TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

Hypergraph Construction and Its Application to the Compositional Modelling of Concurrency

Barbara König

TUM-I0003
SFB-Bericht Nr. 342/02/00 A
Februar 00

TUM–INFO–02-I0003-0/1.–FI

Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©2000 SFB 342 Methoden und Werkzeuge für
die Nutzung paralleler Architekturen

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

Hypergraph Construction and Its Application to the Compositional Modelling of Concurrency (Extended version)

Barbara König (koenigb@in.tum.de)

Fakultät für Informatik, Technische Universität München

Abstract. We define a construction operation on hypergraphs using a colimit and show that its expressiveness concerning graph rewriting is equal to the graph expressions of Courcelle and the double-pushout approach of Ehrig. With an inductive way of representing graphs, graph rewriting arises naturally as a derived concept. The usefulness of our approach for the compositional modelling of concurrent systems is then shown by defining the semantics of a process calculus with mobility and of Petri nets.

1 Introduction

Graph rewriting is one adequate approach for modelling the semantics of concurrent systems: multi-dimensional structures describing interconnected computers or other components can be naturally described by graphs. When trying to model semantic frameworks for concurrency (such as process algebras) with graph rewriting, we run into problems since it is hard to represent compositionality and modularity inherent in these formalisms in the world of graphs. Process algebras rely on compositionality in the definition of their syntax and their semantics, and in almost all proofs. Compositionality is easy to achieve in a "string-based" syntax: systems are constructed out of smaller systems by concatenating their descriptions, connections are established by having common channel names.

We propose an analogue to concatenation in the world of hypergraphs: hypergraphs have so-called "external nodes", their interface to the outside, and in order to attach two (or more) hypergraphs, information is needed on how these external nodes should be merged. Such a form of graph concatenation was proposed in [1] in the form of "graph expressions", where three operators (disjoint sum, node fusion, redefinition of external nodes) were introduced. These operators can be used in order to gradually construct hypergraphs out of smaller hypergraphs.

We propose a similar approach of graph construction where the information on how to concatenate graphs is not provided by operators, but rather by part of a colimit. The construction itself consists of completing the colimit and is related to the double-pushout approach [3]. The expressive power of graph rewriting in our approach is the same as for graph expressions [1] (and thus the same as in the double-pushout approach [8]). Compared to [1] we have additional information in the uniqueness property of the colimit and embeddings of the subgraphs into the constructed graph which are provided by the colimit. This information can be helpful when we want to reason about concurrent systems or if we want to extend the formalism. We still have a description of the graph in terms of edge and node sets, which makes it easy to add additional labels or annotations.

We demonstrate the usefulness of our approach by giving a compositional semantics of a process calculus with mobility and of Petri nets. Note that, although we model concurrent systems, the semantics itself will not be concurrent in the sense of true concurrency (as in [6]).

2 Hypergraphs and Hypergraph Construction

We first define some basic notions, namely hypergraph, hypergraph morphism, and isomorphism (see also [8]). Hypergraphs are a generalization of directed graphs where an arbitrarily long sequence of nodes is assigned to every edge. The order of nodes with respect to a certain edge is

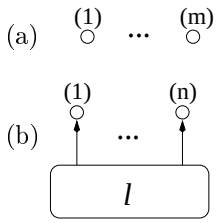
relevant. Intuitively we construct hypergraphs by drawing edges (with nodes) and then merging the nodes, rather than by drawing nodes and then connecting them by edges. This intuition will also guide our choice of a hypergraph construction operator.

Definition 1. (Hypergraph, Hypergraph Morphism, Isomorphism) *Let L be a fixed set of labels. A* hypergraph *H is a tuple $H = (V_H, E_H, s_H, l_H, \chi_H)$ where V_H is a set of* nodes*, E_H is a set of* edges *disjoint from V_H, $s_H \colon E_H \to V_H^*$ maps each edge to a string of* source nodes*, $l_H \colon E_H \to L$ assigns a label to each edge, and $\chi_H \in V_H^*$ is a string of* external nodes*.*

Let H, H' be two hypergraphs. A hypergraph morphism *$\phi \colon H \to H'$ consists of two mappings $\phi_E \colon E_H \to E_{H'}$, $\phi_V \colon V_H \to V_{H'}$ satisfying*[1] *$\phi_V(s_H(e)) = s_{H'}(\phi_E(e))$ and $l_H(e) = l_{H'}(\phi_E(e))$ for all $e \in E_H$. If furthermore $\phi_V(\chi_H) = \chi_{H'}$ we call ϕ a* strong *hypergraph morphism and denote it by $\phi : H \twoheadrightarrow H'$. The hypergraphs H and H' are called* isomorphic *($H \cong H'$) if there exists a bijective strong morphism (= isomorphism) from one hypergraph into the other.*

The arity of a hypergraph H is defined as $ar(H) = |\chi_H|$ while the arity of an edge e of H is $ar(e) = |s_H(e)|$. We can regard hypergraphs and hypergraph morphisms (respectively strong hypergraph morphisms) as objects respectively morphisms of a category.

