2a und 2b haben beide Gates mit dem Label *Anfrage Verfügbarkeit*. In Abbildung 3a verschmelzen die beiden Gates zu einem Element im Inneren von *Auftragsverwaltung* • *Bestandsverwaltung*. Entsprechendes gilt für die beiden Gates mit dem Label *Bestätigung Verfügbarkeit*. Die anderen drei Gates der *Auftragsverwaltung* und die anderen beiden Gates der *Bestandsverwal-*



(a) die Auftragsverwaltung

(b) die Kunden

(c) der Lieferant

Abb. 2: Abstrakte Fassungen der drei Abteilungen des Handelsbetriebes

tung bilden dann die Oberfläche des komponierten Systems *Auftragsverwaltung* • *Bestandsverwaltung*.

In Abbildung 3b entsteht entsprechend das System

$$\mathit{Bestandsverwaltung} \bullet \mathit{Lager}.$$

Beide komponierten Module kann man nun durch die jeweils fehlende dritte Abteilung des Handelsbetriebes ergänzen und die beiden Module

$$(\mathit{Auftragsverwaltung} \bullet \mathit{Bestandsverwaltung}) \bullet \mathit{Lager}$$

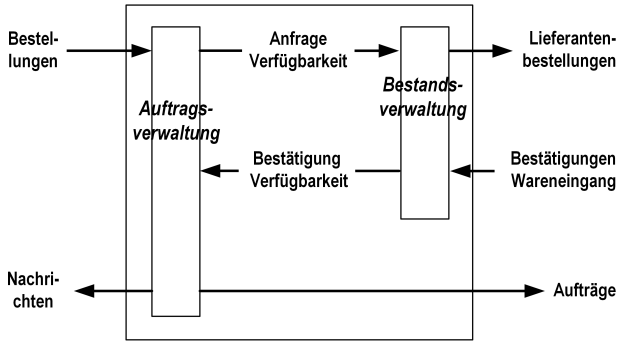
sowie

$$\mathit{Auftragsverwaltung} \bullet (\mathit{Bestandsverwaltung} \bullet \mathit{Lager})$$

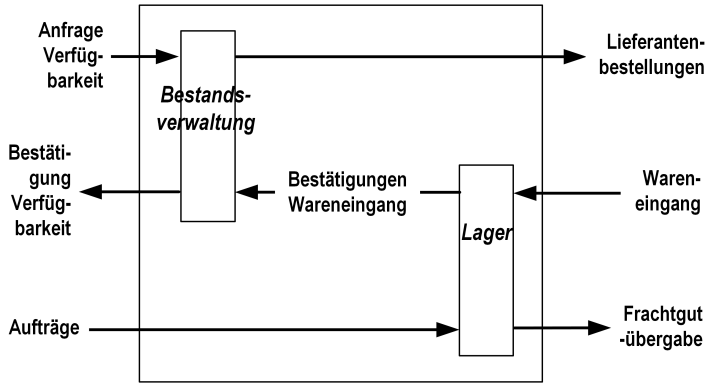
bilden. Diese beiden Module sind identisch; deshalb kann man in Abbildung 3c die Klammern weglassen.

Einige Einzelheiten der Definition der Komposition beschreibt der Anhang und vor allem das HERAKLIT-Handbuch. Ganz zentral sind zwei Eigenschaften des HERAKLIT-Kompositionsoperators: er ist

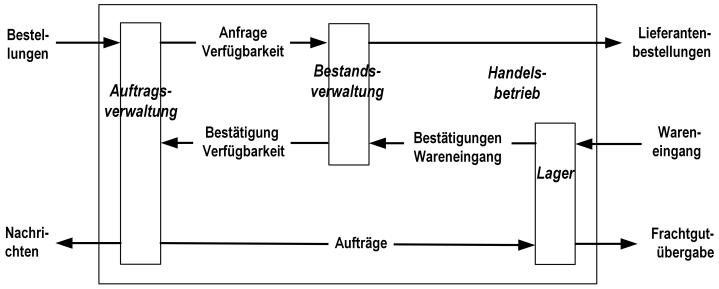
- *total*, er bildet also aus beliebigen HERAKLIT-Modulen L und M das HERAKLIT-Modul $L \bullet M$, und er ist
- *assoziativ*, es gilt also für beliebige Module L, M und N: $(L \bullet M) \bullet N = (L \bullet M) \bullet N$; deshalb muss man nie klammern.



(a) die Komposition $Auftragsverwaltung \bullet Bestandsverwaltung$



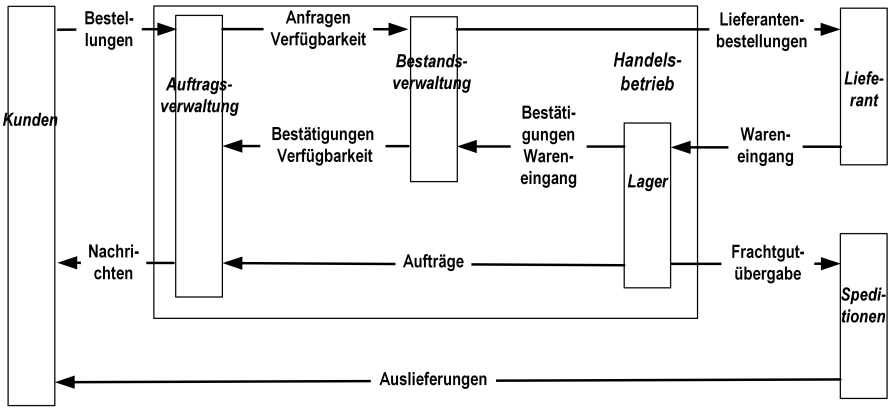
(b) das Modul $Bestandsverwaltung \bullet Lager$



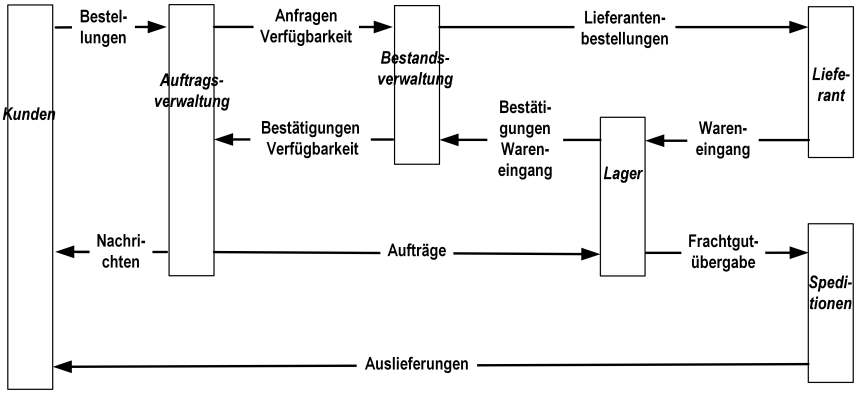
(c) der Handelsbetrieb, definiert als $Auftragsverwaltung \bullet Bestandsverwaltung \bullet Lager$

Abb. 3: Komposition von Abteilungen

Beide Eigenschaften sind fundamental wichtig für einen in der Praxis nützlichen Kompositionsoperator für Module und damit für Modelle realer Systeme.



(a) die Komposition $Kunden \bullet Handelsbetrieb \bullet Lieferant \bullet Speditionen$



(b) die Komposition $Kunden \bullet Auftragsverwaltung \bullet Bestandsverwaltung \bullet Lager \bullet Lieferant \bullet Speditionen$

Abb. 4: der Handelsbetrieb im Kontext seiner Geschäftspartner

Abbildung 4a komponiert das Handelsbetrieb-Modul mit den Modulen für die Kunden, den Lieferanten und den Speditionen zum *Gesamtmodul* der Fallstudie. Die beteiligten Module sind dann *Teilmodule* des Gesamtmoduls.

Jedes Gate des Handelsbetrieb-Moduls ist per Konstruktion zugleich auch ein Gate eines seiner drei Teilmodule. Deshalb kann man in Abbildung 4a das Handelsbetrieb-Modul ersetzen durch seine drei Teilmodule, wie in Abbildung 4b. Wie am Ende von Abschnitt 3 erläutert, deutet eine Pfeilspitze an einem Gate intuitiv eine Flussrichtung an. Solche Pfeilspitzen sind auch in komponierten Modulen intuitiv nützlich.

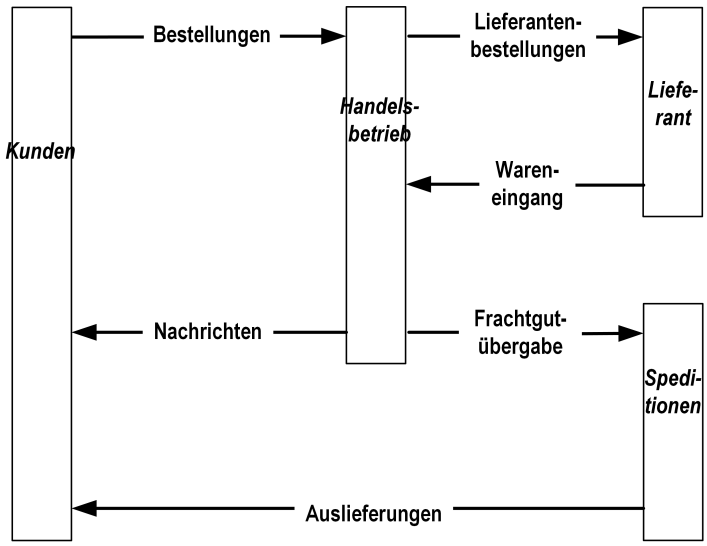


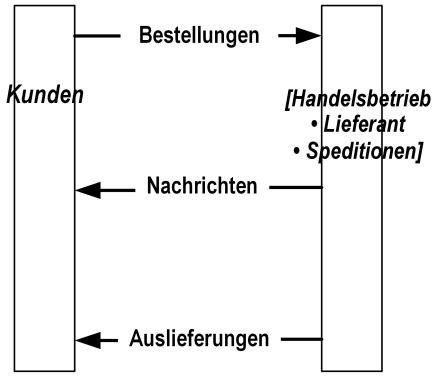
Abb. 5: Komposition der Abstraktionen: $[Kunden] \bullet [Handelsbetrieb] \bullet [Lieferant] \bullet [Speditionen]$

5. Abstraktion von HERAKLIT-Modulen

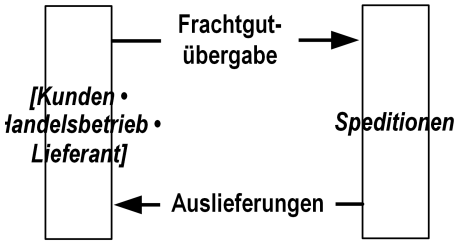
Das Modul in Abbildung 3c ist nicht abstrakt, sondern hat eine innere Struktur. Wenn man von dieser inneren Struktur abstrahiert, erhält man Abbildung 1a, die abstrakte Version des Handelsbetriebes. Generell hat jedes Modul seine eindeutig bestimmte *abstrakte Version*: Die Oberfläche bleibt bestehen, das Innere wird gelöscht, bis auf den Namen des Moduls. Für ein Modul L bezeichnet $[L]$ seine abstrakte Version. Abbildung 3c trägt den Namen „Handelsbetrieb“, folglich muss Abbildung 1a streng genommen „$[Handelsbetrieb]$“ heißen. Nun sieht man aber, dass die Module in Abbildung 1 alle abstrakt sind; deshalb bleibt die Darstellung auch ohne den Abstraktionsoperator $[\cdot]$ eindeutig. Wenn man ein Modul als Komposition gegebener Module bildet und die abstrakte Version meint, schafft der Abstraktionsoperator Eindeutigkeit. Ein Beispiel zeigt der Vergleich von Abbildung 4a mit Abbildung 5.

Den Schritt von einem Modul L zu seiner abstrakten Version $[L]$ kann man umdrehen und L als eine von vielen möglichen Verfeinerungen von $[L]$ auffassen. So ist beispielsweise Abbildung 3c eine Verfeinerung von Abbildung 1a. Wenn man in einem komponierten Modul $L = L_1 \bullet L_2 \bullet ... \bullet L_n$ ein Teilmodul L_i verfeinert oder abstrahiert, ändert sich nichts an den anderen Teilmodulen.

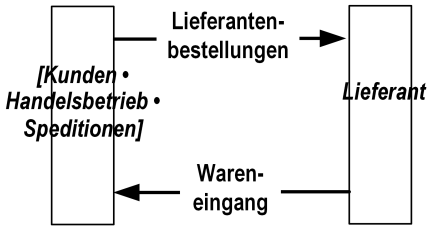
Die drei Teilmodule aus Abbildung 3c werden später weiter verfeinert, ebenso die Geschäftspartner aus Abbildung 1.



