Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph

Citation for published version (APA):

Document status and date:
Published: 01/01/1994

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph

by

T. Kloks and D. Kratsch

94/41
Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph

T. Kloks
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands

D. Kratsch *
Friedrich-Schiller-Universität
Fakultät für Mathematik und Informatik
07740 Jena, Germany

Abstract: We present efficient algorithms for chordal bipartite graphs. Both algorithms use a doubly lexical ordering of the bipartite adjacency matrix. The first algorithm computes a perfect edge without vertex elimination ordering and the second one lists all maximal complete bipartite subgraphs.

Keywords: Design of algorithms, efficient algorithms, chordal bipartite graphs.

Introduction

Chordal bipartite graphs form a large class of perfect graphs containing, for example, convex and biconvex bipartite graphs, bipartite permutation graphs, and bipartite distance hereditary graphs (or (6, 2)-chordal bipartite graphs). For an overview on graph classes the reader is referred to [2, 6].

Recognizing chordal bipartite graphs can be done in time $O(\min(m \log n, n^2))$ [7, 9, 11, 12]. All these recognition algorithms use the same underlying idea. First compute a doubly lexical ordering of the bipartite adjacency matrix of the given bipartite graph and then check if this is $\Gamma$-free (see, e.g., [1, 4, 7]).

Chordal bipartite graphs can also be represented by a so-called perfect edge without vertex elimination ordering. We show that such an ordering can easily be computed from any doubly lexical ordering of the bipartite adjacency matrix. Thus we find an $O(\min(m \log n, n^2))$ algorithm computing a perfect edge from the given chordal bipartite graph. Furthermore, we present an algorithm computing a list of all maximal complete bipartite subgraphs of a chordal bipartite graph in time $O(\min(m \log n, n^2))$.

We improve upon the best known time bounds for computing the perfect edge without vertex elimination ordering as well as for computing the list of maximal complete bipartite subgraphs. Such algorithms are interesting since a perfect edge without vertex elimination ordering and the list of maximal complete bipartite subgraphs of a chordal bipartite graph are useful tools for designing efficient algorithms on chordal bipartite graphs and bi-interval graphs [8, 10].

Background

In this section we start with some definitions and easy lemmas. For more information the reader is referred to [2] or [6].

Definition 1 A graph is called chordal bipartite if it is bipartite and each cycle of length at least six has a chord.

Throughout the paper we assume that the input chordal bipartite graph has no isolated vertex. Furthermore, we denote by $n$ the number of vertices and by $m$ the number of
edges of the input graph. If \( x \) is a vertex of a graph \( G = (V, E) \), we denote by \( N(x) \) the set of neighbors of \( x \).

**Definition 2** Let \( G = (X, Y, E) \) be a bipartite graph. Then \( (x, y) \in E \) is called a bisimplicial edge if \( N(x) \cup N(y) \) induces a complete bipartite subgraph of \( G \).

The notion of a ‘perfect edge without vertex elimination ordering’ appears for example in [2]. It refers to an edge elimination ordering such that no vertices are deleted in the process.

**Definition 3** Let \( G = (X, Y, E) \) be a bipartite graph. Let \( (e_1, \ldots, e_m) \) be an ordering of the edges of \( G \). For \( i = 0, \ldots, m \) define the subgraph \( G_i = (X, Y, E_i) \) as follows. \( G_0 = G \) and for \( i \geq 1 \) \( G_i \) is the subgraph of \( G_{i-1} \) with vertex set \( X \cup Y \) and with edge set \( E_i = E_{i-1} \setminus \{e_i\} \) (i.e. the edge \( e_i \) is removed but not the endvertices). The ordering \( (e_1, \ldots, e_m) \) is a perfect edge without vertex elimination ordering for \( G \) if each edge \( e_i \) is bisimplicial in \( G_{i-1} \).

We use pewveo as a shorthand for perfect edge without vertex elimination ordering.

The following lemma appears for example in [2].

**Lemma 1** \( G \) is chordal bipartite if and only if there is a pewveo of \( G \).

We denote complete bipartite subgraphs of a chordal bipartite graph \( G = (X, Y, E) \) by \( (A, B) \), where \( A \) and \( B \) are nonempty subsets of \( X \) and \( Y \), respectively. We use mcs as a shorthand for maximal complete bipartite subgraph.

