(Abschnitt: BfArM-Daten – Umsteiger-Algorithmen)

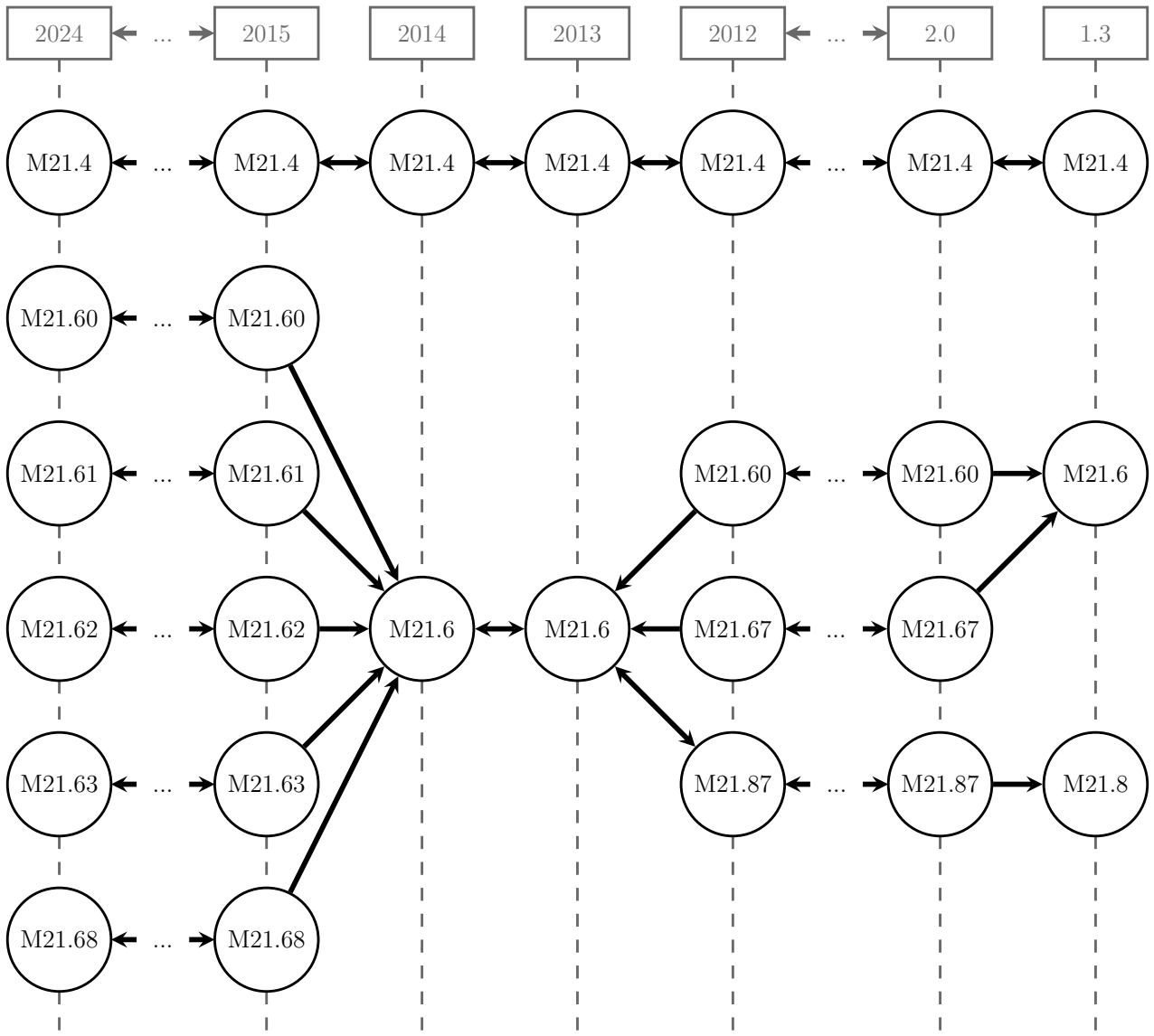


Abbildung 1: caption

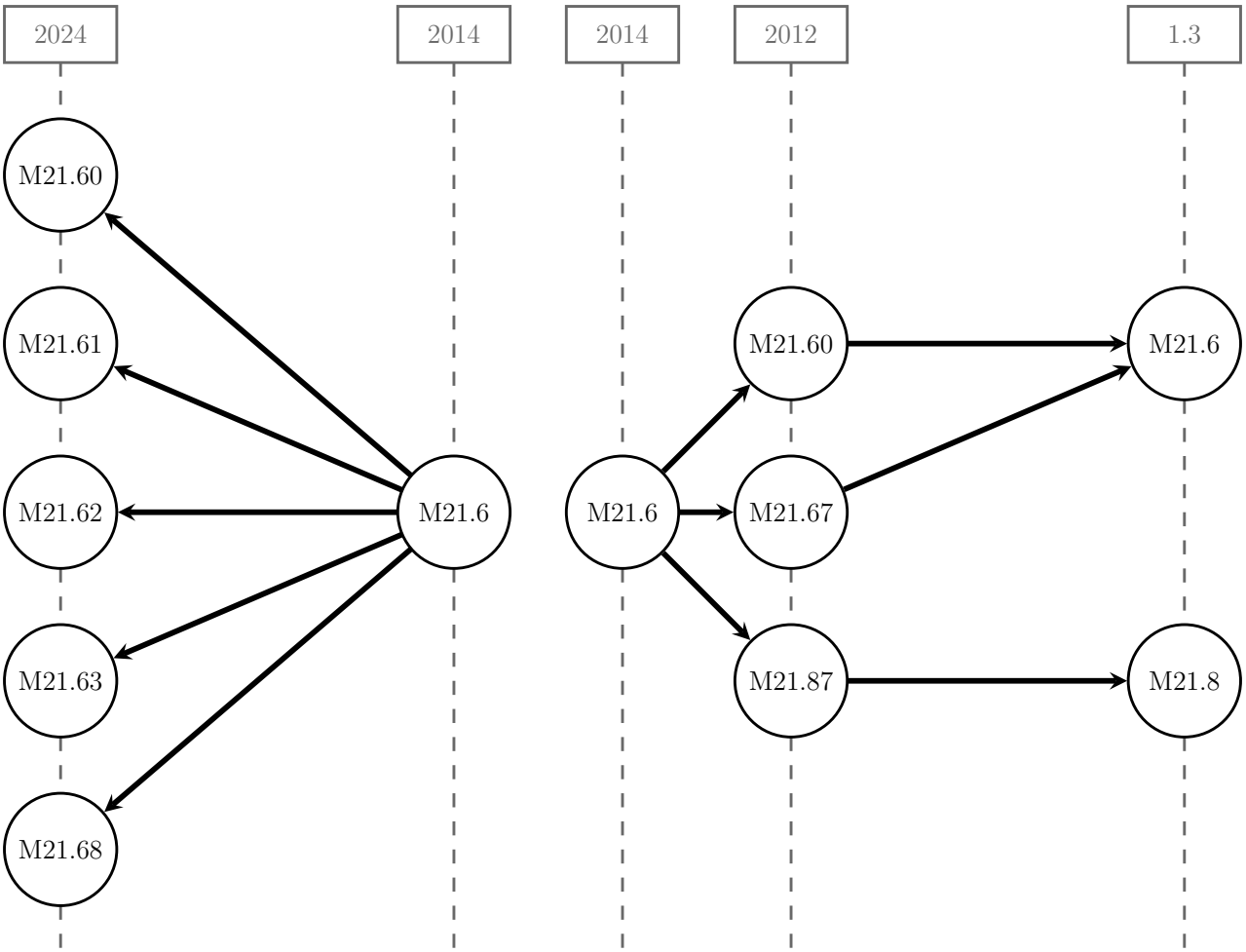


Abbildung 2: caption

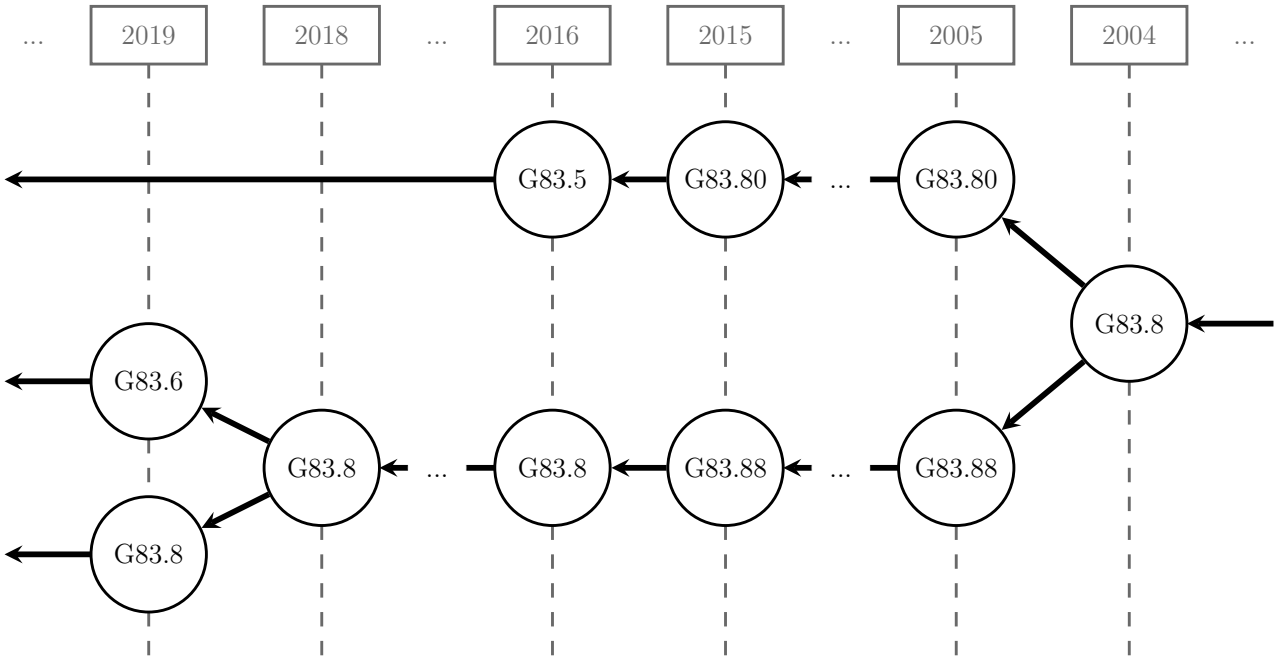