(a) das System aus Sicht der Kunden: $Kunden \bullet [Handelsbetrieb \bullet Großhandel \bullet Speditionen]$



(b) das System aus Sicht der Speditionen $[Kunden \bullet Handelsbetrieb \bullet Großhandel] \bullet Speditionen$

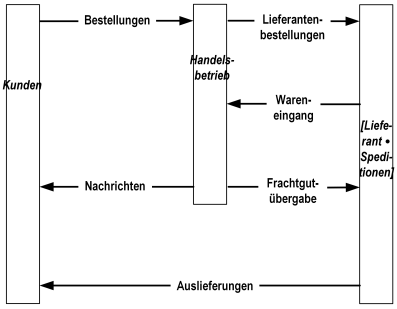


(c) das System aus Sicht der Lieferanten: $[Kunden \bullet Handelsbetrieb \bullet Speditionen] \bullet Lieferant$

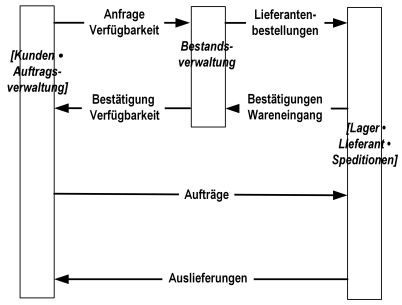
Abb. 6: das System aus Sicht der drei Geschäftspartner

6. Sichten

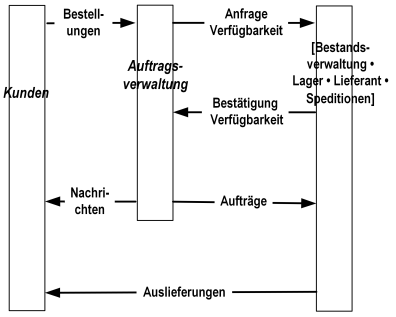
Abstraktion und Komposition von Teilmodulen ergeben verschiedene Sichten auf das Gesamtsystem. Insbesondere hat jeder Geschäftspartner seine Sicht, nämlich



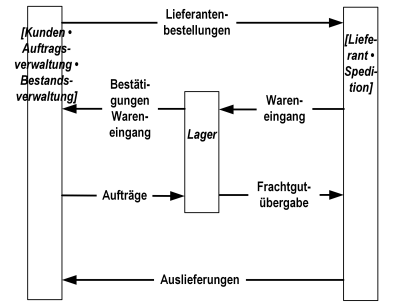
(a) Sicht des Handelsbetriebes: $[Kunden] \bullet Handelsbetrieb \bullet [Lieferant \bullet Speditionen]$



(b) Sicht der Bestandsverwaltung: $[Kunden \bullet Auftragsverwaltung] \bullet Bestandsverwaltung \bullet [Lager \bullet Speditionen]$



(c) Sicht der Auftragsverwaltung: $[Kunden] \bullet Auftragsverwaltung \bullet [Bestandsverwaltung \bullet Lager \bullet Lieferant \bullet Speditionen]$



(d) Sicht des Lagers: $[Kunden \bullet Auftragsverwaltung \bullet Bestandsverwaltung] \bullet Lager \bullet [Lieferant \bullet Speditionen]$

Abb. 7: Sichten des Handelsbetriebes und seiner Abteilungen

die Abstraktion der anderen komponierten Teilmodule. Abbildung 6 zeigt diese drei Sichten.

Ein Modul (in beliebiger Detaillierungstiefe) „sieht“ den Rest des Systems im Allgemeinen in zwei Teilen: das Teilmodul an seiner linken und das an seiner rechten Seite. Abbildung 7a zeigt diese Sicht des Handelsbetriebes. Aber auch jedes seiner Teilmodule hat seine Sicht, wie die Abbildungen 7b, 7c und 7d zeigen.

III Daten und Gegenstände in HERAKLIT-Modellen

7. Daten, Gegenstände, Typen, Multimengen

Um die Ausdrucksstärke von HERAKLIT zu zeigen, beleuchten wir exemplarisch einige Daten-Aspekte der Fallstudie besonders detailliert. Eine *Bestellung* besteht aus einem *Kunden* (genauer: einem Kundennamen) und einer *Artikelliste*.

Die wiederum ist eine Menge von *Artikelpositionen*. Und jede Position besteht aus einem *Artikel* und der *Anzahl* bestellter Exemplare.

Kundennamen, Artikel, Artikelpositionen und Artikellisten sind *Daten*; man kann sie beispielsweise in einem Katalog darstellen, digital versenden, auf Papier drucken et cetera. Insbesondere kann man sie in digitaler Form rechnergestützt verarbeiten.

Daneben gibt es konkrete, realweltliche *Gegenstände*, im Beispiel die Schuhe, Hosen, Hemden und Hüte, also die Waren, die der Lieferant dem Lager sendet, die das Lager verpackt und als Frachtgut den Speditionen übergibt und die dann dieses Frachtgut an die Kunden ausliefern.

Daten und Gegenstände haben ganz unterschiedliche Eigenschaften: Ein Datenelement wie beispielsweise eine Artikelliste kann man problemlos kopieren oder zerlegen; einen Gegenstand nicht. Mit einem Gegenstand kann man kaum mehr machen als ihn zusammen mit anderen in Frachtgut bündeln, ihn von Ort zu Ort transportieren und das Bündel wieder zu öffnen. Daraus folgt eine interessante Eigenschaft, die Daten nicht haben: ein vorhandener Gegenstand, beispielsweise ein Hut, befindet sich immer an genau einem Platz. Daten und Gegenstände sind *Objekte*, die HERAKLIT im selben Formalismus repräsentiert.

Im systematischen Aufbau von HERAKLIT sind Objekte Elemente von *Mengen*; sie sind also nicht im speziellen Sinne der objektorientierten Modellierung oder Programmierung zu verstehen. Oftmals sind es *Multimengen*: in einer Multimenge kann ein Element mehrfach vorkommen, beispielsweise kann eine Lieferung für eine Bestellung zwei Hüte und drei Paar Schuhe enthalten, ohne dass zwischen den beiden Hüten oder den Schuh-Paaren unterschieden wird. Wir schreiben diese Bestellung als Multimenge $[Hut, Hut, Schuhe, Schuhe, Schuhe]$. Formal ist eine Multimenge mit Elementen aus einer Menge A eine Abbildung $M : A \to N$, die jedem Element von A seine Anzahl in M zuordnet; im Beispiel also $M(Hut) = 2$ und $M(Schuhe) = 3$.

Multimengen L und M über einer Menge A kann man addieren:

$$(L + M)(a) \coloneqq L(a) + M(a),$$

mit einem Skalar (einer natürlichen Zahl n) multiplizieren:

$$(n \cdot M)(a) \coloneqq n \cdot (M(a)),$$

und miteinander vergleichen:

$$L \leqslant M \coloneqq \forall a \in A : L(a) \leqslant M(a).$$

Die Potenzmenge $P(M)$ ist die Menge aller Mengen N mit $N < M$. $M(A)$ ist die Menge aller Multimengen über A.

Schließlich verwenden wir noch *Prädikate*: ein Prädikat p trifft auf ein Element zu oder es trifft nicht zu. Für eine Menge M schreiben wir $p(elm(M))$, wenn p auf jedes Element von M zutrifft.

Abbildung 8 stellt die in diesem Dokument verwendeten Notationen zusammen.

Grundsorten
ℕ
endliche Mengen

Abgeleitete Sorten
A × B *kartesisches Produkt von Mengen A und B*
P(_) *Potenzmenge*

Funktions-Symbole:
_+_ : Addition auf ℕ
_-_ : Subtraktion auf ℕ
_<_ : Ordnung auf ℕ

Multimengen
intuitiv: eine Menge, in der einzelne Elemente mehrfach vorkommen können.
Notation: Beispiel: [a, a, b]
Formal: für eine gegebene endliche Menge A: eine Abbildung M: A⟶ ℕ „Multimenge über A"

Operationen auf Multimengen
für Multimengen L und M:
- Summe (Vereinigung) L+M,
- Für n ∈ ℕ: n•M: n-fache Addition von M,
- Halbordnung L < M: komponentenweise,
P(M) Potenzmenge von M (alle Mengen N kleiner oder gleich M)
M(A) Menge aller Multimengen über A

Eine Notation für Prädikate p
Für Menge A = {a,b,c}: *„p trifft zu auf elm(A)"* ist eine Kurzschreibweise für *„p trifft zu auf a und auf b und auf c"*. Für Multimengen: entsprechend mehrfach

Abb. 8: HERAKLIT-Multimengen

8. Die Struktur S und das Schema des Handelsbetriebes

Das Modell des Handelsbetriebes besteht aus vier unterschiedlichen Arten von Objekten:

- *Grundsorten*: das sind *Kunden*, *Artikel*, *Termine*, *Waren*, *Speditionen*;
- *Abgeleitete Sorten*: eine *Artikelposition* besteht aus einem Artikel und einer Anzahl, eine *Artikelliste* ist eine Multimenge von Artikelpositionen, eine *Artikelmenge* ist eine Multimenge von Artikeln und eine *Warenmenge* ist eine Multimenge von Waren;
- *Konstanten*, die im Modell explizit vorkommen: spezifische Artikelpositionen, Kunden sowie anfangs gelistete Artikel, Waren und Speditionen;
- *Funktionen*, die Grundsorten und abgeleitete Sorten aufeinander beziehen: Jede Artikelposition p und jede Artikelliste a entspricht einer Multimenge p beziehungsweise a von Artikeln und jede Ware W entspricht einem Artikel $f(w)$;
- *Prädikate* über den Grundsorten und den abgeleiteten Sorten: $bestellende\ Kunden$, $abgeschickte\ Bestellungen$, $Kopien\ abgeschickter\ Bestellungen$, $geliefertе\ Waren$ et cetera sind Prädikate. Auf ein Element einer Menge trifft ein Prädikat zu oder es trifft nicht zu. Beispiel: Das Prädikat *abgeschickte Bestellungen* trifft auf alle Bestellungen zu, die von Kunden abgeschickt, aber von dem Handelsbetrieb noch nicht bearbeitet worden sind.

Daneben verwenden wir noch eine Reihe von Variablen für die verschiedenen Sorten. Grund- und abgeleitete Mengen, sowie Funktionen und Prädikate über diesen Mengen bilden eine *(Tarski-) Struktur*. Solche Strukturen bilden die Grundlage zur Formulierung dynamischen Verhaltens mit HERAKLIT-Verhaltensmodellen.

Grundsorten

KN = {Ute, Max} *Kunden*
AR = {„Schuhe“, „Hut“, „Hose“} *Artikel*
WA = {Schuhe, Hut, Hose} *Waren*
TE = {23.12., 24.12.} *Termine*
SP = {Maier, Müller, Schulz} *Spediteure*

Abgeleitete Sorten

AP = **AR** × ℕ *Artikelpositionen*
AL = ***M*(AP)** *Artikellisten*
AM = ***M*(AR)** *Artikelmengen*
WM = ***M*(WA)** *Warenmengen*

Konstanten

p1: AP = („Schuhe“, 2),
p2: AP = („Hut“, 1)
K: P(KN) = {Ute, Max} *bestellende Kunden*
G: AL = {(„Schuhe“, 3), („Hut“, 1)} *anfangs gelistete Artikel*
H: WM = {Schuhe, Schuhe, Schuhe, Hut} *anfangs verfügbare Waren*
R: P(SP) = {Maier, Müller} *anfangs verfügbare Speditionen*

Funktionen

(a,n)‘ = n[a] für (a,n) ∈ **AP**
[p1, … , pn]‘ = p1‘ + … + pn‘ für [p1, … , pn] ∈ **AL**
Beispiel: [p1, p2, p2]‘ =
{Schuhe, Schuhe, Hut, Hut}

f(w) = „w“, für w ∈ **WA**
f([a1, … , an]) = [f(a1), … , f(an)] für [p1, … , pn] ∈ **AM**

Abb. 9: die Heraklit-Struktur S des Handelsbetriebes

(Tarski-) Strukturen sind zugleich der semantische Bereich, für den die Prädikatenlogik Aussagen formuliert. Prädikatenlogik und temporale Logik liegen dann nahe zur Formulierung und zum Nachweis von Eigenschaften von Heraklit-Modellen.

Abbildung 9 zeigt die Struktur und die Variablen, die wir im Weiteren zur Beschreibung der Fallstudie verwenden.

IV Verteilte Abläufe

Kunden schicken Bestellungen an den Handelsbetrieb und empfangen von ihm Lieferungen; der Betrieb bestellt knapp werdende Artikel bei einem Lieferanten nach, er beauftragt eine Spedition, Fracht aus dem Lager an Kunden auszuliefern et cetera. Solcherart Verhalten, zusammengesetzt aus einzelnen als elementar angesehenen Ereignissen, wird in Heraklit-*Verhaltensmodulen* beschrieben.

9. Lokale Zustände und Ereignisse

In den Natur- und Technikwissenschaften wird das Verhalten eines Systems sehr häufig modelliert als kontinuierlicher Prozess entlang einer Zeitachse reeller Zahlen. Hier fassen wir *Verhalten* fundamental anders auf: Das Verhalten eines Systems wird beschrieben mit *lokalen Zuständen*, die von diskreten *Ereignissen* aktualisiert werden. Das Resultat eines Ereignisses kann Anlass für weitere Ereignisse sein.

