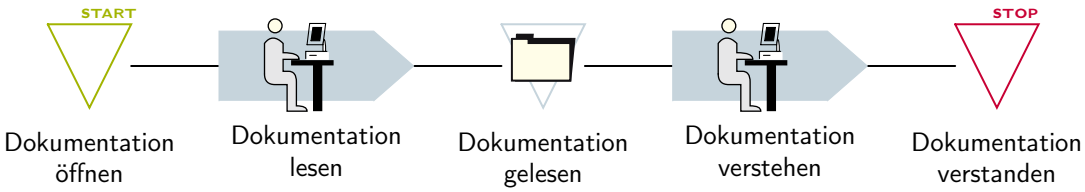
PGF / TikZ Erweiterungspaket `memoorgml`

Fabian Schneider (fabian.schneider@studium.fernuni-hagen.de)

1. Juni 2016



1 Einführung

Dieses Erweiterungspaket für *PGF* bzw. *TikZ* dient der Erstellung von *MEMO OrgML* Kontrollfluss- und Dekompositionsdiagrammen in TEX bzw. LATEX Dokumenten [1].
Das Paket stellt dabei Shapes für *TikZ*-Knoten bereit, welche die Notationssymbole der *MEMO OrgML* Sprachkonzepte abbilden [2].

Zur Verwendung des Pakets muss zuerst *TikZ* über den `usepackage`-Befehl geladen werden: `\usepackage{tikz}`, anschließend kann dann das Erweiterungspaket geladen werden: `\usetikzlibrary{memoorgml}`.

Das Erweiterungspaket besteht aus drei Dateien:

- `pgflibrarymemoorgmlshapes.code.tex`: Diese Bibliotheksdatei enthält die Definition aller benötigter Shapes.
- `tikzlibrarymemoorgmlstyles.code.tex`: Diese Bibliotheksdatei enthält die vordefinierten Styles zur Formatierung der Shapes.
- `tikzlibrarymemoorgml.code.tex`: Diese Bibliotheksdatei stellt Makros zur Unterstützung der Diagrammerstellung bereit.

Das Paket wurde gegen die TEX-Live Distributionen 2015 und 2016 getestet.

2 Installation

Das Erweiterungspaket kann von `https://github.com/Lehrstuhl-BWL-EvIS/TikZ` heruntergeladen werden. Das heruntergeladene Verzeichnis enthält neben den eigentlichen Bibliotheksdateien noch ein `Makefile` zur Installation der des Erweiterungspaketes unter Mac OS X und Linux.

Zur Installation des Erweiterungspaketes unter `Mac OS X` und `Linux` muss GNU Make installiert sein. Das `Makefile` muss dann als Super-User mit folgendem Befehl ausgeführt werden:

```
1       sudo make
```

Das `Makefile` selbst hat nur ein Target und kopiert die Bibliotheksdateien mittels `install` in das entsprechende Verzeichnis. Anschließend wird der `texmf`-Tree aktualisiert.

3 Bereitgestellte Notationssymbole

Das Erweiterungspaket stellt zum aktuellen Zeitpunkt die Shapes für eine Untermenge der Notationssymbole der *MEMO OrgML* bereit. Alle bereitgestellten Shapes sind in der folgen Tabelle aufgeführt:

Konzept	Bezeichnung Notationssymbol	Suffix
Bassisshape Ereignis	event	-
Startereginis	startevent	event
Endereignis	endevent	event
Nachricht eingegegangen	messageevent	event
Relevante Änderung des Informationszustandes	informationchangeevent	event
Basisshape zeitliches Ereignis	timeevent	event
Zeit. Ereignis - Zeitpunkt erreicht	pointintimeevent	event
Zeit. Ereignis - Zeitdauer erreicht	timespanevent	event
Änderung - Neu	newevent	event
Änderung - Modifiziert	modifiedevent	event
Änderung - Abgebrochen	canceledevent	event
Basisshape Softwarebanchrichtigung	softwareevent	event
Softwarebenachrichtigung - Publish	publishsoftwareevent	event
Softwarebenachrichtigung - Poll	pollsoftwareevent	event
Geschäftsprozess Prozess	businessprocess	process
Unspezifizierter Prozess	unspecifiedprocess	process
Manueller Prozess	manualprocess	process
Automatisierter Prozess	automatedprocess	process
Computergestützter Prozess	computersupportedprocess	process
Dekomponierbarer Prozess	decompositionprocess	process
Extern ausgeführter Prozess	externalprocess	process