Abbildung 3: caption

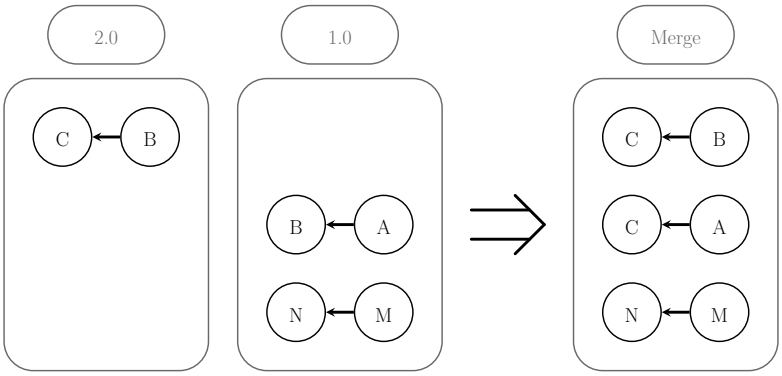


Abbildung 4: caption

D26: The ***transitive closure*** R^* of a binary relation R is the relation R^* defined by $(x, y) \in R^*$ if and only if there exists a sequence $x = v_0, v_1, v_2, \ldots, v_k = y$ such that $k \geq 1$ and $(v_i, v_{i+1}) \in R$, for $i = 0, 1, \ldots, k-1$. Equivalently, the transitive closure R^* of the relation R is the smallest transitive relation that contains R.