The following lemma indicates how to compute the list of all mcs from a pewveo of the chordal bipartite graph.

**Lemma 2** If \( G = (X, Y, E) \) is chordal bipartite, then it contains at most \( m^2 \) maximal complete bipartite subgraphs.

**Proof.** \( G \) is chordal bipartite, hence there is a perfect edge without vertex elimination ordering \( (e_1, \ldots, e_m) \). Consider a maximal complete bipartite subgraph \( (A, B) \). Let \( e_i \) be the first edge in the ordering which is an edge of \( (A, B) \). Let \( e_i = (x, y) \) with \( x \in A \) and \( y \in B \). Since \( e_i \) is bisimplicial and \( (A, B) \) is maximal we have \( A = N(y) \) and \( B = N(x) \). \( \Box \)

Recently the importance of a fast algorithm computing a pewveo of a chordal bipartite graph was recognized in [8, 10]. In both papers the ordering is used for the computation of the list of all mcs.

An \( O(n^2 m) \) algorithm for computing a pewveo follows from results of [5] and an \( O(n^2 + m^2) \) algorithm is given in [8]. Both algorithms iteratively compute a bisimplicial edge in the graph \( G_i \) for \( i = 0, 1, \ldots, m - 1 \).

The list of all mcs of a connected chordal bipartite graph can be computed in time \( O(nm^2) \) from a pewveo in a straightforward manner (see [8]). An alternative \( O(m^6) \) algorithm using fast matrix multiplication is given in [8], where \( O(m^6) \) is the best known time bound for multiplying two \( m \times m \) matrices.

**Doubly lexical ordering and perfect edge without vertex elimination ordering**

Our new algorithm for computing a pewveo exploits the information contained in a doubly lexical ordering of the bipartite adjacency matrix of the given chordal bipartite graph.

**Definition 4** Let \( G = (X, Y, E) \) be a bipartite graph with \( X = \{x_1, x_2, \ldots, x_s\} \) and \( Y = \{y_1, y_2, \ldots, y_t\} \). The bipartite adjacency matrix of \( G \) is the binary \( s \times t \) matrix \( A = (a_{ij}) \) such that \( a_{ij} = 1 \) if and only if \( (x_i, y_j) \in E \).

For the recognition of chordal bipartite graphs it is of importance to obtain a doubly lexical ordering of its bipartite adjacency matrix [1, 4, 7].
**Definition 5** A doubly lexical ordering of a binary matrix is an ordering of the columns and of the rows such that both the columns and the rows, as vectors, are lexically increasing.

The term ‘increasing’ is to be understood as ‘non-decreasing’. Here the vectors are read backwards, i.e., a vector $x$ is less than another vector $y$ if in the last different entry $x$ has a zero and $y$ a one.

**Definition 6** A binary matrix is $\Gamma$-free if it does not contain the matrix

\[
\begin{pmatrix}
1 & 1 \\
1 & 0
\end{pmatrix}
\]

as a submatrix.

The following lemma shows a strong relation between $\Gamma$-free matrices and chordal bipartite graphs (see [7, 9]).

**Lemma 3** A graph is chordal bipartite if and only if a doubly lexical ordering of its bipartite adjacency matrix is $\Gamma$-free.

Hence, chordal bipartite graphs can be recognized by determining a doubly lexical ordering of its bipartite adjacency matrix and checking whether this matrix is $\Gamma$-free. This approach leads to $O(m \log n)$ and $O(n^2)$ recognition algorithms for chordal bipartite graphs, where the computation of a doubly lexical ordering is the most time consuming step [11, 12].

We describe an algorithm to compute a pewveo of a chordal bipartite graph $G = (X, Y, E)$. We assume that the bipartite $s \times t$ adjacency matrix of the graph is stored by lists of nonzero entries of the columns and rows, respectively, as described in [9, 11]. This will allow us to inspect the nonzero entries of the matrix by using a pointer for each list of the nonzero entries of a row.

Using the algorithm of [11] or [12], a doubly lexical ordering of this matrix is computed in time $O(\min(m \log n, n^2))$. The resulting matrix is $\Gamma$-free. We denote it by $A$ and assume that $X$ and $Y$ are labeled such that $a_{ij} = 1$ if and only if $(x_i, y_j) \in E$.