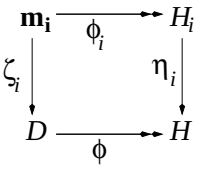
Notation: We call a hypergraph *discrete*, if its edge set is empty. $\mathbf{m}$ denotes a discrete graph of arity $m \in \mathbb{N}$ with m nodes where every node is external (see (a), external nodes are labelled (1), (2), ... in their respective order).
$H = [l]_n$ is the hypergraph with exactly one edge e with label l where $s_H(e) = \chi_H$, $|\chi_H| = n$, χ_H contains no duplicates and $V_H = Set(\chi_H)$, where $Set(s)$ is the set of all elements of a string s (see (b), nodes are ordered from left to right).



We now present the "concatenation operation" discussed in the introduction. The construction plan telling us how this concatenation is supposed to happen is represented by hypergraph morphisms mapping discrete graphs to discrete graphs. The following definitions use concepts from category theory, namely categories and colimits. For an introduction to these concepts see [3, 2].

Definition 2. (Hypergraph Construction) *Let $H_1, \ldots, H_n$ be hypergraphs and let*[2] *$\zeta_i : \mathbf{m_i} \to D$, $i \in [n]$ be morphisms where $ar(H_i) = m_i \in \mathbb{N}$ and D is a discrete graph. There is always a unique strong morphism $\phi_i : \mathbf{m_i} \twoheadrightarrow H_i$ for every $i \in [n]$.*

Let H (with morphisms $\phi : D \to H$, $\eta_i : H_i \to H$) be the colimit of $\zeta_1, \ldots, \zeta_n, \phi_1, \ldots, \phi_n$ such that ϕ is a strong morphism. We define: $\bigotimes_{i=1}^{n}(H_i, \zeta_i) = H$. (Alternatively we write $(H_1, \zeta_1) \otimes \ldots \otimes (H_n, \zeta_n)$—or $\otimes(H_1, \zeta_1)$, if $n = 1$—instead of $\bigotimes_{i=1}^{n}(H_i, \zeta_i)$.)



Generally, colimits do not necessarily exist, but they always exist in our case. The colimit is unique up to bijective morphisms, but *not* unique up to isomorphism. Therefore we demand above that the morphism ϕ from D into the colimit be a strong morphism and thereby determine the string of external nodes of the result. Furthermore the morphisms η_i generated by the colimit are always embeddings (as defined in [1]).

Hypergraph construction without category theory: although the characterization of hypergraph construction is more elegant in the categorical setting, we can also describe it without category theory. We assume that the node and edge sets of D and H_1, ..., H_n are pairwise disjoint. Furthermore let $\approx$ be the smallest equivalence on their nodes satisfying

$$\zeta_i(v) \approx \phi_i(v) \text{ if } i \in [n],\ v \in V_{\mathbf{m_i}}$$

[1] Application of morphisms to sequences of nodes is conducted pointwise.
[2] $[n]$ stands for the set $\{1, \ldots, n\}$.

The nodes of the constructed graph are the equivalence classes of $\approx$. Thus $\bigotimes_{i=1}^{n}(H_i, \zeta_i)$ is isomorphic to

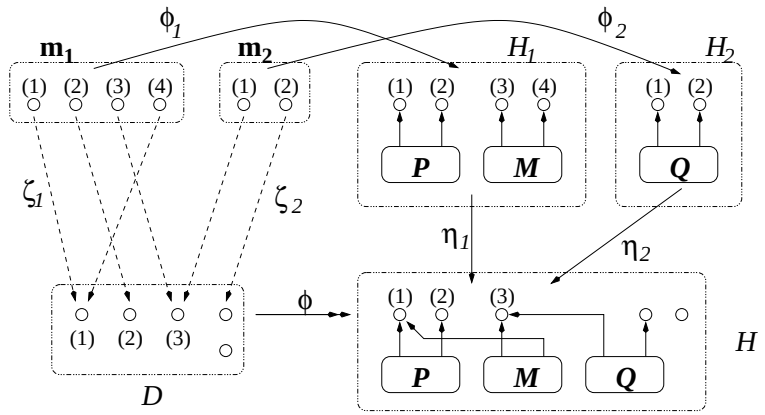
$$((V_D \cup \bigcup_{i=1}^{n} V_{H_i})/\approx, \bigcup_{i=1}^{n} E_{H_i}, s_H, l_H, \chi/\approx)$$

where $s_H(e) = [v_1]_\approx \dots [v_k]_\approx$ if $e \in E_{H_i}$ and $s_{H_i}(e) = v_1 \dots v_k$. Furthermore $l_H(e) = l_{H_i}(e)$ if $e \in E_{H_i}$. And we define $\chi/\approx = [v_1]_\approx \dots [v_k]_\approx$ if $\chi_D = v_1 \dots v_k$.

