Graphische Repräsentation von Interaktionsausdrücken

Christian Heinlein, Abt. DBIS

Juni 1997

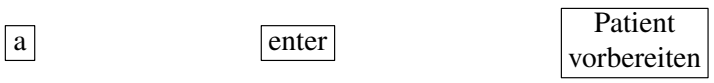
1. Einleitung

Dieser Bericht stellt eine einfache graphische Notation für *Interaktionsausdrücke* vor, wie sie in den Berichten „Grundlagen von Interaktionsausdrücken“ (UIB 97-09) und “Interaction Expressions – A Powerful Formalism for Describing Inter-Workflow Dependencies” (UIB 97-04) beschrieben sind.

2. Graphische Repräsentation

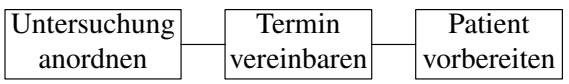
2.1 Aktionen

Aktionen werden als *Rechtecke* dargestellt, die mit dem Namen der Aktion beschriftet werden:

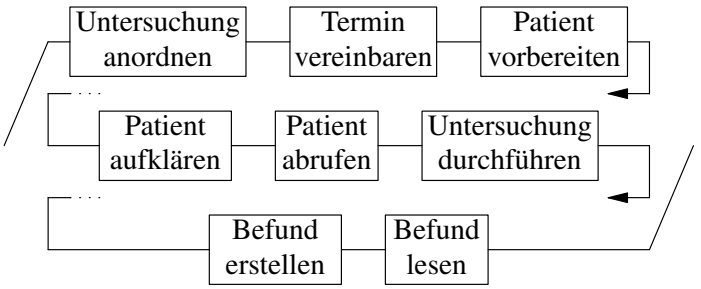


2.2 Sequentielle Komposition

Die Komponenten einer sequentiellen Komposition werden grundsätzlich *von links nach rechts* angeordnet und durch *waagrechte Linien* (evtl. auch Pfeile) verbunden:

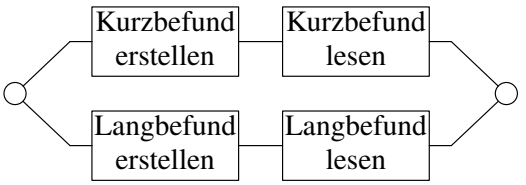


Für den Fall, daß der horizontal zur Verfügung stehende Platz für eine lange Sequenz nicht ausreicht, können an beliebigen Stellen „Zeilenumbrüche“ vorgenommen werden:



2.3 Verzweigungen

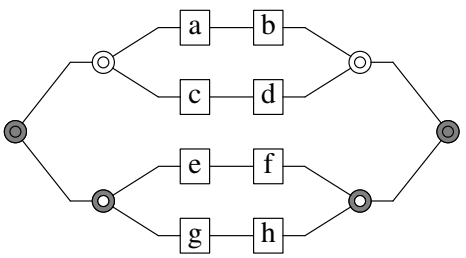
Die Komponenten einer *Verzweigung*, d. h. einer parallelen Komposition, einer Selektion, einer Konjunktion oder einer Synchronisation, werden grundsätzlich *von oben nach unten* angeordnet und mit einem *Verzweigungssymbol* zur linken und einem *Vereinigungssymbol* zur rechten verbunden:



Die Verzweigungs- bzw. Vereinigungssymbole sind hierbei wie folgt definiert:

- ○ für Selektion,
- ◎ für parallele Komposition,
- ◉ für Synchronisation und
- ◉ für Konjunktion.

Ein *einzelner* Kreis ○ (Selektion) besagt also, daß genau *ein* Zweig der Verzweigung zu wählen ist, während ein Doppelkreis (alle übrigen Symbole) besagt, daß *alle* Zweige zu „beschreiten“ sind. Der Grad der „Schwärzung“ des Symbols korrespondiert hierbei mit dem „Kopplungsgrad“ der Zweige: ◎ (parallele Komposition) besagt, daß die Zweige vollkommen unabhängig voneinander ausgeführt werden können, während ◉ (Synchronisation) eine teilweise und ◉ (Konjunktion) eine vollständige Abhängigkeit der Zweige voneinander symbolisiert (vgl. die Definition der Operatoren im Grundlagenbericht). Die Darstellung

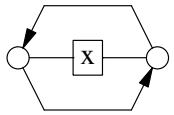


entspricht also dem Ausdruck

$$(a - b + c - d) \;\&\; (e - f \;@\; g - h).$$

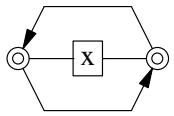
2.4 Iterationen

Stellt man sich die Ausführung eines Ausdrucks als ein Durchlaufen des zugehörigen Grahpen vor, so kann eine sequentielle Iteration der Gestalt *x* * wie folgt mit Hilfe von Selektionen dargestellt werden:



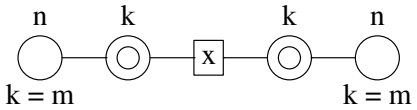
Beim linken ○-Knoten hat man die Wahl, den „Schleifenkörper“ *x* auszuführen oder aber die Schleife komplett zu überspringen. Hat man *x* ausgeführt, so hat man beim rechten ○-Knoten die Wahl, zurückzuspringen und *x* erneut auszuführen oder aber die Schleife zu beenden und im Graphen fortzufahren. (Prinzipiell hat man natürlich auch die Möglichkeit, ständig zwischen den beiden ○-Knoten hin- und herzuspringen.)

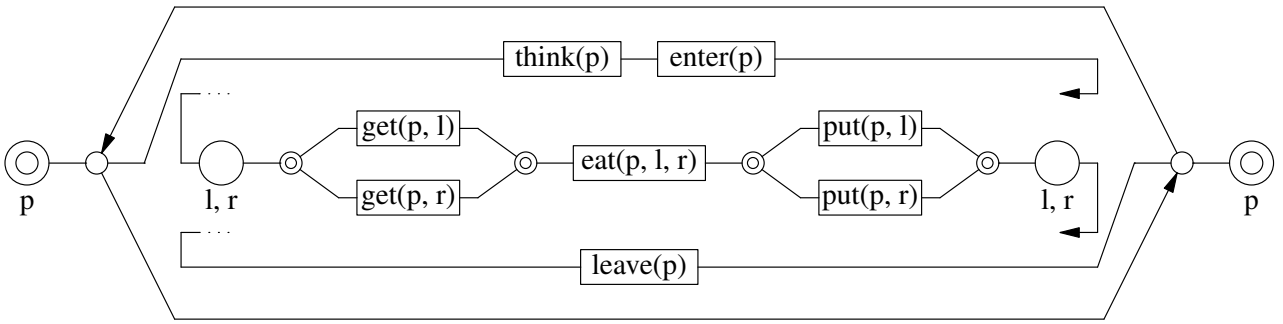
Die parallele Iteration läßt sich intuitiv zwar nicht so direkt interpretieren wie ihr sequentielles Pendant, aus Gründen der „optischen Analogie“ wird sie jedoch einfach wie folgt dargestellt:



2.5 Multiplikatoren und Quantoren

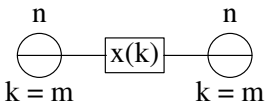
Ganz analog zur formalen Notation werden Multiplikatoren und Quantoren mit denselben Symbolen dargestellt wie die zugrundeliegenden binären Operatoren, die lediglich etwas vergrößert und mit den gewünschten Bereichsgrenzen versehen werden. Aus Symmetriegründen werden die Bereichsgrenzen sowohl beim „Verzweigungs-“ als auch beim „Vereinigungssymbol“ angegeben, was außerdem das Erkennen zusammengehörender Symbolpaare erleichtert:





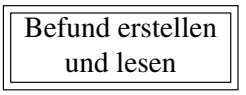
Sequentielle Komposition

Da die sequentielle Komposition in ihrer Grundform nicht mit Hilfe von Verzweigungs- und Vereinigungssymbolen dargestellt wird, ist die soeben eingeführte Multiplikator-Darstellung für sie nicht anwendbar. Aus diesem Grund wird die folgende Ersatz-Darstellung verwendet:

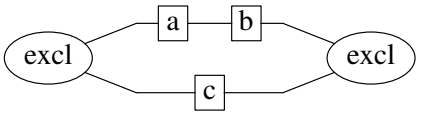


2.6 Makros und benutzerdefinierte Operatoren

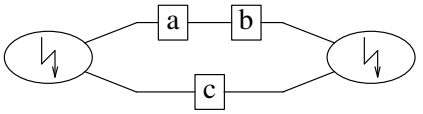
Makros werden typischerweise auf zwei unterschiedliche Arten verwendet: Ein parameterloses Makro dient i. d. R. als Abkürzung für einen an anderer Stelle definierten komplexen Ausdruck, stellt also im Prinzip nur eine „komplexe“ Aktion dar. Aus diesem Grund wird der Aufruf eines solchen Makros, ähnlich wie eine Aktion, als Rechteck dargestellt, allerdings mit einem doppelten Rahmen, der die „Komplexität“ andeuten soll:



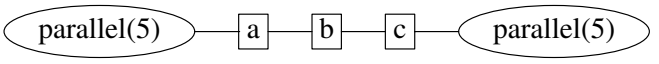
Makros (oder auch benuzterdefinierte Operatoren), deren Parameter (bzw. Operanden) Ausdrücke sind, stellen i. d. R. eine (komplexe) Verknüpfung dieser Ausdrücke dar und werden daher analog zu (vordefinierten) Operatoren wie folgt dargestellt:



Als Verzweigungs- und Vereinigungssymbol wird also jeweils eine Ellipse verwendet, die mit dem Namen des Makros bzw. Operators beschriftet ist. Anstelle eines Namens kann in einer graphischen Darstellung natürlich auch ein sprechendes Symbol verwendet werden, wie z. B.:

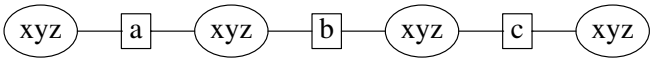


Parameter, die keine Ausdrücke darstellen (wie z. B. Bereichsgrenzen von Multiplikatoren), werden – ähnlich wie Parameter von Aktionen – in einer textuellen Parameterliste des Makronamens angegeben:



Eindeutigkeit

Wenn ein Makro mit genau einem Parameter mehrfach in einem Ausdruck auftritt, ist die soeben eingeführte Darstellung von Makroaufrufen u. U. nicht eindeutig. Beispielsweise kann die Darstellung



entweder als sequentielle Komposition

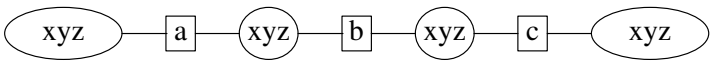
$$xyz(a) - b - xyz(c)$$

oder aber als Verschachtelung von Makroaufrufen

$xyz(a - xyz(b) - c)$

interpretiert werden.

Um diese Mehrdeutigkeit zu beseitigen, vereinbaren wir, daß im Falle verschachtelter Makroaufrufe die äußeren Ellipsen etwas breiter gezeichnet werden als die inneren:



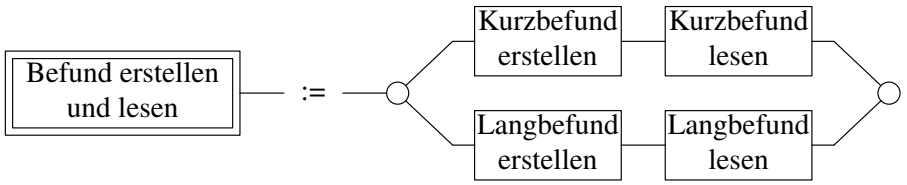
Das obige Bild, bei dem alle Ellipsen gleich groß sind, ist somit als sequentielle Komposition zu interpretieren.

Anmerkung

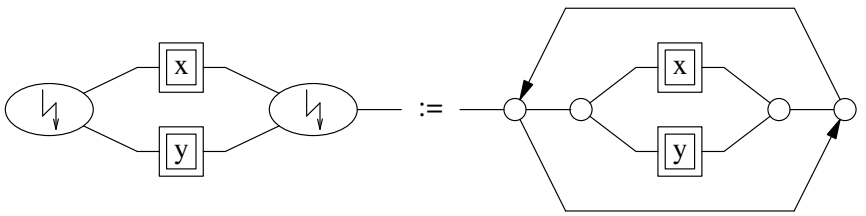
Prinzipiell existiert dieses Mehrdeutigkeitsproblem auch bei der Darstellung von Multiplikatoren und Quantoren, da diese optisch analog zu Makros mit einem Parameter dargestellt werden. Da Multiplikatoren/Quantoren jedoch mit ihrem Indexbereich/Parameter dargestellt werden, tritt das Problem nur dann wirklich auf, wenn ein innerer Multiplikator/Quantor denselben Indexbereich/Parameter hat wie ein äußerer. Obwohl eine solche Konstellation prinzipiell zulässig ist (der innere Index/Parameter verbirgt dann, ähnlich wie bei blockorientierten Programmiersprachen, den äußeren), tritt sie in praktischen Anwendungen wohl nie auf, weil der resultierende Ausdruck nur sehr schwer verständlich ist.