The procedure $\text{pewveo}(A)$ given in Figure 1 computes a pewveo of $G$.

**Figure 1:** computing a pewveo

![Computation of a pewveo](image)

The procedure $\text{pewveo}(A)$ can obviously be implemented such that it runs in $O(n + m)$ time.

**Lemma 4** Let $A$ be a $\Gamma$-free bipartite adjacency matrix of a chordal bipartite graph $G$. Then the procedure $\text{pewveo}(A)$ computes a pewveo of $G$.

**Proof.** The crucial point is that when inspecting $a_{ij}$ then by the order of passing the matrix we have $a_{pj} = 0$ for all $p < i$ and $a_{iq} = 0$ for all $q < j$.

Let us denote the bipartite graph which corresponds to the matrix when inspecting $a_{ij}$ by $G' = (X, Y, E')$, i.e., $G'$ is the graph resulting from $G$ by the removal of those edges which are already inserted in the pewveo. Let $N'(u)$ denote the set of neighbours of a vertex $u$ in $G'$.

We conclude the proof by showing that $(x_i, y_j)$ is bisimplicial in $G'$. Let $y_p \in N'(x_i) \setminus \{y_j\}$ and $x_q \in N'(y_j) \setminus \{x_i\}$. Then $p > j$ and $q > i$. Hence $a_{ij} = 1$, $a_{ip} = 1$ and $a_{qj} = 1$ which implies that $a_{pq} = 1$ since $A$ is $\Gamma$-free.
Consequently \( N'(x_i) \cup N'(y_j) \) induces a complete bipartite subgraph of \( G' \). Hence, by definition, \((x_i, y_j)\) is bisimplicial in \( G' \). \( \square \)

**Theorem 1** The procedure \texttt{peveeo}(\(A\)) is an \( O(n + m) \) algorithm computing a perfect edge without vertex elimination ordering of a chordal bipartite graph \( G \) which is given by a \( \Gamma \)-free bipartite adjacency matrix \( A \).

**Theorem 2** There is an algorithm computing a perfect edge without vertex elimination ordering of a given chordal bipartite graph in time \( O(\min(m \log n, n^2)) \).

### Listing all maximal complete bipartite subgraphs

Notice that, for bipartite graphs in general, a list of all mcbs can be computed as follows. First make a clique of the two color classes and then use a clique listing algorithm (see, e.g., [3]). In this section we show that there is a faster method for chordal bipartite graphs using a doubly lexical ordering of the bipartite adjacency matrix.

Let \( A \) be the \( \Gamma \)-free bipartite adjacency matrix of the chordal bipartite graph \( G = (X, Y, E) \). Let the vertices of \( G \) be labeled such that \((x_i, y_j) \in E\) if and only if \( a_{ij} = 1 \).

Consider the \texttt{peveeo}(\(A\)). Given the \texttt{peveeo}, there is a compact presentation of the list of mcbs. For any edge \( e_r = (x, y) \) in the \texttt{peveeo} consider the graph \( G_r = G \setminus \{e_1, \ldots, e_{r-1}\} \). Let \( A_r \) and \( B_r \) be the sets of neighbors of \( y \) and \( x \), respectively, in \( G_r \). Then \((A_r, B_r)\) induces a complete bipartite subgraph in \( G_r \). Hence, given the \texttt{peveeo}, the mcbs algorithm needs only to output those edges \( e_r \) of \( G \) for which \((A_r, B_r)\) is a mcbs in \( G \). Equivalently, our algorithm labels those pairs \((i, j)\) for which \( a_{ij} \) of \( A \) is nonzero and for which the corresponding edge \((x_i, y_j)\) gives a mcbs of \( G \).

Let \( a_{ij} \) be a nonzero entry of \( A \). We denote by \((A_{ij}, B_{ij}) := \{(x_k \mid k \geq i \wedge a_{kj} = 1 \}, \{y_\ell \mid \ell \geq j \wedge a_{\ell i} = 1\}\) the complete bipartite subgraph of \( G \) corresponding with the edge \((x_i, y_j)\) with respect to \((e_1, e_2, \ldots, e_m)\).