In other words: we join all graphs $D, H_1, \dots, H_n$ and fuse exactly the nodes which are the image of one and the same node in the $\mathbf{m_i}$. χ_D becomes the new sequence of external nodes.

Now the morphisms $\phi : D \twoheadrightarrow H$ and $\eta_i : H_i \to H$ can be defined as follows: $\phi(v) = [v]_\approx$ if $v \in V_D$. Furthermore $\eta_i(v) = [v]_\approx$ if $v \in V_{H_i}$ and $\eta_i(e) = e$ if $e \in E_{H_i}$.

Example: we want to construct a graph H consisting of a hyperedge representing a message (labelled M) and two hyperedges representing processes (labelled P, Q). (We will show in section 5 how to model the reception of a message by a process with graph rewriting.)
H consists of two subgraphs H_1 (containing the process labelled P and a message) and H_2 (containing a process labelled Q) which are concatenated according to the "construction plan" given by the discrete morphisms ζ_1, ζ_2. E.g. the third external node of H_1 is fused with the third external node of D. This, in turn, is fused with the first external node of H_2. Isolated nodes in H are generated by nodes in D which are not in the range of either of the ζ_i.



Example: another example is an operator which takes two hypergraphs of the same arity and attaches them at their external nodes, i.e. the nodes are merged in their respective order. Let m be a fixed natural number and let $\zeta : \mathbf{m} \twoheadrightarrow \mathbf{m}$ be a strong morphism. We define $H_1 \square H_2 = (H_1, \zeta) \otimes (H_2, \zeta)$ for H_1, H_2 satisfying $ar(H_1) = ar(H_2) = n$.

3 Graph Expressions and the Double-Pushout Approach

3.1 Graph Expressions

Graph expressions were introduced by Michel Bauderon and Bruno Courcelle in [1] as an algebraic structure for graph construction. They introduced three operators (explained below) and a complete set of equations relating hypergraphs if and only if they are isomorphic. We will now introduce the three operators and give their corresponding version in our framework in terms of the discrete morphisms ζ_i.

Disjoint Sum: $H_1 \oplus H_2$ is the hypergraph resulting from the disjoint union of the node and edge sets and of the source and labelling functions of H_1 and H_2. Furthermore χ_{H_1} and χ_{H_2} are concatenated to form the sequence of external nodes of $H_1 \oplus H_2$. $H_1 \oplus H_2 \cong \bigotimes_{i=1}^{2}(H_i, \zeta_i)$ where ζ_1, ζ_2 are defined as in figure 1 (a) ($m = ar(H_1)$, $n = ar(H_2)$).

Redefinition of External Nodes: Let $\alpha : [p] \to [m]$ and $m = ar(H)$. Then $\sigma_\alpha(H)$ is the hypergraph resulting from redefining the external nodes of H according to α, i.e. χ_H is replaced by[3] $\lfloor \chi_H \rfloor_{\alpha(1)} \cdots \lfloor \chi_H \rfloor_{\alpha(p)}$. The rest of the hypergraph stays unchanged.

We exploit the fact that α can always be decomposed into $\alpha = \alpha_1 \circ \ldots \circ \alpha_k$ where each α_i has one of the following three forms listed below. According to [1] $\sigma_\alpha(H) \cong \sigma_{\alpha_k}(\ldots \sigma_{\alpha_1}(H) \ldots)$. Thus we only have to consider these three cases.

In each of the following cases $\sigma_\alpha(H) \cong \otimes(H, \zeta)$.

Permutation of External Nodes: $\alpha : [m] \to [m]$ is a bijection. ζ is defined in (b).

Hiding an External Node: $\alpha : [m] \to [m+1]$ where $\alpha(i) = i$. ζ is defined in (c).

Duplicating an External Node: $\alpha : [m+1] \to [m]$ where $\alpha(i) = i$ if $i \in [m]$ and $\alpha(m+1) = m$. Then ζ is defined as in figure 1 (d).