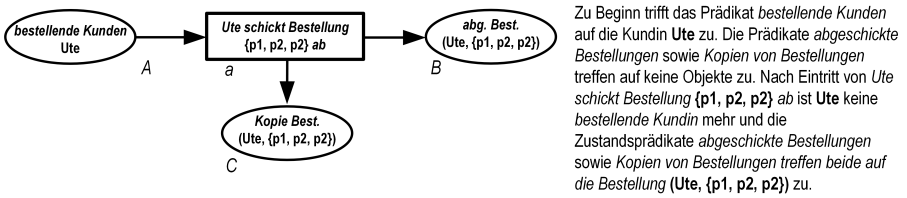


Abb. 10: Ereignis *Ute schickt Bestellung* $\{p1, p2, p2\}ab$

Die Beschreibung lokaler Zustände basiert auf Prädikaten (siehe Abschnitt 7): ein *lokaler Zustand* ist ein Prädikat p zusammen mit einem Objekt o. Dynamik entsteht, indem das Objekt o den lokalen Zustand *erreicht*: dann trifft p auf o zu. Wenn o den lokalen Zustand *verlässt*, trifft p nicht mehr auf o zu. Das Erreichen und Verlassen lokaler Zustände wird durch den Eintritt von *Ereignissen synchronisiert*: Einige Objekte erreichen oder verlassen einige lokale Zustände.

Ein typisches Beispiel ist die Bestellung von Artikeln: indem beispielsweise die Kundin *Ute* eine Bestellung abschickt, verlässt sie den lokalen Zustand *bestellende Kunden* und die Bestellung erreicht die beiden lokalen Zustände *abgeschickte Bestellungen* und *Kopien abgeschickter Bestellungen*. Abbildung 10 stellt das Ereignis *Ute schickt Bestellung* $\{p1, p2, p2\}$ *ab* grafisch dar: jede der drei Ellipsen repräsentiert eines der beteiligten Prädikate zusammen mit dem entsprechenden Objekt, auf das das Prädikat zutrifft. Das Rechteck t enthält den Namen des Ereignisses. Ein Pfeil zwischen einer Ellipse und dem Rechteck zeigt, ob das Objekt bei Eintritt des Ereignisses den entsprechenden lokalen Zustand verlässt (Pfeilspitze am Rechteck) oder erreicht (Pfeilspitze an der Ellipse).

Formal hat das Ereignis *Ute schickt Bestellung* $\{p1, p2, p2\}$ *ab* die Struktur eines *Netzes*: jede Ellipse ist ein *Platz*, das Rechteck ist eine *Transition*, die Pfeile bilden die *Flussrelation*.

Zwei Ereignisse e und f können zu $e \bullet f$ *komponiert* werden, wenn ein Objekt o einen lokalen Zustand p mit Eintritt von e erreicht und dann mit Eintritt von f verlässt. Als Beispiel zeigt Abbildung 11 das Ereignis *Ute empfängt Nachricht*. Wie oben dargestellt, erreicht die Bestellung $(Ute, \{p1, p2, p2\})$ den lokalen Zustand *Kopie Bestellungen* mit dem Ereignis $Ute\ schicktBestellung\{p1, p2, p2\}ab$ und verlässt ihn mit dem Ereignis *Ute empfängt Nachricht*. Die beiden Ereignisse werden komponiert, indem die beiden Instanzen des lokalen Zustandes *Kopie Bestellungen* miteinander verschmelzen, wie Abbildung 12 zeigt.

Wie schon ein einzelnes Ereignis, hat auch die Komposition zweier Ereignisse die Struktur eines Netzes; bestehend aus Plätzen, Transitionen und Pfeilen.

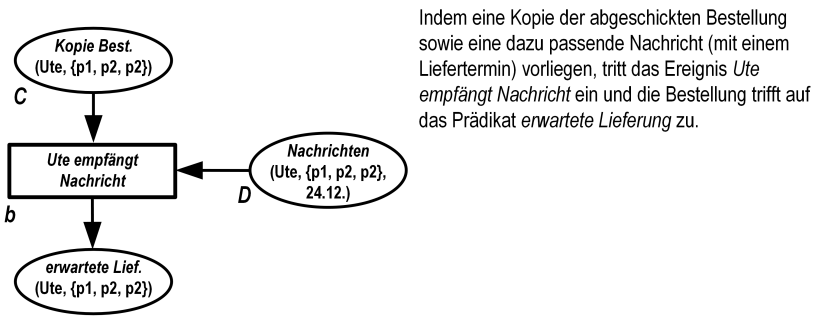


Abb. 11: Ereignis *Ute empfängt Nachricht*: Lieferung kommt an Weihnachten

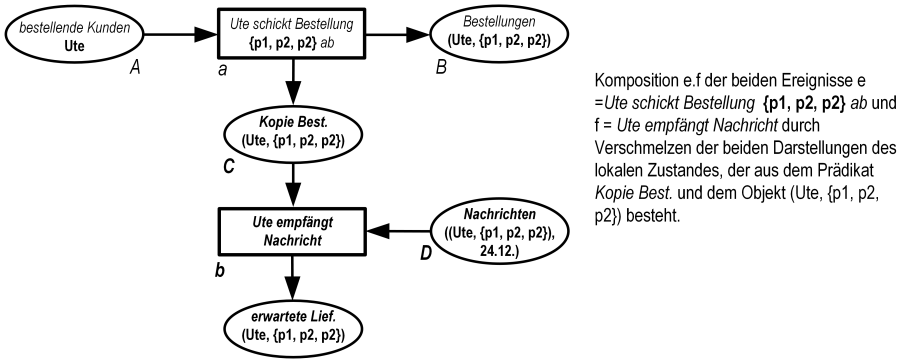
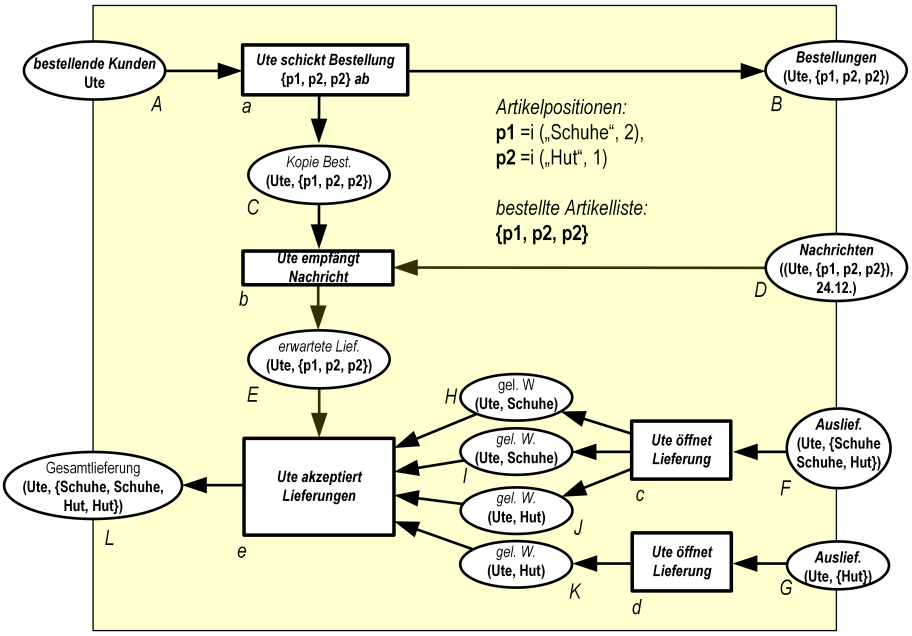


Abb. 12: Komposition: Verschmelzen lokaler Zustände

10. Verteilte Abläufe von Modulen

Die Komposition mehrerer Ereignisse beschreibt ein Stück *Verhalten*; ein *Ablauf* entsteht. Die Komposition in Abbildung 12 beschreibt ein Anfangsstück eines möglichen *Ablaufs* des Kunden-Moduls. Abbildung 13 ergänzt dieses Anfangsstück zu einem vollständigen Ablauf des Kunden-Moduls. Ein Modul kann sich auf verschiedene Weise verhalten; beispielsweise kann die Kundin Ute für ihre Bestellung zwischen ganz unterschiedlichen Artikellisten wählen.

Im Ablauf des Moduls $Auftragsverwaltung$ in Abbildung 14 wird keine Reihenfolge angenommen, in der die Bestätigungen der Reservierung der Artikelpositionen das Modul erreichen oder in der das Modul die beiden Pakete beauftragt. Allerdings wird die Kundin erst benachrichtigt, nachdem sichergestellt ist, dass das Lager für jede gewünschte Artikelposition eine entsprechende Ware vorrätig hat.



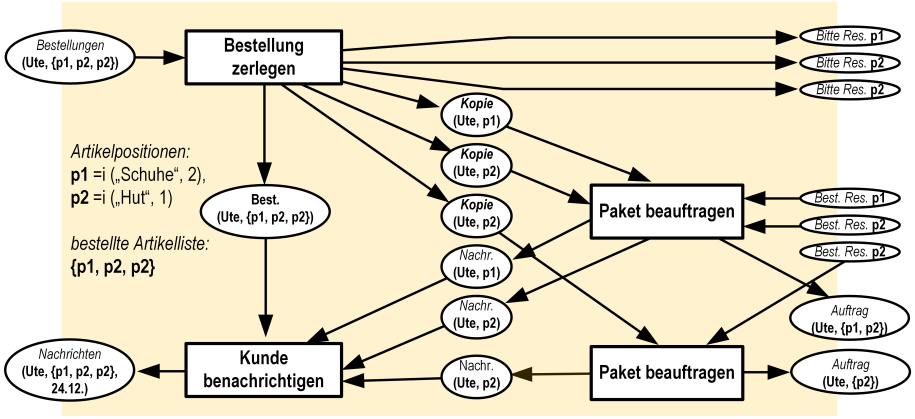
Nachdem die Bestellung abgeschickt wurde, erreichen eine Nachricht und zwei Lieferungen das Modul (lokale Zustände „*Auslief.*“ in der rechten Schnittstelle). Ute öffnet beide Lieferungen. Zusammen enthalten sie zwei Paar Schuhe und zwei Hüte. Damit akzeptiert die Kundin Ute die gelieferte Ware.

Abb. 13: ein Ablauf des Moduls *Kunden*

Das Modul *Bestandsverwaltung* muss eine Bitte um zwei Paar Schuhe und eine Bitte um zwei Hüte beantworten. Im Ablauf in Abbildung 14 nehmen wir an, die Liste der verfügbaren Waren des Handelsbetriebs vermerkt am Lager drei Paar Schuhe und einen Hut (diese Liste wird verkürzt und technisch unspezifisch als Datenbank bezeichnet). Damit muss ein Hut beim Lieferanten besorgt werden.

Die Abbildung 16 zeigt, dass das Lager vom Lieferanten zwei Hüte entgegennimmt und diese Lieferung der Bestandsverwaltung meldet. Im gezeigten Ablauf werden zwei Aufträge erledigt: Einer verpackt einen der beiden gelieferten Hüte als Paket und übergibt das Paket als Fracht den Speditionen. Für den zweiten Auftrag sind zwei Paar Schuhe und ein Hut auf Lager; sie werden verpackt und den Speditionen übergeben.

Wie die Abbildung 17 zeigt, empfängt der Lieferant von der Bestandsverwaltung die Bestellung zweier Hüte und liefert die Hüte an das Lager. Wie die Abbildung 18 zeigt, werden die beiden Pakete unabhängig voneinander an die Kundin Ute ausgeliefert.



Die Artikelliste der Bestellung wird in ihre 3 Artikelpositionen zerlegt. Für jede Artikelposition wird eine Bitte um Reservierung entsprechender Waren an die Bestandsverwaltung geschickt (ohne den Kunden zu nennen) und eine Kopie (mit Kundennamen) gespeichert (lokale Zustände *Best. Verf. pi*).

Es werden zwei Pakete beauftragt: Eins mit den Artikelpositionen p1 und p2, und eins mit der zweiten Artikelposition p2. Schließlich wird die Kundin benachrichtigt.

Abb. 14: ein Ablauf des Moduls *Auftragsverwaltung*

11. Komposition der Abläufe der Module: Ein Ablauf des Handelsbetriebes

In Abschnitt 10 haben wir gesehen, wie einzelne Ereignisse komponiert werden können und so ein mögliches Verhalten jedes der sechs Module dargestellt wird. Nun komponieren wir dieses Verhalten der Module zu einem Ablauf des Handelsbetriebs. Dieser Ablauf beginnt mit einer Bestellung eines Kunden, verläuft durch alle sechs Module und kehrt schließlich zurück zum Kunden mit der Auslieferung der bestellten Waren. Abbildung 19 zeigt die Komposition der Abläufe der sechs Module aus Abschnitt 10. Dabei entstehen zwei Probleme: 1. Die grafische Anordnung der zu verschmelzenden Schnittstellen-Elemente passt im Allgemeinen nicht überein. Zudem sollen teilweise Schnittstellen-Elemente verschmolzen werden, deren Label nicht übereinstimmen. Technisch organisieren wir das mit *Adapter-Modulen*, grafisch dargestellt als eine Linie mit einem schwarzen Quadrat, notiert als Module *p1-Schuhe2* beziehungsweise *p2-Hut1*.