We call a nonzero entry \( a_{ij} \) maximal if \((A_{ij}, B_{ij})\) is a mcbs of \( G \). Recall that, by Lemma 2, for any mcbs \((A, B)\) of \( G \) there is a nonzero entry \( a_{ij} = 1 \) such that \((A, B) = (A_{ij}, B_{ij})\).

**Lemma 5** A nonzero entry \( a_{pq} \) of \( A \) is not maximal if and only if there is another nonzero entry \( a_{ij} \) such that \( i \leq p, j \leq q \) and

\[
\begin{align*}
(1) & \quad a_{kj} = 1 \wedge k \geq p \quad \Rightarrow \quad a_{kj} = 1 \\
(2) & \quad a_{\ell i} = 1 \wedge \ell \geq q \quad \Rightarrow \quad a_{\ell i} = 1
\end{align*}
\]

**Proof.** Clearly, the nonzero entry \( a_{pq} \) is not maximal if and only if there is a nonzero entry \( a_{ij} \) with \( i \leq p \) and \( j \leq q \), such that \( A_{pq} \subseteq A_{ij} \) and \( B_{pq} \subseteq B_{ij} \), which is equivalent to \( \{x_k \mid k \geq p \wedge a_{kj} = 1\} \subseteq \{x_k \mid k \geq i \wedge a_{kj} = 1\} \) and \( \{y_\ell \mid \ell \geq q \wedge a_{\ell i} = 1\} \subseteq \{y_\ell \mid \ell \geq j \wedge a_{\ell i} = 1\} \). This is equivalent with (1) and (2). \( \square \)

We will say that \( a_{ij} \) covers \( a_{pq} \) if the conditions of Lemma 5 are fulfilled.

**Definition 7** Let \( A \) be a \( \Gamma \)-free matrix. For every nonzero entry \( a_{ij} \) of \( A \) we define

\[
\begin{align*}
d_{ij} & = |\{k \geq i \mid a_{kj} = 1\}| \\
r_{ij} & = |\{\ell \geq j \mid a_{\ell i} = 1\}|
\end{align*}
\]

Consequently \( d_{ij} = \sum_{k=i}^n a_{kj} \) and \( r_{ij} = \sum_{\ell=j}^m a_{\ell i} \). (Notice that \( d_{ij} \) and \( r_{ij} \) are undefined in case \( a_{ij} = 0 \).)

**Proposition 1** Let \( A \) be a \( \Gamma \)-free matrix. For any row \( i \), the sequence \( d_{ij} \) (of its nonzero entries) is increasing in \( j \). For any column \( j \), the sequence \( r_{ij} \) (of its nonzero entries) is increasing in \( i \).

**Proof.** Let \( a_{ij} = 1, a_{iq} = 1 \) and \( j < q \). Then \( a_{kj} = 1 \) implies \( a_{kq} = 1 \) for any \( k > i \) since \( A \) is \( \Gamma \)-free. Hence \( d_{ij} \leq d_{iq} \). Analogously, \( a_{ij} = 1, a_{pj} = 1 \) and \( i < p \) imply that \( r_{ij} \leq r_{pj} \). \( \square \)
The following lemma indicates how we can determine the nonzero entries of $A$ which are maximal.

**Lemma 6** The nonzero entry $a_{pq}$ of the $\Gamma$-free bipartite adjacency matrix $A$ of $G$ is maximal if and only if

\begin{align*}
(3) & \quad \forall k < p \ [r_{kq} < r_{pq}] \quad \text{and} \\
(4) & \quad \forall \ell < q \ [d_{p\ell} < d_{pq}]
\end{align*}

**Proof.** Assume $a_{pq}$ is not maximal and $a_{ij}$ covers $a_{pq}$. Then $i \leq p$, $j \leq q$, $a_{ij} = 1$ and $a_{pq} = 1$ by Lemma 5. Hence $d_{pq} = d_{p\ell}$ and $r_{ij} = r_{pq}$ by Lemma 5 and Proposition 1. Since $i \neq p$ or $j \neq q$, either (3) or (4) is violated.

Now assume that $a_{kq}$ is nonzero and $r_{kq} = r_{pq}$ for some $k < p$. We show that $a_{kq}$ covers $a_{pq} = 1$. We verify the conditions of Lemma 5. Clearly, (1) is fulfilled. Furthermore, $r_{kq} = r_{pq}$ and the $\Gamma$-freeness of $A$ imply that $\{\ell \mid \ell \geq q \land a_{\ell k} = 1\} = \{\ell \mid \ell \geq q \land a_{\ell p} = 1\}$, thus (2) is also fulfilled.