Fusing External Nodes: Let δ be an equivalence relation on $[m]$ where $m = ar(H)$. $\theta_\delta(H)$ is obtained by fusing all external nodes which are related by δ. The arity of the hypergraph is not changed.

According to [1] $\theta_\delta(H) \cong \theta_{\delta_k}(\ldots \theta_{\delta_1}(H) \ldots)$ where each δ_i is an equivalence generated by a single pair (i, j) with $i, j \in [m]$. With the permutation operation defined above it suffices to define a colimit construction fusing the last two nodes of a hypergraph. Let δ' be the equivalence on $[m]$ generated by the pair $(m-1, m)$. Then $\theta_{\delta'}(H) \cong \otimes(H, \zeta)$ where ζ is defined as in figure 1 (e).

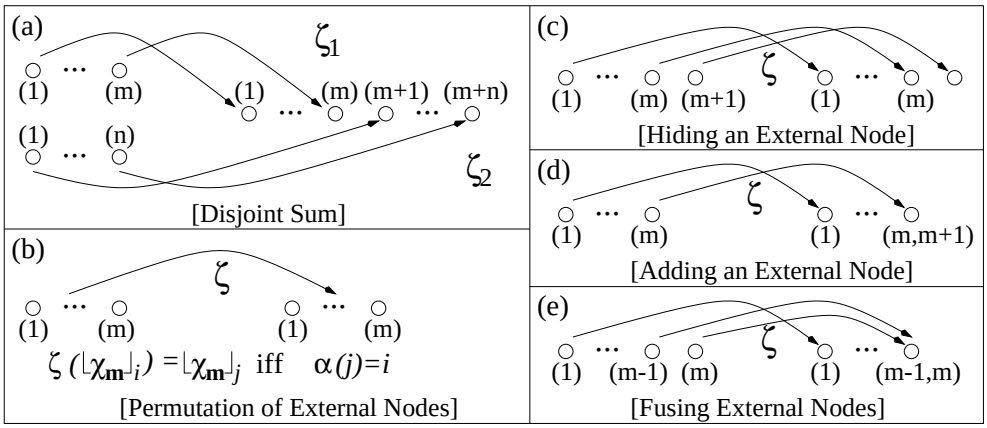


Fig. 1. Converting a graph expression into the corresponding colimit construction.

We have shown how to emulate all three operators by colimits. It is still left to show how the subsequent application of colimits can be converted into one single colimit construction. We will delay this until section 4.

3.2 The Double-Pushout Approach

The double-pushout approach to graph rewriting was introduced by Ehrig [3], the double-pushout approach to *hypergraph* rewriting is presented in [8]. Our colimit construction is closely related to this approach. We will now make this connection precise. If, in the diagram in definition 2, we set $n = 1$ and allow D to be an arbitrary non-discrete graph, we obtain exactly the right-hand side of a double-pushout.

This does not yet reveal anything about the expressive power of our approach. We will now define the notion of a rewriting step: let $r = (L, R)$ be a rewriting rule, where L, R are hypergraphs

[3] $\lfloor s \rfloor_i$ denotes the i-th element of the string s.

with $ar(L) = ar(R)$. Then $\overset{r}{\Longrightarrow}$ is the smallest relation which is generated by the following two rules and is closed under isomorphism.

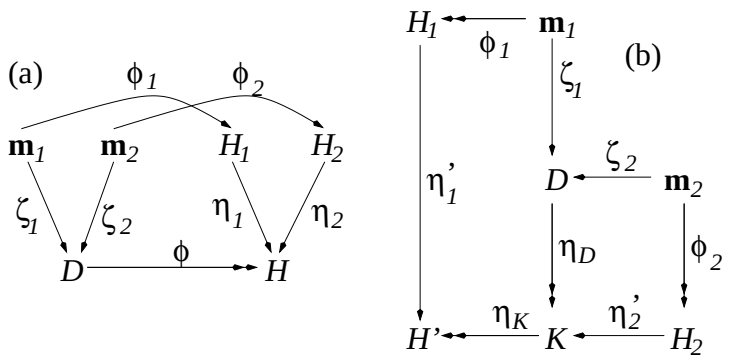
$$L \overset{r}{\Longrightarrow} R \qquad \frac{H_1 \overset{r}{\Longrightarrow} H_1'}{(H_1, \zeta_1) \otimes (H_2, \zeta_2) \overset{r}{\Longrightarrow} (H_1', \zeta_1) \otimes (H_2, \zeta_2)}$$

Proposition 1. *Let G, H be hypergraphs and let $r = (L, R)$ be a rewriting rule. Then $G \overset{r}{\Longrightarrow} H$ if and only if G can be transformed into H by r in the double-pushout approach (with a production span $L \twoheadleftarrow \mathbf{m} \twoheadrightarrow R$ where $m = ar(L) = ar(R)$).*

Proof. It was already shown in [1] that the expressive power of rewriting in terms of graph expressions is equal to the expressive power of the double-pushout approach (where components of a production span might be non-injective). The proposition follows from the fact that graph expressions are equivalent to our form of graph construction (see section 3.1).