Aufgefasst als Graph ist Abbildung 19 azyklisch: Keine Pfeilkette schließt sich zu einem Kreis. Manche Ereignisse sind dadurch in einer „vor-nach“-Beziehung geordnet: Wenn ein Ereignis e durch eine Kette anderer Ereignisse vor einem Ereignis f liegt, dann liegt f sicher nicht vor e. Allerdings treten zwei Ereignisse gegebenenfalls nebeneinander ohne Ordnung ein. Diese Halbordnung wird

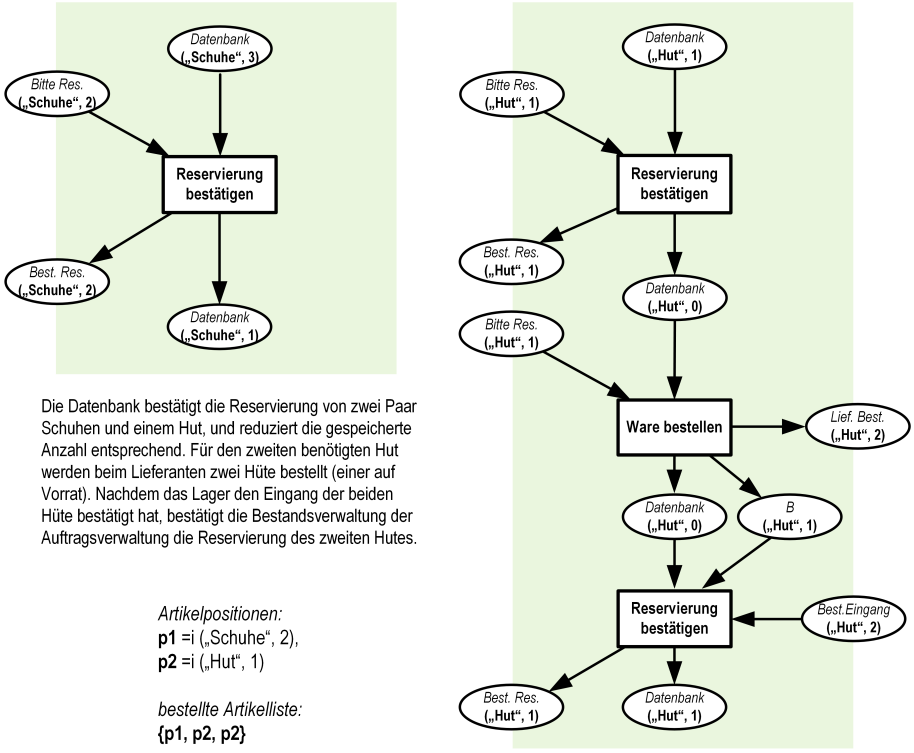


Abb. 15: ein Ablauf (bestehend aus zwei Teilen) des Moduls *Bestandsverwaltung*

anschaulich deutlich durch die Anordnung der Knoten des Gesamtablaufs in Abbildung 20: jeder Pfeil läuft von links nach rechts, allerdings sind jetzt Teile einzelner Module nicht mehr lokal unmittelbar zusammen angeordnet. Der farbige Hintergrund zeigt jeweils das beitragende Modul. Der Ablauf mancher Module ist in mehrere Teile zerlegt. Gelegentlich ist auch die links/rechts-Orientierung von Schnittstellen-Elementen der Module vertauscht.

V Prädikate und Ereignis-Schemata

Im vierten Teil haben wir *einen* konkreten Ablauf, ein Stück Verhalten eines Handelsbetriebs vorgestellt, indem die Kundin *Ute* eine Bestellung mit der Artikelliste $[p1, p2, p2]$ abschickt. Das soll nun allgemeiner gefasst werden: es gibt unendlich viele mögliche Artikellisten für eine Bestellung von Ute; außer Ute kann auch Max Bestellungen abschicken; zu Beginn können im Lager die drei Artikel Hüte, Schuhe, Hosen in unterschiedlichen Stückzahlen vorliegen und die Auftragsverwaltung hat viele unterschiedliche Möglichkeiten, Pakete zusammen-

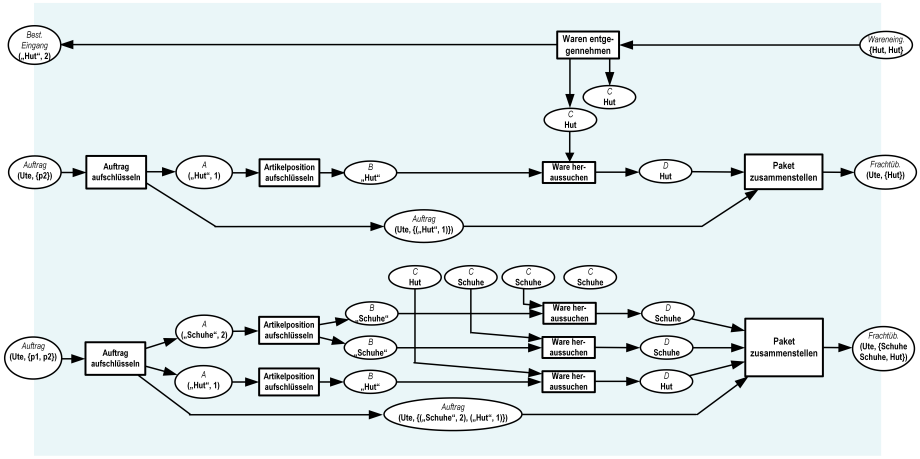


Abb. 16: Ein Ablauf des Moduls *Lager*

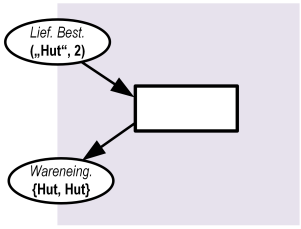


Abb. 17: ein Ablauf des Moduls $Lieferant$

zustellen. Dadurch gibt es unendlich viele mögliche Abläufe. Alle diese Abläufe sollen nun in *einer* Darstellung sichtbar werden. Die Idee dabei: Ereignisse mit gleichen Prädikaten werden zusammengefasst.

12. Ereignisse mit gleichen Prädikaten

Zunächst stellen wir das Ereignis *Ute schickt Bestellung* $\{p1, p2, p2\}$ *ab* aus Abbildung 10 anders dar: wie Abbildung 21 zeigt, werden dabei die beiden Konfigurationen vor und nach Eintritt des Ereignisses in zwei verschiedenen Grafiken repräsentiert. In Abbildung 21b repräsentiert der linke Platz nun nicht den lokalen Zustand, der aus dem Prädikat *bestellende Kunden* und der Kundin *Ute*

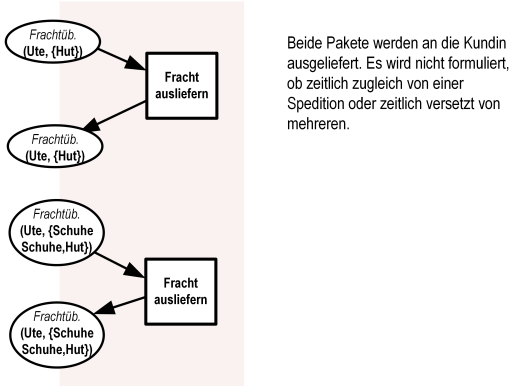


Abb. 18: ein Ablauf des Moduls *Speditionen*

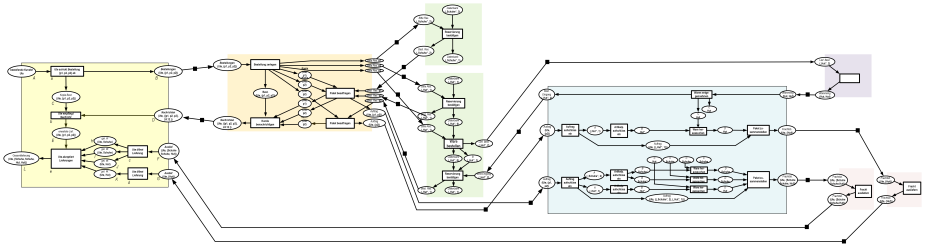


Abb. 19: Komposition der Abläufe der Module: ein Ablauf der komponierten Module

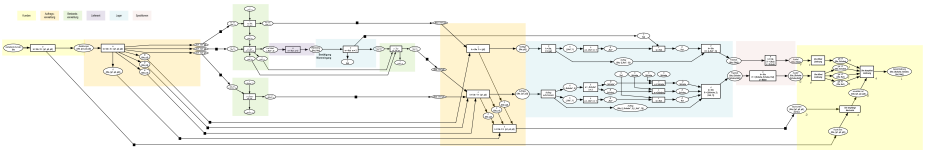
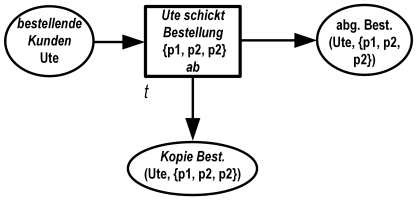
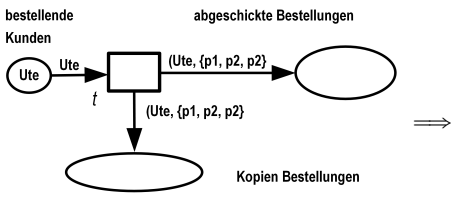


Abb. 20: der Ablauf, dargestellt von links nach rechts

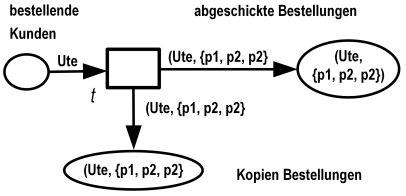
besteht, sondern er repräsentiert das Prädikat *bestellende Kunden*. Dass dieses Prädikat auf *Ute* zutrifft, zeigt dann die Marke „Ute“ innerhalb des Platzes. Entsprechend zeigen die *Prädikate abgeschickte Bestellungen* und *Kopien Bestellungen*, dass beide Prädikate vor Eintritt des Ereignisses auf keine Objekte zutreffen. Abbildung 21c zeigt die Situation nach Eintritt: das Prädikat *bestellende Kunden* trifft nun auf kein Objekt zu, die Prädikate *abgeschickte Bestellungen* und *Kopien Bestellungen* treffen nun beide auf das Objekt $(Ute, \{p1, p2, p2\}$ zu. Aus Abbildung21b allein kann man Abbildung 21c ableiten: Intuitiv formuliert, legt die Beschriftung an jedem Pfeil fest, welche Objekte „durch den Pfeil flie-



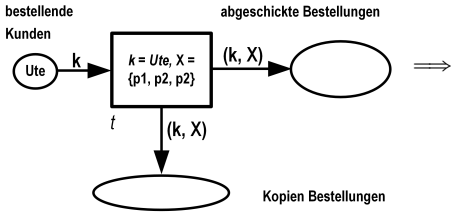
(a) Wiederholung von Abbildung 10: Darstellung eines Ereignisses mit Aussagen



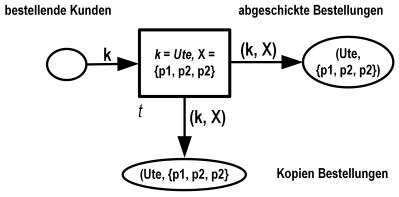
(b) Darstellung mit Prädikaten: vor Eintritt des Ereignisses



(c) Darstellung mit Prädikaten: nach Eintritt des Ereignisses



(d) Darstellung mit Variablen: vor Eintritt des Ereignisses

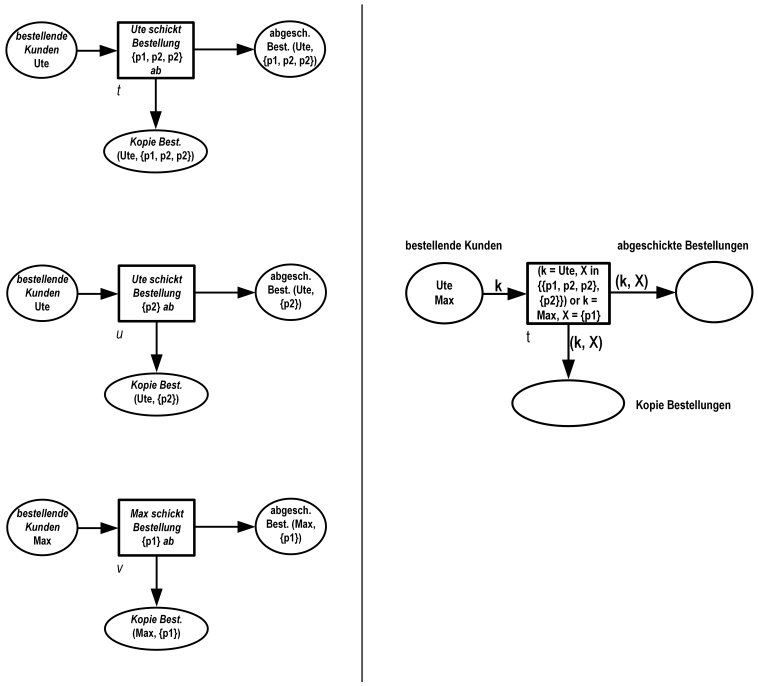


(e) Darstellung mit Prädikaten: vor Eintritt des Ereignisses

Abb. 21: Übergang von lokalen Zuständen zu Prädikaten

ßen“, wenn das Ereignis eintritt. Damit repräsentiert Abbildung 21b dasselbe Verhalten wie Abbildung 21a.

In einem weiteren Schritt ersetzen wir die Pfeilbeschriftungen durch Variablen (Abbildungen 21d und 21e), und erläutern die Belegung der Variablen in



(a) drei Ereignisse dargestellt mit Aussagen

(b) gemeinsame Darstellung der drei Ereignisse mit Prädikaten

Abb. 22: Darstellung von Ereignissen mit Hilfe von Prädikaten

einer Beschriftung der Transition. Auch die Darstellung in Abbildung 21d repräsentiert damit das Verhalten von Abbildung 21a.