D27: Let G be the digraph representing a relation R. Then the digraph G^* representing the transitive closure R^* of R is called the ***transitive closure of the digraph*** G. Thus, an arc (x, y), $x \neq y$, is in the transitive closure G^* if and only if there is a directed x-y path in G. Similarly, there is a self-loop in digraph D^* at vertex x if and only if there is a directed cycle in digraph G that contains x.

Abbildung 5: aus (Gross et al., 2013, Seite 172)

E7: Suppose a relation R on the set $S = \{a, b, c, d\}$ is given by

$$\{(a, a), (a, b), (b, c), (c, b), (c, d)\}$$

Then the digraph G representing the relation R and the transitive closure G^* are as shown in Figure 3.1.7.

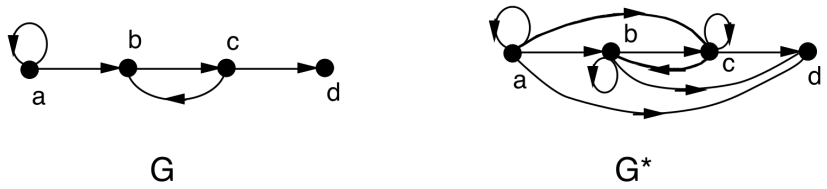


Abbildung 6: aus (Gross et al., 2013, Seite 172)

5.3.1 The Purdom Algorithm

Purdom, in [Pur70], made two key observations:

1. During the computation of transitive closure of a directed acyclic graph, if node $s \prec_i t$, then additions to the successor list of node s cannot affect the successor list of node t. One should therefore compute the successor list of t first and then that of s. By processing nodes in reverse topological order, one need add to a node only the successor lists of its immediate successors since the latter would already have been fully expanded. We call this idea the *immediate successor* optimization [AgJ90].

2. All nodes within a strongly connected component (SCC) in a graph have identical reachability properties, and the condensation graph obtained by collapsing all the nodes in each strongly connected component into a single node is acyclic.

Abbildung 7: aus (Dar, 1993, Seite 76f)

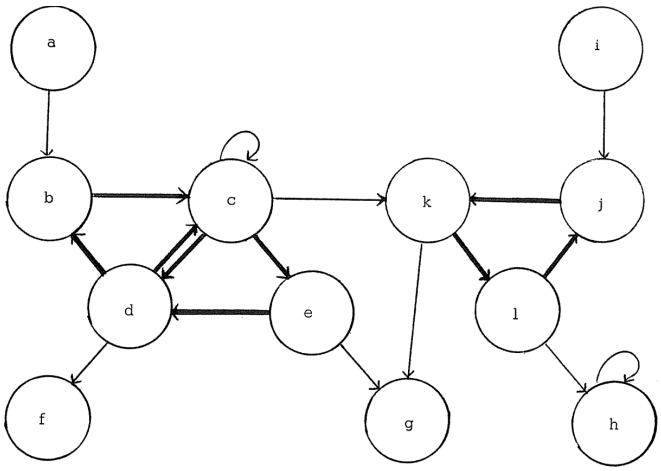


Abbildung 8: aus (Purdom, 1970, Seite 78)

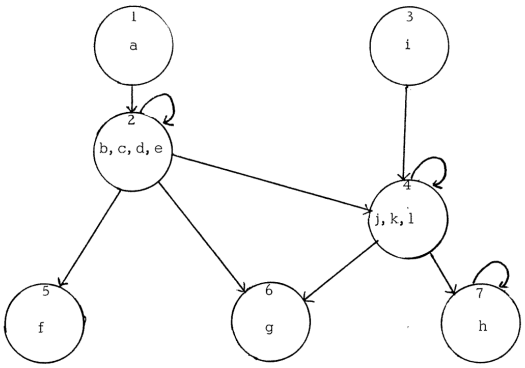


Abbildung 9: aus (Purdom, 1970, Seite 78)