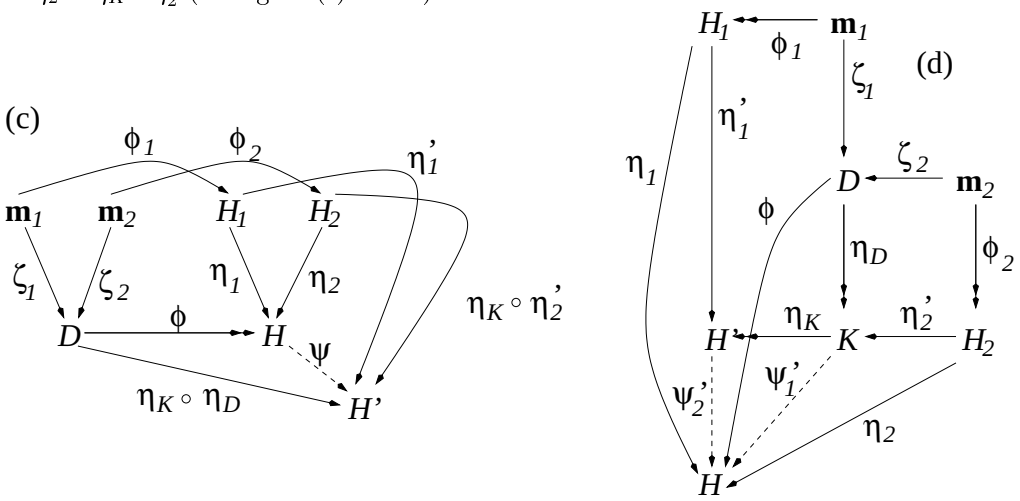
We now show the proposition in a more direct way and explain in detail how the double-pushout approach relates to our approach:

– first, we prove that if H is a colimit of $\zeta_1, \zeta_2, \phi_1, \phi_2$ in diagram (a) and diagram (b) consists of two pushouts, then $H \cong H'$.



Diagrams (a) and (b) can always be completed to form colimits, since we can always explicitly construct the colimit as in the remark after definition 2.

- We first construct a morphism $\psi : H \to H'$. From diagram (b) it follows that $\eta_K \circ \eta_D : D \to H'$, $\eta_1' : H_1 \to H'$ and $\eta_K \circ \eta_2' : H_2 \to H'$. Since diagram (a) is a colimit this implies the existence of a morphism $\psi : H \to H'$ satisfying $\psi \circ \phi = \eta_K \circ \eta_D$, $\psi \circ \eta_1 = \eta_1'$ and $\psi \circ \eta_2 = \eta_K \circ \eta_2'$ (see figure (c) below).

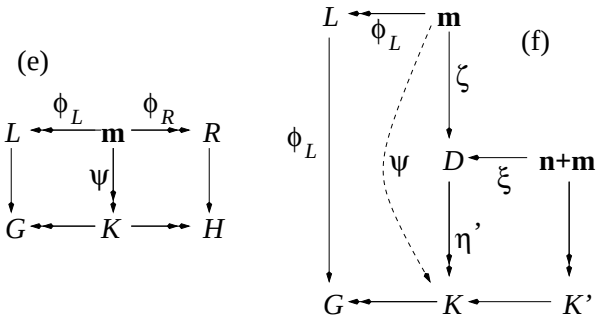


- In the same way we can show that there is a morphism $\psi_2' : H' \to H$: from diagram (a) it follows that $\phi : D \to H$ and $\eta_2 : H_2 \to H$. This implies the existence of $\psi_1' : K \to H$ satisfying $\psi_1' \circ \eta_2' = \eta_2$ and $\psi_1' \circ \eta_D = \phi$. Since there is a morphism $\eta_1 : H_1 \to H$, it follows that there exists a morphism $\psi_2' : H' \to H$ satisfying $\psi_2' \circ \eta_1' = \eta_1$ and $\psi_2' \circ \eta_K = \psi_1'$.