Basisshape Synchronisation	sync	-
Synchronisation nach Beendigung aller Prozess	andsync	sync
Synchronisation nach Beendigung des ersten Prozesses	orsync	sync
Start der Parallelisierung	paraconn	-
Start einer Iteration	startiteratoin	-
Verschlungene Iterationspfeile	iteration	-
Nicht spezifizierte Ausnahme	unspecifiedexception	exception
Manuelle Ausnahme	manualexception	exception
Zeit abgelaufen	timeexpiredexception	exception
Technischer Fehler	technicalexception	exception
Automatische Ausnahme	automaticexception	exception
Natürlichsprachliche Ausnahme	naturallanguageexception	exception
Nicht spezifizierte Entscheidung	unspecifieddecision	decision
Manuelle Entscheidung	manualdecision	decision
Maschinelle Entscheidung	automaticdecision	decision
Manuelle Entscheidung mit eindeutigen Regeln	manualexplictdecision	decision
Plussymbol Dekomposition	decompositionplus	-

Das Suffix definiert zu welcher Gruppe von Notationssymbolen es gehört. Beispielsweise enden alle Shapes für Ereignisse mit dem Suffix `event`.

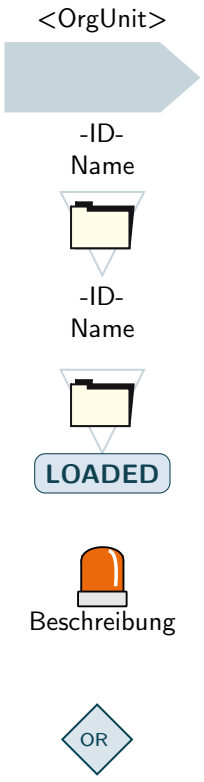
4 Makros

Das Erweiterungspaket stellt eine Reihe von Makros bereit, die den Benutzer bei der Erstellung der Diagramme unterstützen:

`\memoprocess` ⟨*Shape*⟩⟨*Name*⟩⟨*x-Pos.*⟩⟨*y-Pos.*⟩⟨*Org.-Einheit*⟩⟨*ID*⟩⟨*Bezeichnung*⟩
`\memoevent` ⟨*Shape*⟩⟨*Name*⟩⟨*x-Pos.*⟩⟨*y-Pos.*⟩⟨*ID*⟩⟨*Bezeichnung*⟩
`\memoeventloaded` ⟨*Shape*⟩⟨*Name*⟩⟨*x-Pos.*⟩⟨*y-Pos.*⟩
`\memoexception` ⟨*Shape*⟩⟨*Name*⟩⟨*x-Pos.*⟩⟨*y-Pos.*⟩⟨*Beschreibung*⟩

Diese Makros erzeugen einen Knoten mit dem übergebenen Shape an der definierten Position. Den Makros muss im Parameter ⟨*Shape*⟩ der Name des Shapes ohne das Suffix angegeben werden. Dieser wird vom Makro angehängt um sicherzustellen, dass nur die passenden Shapes für das jeweilige Makro verwendet werden. Die so erzeugten Knoten bzw. Notationssymbole können dann über den im Parameter ⟨*Name*⟩ vergebenen Namen referenziert werden.

Abbildung 1: Beispiel: Erstellung von (Teil-) Prozess-, Ereignis- und Ausnahmenotationssymbolen.




```
1 \begin{tikzpicture}
2 \memoprocess{unspecified}{p1
    }{0}{0}{<OrgUnit>}{-ID-}{Name
    };
3 \end{tikzpicture}
```

```
1 \begin{tikzpicture}
2 \memoevent{informationchange}{e1
    }{0}{0}{-ID-}{Name};
3 \end{tikzpicture}
```

```
1 \begin{tikzpicture}
2 \memoeventloaded{
    informationchange}{e1}{0}{0};
3 \end{tikzpicture}
```

```
1 \begin{tikzpicture}
2 \memoexception{unspecified}{ex1
    }{0}{0}{Beschreibung};
3 \end{tikzpicture}
```

```
1 \begin{tikzpicture}
2 \memosync{or}{sy1}{0}{0};
3 \end{tikzpicture}
```

Die Parameter für die textuellen Komponenten des Notationssymbols, ⟨*Org.-Einheit*⟩,⟨*ID*⟩, ⟨*Bezeichnung*⟩, sind dabei optional was bedeutet, dass sie leer übergeben werden können.

`\memoconn` ⟨*Name linker Knoten*⟩⟨*Name rechter Knoten*⟩

Dieses Makro erzeugt mittels des `\draw`-Befehls eine gerade Kante zwischen den beiden übergebenen Knoten. Die Knoten bzw. deren Position wird dabei über die Namen ermittelt, welche mit den zuvor vorgestellten Makros vergeben wurden.