Analogously, $d_{p\ell} = d_{pq}$ for some $\ell < q$ implies that $a_{pq}$ covers $a_{pq}$. \[\square\]

Consequently, the maximal nonzero entries of $A$ can be determined using the procedure given in Figure 2.

The correctness of the procedure follows from Lemma 6.

It is not hard to implement $\text{mcbs}(A)$ such that it runs in $O(n + m)$. The computation of the values $r_{ij}$ and $d_{ij}$ can be done by scanning all rows and all columns once.

**Theorem 3** The procedure $\text{mcbs}(A)$ is an $O(n + m)$ algorithm computing the list of all maximal complete bipartite subgraphs of a chordal bipartite graph which is given by a $\Gamma$-free bipartite adjacency matrix $A$.

**Theorem 4** There is an algorithm computing the list of all maximal complete bipartite subgraphs of a given chordal bipartite graph in time $O(\min(m \log n, n^2))$.

\begin{algorithm}
\begin{algorithmic}
\Function{mcbs}{$A$}
\State Compute $r_{ij}$ and $d_{ij}$ for nonzero entries;
\For {$i = 1$ to $s$}
\DoBegin\While{\text{Pass through all nonzero entries of row $i$ in increasing order of $j$ and label those $a_{ij}$ with $\mathcal{D}$ for which $d_{ij} > \max\{d_{\ell j} \mid \ell < i\}$}}
\End\While{\text{for $j = 1$ to $t$}}
\DoBegin\While{\text{Pass through all nonzero entries of column $j$ in increasing order of $i$ and label those $a_{ij}$ with $\mathcal{R}$ for which $r_{ij} > \max\{r_{kj} \mid k < i\}$}}
\End\While{\text{Output $(x_i, y_j)$ for all entries $a_{ij}$ labeled $\mathcal{D}$ and $\mathcal{R}$}}
\End
\End
\End
\End
\end{algorithmic}
\caption{computing the list of mcb5}
\end{algorithm}

**Conclusions**

We have shown that a perfect edge without vertex elimination ordering and a list of all maximal complete bipartite subgraphs of a chordal bipartite graph can be computed in linear time when the graph is given by a $\Gamma$-free bipartite adjacency matrix. Such a matrix is the output of the best known recognition algorithms for chordal bipartite graphs [11, 12].

However, it would be interesting to know whether the bipartite adjacency matrix of a given chordal bipartite graph can be transformed into a $\Gamma$-free bipartite adjacency matrix by a linear time algorithm.

1 Acknowledgements

We thank Jeremy Spinrad for reading the manuscript and for his useful comments and remarks.
References


[10] Müller, H., Recognizing interval digraphs and bi-interval graphs in polynomial time, manuscript.


<table>
<thead>
<tr>
<th>Volume</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/02</td>
<td>R.P. Nederpelt, H.C.M. de Swart</td>
<td>Implication. A survey of the different logical analyses &quot;if...,then...&quot;, p. 26.</td>
</tr>
<tr>
<td>91/03</td>
<td>J.P. Katoen, L.A.M. Schoenmakers</td>
<td>Parallel Programs for the Recognition of P-invariant Segments, p. 16.</td>
</tr>
<tr>
<td>91/05</td>
<td>D. de Reus</td>
<td>An Implementation Model for GOOD, p. 18.</td>
</tr>
<tr>
<td>91/06</td>
<td>K.M. van Hee</td>
<td>SPECIFICATIEMETHODEN, een overzicht, p. 20.</td>
</tr>
<tr>
<td>91/07</td>
<td>E.Poll</td>
<td>CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.</td>
</tr>
<tr>
<td>91/12</td>
<td>E. van der Sluis</td>
<td>A parallel local search algorithm for the travelling salesman problem, p. 12.</td>
</tr>
<tr>
<td>91/14</td>
<td>P. Lemmens</td>
<td>The PDB Hypermedia Package. Why and how it was built, p. 63.</td>
</tr>
<tr>
<td>91/16</td>
<td>A.J.J.M. Marcelis</td>
<td>An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.</td>
</tr>
</tbody>
</table>
91/17 A.T.M. Aerts  
P.M.E. de Bra  
K.M. van Hee  
Transforming Functional Database Schemes to Relational Representations, p. 21.