- We will now show that $\psi_2' \circ \psi = id_{H'}$: we know that $(\psi_2' \circ \psi) \circ \phi = \psi_2' \circ \eta_K \circ \eta_D = \phi$, $(\psi_2' \circ \psi) \circ \eta_1 = \psi_2' \circ \eta_1' = \eta_1$ and $(\psi_2' \circ \psi) \circ \eta_2 = \psi_2' \circ \eta_K \circ \eta_2' = \eta_2$. Since a morphism satisfying these conditions is unique and there is already one morphism ($id_{H'}$) satisfying them, it follows that $\psi_2' \circ \psi = id_{H'}$.
- The next step is to show that also $\psi \circ \psi_2' = id_H$. We first show that $\psi \circ \psi_1' = \eta_K$ by regarding the pushout on the left-hand side in diagram (b) and the hypergraph H': we know that $\eta_K \circ \eta_D = \psi \circ \phi = (\psi \circ \psi_1') \circ \eta_D$ and $\eta_K \circ \eta_2' = \psi \circ \eta_2 = (\psi \circ \psi_1') \circ \eta_2'$. Since the morphism satisfying these conditions is unique it follows that $\eta_K = \psi \circ \psi_1'$.
  Furthermore it follows that $(\psi \circ \psi_2') \circ \eta_1' = \psi \circ \eta_1 = \eta_1'$ and $(\psi \circ \psi_2') \circ \eta_K = \psi \circ \psi_1' = \eta_K$. Since (because of the pushout on the right-hand side in diagram (b)) there is a unique morphism satisfying these conditions, and we know that id_H is satisfying them, it follows that $\psi \circ \psi_2' = id_H$.
- Since $\psi_2' \circ \psi = id_{H'}$ and $\psi \circ \psi_2' = id_H$ it follows that ψ is bijective. It is left to show that it is a strong morphism: $\psi(\chi_H) = \psi(\phi(\chi_D)) = \eta_K(\eta_D(\chi_D)) = \chi_{H'}$. ($\phi, \eta_K, \eta_D$ are strong by definition.)

– We now assume that $G \overset{(L,R)}{\Rightarrow} H$. Because of proposition 2 in section 4 we can assume that each of the two rewrite rules generating $\overset{(L,R)}{\Rightarrow}$ is used exactly once. Thus we can conclude that $G \cong (L, \zeta_1) \otimes (J, \zeta_2)$ and $H \cong (R, \zeta_1) \otimes (J, \zeta_2)$. We now set $K = \otimes(J, \zeta_2)$ (as in the pushout on the right-hand side in diagram (b)). Then G is the pushout of $\eta_D \circ \zeta_1$ and ϕ_1 (the canonical strong morphism) in diagram (b). (We assume that $H_1 = L$, $H_2 = J$ and $H' = G$.) In the same way we can describe H as a pushout of $\eta_D \circ \zeta_1$ and the canonical strong morphism from $\mathbf{m_1}$ into R.

– Now assume we can transform G into H by applying the production span $L \twoheadleftarrow \mathbf{m} \twoheadrightarrow R$ in the double-pushout approach. (In [4] it was shown that every double-pushout can be converted into a double-pushout where the middle graph in the production span is discrete. We can also assume that all its nodes are external, that it is therefore isomorphic to $\mathbf{m}$ and that there are strong morphisms $\phi_L : \mathbf{m} \twoheadrightarrow L$ and $\phi_R : \mathbf{m} \twoheadrightarrow R$.) A double-pushout has the form shown in figure (e).



We assume that K' is the same hypergraph as K with a different sequence of external nodes. We obtain K' by replacing the sequence of external nodes of K by $\chi_K \circ \psi(\chi_{\mathbf{m}})$. Let $\xi : \mathbf{n} + \mathbf{m} \to D$ (where $n = ar(K)$) be a discrete morphism such that $K \cong \otimes(K', \xi)$, i.e. ξ hides the last m nodes of K' (compare with section 3.1). Now let $\zeta : \mathbf{m} \to D$ be a discrete morphism such that[4] $\zeta(\chi_{\mathbf{m}}) = \xi(\lfloor\chi\rfloor_{n+1\ldots n+m})$, therefore $\eta'(\zeta(\chi_{\mathbf{m}})) = \eta'(\xi(\lfloor\chi\rfloor_{n+1\ldots n+m})) = \psi(\chi_{\mathbf{m}})$ and $\eta' \circ \zeta = \psi$. (η' is generated by the pushout in figure (f).)
This implies that $G \cong (L, \zeta) \otimes (K', \xi)$. And in the same way we can show that $H \cong (R, \zeta) \otimes (K', \xi)$. It thus follows that $G \overset{(L,R)}{\Rightarrow} H$.