Den Vorteil der neuen Darstellung verdeutlicht Abbildung 22. Teil 22a zeigt drei verschiedene Bestellungen; zwei von Ute und eine von Max. Teil 22b repräsentiert diese drei Bestellungen in *einem* Schema. Kernstück der Darstellung sind die *Variablen* k und X. Sie können mit konkreten Objekten *belegt* werden: k mit einem Kunden (Ute oder Max), X mit einer Artikelliste. Drei dieser Belegungen erfüllen die Beschriftung der Transition t. Jede von ihnen repräsentiert eines der drei Ereignisse in Abbildung 22a.

Mit Variablen und Bedingungen in der Transition t kann man also eine „gemeinte“ Menge von Bestellungen charakterisieren. Besonders einfach ist die Charakterisierung *aller* Bestellungen, die in der gegebenen Struktur möglich sind: dafür verzichtet man auf jegliche zusätzliche Bedingung.

Abbildung 23 zeigt den Umgang der Kunden mit Nachrichten des Handelsbetriebs: Der Zustellungstermin der letzten Teillieferung zu einer Bestellung wird avisiert: die Transition tritt ein, wenn die Variablen k und X so mit einem Kunden k_0 und einer Bestellliste X_0 belegt werden können, dass (k_0, X_0) als Marke in *Kopien Bestellungen* und für irgend einen Termin d_0 die Marke (k_0, X_0, d_0)

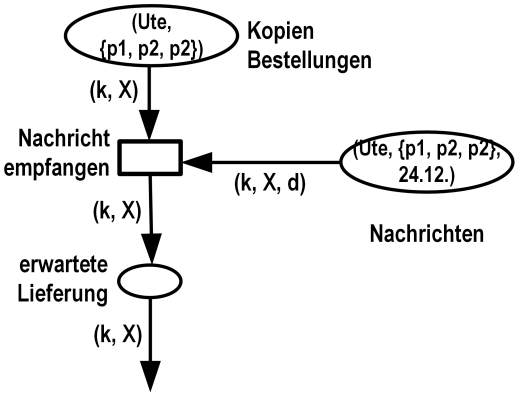
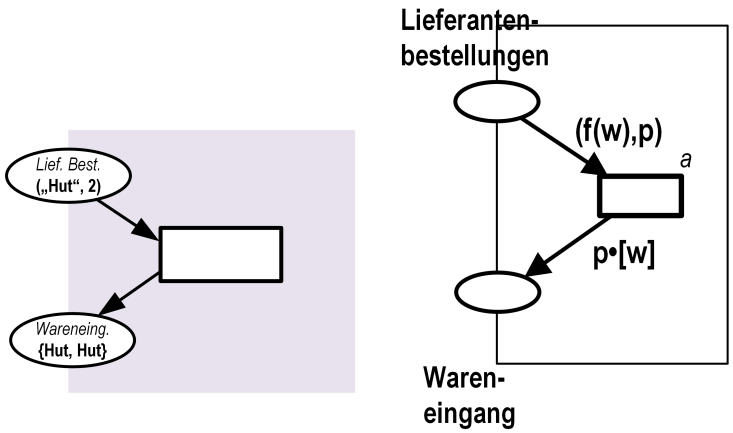


Abb. 23: aktivierte Transition *Nachricht empfangen*



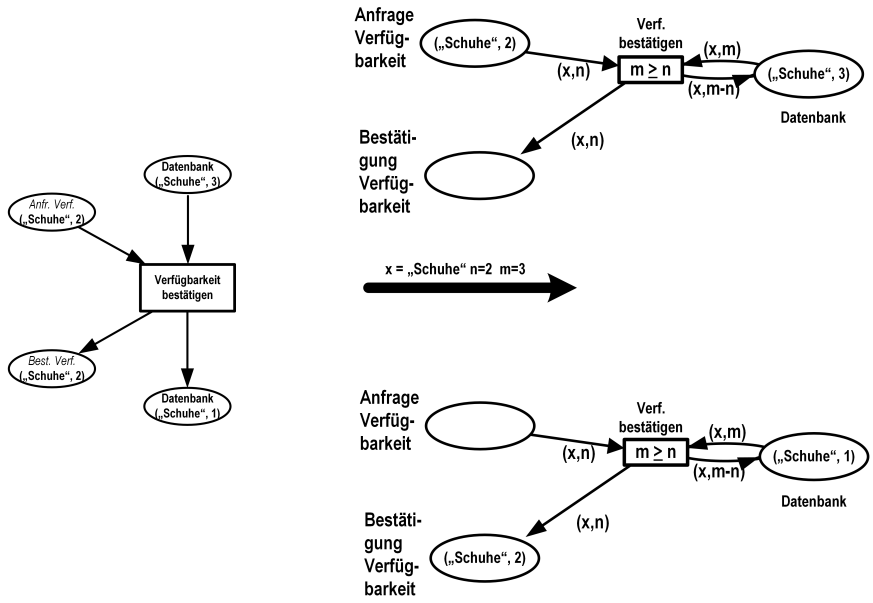
(a) Ereignis des Lieferanten (b) Verwendung der Funktion f

Abb. 24: System-Funktionen

in *Nachrichten* liegt. In Abbildung 23 sind k, X und d mit $Ute, (p1, p2, p2)$ und 24.12. belegt.

13. Funktionen

Abbildung 24a wiederholt die Abbildung 17: der Lieferant empfängt die Artikelposition $(„Hut“, 2)$ und liefert die Multimenge $[Hut, Hut]$. Wir benötigen ein generelles Prinzip, um aus einer Artikelposition eine Multimenge von Waren abzuleiten. Dafür verwenden wir eine Funktion, die jeder Ware einen Artikel zuordnet. In unserer Fallstudie ist das die Funktion f mit $f(Hut) = „Hut“$ und



(a) Ereignis der Bestandsverwaltung im Beispielablauf

(b) schematische Darstellung

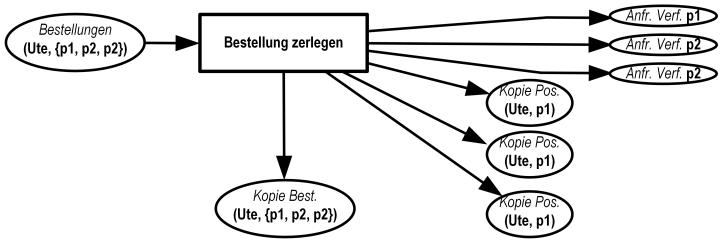
Abb. 25: Ereignis und zugehörige schematische Darstellung mit einer Schlinge

$f(Schuhe) =$ „$Schuhe$“. Im Allgemeinen ist f nicht injektiv; beispielsweise bietet ein Autoverleiher kleine, mittlere und großes Fahrzeuge, also drei Waren an, hat aber viele reale Fahrzeuge. Jedes ist klein, mittel oder groß.

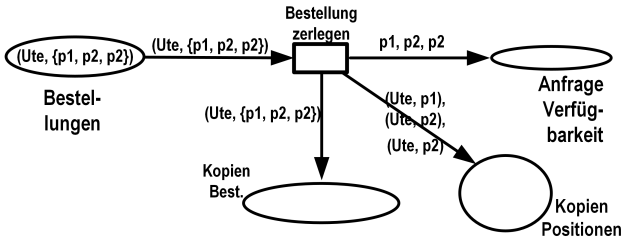
Zur schematischen Charakterisierung des Ereignisses des Lieferanten verwendet Abbildung 24b eine Variable w für die zu liefernde Ware und die Variable p für ihre Anzahl. Die einelementige Multimenge $[w]$ mit dem Faktor p ergibt die Multimenge $p \bullet [w]$, die p Exemplare der Ware w enthält. Die Funktion X' liefert zu einer *Artikelliste* oder *Artikelposition* die entsprechenden Artikel.

14. Schlingen

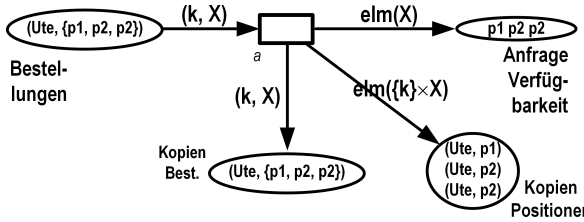
Es kommt vor, dass bei einem Ereignis ein Objekt einen lokalen Zustand verlässt und ein anderes (oder dasselbe) Objekt einen lokalen Zustand mit demselben Prädikat erreicht. Ein Beispiel für ein solches Prädikat ist die Datenbank der Bestandsverwaltung, die Abbildung 25a wiederholt. In einer schematischen Darstellung kommt jedes Prädikat nur einmal vor; deshalb entsteht in Abbildung 25b



(a) Ereignis im Modul $Auftragsverwaltung$



(b) dasselbe Ereignis in schematischer Darstellung



(c) dasselbe Ereignis mit Variablen im Modus $k = Ute, X = \{p1, p2, p3\}$

Abb. 26: die Verwendung von elm

eine Schlinge zwischen dem Platz *Datenbank* und der Transition *Verfügbarkeit bestätigen*.

15. elm

In den bisherigen Beispielen schematischer Darstellungen von Ereignissen verlässt oder erreicht höchstens *ein* Objekt einen Platz. Das ist nicht immer so; beispielsweise generiert das Ereignis *Bestellung zerlegen* im Modul *Auftragsverwaltung* drei lokale Zustände mit dem Prädikat *Anfrage Verfügbarkeit* (Abbildung 26a). In der schematischen Darstellung (Abbildung 26b) stehen deshalb die drei Objekte $p1, p2, p2$ am Pfeil hin zum Platz *Anfrage Verfügbarkeit*. Bei Eintritt von *Bestellung zerlegen* erreichen die drei Objekte zugleich diesen Platz. Bei

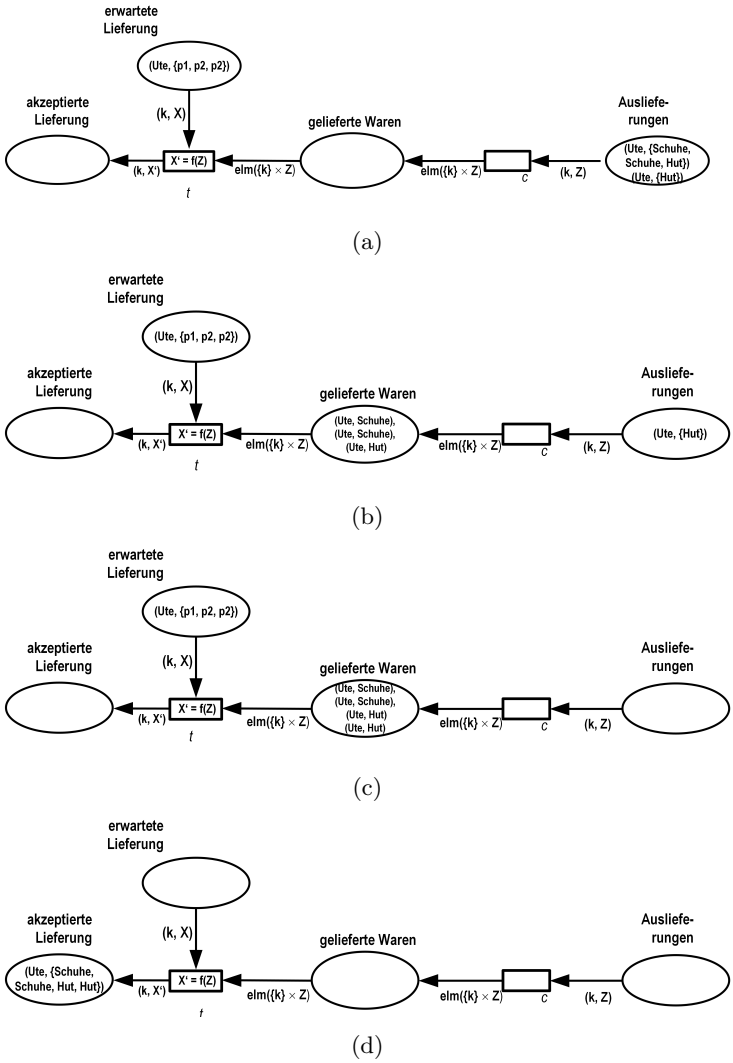


Abb. 27: Ware empfangen

der Verwendung der Variable X mit der Belegung $X = [p1, p2, p2]$ würde die Pfeilbeschriftung X die Menge $[p1, p2, p2]$ als eine Marke auf dem Platz ablegen anstatt der gemeinten drei Marken $p1, p2, p2$, also der Elemente dieser Menge (Abbildung 26c). Dass nach Eintritt von *Bestellung zerlegen* die Elemente der Menge X den Platz *Anfrage Verfügbarkeit* erreichen und nicht etwa die Menge selbst, wird durch die Notation $elm(X)$ erreicht. Entsprechendes gilt für den Platz *Kopien Position* und dem dort endenden Pfeil.