91/18 Rik van Geldrop  
Transformational Query Solving, p. 35.

91/19 Erik Poll  
Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.

91/20 A.E. Eiben  
R.V. Schuwer  

91/21 J. Coenen  
W.-P. de Roever  
J. Zwiers  
Assertional Data Reification Proofs: Survey and Perspective, p. 18.

91/22 G. Wolf  

91/23 K.M. van Hee  
L.J. Somers  
M. Voorhoeve  
Z and high level Petri nets, p. 16.

91/24 A.T.M. Aerts  
D. de Reus  
Formal semantics for BRM with examples, p. 25.

91/25 P. Zhou  
J. Hooman  
R. Kuiper  
A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and completeness, p. 52.

91/26 P. de Bra  
G.J. Houben  
J. Paredaens  
The GOOD based hypertext reference model, p. 12.

91/27 F. de Boer  
C. Palamidessi  
Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.

91/28 F. de Boer  
A compositional proof system for dynamic process creation, p. 24.

91/29 H. Ten Eikelder  
R. van Geldrop  
Correctness of Acceptor Schemes for Regular Languages, p. 31.

91/30 J.C.M. Baeten  
F.W. Vaandrager  
An Algebra for Process Creation, p. 29.

91/31 H. ten Eikelder  
Some algorithms to decide the equivalence of recursive types, p. 26.

91/32 P. Struik  

91/33 W. v.d. Aalst  
The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.

91/34 J. Coenen  
Specifying fault tolerant programs in deontic logic, p. 15.
<table>
<thead>
<tr>
<th>No.</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>92/01</td>
<td>J. Coenen, J. Zwiers, W.-P. de Roever</td>
<td>A note on compositional refinement, p. 27.</td>
</tr>
<tr>
<td>92/02</td>
<td>J. Coenen, J. Hooman</td>
<td>A compositional semantics for fault tolerant real-time systems, p. 18.</td>
</tr>
<tr>
<td>92/03</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Real space process algebra, p. 42.</td>
</tr>
<tr>
<td>92/05</td>
<td>J.P.H.W.v.d.Eijnde</td>
<td>Conservative fixpoint functions on a graph, p. 25.</td>
</tr>
<tr>
<td>92/06</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Discrete time process algebra, p.45.</td>
</tr>
<tr>
<td>92/07</td>
<td>R.P. Nederpelt</td>
<td>The fine-structure of lambda calculus, p. 110.</td>
</tr>
<tr>
<td>92/10</td>
<td>P.M.P. Rambags</td>
<td>Composition and decomposition in a CPN model, p. 55.</td>
</tr>
<tr>
<td>92/13</td>
<td>F. Kamareddine</td>
<td>Set theory and nominalisation, Part II, p.22.</td>
</tr>
<tr>
<td>92/14</td>
<td>J.C.M. Baeten</td>
<td>The total order assumption, p. 10.</td>
</tr>
<tr>
<td>92/15</td>
<td>F. Kamareddine</td>
<td>A system at the cross-roads of functional and logic programming, p.36.</td>
</tr>
<tr>
<td>92/16</td>
<td>R.R. Seljée</td>
<td>Integrity checking in deductive databases; an exposition, p.32.</td>
</tr>
<tr>
<td>92/17</td>
<td>W.M.P. van der Aalst</td>
<td>Interval timed coloured Petri nets and their analysis, p. 20.</td>
</tr>
<tr>
<td>92/18</td>
<td>R.Nederpelt, F. Kamareddine</td>
<td>A unified approach to Type Theory through a refined lambda-calculus, p.30.</td>
</tr>
<tr>
<td>92/20</td>
<td>F.Kamareddine</td>
<td>Are Types for Natural Language? P. 32.</td>
</tr>
</tbody>
</table>
92/21 F. Kamareddine
Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.

92/22 R. Nederpelt
A useful lambda notation, p. 17.
F. Kamareddine

92/23 F. Kamareddine
Nominalization, Predication and Type Containment, p. 40.
E. Klein

92/24 M. Codish
Bottom-up Abstract Interpretation of Logic Programs, p. 33.
D. Dams
Eyal Yardeni

92/25 E. Poll
A Programming Logic for Fp, p. 15.