[4] Let $s = a_1 \ldots a_n$ be a string of elements. Then $\lfloor s \rfloor_{i_1 \ldots i_m} = a_{i_1} \ldots a_{i_m}$

□

4 Some Properties of Hypergraph Construction

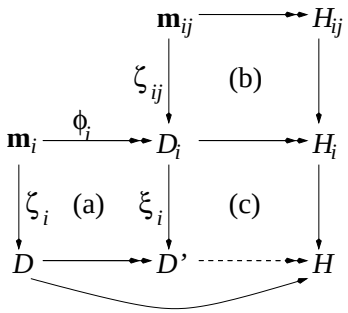
As promised in the previous section we now introduce a mechanism for combining several construction operations into one by collapsing hierarchies of graph construction. In the world of strings this has a rough analogue in the associativity of concatenation, which does not hold for graph construction.

Proposition 2. *In the following let i range over $[n]$ and j range over $[n_i]$. Let $\zeta_{ij} : \mathbf{m_{ij}} \to D_i$ and $\zeta_i : \mathbf{m_i} \to D$ be morphisms with $m_i = ar(D_i)$. Let $\phi_i : \mathbf{m_i} \twoheadrightarrow D_i$ be the unique strong morphisms and let the ξ_i be the morphisms generated by colimit (a) in the figure below. Then it holds for arbitrary hypergraphs H_{ij} with $m_{ij} = ar(H_{ij})$ that*

$$\bigotimes_{i=1}^{n}(\bigotimes_{j=1}^{n_i}(H_{ij}, \zeta_{ij}), \zeta_i) \cong \bigotimes_{i,j}(H_{ij}, \xi_i \circ \zeta_{ij}) \tag{1}$$

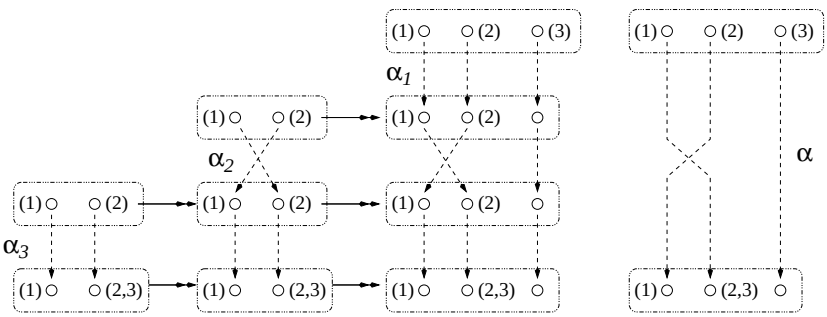
Proof.

The proof of the proposition is shown in the figure. While the left-hand side of equation (1) is formed by applying first colimits (b) to the H_{ij} followed by an application of colimit (a)+(c), the same goal can be achieved by forming colimit (a) first and applying colimits (b)+(c) afterwards. This argumentation is valid since the combination of two colimits always yields another colimit.



□

Example: we translate a sequence of redefinitions of external node into graph constructions and collapse them into one single application of a colimit. Assume we want to compute $\sigma_\alpha(H)$ where $ar(H) = 3$, $\alpha : [3] \to [3]$ and $\alpha(1) = 2$, $\alpha(2) = \alpha(3) = 1$. As explained in section 3.1 we can decompose α into $\alpha = \alpha_1 \circ \alpha_2 \circ \alpha_3$ with $\alpha_1 : [3] \to [2]$ (hiding an external node), $\alpha_2 : [2] \to [2]$ (permutation, exchanging the first and the second node) and $\alpha_3 : [2] \to [3]$ (duplicating an external node).
We can now construct the respective discrete morphisms ζ_1, ζ_2 and ζ_3 (according to figure 1) and combine them according to proposition 2. The result is ζ shown in the figure below. That is $\sigma_\alpha(H) \cong \otimes(H, \zeta)$ for every hypergraph H of arity 3.



Hyperedges are the basic units of graph construction. Just as every element of a vector space can be decomposed into base vectors in a unique way, there is a unique decomposition of every hypergraph into hyperedges. Note that isolated nodes are created by nodes of the discrete hypergraph D which are not in the range of any of the ζ_i.

Proposition 3. (Unique Factorization) *Let H be a hypergraph. Then there exists a natural number n, labels l_i and morphisms $\zeta_i : \mathbf{m_i} \to D$ (where $i \in [n]$ and D is a discrete hypergraph) such that $H \cong \bigotimes_{i=1}^{n}([l_i]_{m_i}, \zeta_i)$. This factorization is unique up to isomorphism and index permutation.*

Proof. With section 3.1 and [1] it is straightforward to see that such a factorization of a hypergraph H is always possible (isolated nodes in H are generated by nodes of the discrete graph G which are not in the range of any of the ζ_i.)