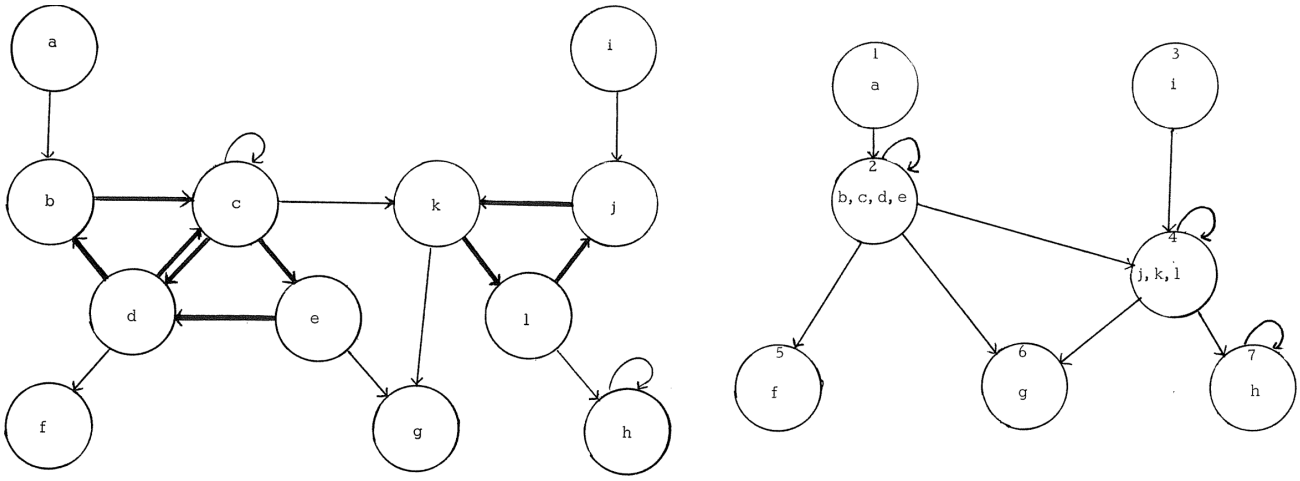


Abbildung 10: aus (Purdom, 1970, Seite 78)

2.2 Search

A second approach for computing the transitive closure is to search the graph n times, each time starting from a different node, thereby determining what can be reached from that node. In this approach, we completely disregard any parts of the TC-relation that have already been computed for other starting nodes. Instead, we search the entire part of the graph that is reachable from the starting node by following every edge we can get to.

Abbildung 11: aus (Jakobsson, 1991, Seite 200)

1 Allgemeines

Angenommen eine Funktion, die Daten liest:

```
lies_daten (system, version, typ, kode)
```

Beispiel:

```
lies_daten ('icd10gm', '2014', 'codes', 'M21.4')
lies_daten ('icd10gm', '2014', 'codes', 'M21.6')
```

Kode	Titel
M21.4	Plattfuß [Pes planus] (erworben)
M21.6	Sonstige erworbene Deformitäten des Knöchels und des Fußes

Analog dazu die Umsteiger, wobei sich diese immer von der angebebenen Version auf die nächstältere Version beziehen, zum Beispiel:

```
lies_daten ('icd10gm', '2014', 'umsteiger', 'M21.4')
lies_daten ('icd10gm', '2014', 'umsteiger', 'M21.6')
```

new (2014)	old (2013)	←Auto→	
M21.4	M21.4	A	A
M21.6	M21.6	A	A

"Auto" steht für die automatische Überleitbarkeit in die jeweilige Richtung, das heißt 2014←2013 und 2014→2013.