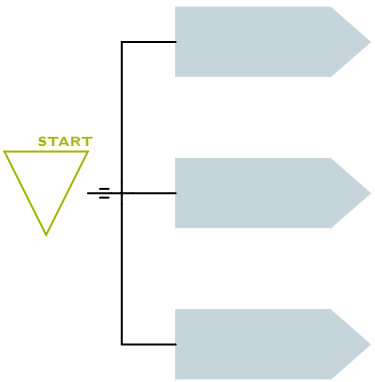
Abbildung 2: Beispiel: Erzeugung einer geraden Kante.


```
1  \begin{tikzpicture}
2  \memoevent{start}{s1}{0}{0}{}{};
3  \memoprocess{unspecified}{p1
      }{2.5}{0}{}{}{};
4  \memoevent{end}{e1}{4.8}{0}{}{};
5
6  \memoconn{s1}{p1};
7  \memoconn{p1}{e1};
8  \end{tikzpicture}
```

`\memoparaconn` ⟨*Name Ereignis*⟩⟨*Name Prozess*⟩⟨*Anker*⟩
`\memoconnsync` ⟨*Name Ereignis*⟩⟨*Name Verküpfung*⟩⟨*Anker*⟩

Diese Makros dienen der Erstellung der Kontrollstrukturen für die *parallele Ausführung* von Prozessen und die *Synchronistation nach Beendigung aller Prozesse* bzw. *Synchronisation nach Beendigung des ersten Prozess.*
Das Makro `\memoconnpara` erzeugt eine rechtwinklige Kante zwischen dem übergebenen Knoten für das Ereignis von dem die Parallelisierung ausgeht und dem übergebenen Knoten für den Prozess. Der Knoten, der den Beginn der Parallelisierung darstellt, wird automatisch vor dem Ereignisknoten erzeugt. Der Anker gibt dabei an, von welcher Position aus die Kante aus dem Prallelisierungs-Knoten ausgeht. Sinnvolle Anker: `east`, `south`, `north`
Dieses Makro muss für jeden Knoten, der einen (Teil-) Prozess darstellt, und Teil der Parallelisierung sein soll aufgerufen werden.

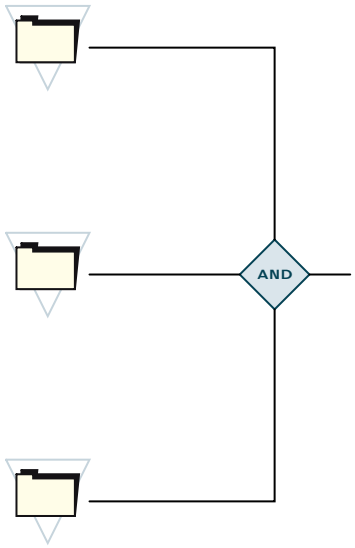
Abbildung 3: Beispiel: Parallele Ausführung von drei Prozessen.




```
1  \begin{tikzpicture}
2  \memoevent{start}{s1}{0}{0}{}{};
3
4  \memoprocess{unspecified}{u1
      }{3}{0}{}{}{};
5  \memoprocess{unspecified}{u2
      }{3}{2}{}{}{};
6  \memoprocess{unspecified}{u3
      }{3}{-2}{}{}{};
7
8  \memoconnpara{s1}{u1}{east};
9  \memoconnpara{s1}{u2}{north};
10 \memoconnpara{s1}{u3}{south};
11 \end{tikzpicture}
```

Das Makro \memoconnsync erzeugt eine rechtwinklige Kante zwischen einem der Knoten die das Ende, der Parallelisierung darstellen und zwischen dem mit dem Makro \memosync erzeugten Knoten für das Notationssymbol, dass die logische Verknüpfungsoperation darstellt. Der übergebene Anker definiert die Position der eingehenden Kante des Knotens, der die logische Verknüpfungsoperation darstellt. Sinnvolle Anker: `west`, `south`, `north`.
Die Knoten für die Ereignisse und Prozesse, die Teil der Parallelisierung sind, müssen zuvor mit den eingangs genannten Makros erstellt worden sein.

Abbildung 4: Beispiel: Synchronisation parallel ausgeführter Prozesse.