2.7 Definition von Makros und Operatoren

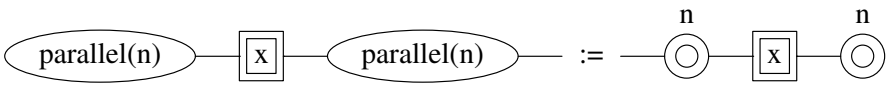
Ein parameterloses Makro wird im Prinzip durch eine „Zuweisung“ eines beliebigen Ausdrucks an einen Bezeichner definiert:



Ein Makro (oder Operator) mit Parametern wird analog zu seinem Aufruf definiert. Da seine Parameter im Prinzip Platzhalter für beliebige (d. h. komplexe) Ausdrücke sind, werden sie auch wieder als „komplexe Aktionen“ dargestellt:



Parameter, die keine Ausdrücke sind, werden wieder in einer textuellen Parameterliste angegeben:



2.8 Anmerkung

Die vorgestellte Notation ist für eine graphische Darstellungsform relativ strikt: Komponenten einer sequentiellen Komposition werden grundsätzlich von links nach rechts, Komponenten der übrigen binären Operatoren grundsätzlich von oben nach unten angeordnet. Auf den ersten Blick mag dies als unnötige Einschränkung der „künstlerischen Freiheit begabter Designer“ erscheinen, auf den zweiten Blick hat eine solche „Reglementierung“ jedoch einige Vorteile gegenüber den sonst üblichen „Freistil-Darstellungen“, bei denen Linien, Pfeile etc. in beliebigen Richtungen („kreuz und quer“) gezogen werden dürfen:

- Die resultierenden Darstellungen wirken wesentlich homogener und sind dadurch leichter visuell erfaßbar als unstrukturierte Graphiken. Dies gilt insbesondere, wenn „Autor“ und „Leser“ einer Graphik verschiedene Personen sind.
  Zum Vergleich betrachte man die Vorteile von strukturierter Programmierung gegenüber „Spaghetti-Code“ bzw. von Struktogrammen (Nassi-Shneiderman-Diagrammen) gegenüber Programmablaufplänen (Flußdiagrammen).
- Die manuelle Erstellung von Graphiken kann durch einen syntaxgesteuerten Editor unterstützt werden, der diese „Layout-Policy“ implementiert. Dadurch kann sich der „Autor“ einer Graphik voll auf deren Inhalt konzentrieren, weil er nicht ständig mit der „Optimierung“ der äußeren Form beschäftigt ist (die ohnehin nach jeder Änderung wieder neu vorzunehmen ist).
- Mit Hilfe eines „Graphik-Beschreibungsprogramms“ (wie z. B. *Pic* für Troff oder TEX/$LATEX$), bei dem eine Graphik nicht interaktiv, sondern durch eine programmiersprachenähnliche Beschreibung erstellt wird, können Graphiken relativ leicht automatisch generiert werden.
  Mit Hilfe geeigneter Makrodefinitionen kann das erste Beispiel aus Abschnitt 2.3 z. B. wie folgt „gezeichnet“ werden:[1]

```
disj(
    seqc(
        act("Kurzbefund" "erstellen", 10)
        act("Kurzbefund" "lesen", 10)
    )
    seqc(
        act("Langbefund" "erstellen", 10)
        act("Langbefund" "lesen", 10)
    )
)
```

  Mit Hilfe eines Parsers für Interaktionsausdrücke (den man für eine Implementierung natürlich ohnehin braucht) kann prinzipiell auch diese Graphikbeschreibung selbst wieder aus dem zugrundeliegenden Ausdruck generiert werden.

[1] `disj` steht für Disjunktion/Selektion, `seqc` für sequentielle Komposition und `act` für Aktivität.
Da das Graphikprogramm als Präprozessor des eigentlichen Textverarbeitungsprogramms nichts über die Breite von Texten weiß, muß deren ungefähre Abmessung jeweils als Parameter angegeben werden; ein guter Schätzwert hierfür ist die Anzahl der Zeichen, die intern mit der durchschnittlichen Breite eines Zeichens multipliziert wird.

3. Ausblick

Wie bereits erwähnt, kann die vorgestellte graphische Notation mittels eines *syntaxgesteuerten Editors* implementiert werden. Um bei der Erfassung eines Ausdrucks sicherzustellen, daß die verwendeten Aktionsnamen in der „umgebenden Welt“ auch definiert sind, ist der Anschluß an ein entsprechendes *Repository* denkbar. Außerdem könnte es sinnvoll sein, wenn die so erstellte graphische Darstellung eines Ausdrucks am Bildschirm *animiert* werden kann, damit ein weniger geübter Benutzer die Möglichkeit hat, verschiedene Ausführungsreihenfolgen interaktiv „durchzuspielen“ und so ein Gefühl für die Wirkungsweise des Ausdrucks zu erhalten.