Insgesamt hat M21.4 über alle Versionen keine Umsteiger.

M21.6 hat folgende:

new (2015)	old (2014)	←Auto→	
M21.4	M21.4	A	A
M21.60	M21.6		A
M21.61	M21.6		A
M21.62	M21.6		A
M21.63	M21.6		A
M21.68	M21.6		A

new (2013)	old (2012)	←Auto→	
M21.4	M21.4	A	A
M21.6	M21.60	A	
M21.6	M21.67	A	
M21.6	M21.87	A	A

test1

new (2.0)	old (1.3)	←Auto→	
M21.4	M21.4	A	A
M21.60	M21.6		A
M21.67	M21.6		A
M21.80	M21.8		A
M21.81	M21.8		A
M21.82	M21.8		A
M21.83	M21.8		A
M21.84	M21.8		A
M21.85	M21.8		A
M21.86	M21.8		A
M21.87	M21.8		A
M21.88	M21.8		A
M21.89	M21.8	A	A

test2

Purdom einfach:

1. Ausgehend von einem Graphen G: Bestimme die stark zusammenhängenden Komponenten (SCC) in G und vereinige diese in jeweils einen einzelnen Knoten. Das ergibt einen zyklenfreien, verdichteten Graphen G_c.
2. Sortiere G_c topologisch.
3. Ermittele die transitive Hülle von G_c über dessen Adjazenzlisten in rückwärts topologischer Reihenfolge.
4. Bestimme die transitive Hülle des ursprünglichen Graphens G über die Nachbarschaftslisten der stark zusammenhängenen Komponenten in G_c: ein Knoten y ist ein erreichbarer Nachbar von Knoten x, falls $SCC(y) = SCC(x)$ oder falls $SCC(y)$ ein erreichbarer Nachbar von $SCC(x)$ ist.

2 Horizontale Suche

data['key'] = 'value'

```
'key' => 'value'
```

test

Parameter:

iPar { an int parameter with the meaning described here}

local Variables:

iVar {an int variable with the meaning described here}

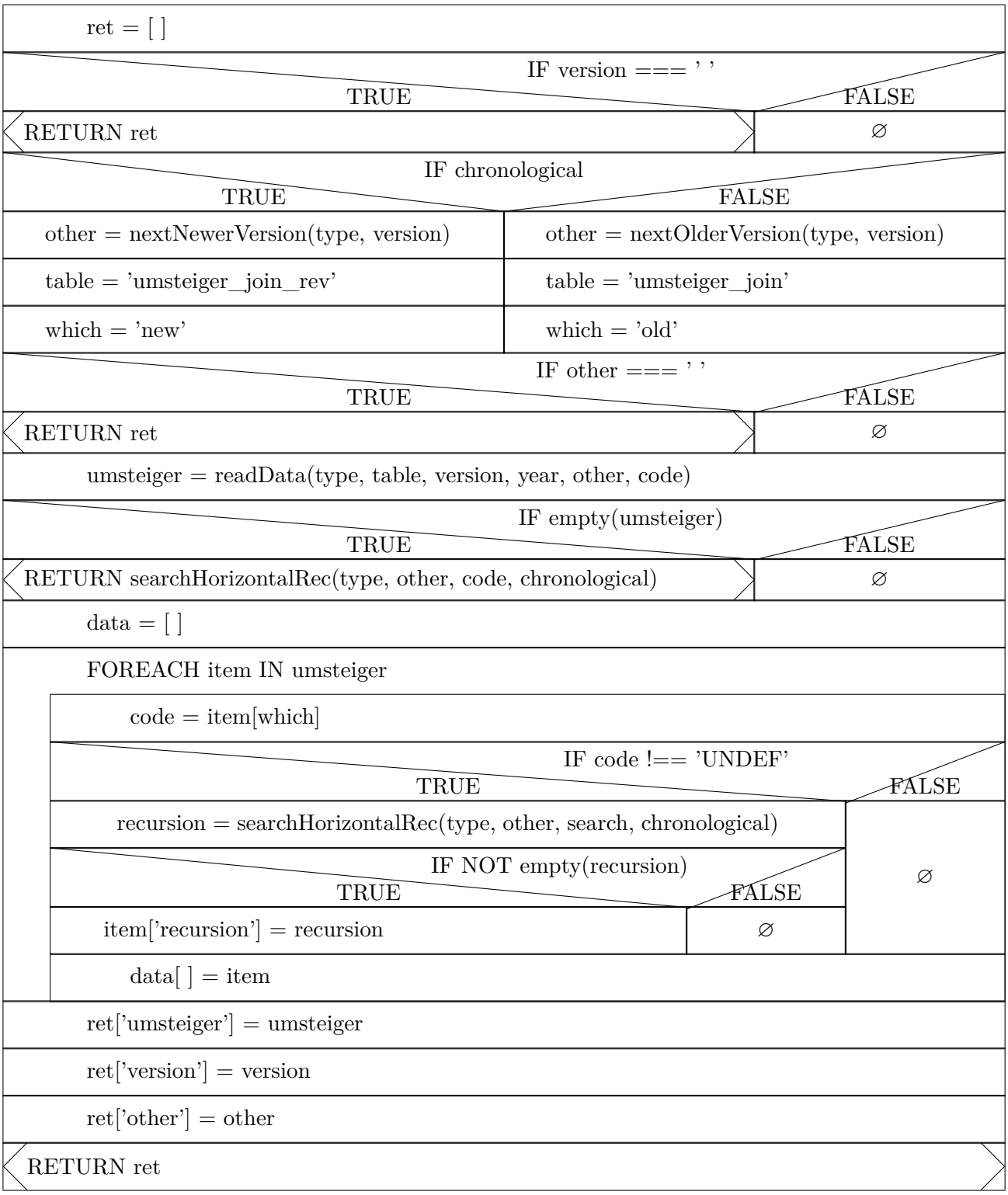
dVar {a double variable with the meaning described here}

	2024	2023	...	2004	2.0	1.3	

searchUmsteigerHorizontal

```
    data = [ ]
    data [ 'fwd' ] = searchUmsteigerHorizontal_recursion (type, version, code, true)
    data [ 'rev' ] = searchUmsteigerHorizontal_recursion (type, version, code, false)
RETURN data
```

searchUmsteigerHorizontal_recursion



Q

3 Horizontale Suche

searchUmsteigerVertical

data = [ ]
data += searchUmsteigerVertical_subroutine (type, target_version, false, function)
data += searchUmsteigerVertical_subroutine (type, target_version, true, function)
RETURN data