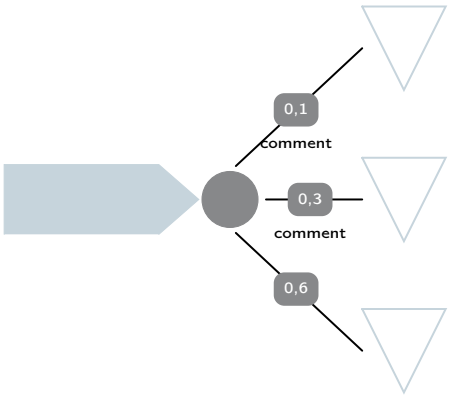



```
1  \begin{tikzpicture}
2  \memoevent{informationchange}{i1
       }{0}{0}{}{};
3  \memoevent{informationchange}{i2
       }{0}{3}{}{};
4  \memoevent{informationchange}{i3
       }{0}{-3}{}{};
5
6  \memosync{and}{a1}{3}{0};
7
8  \memoconnsync{i1}{a1}{west};
9  \memoconnsync{i2}{a1}{north};
10 \memoconnsync{i3}{a1}{south};
11 \end{tikzpicture}
```

`\memoconnxor` ⟨*Shape Entscheidungstyp*⟩⟨*Name Prozess*⟩⟨*Name Ereignis*⟩⟨*Anker*⟩
`\memoconnxorprobability` ⟨*Shape Entscheidungstyp*⟩⟨*Name Prozess*⟩⟨*Name Ereignis*⟩⟨*Anker*⟩ ⟨*Wahrscheinlichkeit*⟩⟨*Kommentar*⟩

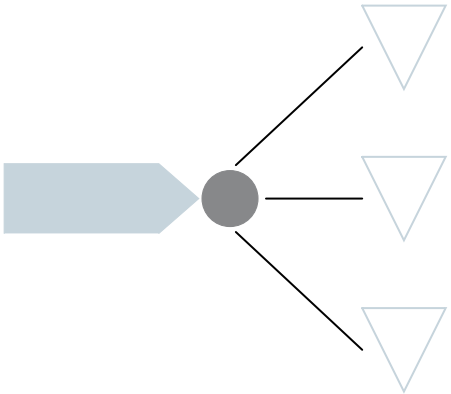
Mit diesen Makros können die Kontrollstrukturen *<Prozess> produziert alternatives <Ereignis> mit Wahrscheinlichkeiten (exklusives ODER)* und *<Prozess> produziert alternatives <Ereignis> ohne Wahrscheinlichkeiten (exklusives ODER)* erzeugt werden. Die beiden Makros erzeugen vor dem Knoten, der den (Teil-) Prozess darstellt, einen Knoten mit dem Shape des jeweiligen Entscheidungstyps. Weiterhin erzeugt das Makro ausgehend von dem Knoten, der den Entscheidungstyp darstellt, eine gerade Kante zum übergebenen Knoten, der das Ereignis darstellt. Der übergebene Anker definiert die Position der ausgehenden Kante des Knotens des Entscheidungstyps. Sinnvolle Anker: `east`, `south`, `north`.
Dem Makro \memoconnxorprobability können zusätzlich noch die Parameter für die Angabe der Wahrscheinlichkeit ($0.1-0.9$) sowie einen Kommentar, der bei der Wahrscheinlichkeit angezeigt wird, übergeben werden. Die Darstellung der Wahrscheinlichkeit erfolgt als Knoten auf der Kante zwischen dem Knoten für den Entscheidungstyp und dem Knoten für das Ereignis. Der Kommentar ist als Label implementiert.

Abbildung 5: Beispiel: Produktion eines alternativen Ereignisses mit Wahrscheinlichkeit.




```
\memoprocess{unspecified}{p2
    }{0}{0}{}{}{};

\memoevent{}{e4}{4}{2}{}{};
\memoevent{}{e5}{4}{-2}{}{};
\memoevent{}{e6}{4}{0}{}{};

\memoconnxorprobability{
    unspecified}{p2}{e4}{north
    }{0,1}{comment};
\memoconnxorprobability{
    unspecified}{p2}{e5}{south
    }{0,6}{};
\memoconnxorprobability{
    unspecified}{p2}{e6}{east
    }{0,3}{comment};
```

Abbildung 6: Beispiel: Produktion eines alternativen Ereignisses ohne Wahrscheinlichkeit.