Als weiteres Beispiel zeigt Abbildung 27, wie Teillieferungen mit der Bestellung abgeglichen und als Gesamtlieferung akzeptiert werden: Wir nehmen in Abbildung 27a zwei Teillieferungen an. Zunächst wird eine ankommende Lieferung geöffnet und jede der Waren einzeln mit der Kundin *Ute* gekennzeichnet (Abbildung 27b). Nach der zweiten Teillieferung liegen vier Waren im Platz *gelieferte Waren* (Abbildung 27c). Mit der Menge dieser Waren wird die Variable Z belegt; $f(Z)$ kennzeichnet damit die Menge der bestellten Artikel. Diese Menge wird mit X' aus der Artikelliste $X = [p1, p2, p2]$ abgeleitet. Damit ist die Bedingung in der Transition erfüllt, die Transition tritt ein und die Gesamtlieferung wird akzeptiert (Abbildung 27d).

VI Der Handelsbetrieb für die Struktur S

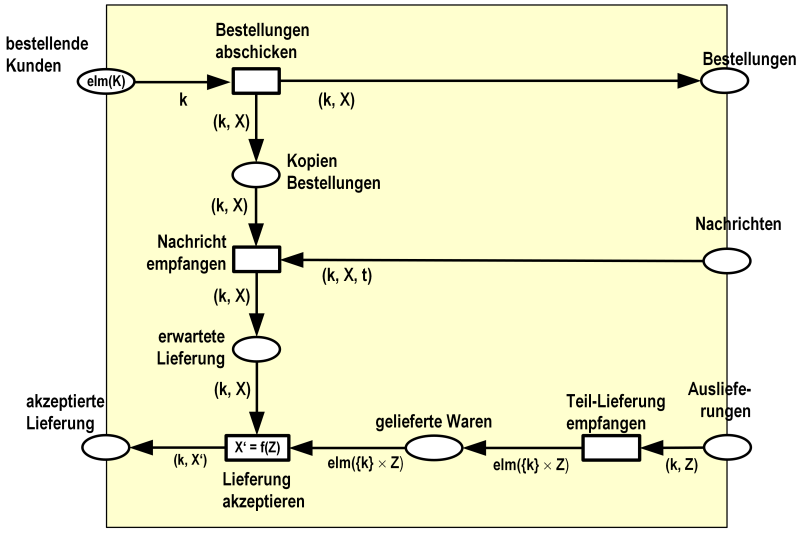
Wir haben nun alle Ausdrucksmittel beisammen, um alle Aspekte der Fallstudie zu modellieren. In der Reihenfolge der Logik einer Bestellung und ihrer Bearbeitung beginnen wir mit dem Kundenmodell. Dann folgen die drei Abteilungen des Handelsbetriebes (Auftragsverwaltung, Bestandsverwaltung und Lager) und schließlich die zwei anderen Geschäftspartner: Der Lieferant und die Speditionen. Die Modelle basieren auf der Struktur S, die in Abschnitt 8 in Abbildung 9 vorgestellt wurde.

16. Das Kundenmodell

Abbildung 28 zeigt das Modul *Kunden*. Mit der Menge $K = \{Ute, Max\}$ aus der Struktur S, liegen also zu Beginn die beiden Marken Ute und Max auf dem Platz *bestellende Kunden*. Die Transitionen haben wir schon im Einzelnen besprochen. Im Kontext mit den anderen Modulen zeigt Abbildung 29 eine erreichbare Markierung, von der aus Ute eine Nachricht mit dem Liefertermin (24.12.) erhält und danach in zwei Teillieferungen die Gesamtlieferung erhält und akzeptiert, wie in Abbildung 27 diskutiert. Unabhängig davon kann Max eine Bestellung abschicken.

17. Das Modell der Auftragsverwaltung

Die Transition *Bestellung zerlegen* der Auftragsverwaltung (Abbildung 30) haben wir schon in der Abbildung 26 diskutiert. Die Auftragsverwaltung bittet um die Betätigung der Verfügbarkeit für jede einzelne Warenposition. Das kann sich verzögern, wenn beispielsweise einige Exemplare erst beim Lieferanten nachbestellt werden müssen. Sobald einige Positionen den Platz *Bestätigung Verfügbarkeit* erreicht haben, kann die Auftragsverwaltung das Lager mit einer entsprechenden Teillieferung beauftragen. Auf dem Platz C werden die in Teillieferungen gesendeten Positionen notiert; wenn alle Positionen einer Bestellung in Teillieferungen beauftragt sind, verschickt die Transition *Kunde benachrichtigen* eine Nachricht an den Kunden mit dem vorgesehenen Termin für die letzte Teillieferung der Bestellung. Technisch kann die Auftragsverwaltung die Variable t mit einem ganz beliebigen Termin belegen.



Platz **bestellende Kunden**: Für eine Menge B von Bestellungen enthält der Platz A anfangs für jede einzelne Bestellung aus B eine Marke. Eine Bestellung besteht aus einem Kunden k und einer Artikelliste X.

Transition **Bestellung abschicken**: Ein Kunde k gibt eine Bestellung nach außen (Schnittstellenplatz **Bestellungen**) und behält eine Kopie im Platz **Kopien Bestellungen**.

Transition *Nachricht empfangen:* Eine Nachricht wird empfangen, wenn sie an eine Kopie einer Bestellung gerichtet ist und zusätzlich ein Datum enthält.

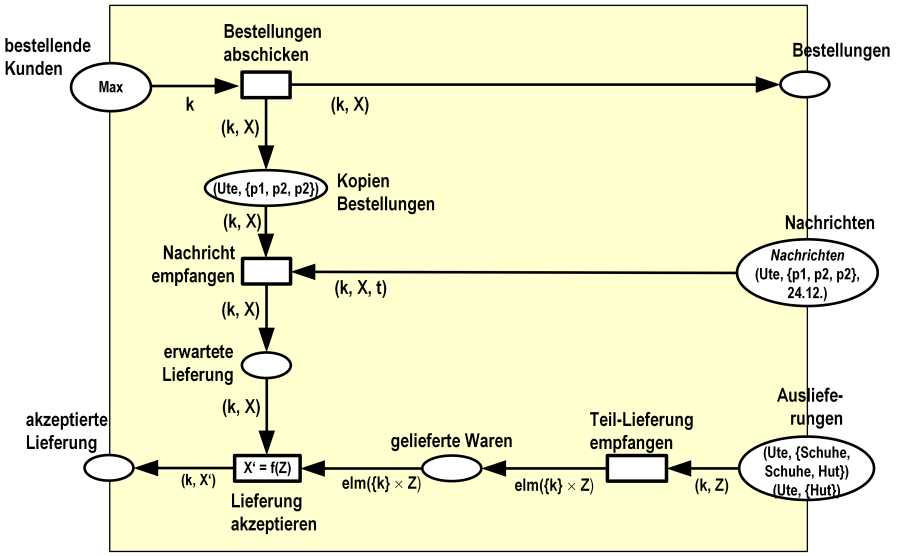
Transition **Teil-Lieferung empfangen**: Eine Teillieferung Z für den Kunden k wird „geöffnet" und jede einzelne Ware in Z als Marke auf dem Platz D abgelegt. Dabei können durchaus mehrere Teillieferungen für denselben Kunden k empfangen werden. Ihre Waren sammeln sich auf dem Platz **gelieferte Waren**.

Transition **Lieferung akzeptieren**: Wenn ein Kunde k eine Lieferung mit der Artikelliste X erwartet und für jeden Artikel der Artikelliste X auf dem Platz D Waren in entsprechender Stückzahl für k angekommen sind (Menge X'), endet der Bestellprozess im Platz C mit der Artikelliste X und der Menge X' passender Waren.

Abb. 28: das Modul *Kunden*

18. Das Modell der Bestandsverwaltung

Im Modell der Bestandsverwaltung (Abbildung 31) beschreiben die Einträge der Artikelliste G im Platz A für jede bestellte Artikelposition die Zahl passender verfügbarer Waren im Lager, anfangs also drei Paar Schuhe, ein Hut und kein Hemd. Falls für eine angefragte Position (a, n) im Lager mindestens n Exemplare verfügbar sind, gibt die Transition a für eine Teillieferung n Exemplare frei. Falls nicht, bestellt die Transition b beim Lieferanten die n Exemplare des Artikels



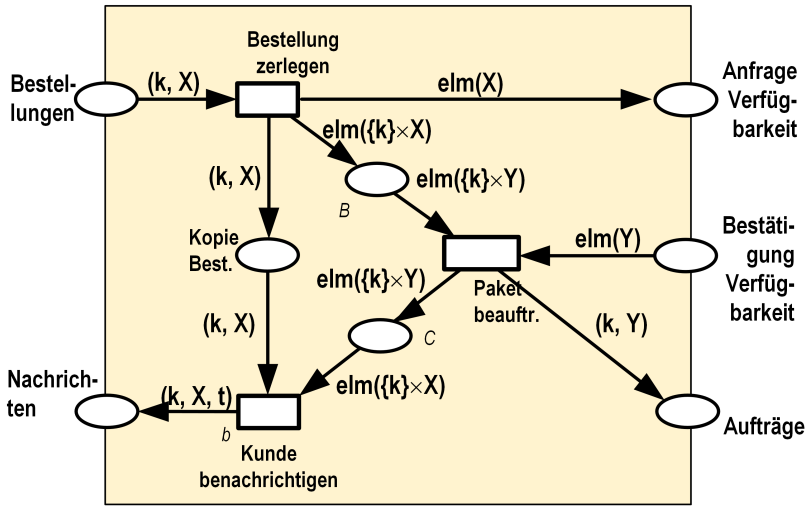
Nachdem die Bestellung abgeschickt wurde, erreichen eine Nachricht und zwei Lieferungen das Modul (lokale Zustände **Auslieferungen** in der rechten Schnittstelle).

Ute öffnet beide Lieferungen. Zusammen enthalten sie zwei Paar Schuhe und zwei Hüte. Damit akzeptiert die Kundin Ute die gelieferte Ware.

Abb. 29: eine erreichbare Markierung des *Kunden*-Modules im Kontext aller Module

a sowie einen Vorrat weiterer p Exemplare. Indem p als Variable definiert ist, kann die Bestandsverwaltung bei jeder einzelnen Nachbestellung den Umfang der Warenbevorratung für den Artikel a frei entscheiden. Wäre p als Konstante deklariert, würde die Struktur S den Umfang der Warenbevorratung für den Artikel a für immer festlegen.

Wenn die Anfrage nach Verfügbarkeit eines Artikels eine Bestellung beim Lieferanten auslöst, wird die Anfrage auf dem Platz B so lange festgehalten, bis das Lager den Erhalt der Exemplare vom Lieferanten bestätigt hat. Die Transition c gibt dann diese Bestätigung an die Auftragsverwaltung weiter und aktualisiert im Platz A die Information über den Bestand. Der Platz A beschreibt also für jeden Artikel, wie viele Exemplare im Lager noch nicht für Lieferungen reserviert sind. Hier erkennt man einen subtilen Unterschied zwischen einer großen und mehreren kleinen Artikelpositionen: wenn beispielsweise drei Hemden verfügbar sind und eine Position vier Hemden bestellt, wird die ganze Position auf Platz B zurückgehalten, bis der Lieferant Hemden geliefert hat. Werden stattdessen zwei Positionen jeweils zwei Hemden bestellen, wird eine von beiden sofort bestätigt und in einer Teillieferung kann der Kunde schon die ersten beiden Hemden erhalten, bevor der Großhandel die weiteren Hemden an das Lager liefert.



Transition **Bestellung zerlegen**: Die Auftragsverwaltung zerlegt die Artikelliste **X** einer eingehenden Bestellung **(k, X)** eines Kunden **k** in ihre Artikelpositionen; jede dieser Artikelpositionen wird als Marke in **Anfrage Verfügbarkeit** und auch in **B** abgelegt. Die Artikelpositionen in **B** werden zusätzlich mit dem Kunden **k** versehen.

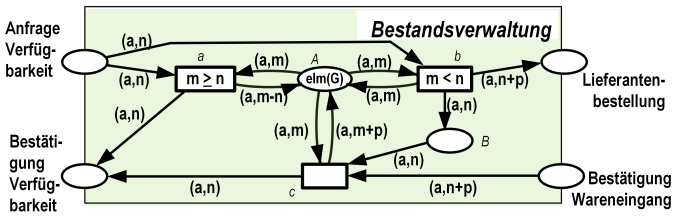
Transition **Kunde benachrichtigen**: Nachdem alle Artikelpositionen – in beliebig zusammengestellten Teillieferungen – beauftragt sind, wird dem Kunden das Lieferdatum mitgeteilt.

Die Transition **Paket beauftr**. ist besonderes interessant: Die Auftragsverwaltung wählt aus den verfügbaren Artikelpositionen eine Teilmenge Y, die sie in einer Teilsendung ausliefern möchte. Dann wird mit **Paket beauftr.** die Teillieferung für Y beauftragt, und die Artikelpositionen von Y werden im Platz C gesammelt. Alternativ kann **Paket beauftr.** auch für jede beliebige Teilmenge von Y eintreten, oder zwei mal für zwei disjunkte Teilmengen von Y. Intuitiv formuliert: Die Auftragsverwaltung hat freie Hand bei der Zusammenstellung von Teillieferungen.

Abb. 30: Die Auftragsverwaltung fragt nach der Verfügbarkeit von Waren und beauftragt Teillieferungen.