92/26 T. H. W. Beelen
A modelling method using MOVIE and SimCon/ExSpect, p. 15.
W. J. J. Stut
P. A. C. Verkoulen

92/27 B. Watson
A taxonomy of keyword pattern matching algorithms, p. 50.
G. Zwaan

93/01 R. van Geldrop
Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.

93/02 T. Verhoeff
A continuous version of the Prisoner’s Dilemma, p. 17

93/03 T. Verhoeff
Quicksort for linked lists, p. 8.

93/04 E. H. L. Aarts
Deterministic and randomized local search, p. 78.
J. H. M. Korst
P. J. Zwietening

93/05 J. C. M. Baeten
A congruence theorem for structured operational semantics with predicates, p. 18.
C. Verhoef

93/06 J. P. Veltkamp
On the unavoidability of metastable behaviour, p. 29

93/07 P. D. Moerland
Exercises in Multiprogramming, p. 97

93/08 J. Verhoosel
A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.

93/09 K. M. van Hee

93/10 K. M. van Hee
Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.

93/11 K. M. van Hee

93/12 K. M. van Hee

93/13 K. M. van Hee
Systems Engineering: a Formal Approach...
93/14 J.C.M. Baeten
J.A. Bergstra


93/15 J.C.M. Baeten
J.A. Bergstra
R.N. Bol


93/16 H. Schepers
J. Hooman

A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27

93/17 D. Alstein
P. van der Stok

Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.

93/18 C. Verhoef

A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.

93/19 G-J. Houben

The Design of an Online Help Facility for ExSpect, p. 21.

93/20 F.S. de Boer


93/21 M. Codish
D. Dams
G. Filé
M. Bruynooghe


93/22 E. Poll

A Typechecker for Bijective Pure Type Systems, p. 28.

93/23 E. de Kogel

Relational Algebra and Equational Proofs, p. 23.

93/24 E. Poll and Paula Severi

Pure Type Systems with Definitions, p. 38.

93/25 H. Schepers and R. Gerth


93/26 W.M.P. van der Aalst

Multi-dimensional Petri nets, p. 25.

93/27 T. Kloks and D. Kratsch

Finding all minimal separators of a graph, p. 11.

93/28 F. Kamareddine and R. Nederpelt

A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.

93/29 R. Post and P. De Bra

GOLD, a Graph Oriented Language for Databases, p. 42.

93/30 J. Deogun
T. Kloks
D. Kratsch
H. Müller

On Vertex Ranking for Permutation and Other Graphs, p. 11.

93/31 W. Körver

Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.