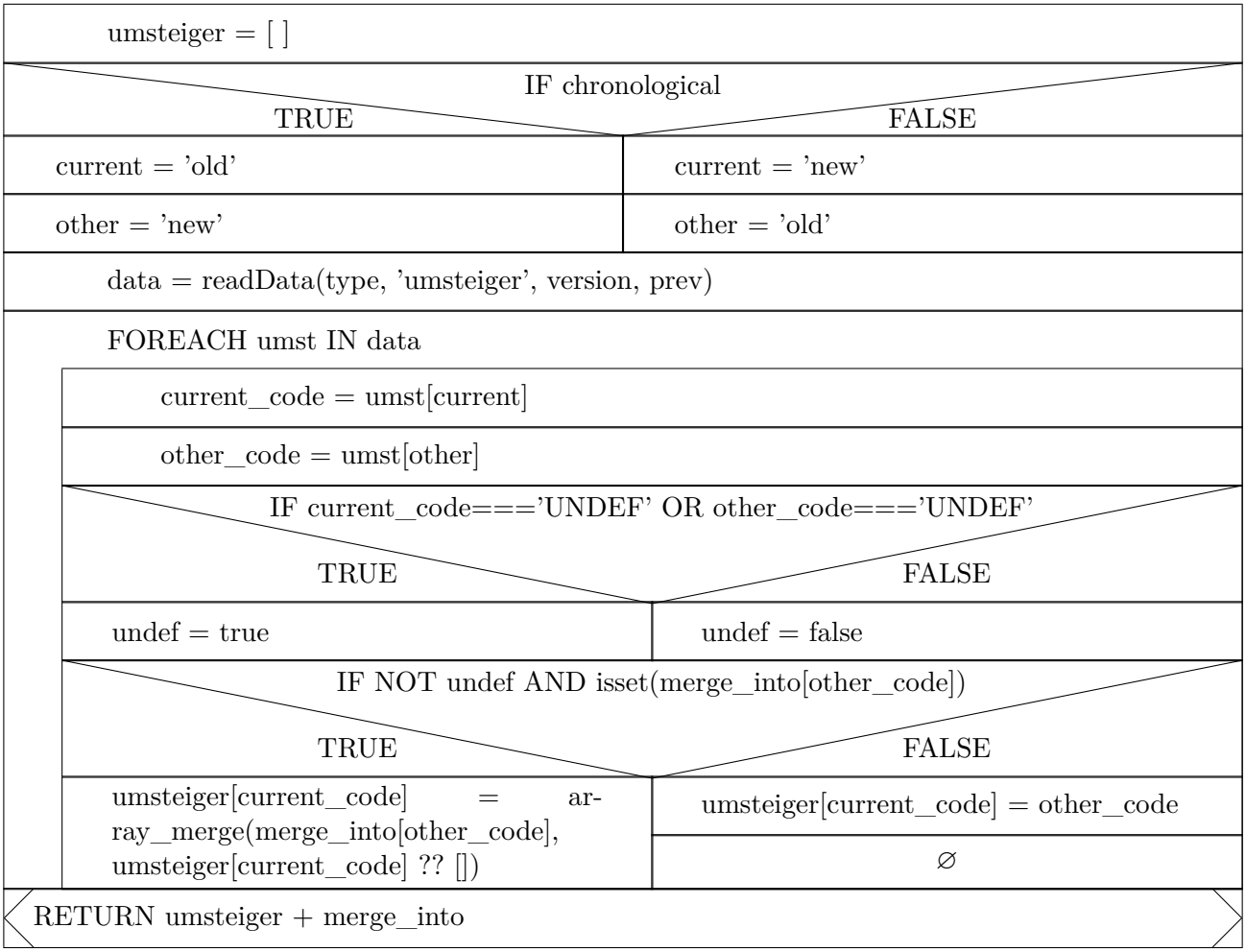
Q

searchUmsteigerVertical_subroutine

data = [ ]	
merge = [ ]	
version = target_version	
WHILE TRUE	
other = version	
IF chronological	
TRUE	FALSE
version = nextOlderVersion(type,other)	version = nextNewerVersion(type,other)
IF version === ' '	
TRUE	FALSE
RETURN data	∅
IF chronological	
TRUE	FALSE
merge = merge_umsteiger(type, chronological, other, version, merge)	merge = merge_umsteiger(type, chronological, version, other, merge)
IF function	
TRUE	FALSE
function(merge, version, target_version)	data['version'] = merge

Q

merge_umsteiger



Q

Literatur

S. Dar, *Augmenting Databases with Generalized Transitive Closure*, ser. Computer sciences technical report. University of Wisconsin–Madison, 1993.

J. Gross, J. Yellen, and P. Zhang, *Handbook of Graph Theory, Second Edition*, ser. Discrete Mathematics and Its Applications. Taylor & Francis, 2013.

H. Jakobsson, "Mixed-approach algorithms for transitive closure," in *Proceedings of the tenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, 1991, pp. 199–205. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/113413.113431

P. Purdom, "A transitive closure algorithm," *BIT Numerical Mathematics*, vol. 10, no. 1, pp. 76–94, 1970.