19. Das Modell des Lagers

Das Lager in Abbildung 32 hat zwei Aufgaben: zu jedem eingehenden Auftrag packt es ein Frachtpaket für die Speditionen; ferner verarbeitet das Lager die eingehenden Lieferungen des Lieferanten. Für die erste Aufgabe bricht die Transition a eine eingehende Artikelliste in ihre Positionen auf. Aus der Position (a, n) eines Artikels a, der in n Exemplaren bestellt wird, erzeugt die Transition b dann n Marken der Sorte a: Zunächst wird mit $n \bullet [a]$ die Multimenge erzeugt, die n Exemplare der Sorte a enthält. Mit $elm(n \bullet [a])$ wird dann jedes ihrer n Elemente zu einer individuellen Marke. Für jede dieser Marken a nimmt die



In Platz A führt die Bestandsverwaltung eine Liste der aktuell im Lager verfügbaren – und daher sofort lieferbaren – Waren. Diese Liste G ist eine Menge von Artikelpositionen; sie nennt zu jedem Artikel a die Anzahl der entsprechenden verfügbaren Waren. Jede dieser Artikelpositionen ist eine Marke im Platz A.

Eine Anfrage zur Verfügbarkeit einer Artikelposition eines Artikels x hat zwei mögliche Resultate: (1) Die Zahl m verfügbarer Exemplare ist größer als die Zahl n angefragter Exemplare. Dann bestätigt die Transition a die Verfügbarkeit der Artikelposition und reduziert den entsprechenden Eintrag in A.

(2) Andernfalls bestellt die Bestandsverwaltung mit der Transition b beim Lieferanten nicht nur die benötigte Zahl von n Exemplaren, sondern auch weitere p Exemplare als Vorrat. Bei jeder Instanziierung von b kann die Bestandsverwaltung eine andere Zahl p wählen.

Irgend wann bestätigt dann das Lager, dass es n+p Exemplare der Ware erhalten hat. Transition c bestätigt darauf hin der Auftragsverwaltung die Verfügbarkeit und aktualisiert die Artikelposition in Platz A.

Abb. 31: Die Bestandsverwaltung bestätigt die Verfügbarkeit von Artikelpositionen.

Transition c aus dem Lager C eine Ware w, die a entspricht, für die also gilt: $f(w) = a$. Diese Ware landet auf dem Platz D.

Für die zweite Aufgabe empfängt die Transition d vom Lieferanten ein Paket $m \bullet [w]$ mit m Exemplaren der Ware w. Die Exemplare werden einzeln ins Lager einsortiert (mit $elm(m \bullet [w])$ werden m Marken der Sorte w erzeugt). Zugleich wird die Bestandsverwaltung über den Eingang der Waren informiert: mit $f(w)$ wird der Artikel der Ware w beschrieben, mit der Position $(f(w), m)$ zusätzlich auch die Anzahl.

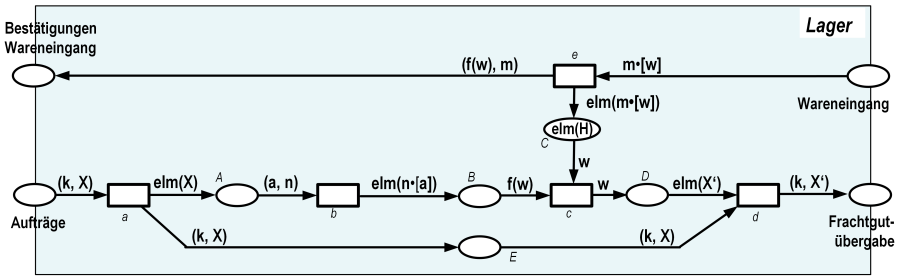
Der Platz C, das eigentliche Warenlager, enthält anfangs die Elemente der Multimenge H, also drei Paar Schuhe, einen Hut und kein Hemd. Das entspricht der Anfangsmarkierung $elm(G)$ des Platzes A der Bestandsverwaltung.

20. Das Modell des Lieferanten

Abbildung 33 zeigt, wie der Lieferant eine Lieferantenbestellung empfängt und eine Ware w heraussucht, die dem bestellten Artikel $f(w)$ entspricht. Dann transportiert der Lieferant ein Paket mit p Exemplaren dieser Ware in den Wareneingang des Lagers (Marke $p \bullet [w]$). Im Modell kann der Lieferant immer liefern; hier besteht natürlich die Möglichkeit, andere Annahmen zu modellieren.

21. Das Modell der Speditionen

Das Modell der Speditionen in Abbildung 34 ist offensichtlich: Es gibt eine Grundsorte R für Speditionen. Der Platz A enthält anfangs einige Speditio-



Dieses Modul hat zwei Aufgaben: 1. Gemäß der von der Auftragsverwaltung eingehenden Aufträge packt dieses Modul Teillieferungen und übergibt sie als Frachtgut den Speditionen. 2. Das Modul nimmt vom Lieferanten Waren entgegen und meldet dies an die Bestandsverwaltung.

Für die erste Teilaufgabe speichert die Transition a jede eingehende Bestellung (k, X) im Platz E und zusätzlich jede Artikelposition der Artikelliste X als Marke in Platz A.

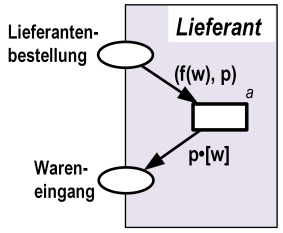
Transition b erzeugt aus einer Artikelposition (a, n) für jeden Artikel a, von dem n Exemplare bestellt wurden, n Marken der Form „a“, und legt sie in Platz B ab.

Für jeden Artikel a in B sucht die Transition c eine „passende“ Ware w im Lager C und legt sie auf Platz D. Welche Waren w zu einem Artikel a in B „passen“, beschreibt die Funktion f, wie im Modul Lieferant erläutert.

Transition d bündelt diese Waren in eine Lieferung und übergibt die Lieferung den Spediteuren.

Für die zweite Teilaufgabe empfängt die Transition e im Wareneingang eine Lieferung mit m Exemplaren der Ware w und sortiert die einzelnen Exemplare in das Lager C ein. Zugleich bestätigt e der Bestandsverwaltung den Eingang der Waren.

Abb. 32: Das Lager packt Frachtpakete und bestätigt die vom Lieferanten gesendeten Waren.



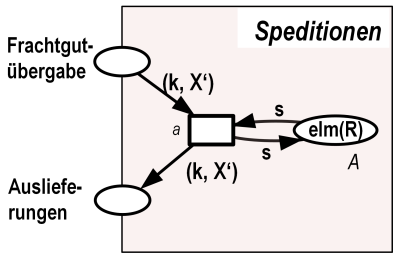
Transition a: Der Lieferant empfängt von der Bestandsverwaltung die Bestellung von p Exemplaren eines Artikels a und liefert p Exemplare einer „passenden“ Ware w. Welche Waren w zum Artikel a „passen“, beschreibt die Funktion f: Eine Ware w „passt“ zum Artikel a, wenn f(w) =a.

Zu einem Artikel a können verschiedene Waren w passen. Diese Waren sind vielleicht verschieden verpackt, haben verschiedene Kontroll-Nummern oder ähnliches.

Abb. 33: Der Lieferant schickt zu jeder Bestellung entsprechende Waren.

nen. Über die Variable r wird eine Spedition ausgewählt. Dieses Modul bietet einen Anknüpfungspunkt für weitere Annahmen zur Auswahl der Speditionen.

Damit sind nun alle Module der Fallstudie modelliert. Die Modelle können komponiert werden, wie in Abbildung 35. Dabei entsteht das Modell des Handelsbetriebes für die Struktur S.



Transition a wählt eine Spedition s, die vom Lager ein Paket übernimmt und es an den Kunden k ausliefert. Einzelheiten dazu werden nicht dargestellt.

Abb. 34: Eine Spedition liefert Frachtpakete aus.

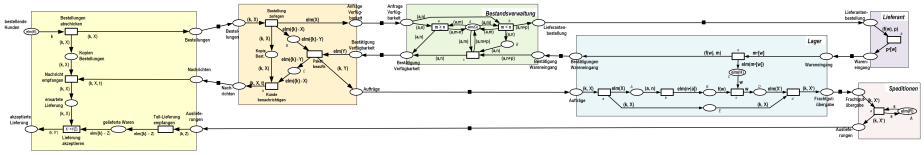


Abb. 35: Modell des Handelsbetriebes, komponiert aus sechs Modulen

VII Das Modell der Fallstudie

Im Modell von Abbildung 35 sind die Grundsorten und Grundfunktionen, also die Kunden, Waren, anfangs verfügbaren Waren, Zuordnung von Artikeln zu Waren und die Speditionen fest vorgegeben. Interessant wäre ein Modell, das alle möglichen Grundsorten und Grundfunktionen umfasst. Zudem sollte das Modell selbst rein *symbolisch* sein, also nicht wie die Modelle in Abschnitt VI mit konkreten Mengen und Funktionen arbeiten, sondern mit Symbolen für Mengen und Funktionen. Dann kann das Modell mit Software-Werkzeugen verarbeitet werden.

Hier können wir wiederum bewährte Konzepte der Prädikatenlogik und vieler Spezifikationssprachen aufgreifen: Wir konstruieren eine *Signatur* Σ für den Handelsbetrieb. Die Grundidee dabei ist einfach: Für jede Grundsorte, jede abgeleitete Sorte, jede Konstante und jede Funktion enthält Σ ein Symbol. Eine *Instanziierung* der Signatur ordnet jedem solchen Symbol eine passende Menge oder Funktion zu. Abbildung 36 zeigt eine solche Signatur. Eine Instanziierung ordnet dann jedem Symbol eine dem Typ entsprechende Menge, ein Objekt oder eine Funktion zu. Die Struktur S ist eine Instanziierung der Signatur Σ.

Zur Beschreibung des Verhaltens der Instanziierungen können wir die Netze aus Teil VI verwenden, so wie sie sind. Denn diese Netze verwenden keine spezifischen Aspekte der Instanziierung S; beispielsweise ist die Anfangsmarkierung von *bestellende Kunden* nicht mit Ute und Max beschriftet, sondern

Grundsorten
KN *Kunden*
AR *Artikel*
WA *Waren*
TE *Termine*
SP *Speditionen*

Abgeleitete Sorten
AP = AR $\times \mathbb{N}$ *Artikelpositionen*
AL = *M*(AP) *Artikellisten*
AM = *M*(AR) *Artikelmengen*
WM = *M*(WA) *Warenmengen*

Konstanten-Symbole
p1, p2: AP *bestellte Artikelpositionen*
K: P(KN) *bestellende Kunden*
G : AL *anfangs gelistete Artikel*
H : WL *anfangs verfügbare Waren*
R : P(SP) *verfügbare Speditionen*

Funktions-Symbole
(_‘): AP $\longrightarrow$ **AM**
(_‘): AL $\longrightarrow$ **AM**
f : WA $\longrightarrow$ **AR**
f : WL $\longrightarrow$ **AM**

Variablen
k: KN
x: AR
X,Y: AL
Z: M(WA)
t: TE
w: WA
s: SP
m, n, p: $\mathbb{N}$

Eigenschaften

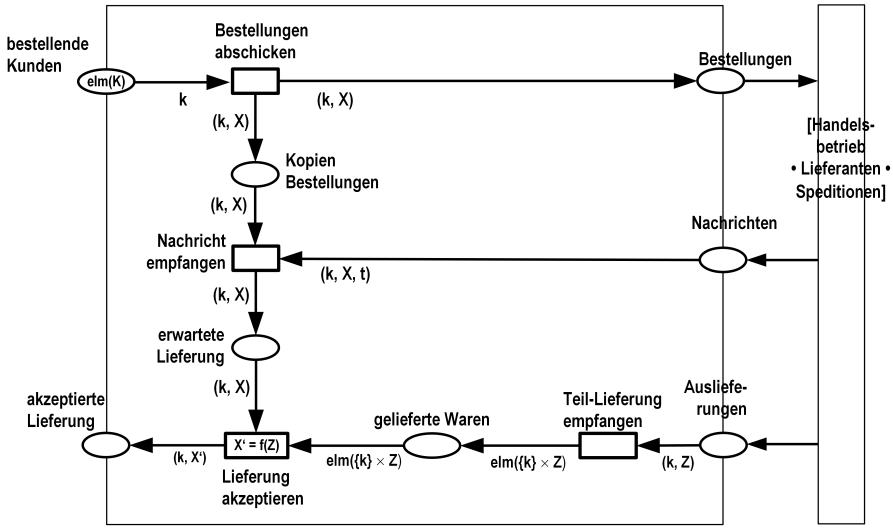
(a,n)‘ = n[a] für (a,n) $\in$ **AP**
[p1, … , pn]‘ = p1‘ + … + pn‘ für
[p1, … , pn] $\in$ **AL**

G‘ = f(H)

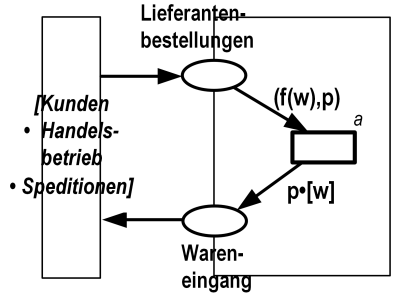
Abb. 36: Signatur Σ

mit $elm(K)$. Die Instanziierung S ergibt dann Ute und Max als Anfangsmarkierung von *bestellende Kunden*. Etwas schwieriger ist die allgemeine Formulierung des Zusammenhangs zwischen gespeicherten Positionen verfügbarer Artikel in der Bestandsverwaltung und den verfügbaren Waren im Lager, mit einer frei wählbaren Funktion f, die beschreibt, welche unterschiedlichen Waren einen bestellten Artikelwunsch erfüllen.