93/32 H. ten Eikelder and H. van Geldrop

<p>| 93/33 | L. Loyens and J. Moonen | ILIAS, a sequential language for parallel matrix computations, p. 20. |
| 93/34 | J.C.M. Baeten and J.A. Bergstra | Real Time Process Algebra with Infinitesimals, p. 39. |
| 93/36 | J.C.M. Baeten and J.A. Bergstra | Non Interleaving Process Algebra, p. 17. |
| 93/38 | C. Verhoef | A general conservative extension theorem in process algebra, p. 17. |
| 93/41 | A. Bijlsma | Temporal operators viewed as predicate transformers, p. 11. |
| 93/42 | P.M.P. Rambags | Automatic Verification of Regular Protocols in P/T Nets, p. 23. |
| 93/43 | B.W. Watson | A taxonomy of finite automata construction algorithms, p. 87. |
| 93/44 | B.W. Watson | A taxonomy of finite automata minimization algorithms, p. 23. |
| 93/48 | R. Gerth | Verifying Sequentially Consistent Memory using Interface Refinement, p. 20. |</p>
<table>
<thead>
<tr>
<th>Paper ID</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>94/01</td>
<td>P. America, M. van der Kammen, R.P. Nederpelt, O.S. van Roosmalen, H.C.M. de Swart</td>
<td>The object-oriented paradigm, p. 28.</td>
</tr>
<tr>
<td>94/02</td>
<td>F. Kamareddine, R.P. Nederpelt</td>
<td>Canonical typing and II-conversion, p. 51.</td>
</tr>
<tr>
<td>94/04</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.</td>
</tr>
<tr>
<td>94/06</td>
<td>T. Basten, T. Kunz, J. Black, M. Coffin, D. Taylor</td>
<td>Time and the Order of Abstract Events in Distributed Computations, p. 29.</td>
</tr>
<tr>
<td>94/08</td>
<td>O.S. van Roosmalen</td>
<td>A Hierarchical Diagrammatic Representation of Class Structure, p. 22.</td>
</tr>
<tr>
<td>94/09</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Process Algebra with Partial Choice, p. 16.</td>
</tr>
<tr>
<td>94/10</td>
<td>T. Verhoeff</td>
<td>The testing Paradigm Applied to Network Structure, p. 31.</td>
</tr>
<tr>
<td>94/13</td>
<td>R. Seljé</td>
<td>A New Method for Integrity Constraint checking in Deductive Databases, p. 34.</td>
</tr>
<tr>
<td>94/14</td>
<td>W. Peremans</td>
<td>Ups and Downs of Type Theory, p. 9.</td>
</tr>
<tr>
<td>94/16</td>
<td>R.C. Backhouse, H. Doornbos</td>
<td>Mathematical Induction Made Calculational, p. 36.</td>
</tr>
<tr>
<td>ID</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>94/18</td>
<td>F. Kamareddine, R. Nederpelt</td>
<td>Refining Reduction in the Lambda Calculus, p. 15.</td>
</tr>
<tr>
<td>94/19</td>
<td>B.W. Watson</td>
<td>The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.</td>
</tr>
<tr>
<td>94/20</td>
<td>R. Bloo, F. Kamareddine, R. Nederpelt</td>
<td>Beyond $\beta$-Reduction in Church's $\lambda\to$, p. 22.</td>
</tr>
<tr>
<td>94/22</td>
<td>B.W. Watson</td>
<td>The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.</td>
</tr>
<tr>
<td>94/23</td>
<td>S. Mauw and M.A. Reniers</td>
<td>An algebraic semantics of Message Sequence Charts, p. 43.</td>
</tr>
<tr>
<td>94/24</td>
<td>D. Dams, O. Grumberg, R. Gerth</td>
<td>Abstract Interpretation of Reactive Systems: Abstractions Preserving $\forall$CTL*, $\exists$CTL* and CTL*, p. 28.</td>
</tr>
<tr>
<td>94/25</td>
<td>T. Kloks</td>
<td>$K_{1,3}$-free and $W_4$-free graphs, p. 10.</td>
</tr>
<tr>
<td>94/29</td>
<td>J. Hooman</td>
<td>Correctness of Real Time Systems by Construction, p. 22.</td>
</tr>
<tr>
<td>94/30</td>
<td>J.C.M. Baeten, J.A. Bergstra, Gh. Ştefănescu</td>
<td>Process Algebra with Feedback, p. 22.</td>
</tr>
<tr>
<td>94/31</td>
<td>B.W. Watson, R.E. Watson</td>
<td>A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.</td>
</tr>
<tr>
<td>94/33</td>
<td>T. Laan</td>
<td>A formalization of the Ramified Type Theory, p. 40.</td>
</tr>
<tr>
<td>94/34</td>
<td>R. Bloo, F. Kamareddine, R. Nederpelt</td>
<td>The Barendregt Cube with Definitions and Generalised Reduction, p. 37.</td>
</tr>
<tr>
<td>94/35</td>
<td>J.C.M. Baeten, S. Mauw</td>
<td>Delayed choice: an operator for joining Message Sequence Charts, p. 15.</td>
</tr>
<tr>
<td>94/36</td>
<td>F. Kamareddine, R. Nederpelt</td>
<td>Canonical typing and $\Pi$-conversion in the Barendregt Cube, p. 19.</td>
</tr>
<tr>
<td>No.</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>94/38</td>
<td>A. Bijlsma, C.S. Scholten</td>
<td>Point-free substitution, p. 10.</td>
</tr>
<tr>
<td>94/39</td>
<td>A. Blokhuis, T. Kloks</td>
<td>On the equivalence covering number of splitgraphs, p. 4.</td>
</tr>
<tr>
<td>94/40</td>
<td>D. Alstein</td>
<td>Distributed Consensus and Hard Real-Time Systems, p. 34.</td>
</tr>
</tbody>
</table>