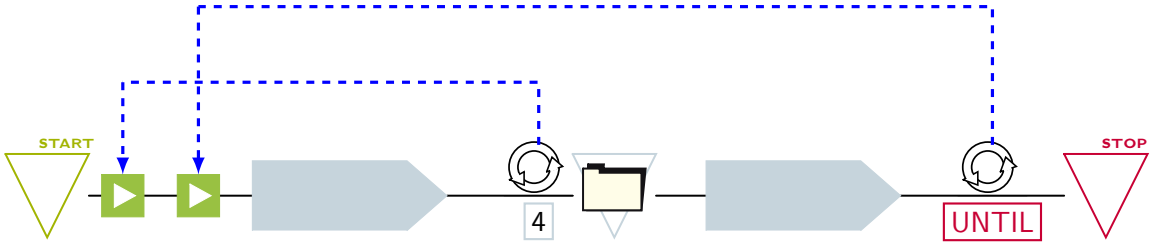


```
\memoprocess{unspecified}{p1
    }{0}{0}{}{}{};

\memoevent{}{e1}{4}{2}{}{};
\memoevent{}{e2}{4}{2}{}{};
\memoevent{}{e3}{4}{0}{}{};

\memoconnxor{unspecified}{p1}{e1
    }{north};
\memoconnxor{unspecified}{p1}{e2
    }{south};
\memoconnxor{unspecified}{p1}{e3
    }{east};
```

`\memoiterationuntil`⟨*Name Startereignis*⟩⟨*Name Endereignis*⟩⟨*Höhe Verbindungslinie*⟩
⟨*Abstand Startereginis*⟩⟨*Abstand Endereignis*⟩
`\memoiterationloop` ⟨*Name Startereignis*⟩⟨*Name Endereignis*⟩⟨*Höhe Verbindungslinie*⟩
⟨*Abstand Startereginis*⟩⟨*Abstand Endereignis*⟩⟨*Anzahl Iterationen*⟩

Diese Makros dienen zur Implementierung von Iterationen mit *beliebiger Wiederholung* und einer *n-fachen Wiederholung (For-Loop)*. Die beiden Makros erzeugen jeweils nach dem Knoten, der das Startereignis der Iteration darstellt und vor dem Knoten der das Endereignis darstellt, jeweils die Knoten mit den Shapes `iterationstart` und `iteration`. Der Abstand nach dem Startereignis und vor dem Endereignis wird über die Parameter ⟨*Abstand Startereignis*⟩ und ⟨*Abstand Endereignis*⟩ definiert. Der Parameter ⟨*Höhe der Verbindungslinie*⟩ definiert die absolute Höhe des blauen Verbindungspfeiles. Dies wurde variabel gestaltet, um auch verschachtelte Itertionen zu ermöglichen ohne, dass die Kanten sich überlappen.
Mit dem Parameter ⟨*Anzahl Iterationen*⟩ des Makros kann die Anzahl der Iterationen definiert werden. Diese wird als Label unterhalb des Knotens, der die verschlungenen Pfeile darstellt, erstellt.

Abbildung 7: Beispiel: Iterationen



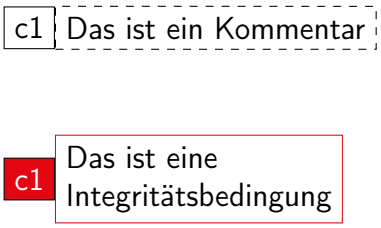

```
1  \memoevent{start}{e1}{0}{0}{}{};
2  \memoprocess{unspecified}{p1}{4}{0}{}{}{};
3  \memoevent{informationchange}{e2}{7.5}{0}{}{};
4  \memoprocess{unspecified}{p2}{10}{0}{}{}{};
5  \memoevent{end}{e3}{14}{0}{}{};
6
7  \memoiterationloop{e1}{e2}{1.5}{1}{1}{4};
8  \memoiterationuntil{e1}{e3}{2.5}{2}{1.5};
9
10 \memoconn{e1}{p1};
11 \memoconn{p1}{e2};
12 \memoconn{e2}{p2};
13 \memoconn{p2}{e3};
```

`memocomment` ⟨*Nummer des Kommentars*⟩⟨*Text*⟩⟨*x-Pos.*⟩⟨*y.-Pos.*⟩
`memoconstraint` ⟨*Nummer der Integritätsbed.*⟩⟨*Text*⟩⟨*x-Pos.*⟩⟨*y.-Pos.*⟩

Mit Hilfe dieser Makros können *natürlichsprachlichen Kommentaren* und *natürlichsprachlichen Integritätsbedingungen* erzeugt werden.
Die beiden Makros erzeugen einen neuen Knoten an der angegebenen Position, der

als Knotentext den String *Cn* enthält, wobei *n* die laufende Nummer des Kommentars bzw. der Integritätsbedingung ist. Der Kommentar- bzw. Bedingungstext wird als Label links neben dem Konten dargestellt.