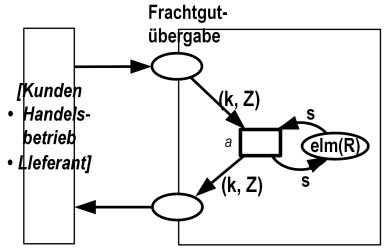
Die Konzepte und Notationen sind in Anlehnung an die Spezifikationssprache Z gewählt. Z bietet Konzepte zur Komposition, Import et cetera von Spezifikationen. Die sehr rudimentären Konzepte zur Beschreibung dynamischen Verhaltens ersetzt HERAKLIT mit seinen ausgefeilten Verhaltensmodulen. Während Prädikatenlogik über Eigenschaften von Strukturen redet, beschreibt HERAKLIT die dynamische Änderung von Strukturen, insbesondere der Extension von



(a) Sicht der Kunden



(b) Sicht des Lieferanten



(c) Sicht der Speditionen

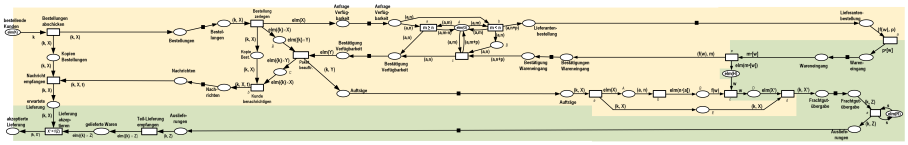
Abb. 37: Sichten der Geschäftspartner auf ihre jeweilige Umgebung

Prädikaten. Damit ist HERAKLIT die natürliche dynamische Ergänzung der statischen Prädikatenlogik.

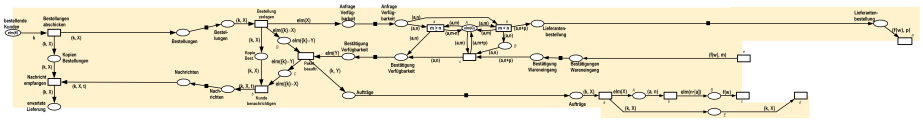
VIII Weitere Sichten auf den Handelsbetrieb

22. Abstrakte Umgebungen für Verhaltensmodelle

Es gibt eine Reihe weiterer Sichten auf den Handelsbetrieb mit seinen Partnern. Zum Verhaltensmodul jedes Partners gibt es ein abstraktes Modul, seine Umgebung, so dass die Komposition beider Module das Gesamtmodul ergibt. Abbildung 37 zeigt das Verhalten der drei Geschäftspartner des Handelsbetriebes in ihrer jeweiligen Umgebung.



(a) Fluss der Bestellungen und Waren



(b) Modul *Bestellungen*

(c) Modul $Warenfluss$

Abb. 38: Module der Bestellungen und des Warenfluss

23. Module für Bestellungen und Warenfluss

Abbildung 35 komponiert die sechs Module, die am Handelsunternehmen beteiligt sind, und beschreibt so das Verhalten des Gesamtmoduls in strukturierter Form: die Komponenten bleiben erkennnbar. Man kann das Gesamtmodul aber auch anders strukturieren. Abbildung 38a separiert jenen Teil des Modells, in dem Waren, also reale Gegenstände, bewegt werden, von dem Teil, der abstrakte Daten der Bestellungen organisiert. Diese Trennung führt zu den beiden Modulen in Abbildung 38b und 38c. Ihre Komposition ergibt wieder das Gesamtmodul aus Abbildung 38a. Es gilt also:

$$Gesamtverhalten = Bestellungen \bullet Warenfluss.$$

Anhang

24. Theoretische, konzeptionelle und methodische Grundlagen

Im zentralen Konzept eines Moduls kombiniert HERAKLIT drei bewährte, intuitiv leicht verständliche und mathematisch gesicherte Techniken, die auch bisher schon zur Spezifikation von Systemen verwendet werden:

1. *Abstrakte Datentypen* und *Algebraische Spezifikationen zur Formulierung konkreter und abstrakter Daten*: seit den 1970er Jahren werden solche Spezifikationen verwendet, in Spezifikationssprachen eingebaut, und oft zur (domänenspezifischen) Modellierung verwendet. Dazu gehören beispielsweise die

älteren Sprachen *VDM* und *Z* [9], [2], die reichhaltige Sprache *RAISE* [6], [1] und viele weitere Sprachen. [10] versuchen, mit der „*Common Algebraic Specification Language*“ CASL, den Kern des Gebietes zu standardisieren. Sehr systematisch stellt das Buch [15] die theoretischen Grundlagen und einige Anwendungsgebiete Algebraischer Spezifikationen dar. In diesen Zusammenhang gehören auch *Abstract State Machines* (ASM) [7]: Ausgehend von einer Signatur Σ gibt es dort globale Zustände; jeder Zustand ist eine Instanziierung von Σ. Ein Schritt ist typischerweise eine bedingte Wertzuweisung an eine Variable oder an einen Term, dessen Wert auf diese Weise aktualisiert wird. Ganz ähnlich beschreibt HERAKLIT Zustände, allerdings *lokale* Zustandskomponenten, mit instanziierten Grundsorte. Für eine Sorte A oder eine Funktion f einer Instanziierung ist dann eine lokale Zustandskomponente eine Teilmenge oder ein einzelnes Element von A oder ein Tupel der Form $(a, f(a))$.

2. *Petrinetze zur Formulierung dynamischen Verhaltens*: HERAKLIT nutzt die zentralen Ideen von Petrinetzen: Ein Schritt eines Systems, insbesondere eines großen Systems, hat *lokal beschränkte* Ursachen und Wirkungen. Damit kann man Abläufe beschreiben, ohne globale Zustände und global wirksame Schritte verwenden zu müssen. Dieses Konzept aus den frühen 1960er Jahren [11] wurde Anfang der 1980er Jahre prädikatenlogisch und mit „*gefärbten Marken*“ verallgemeinert [5], [8]. Den Zusammenhang mit Algebraischen Spezifikationen stellt [12] her. HERAKLIT ergänzt diese Sichtweise um zwei entscheidende Aspekte: uninterpretierte Konstantensymbole für Mengen in Plätzen, die mit der elm-Notation Instanziierungen mit vielen möglichen Anfangsmarkierungen fassen, und die elm-Notation in Pfeilbeschriftungen, um flexiblen Markenfluss zu beschreiben.
3. *Der Composition Calculus zur Strukturierung großer Systeme*: dieses Kalkül mit seinem breit verwendbaren assoziativen Kompositionsoperator ist der jüngste Beitrag zu den Fundamenten von HERAKLIT. Die naheliegende und in der Literatur oft diskutierte Idee, Komposition als Verschmelzung der Schnittstellen-Elemente von Modulen zu modellieren, wird ergänzt um die Unterscheidung linker und rechter Schnittstellen-Elemente und die Komposition $A \bullet B$ als Verschmelzung rechter Schnittstellen-Elemente von A mit linken Schnittstellen-Elementen von B. Nach [13] ist diese Komposition assoziativ (im Gegensatz zur naiven Verschmelzung von Schnittstellen-Elementen); sie hat zudem eine Reihe weiterer nützlicher Eigenschaften. Insbesondere ist diese Komposition verträglich mit Verfeinerung sowie Vergröberung und mit einzelnen Abläufen. Weitere Einzelheiten und Anwendungsbeispiele diskutieren [14] und [3].

Diese drei theoretischen Grundlagen harmonieren miteinander und generieren weitere „best practice“-Konzepte, die zu einem methodischen Vorgehen der Modellierung mit HERAKLIT beitragen, und die in diesem Beitrag nur gestreift werden. Dazu gehören insbesondere *Adapter*: Ein Adapter für zwei Module A und B formuliert Kriterien und Eigenschaften für die Komposition von A und B als weiteres Modul C, so dass $A \bullet C \bullet B$ die gewünschten Eigenschaften garan-

tieren. Weitere Einzelheiten dazu werden in [14] sowie im HERAKLIT-Handbuch beschrieben. Die methodische systematische Verwendung der Komposition von Abstraktion und von einzelnen Abläufen wird in zukünftigen Fallstudien deutlich.

Glossar

Ablauf, verteilt

Ein verteilter Ablauf ist ein Netz von Ereignissen.

Daten (-element)

Daten kann man beispielsweise in einem Katalog darstellen, digital versenden, auf Papier drucken et cetera. Insbesondere kann man sie in digitaler Form rechnergestützt verarbeiten. Wir benutzen das Wort nur im Plural und Datenelement für den Singular, um Verwechselungen mit einer Kalenderangabe zu vermeiden.

Ereignis

Ein Ereignis synchronisiert das Erreichen und Verlassen bestimmter lokaler Zustände.

Gate

Die Oberfläche eines Moduls besteht aus einer Menge von Gates, die jeweils ein Label tragen.

Gegenstand

Etwas aus der realen Welt, auch Ding, allgemeiner Objekt.

Instanziierung

Eine Instanziierung einer Signatur ordnet jedem Symbol der Signatur eine passende Menge oder Funktion zu.

Label

Ein Label bezeichnet die Beschriftung eines Gates.

Modul

Zentrales HERAKLIT-Konzept, um Teile eines Systems zu beschreiben. Ein Modul hat ein *Inneres* und eine *Oberfläche*.

Modul, Ereignis-

Ein Modul, das nur aus einer einzelnen Transitionen im Inneren besteht, und aus Plätzen seiner Schnittstelle.

Modul, abstrakt

Ein Modul, dessen Inneres nur aus seinem Namen besteht.

Objekt

Ein Objekt ist ein Gegenstand oder ein Datenelement.

Prädikat

Ein Prädikat trifft auf ein Element einer Menge zu oder nicht zu.

Schema

Das Gemeinsame verschiedener verteilter Abläufe wird in einem Schema ausgedrückt. Dafür wird eine Signatur mit verschiedenen Variablen verwendet.

Signatur Σ

Eine Zusammenstellung von Symbolen für strukturierte Mengen und Funktionen. Eine *Instanziierung* ordnet jedem Symbol eine entsprechende Menge oder Funktion zu und erzeugt so eine Struktur.

Spezifikation, algebraisch

Die algebraische Spezifikation dient zur Formulierung konkreter und abstrakter Daten.

Struktur, (Tarski-)

Eine Zusammenstellung von Typen von Daten und Gegenständen, und darauf aufbauend von Mengen, Tupeln und Funktionen.

System

Allgemeiner Gegenstand der Realität.

Variable

Variablen können mit konkreten Objekten belegt werden.

Verhalten

Allgemeine Bezeichnung von Veränderungen in einem System.

Zustand, lokal

Ein lokaler Zustand beschreibt eine Menge von Prädikaten, die für ein Objekt erfüllt sind.

Literatur

1. Bjørner, D. : *Domain Science & Engineering: A Foundation for Software Development*. Unpublished manuscript (2020)
2. Bowen, J. P.: Z: A formal specification notation. In: *Software specification methods*. Springer, 2001, S. 3–19
3. Fettke, P. ; Resig, W. : Modelling service-oriented systems and cloud services with Heraklit. In: *CoRR* abs/2009.14040 (2020). `http://arxiv.org/abs/2009.14040`. – presented at the 16th International Workshop on Engineering Service-Oriented Applications and Cloud Services, Heraklion, Greece, September 28-30, 2020
4. Fettke, P. ; Resig, W. : Modellieren mit Heraklit – Handbuch (in Vorbereitung). (2021)
5. Genrich, H. J. ; Lautenbach, K. : The Analysis of Distributed Systems by Means of Predicate/Transition-Nets. In: Kahn, G. (Hrsg.): *Semantics of Concurrent Computation, Proceedings of the International Symposium, Evian, France, July 2-4, 1979* Bd. 70, Springer, 1979, S. 123–147
6. George, C. : The NDB Database Specified in the RAISE Specification Language. In: *Formal Asp. Comput.* 4 (1992), Nr. 1, S. 48–75
7. Gurevich, Y. : Evolving algebras 1993: Lipari guide. In: Börger, E. (Hrsg.): *Specification and validation methods*. Oxford University Press, 1993, S. 9–36
8. Jensen, K. : *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use - Volume 1*. Springer, 1992
9. Jones, C. B.: *Systematic software development using VDM*. 2. Prentice Hall, 1991
10. Mossakowski, T. ; Haxthausen, A. E. ; Sannella, D. ; Tarlecki, A. : Casl - The Common Algebraic Specification Language: Semantics and Proof Theory. In: *Comput. Artif. Intell.* 22 (2003), Nr. 3-4, S. 285–321
11. Petri, C. A.: *Kommunikation mit Automaten*, Technische Hochschule Darmstadt, Diss., 1962
12. Reisig, W. : Petri Nets and Algebraic Specifications. In: *Theor. Comput. Sci.* 80 (1991), Nr. 1, S. 1–34
13. Reisig, W. : Associative composition of components with double-sided interfaces. In: *Acta Informatica* 56 (2019), Nr. 3, 229–253. `http://dx.doi.org/10.1007/s00236-018-0328-7`. – DOI 10.1007/s00236–018–0328–7
14. Reisig, W. : Composition of component models - a key to construct big systems. (2020). – in press
15. Sannella, D. ; Tarlecki, A. : *Foundations of Algebraic Specification and Formal Software Development*. Springer, 2012