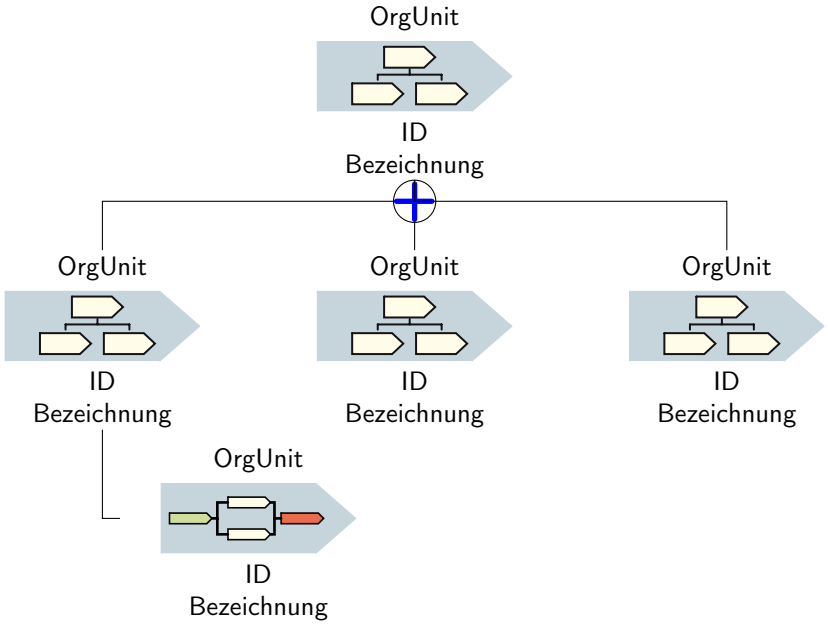
Abbildung 8: Beispiel: Kommentare und Integritätsbedinungen.




```
1 \begin{tikzpicture}
2 \memocomment{1}{Das ist ein
     Kommentar}{0}{0};
3 \memoconstraint{1}{Das ist eine
     \\Integritaetsbedingung
     }{0}{-3};
4 \end{tikzpicture}
```

`\memodecomproot` ⟨*Shape*⟩⟨*Name*⟩⟨*x-Pos.*⟩⟨*y-Pos.*⟩⟨*Org.Einheit*⟩⟨*ID*⟩⟨*Bezeichnung*⟩
`memodecompchild` ⟨*Name*⟩⟨*Org.Einheit*⟩⟨*ID*⟩⟨*Bezeichnung*⟩

Diese Makros dienen der Erstellung von Dekompositionsdiagrammen. Die Erstellung der Dekompositionsdiagramme basiert dabei auf dem `child`-Befehl, mit dem Bäume aufgespannt werden können. Mit dem Makro `\memodecomproot` wird der Wurzelknoten des Baums definiert. Die Wurzel des Baumes kann dabei wiederum an einer beliebigen Stelle des Diagramms platziert werden. Mittels des `child`-Befehls werden dann die Kinknoten unterhalb der Wurzel erzeugt. Der Befehl umschließt dabei das Makro `\memodecompchild`.




```
1  \begin{tikzpicture}[decompositiondiagramstyle]
2      \memodecomproot{decomposition}{d1}{0}{0}{OrgUnit}{ID}{
          Bezeichnung}
3          [edge from parent fork down]
4              child {\memodecompchild{decomposition}{OrgUnit}{
                  ID}{Bezeichnung}
5                  child[subprocess,firstsubprocess] {
                       \memodecompchild{business}{OrgUnit}{ID}{
                       Bezeichnung}}
6                  }
7                  child { \memodecompchild{decomposition}{
                       OrgUnit}{ID}{Bezeichnung}
8                  %~~ Einfuegen des Gabelungssymbols (
                       Pluszeichen)
9                  edge from parent node[shape=
                       decompositionplus, decompositionplus,
                       midway] {}}
10                 child { \memodecompchild{decomposition}{
                       OrgUnit}{ID}{Bezeichnung} };
11 \end{tikzpicture}
```

Die Formatierung des Diagramms muss dabei mit den entsprechenden Einstellungen Erfolgen. Folgendes Listing zeigt ein Beispiel:

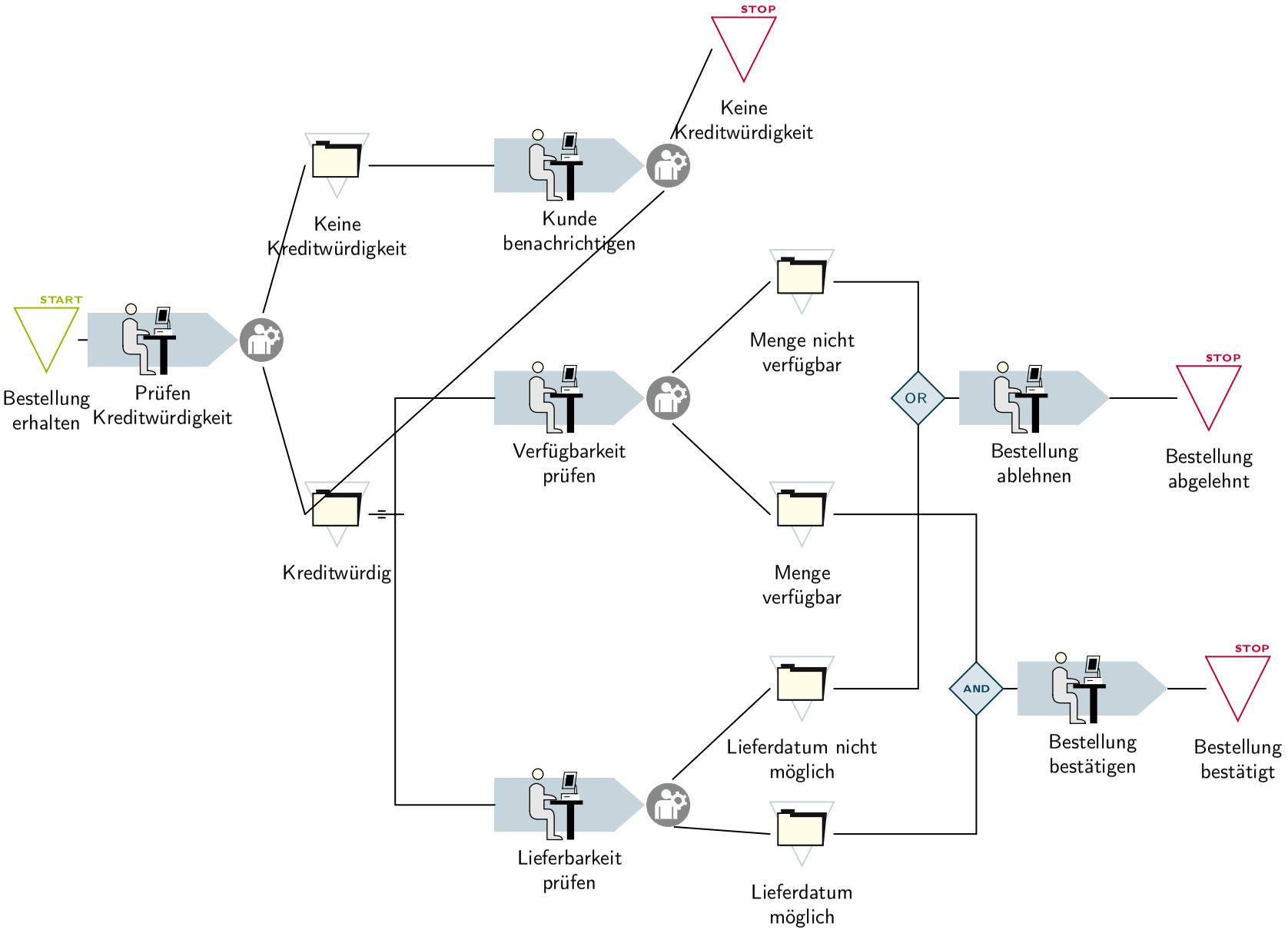
```
1  \tikzset{
2  decompositiondiagramstyle/.style={
3      edge from parent fork down,
4      level distance = 8em,
5      edge from parent/.style={draw, shorten <= 5ex, shorten
           >=3ex},
6          level 1/.style = {node distance=20em, sibling
               distance = 10em},
7          level 2/.style = {node distance=20em, sibling
               distance = 10em},
8          level 3/.style = {node distance=40em, sibling
               distance = 10em},
9          subprocess/.style={grow=down,xshift=5em,
10         edge from parent path={(\tikzparentnode.south) |- (
               \tikzchildnode.west)}},
11     firstsubprocess/.style={level distance=14ex}
12 }
13 }
```

5 Weitere Beispiele

Folgendes Beispiel stellt einen komplettes Kontrollflussdiagramm dar, dass auf [1, S. 92] basiert:

```
\begin{tikzpicture}
\memoevent{start}{start}{0}{0}{}{Bestellung\\erhalten};
\memoprocess{computersupported}{credit_check
   }{2}{0}{}{}{Pruefen\\Kreditwuerdigkeit};
\memoconn{start}{credit_check};

 \memoevent{informationchange}{i1}{5}{3}{}{Keine
    \\Kreditwuerdigkeit};
 \memoevent{informationchange}{i2}{5}{-3}{}{
    Kreditwuerdig};

 \memoconnxor{manualexplicit}{credit_check}{i1}{north};
 \memoconnxor{manualexplicit}{credit_check}{i2}{south};

 \memoprocess{computersupported}{inform_customer
    }{9}{3}{}{}{Kunde\\benachrichtigen};
 \memoevent{end}{end_inform}{12}{5}{}{Keine
    \\Kreditwuerdigkeit};
 \memoconn{i1}{inform_customer};

 \memoconnxor{manualexplicit}{inform_customer}{
    end_inform}{north};
 \memoconnxor{manualexplicit}{inform_customer}{i2}{
    south};

 \memoprocess{computersupported}{avail_check
    }{9}{-1}{}{}{Verfuegbarkeit\\pruefen};
 \memoprocess{computersupported}{deliv_check
    }{9}{-8}{}{}{Lieferbarkeit\\pruefen};

 \memoconnpara{i2}{avail_check}{north};
 \memoconnpara{i2}{deliv_check}{south};

 \memoevent{informationchange}{i3}{13}{1}{}{Menge nicht
    \\verfuegbar};
 \memoevent{informationchange}{i4}{13}{-3}{}{Menge
    \\verfuegbar};

 \memoconnxor{manualexplicit}{avail_check}{i3}{north};
 \memoconnxor{manualexplicit}{avail_check}{i4}{south};

 \memoevent{informationchange}{i5}{13}{-6}{}{
    Lieferdatum nicht\\moeglich};
 \memoevent{informationchange}{i6}{13}{-8.5}{}{
    Lieferdatum\\moeglich};

  \memoconnxor{manualexplicit}{deliv_check}{i5}{north};
 \memoconnxor{manualexplicit}{deliv_check}{i6}{south};

```

```
37        \memosync{or}{sync_or}{15}{-1};
38        \memosync{and}{sync_and}{13}{-6};
39
40        \memoconnsync{i3}{sync_or}{north};
41        \memoconnsync{i5}{sync_or}{south};
42
43        \memoconnsync{i4}{sync_and}{north};
44        \memoconnsync{i6}{sync_and}{south};
45
46        \memoprocess{computersupported}{deny_order
             }{17}{-1}{}{}{Bestellung\\ablehnen};
47        \memoprocess{computersupported}{confirm_order
             }{19}{-6}{}{}{Bestellung\\bestaetigen};
48        \memoconn{sync_or}{deny_order};
49        \memoconn{sync_and}{confirm_order};
50
51        \memoevent{end}{end_deny}{20}{-1}{}{Bestellung
             \\abgelehnt};
52        \memoevent{end}{end_confirm}{21}{-6}{}{Bestellung
             \\bestaetigt};
53
54          \memoconn{deny_order}{end_deny};
55        \memoconn{confirm_order}{end_confirm};
56  \end{tikzpicture}
```


START
Bestellung erhalten
Prüfen Kreditwürdigkeit
Keine Kreditwürdigkeit
Kunde benachrichtigen
STOP
Keine Kreditwürdigkeit
Kreditwürdig
Verfügbarkeit prüfen
Menge nicht verfügbar
Menge verfügbar
OR
Bestellung ablehnen
STOP
Bestellung abgelehnt
Lieferbarkeit prüfen
Lieferdatum nicht möglich
Lieferdatum möglich
AND
Bestellung bestätigen
STOP
Bestellung bestätigt

6 Erweiterung des Pakets

Um weitere Notationssymbole zu dem Erweiterungspaket hinzuzufügen muss die Datei `pgflibrarymemoorgmlshapes.code.tex` erweitert werden. Dort muss mit Hilfe des *PGF* Befehls `\pgfdeclareshape` ein neues Shape definiert werden. Eine Vorlage findet sich in [2, S. 1034]. Zu jedem Shape gehört auch ein Style, der den gleichen Namen wie das Shape trägt. Deshalb muss noch ein entsprechender Style in der Datei `tikzlibrarymemoorgmlstyles.code.tex` definiert werden.

Weitere Makros müssen in der Datei `tikzlibrarymemoorgml.code.tex` implementiert werden.

Literatur

[1] U. Frank. MEMO Organisation Modelling Language (2): Focus on Business Processes. Technical Report 49, Institut für Informatik und Wirtschaftsinformatik (ICB) - Universität Duisburg-Essen, 2011.

[2] T. Tantau. *TikZ & PGF Manual for Version 3.0.1a*. Institut für Theoretische Informatik, Universität zu Lübeck, 3 edition, 7 2015. `https://www.ctan.org/pkg/pgf?lang=de`.