We will now show that the factorization is unique. Let

$$H \cong \bigotimes_{i=1}^{n}([l_i]_{m_i}, \zeta_i) \quad H \cong \bigotimes_{i=1}^{n'}([l'_i]_{m'_i}, \zeta'_i)$$

where $\zeta_i : \mathbf{m_i} \to D$, $i \in [n]$ respectively $\zeta'_i : \mathbf{m'_i} \to D$, $i \in [n']$. Let $\eta_i : [l_i]_{m_i} \to H$, $i \in [n]$ and $\phi : D \twoheadrightarrow H$ respectively $\eta'_i : [l'_i]_{m'_i} \to H$, $i \in [n']$ and $\phi' : D' \twoheadrightarrow H$ be the morphisms generated by the colimit.

Since n respectively n' stand for the number of edges of H it follows that $n = n'$. And since the η_i respectively η'_i are the embeddings of the edges into H, it follows that there is a permutation $\alpha : [n] \to [n]$ such that $\eta_i = \eta'_{\alpha(i)}$, which implies that $m_i = m'_{\alpha(i)}$ and $l_i = l'_{\alpha(i)}$.

It is left to show that ζ_i and $\zeta'_{\alpha(i)}$ are equal up to isomorphism. We prove that ϕ_V is a bijection (the same holds for ϕ'_V with the same arguments). From the remark after definition 2 it follows that a hypergraph isomorphic to H can be constructed by taking the union of D and the edges $[l_i]_{m_i}$ and fusing the nodes according to the equivalence $\approx$. Because of the special nature of the hypergraphs which are concatenated (the $[l_i]_{m_i}$ have no duplicates in their sequences of external nodes and no internal nodes), it follows that every nodes of an edge $[l_i]_{m_i}$ is related to some node in D and that no two different nodes of D are related. That is, the equivalence classes are exactly the nodes of D. Since ϕ_V is isomorphic to a function that maps every node of D to its equivalence class, it follows that $\phi_V : V_D \to V_H$ is a bijection.

It follows that

$$(\zeta_i)_V = \phi_V^{-1} \circ (\eta_i)_V \circ (\phi_i)_V = \phi_V^{-1} \circ (\eta'_{\alpha(i)})_V \circ (\phi'_{\alpha(i)})_V = \phi_V^{-1} \circ \phi'_V \circ (\zeta'_{\alpha(i)})_V$$

$\phi_V^{-1} \circ \phi'_V : V_{D'} \to V_D$ is a bijection and since ζ_i, $\zeta'_{\alpha(i)}$ are defined only on discrete graphs they are the same up to isomorphism. □

As in vector spaces we can define linear mappings on hypergraphs.

Definition 3. (Linear Mapping) *A* linear mapping *L maps hypergraphs to hypergraphs of the same arity and satisfies $L(\bigotimes_{i=1}^{n}(H_i, \zeta_i)) \cong \bigotimes_{i=1}^{n}(L(H_i), \zeta_i)$.*

Proposition 4. (Unique Linear Mapping) *For each mapping of hyperedges $[l]_m$ to hypergraphs of arity m, there is exactly one linear mapping (up to isomorphism) which is an extension of the original mapping.*

Proof. Let L' map hyperedges to hypergraphs of the same arity.

- We first show that there is at most one linear mapping L which extends L': let H be an arbitrary hypergraph and let $H \cong \bigotimes_{i=1}^{n}([l_i]_{m_i}, \zeta_i)$ be the unique factorization of H according to proposition 3. If there is a linear mapping L it satisfies: $L(H) \cong \bigotimes_{i=1}^{n}(L'([l_i]_{m_i}), \zeta_i)$ and $L(H)$ is fixed up to isomorphism.
- We now show that there is at least one linear mapping L. We define L as follows:

$$L(\bigotimes_{i=1}^{n}([l_i]_{m_i}, \zeta_i)) = \bigotimes_{i=1}^{n}(L'([l_i]_{m_i}), \zeta_i)$$

This is well-defined since the factorization of every hypergraph is unique. It is left to show that L is a linear mapping.

Let $H \cong \bigotimes_{i=1}^{n}(H_i, \zeta_i)$. According to proposition 3 $H_i \cong \bigotimes_{j=1}^{n_i}([l_{ij}]_{m_{ij}}, \zeta_{ij})$ for suitable $l_{ij}, m_{ij}, \zeta_{ij}$. It follows that

$$L(H) \cong L(\bigotimes_{i=1}^{n}(\bigotimes_{j=1}^{n_i}([l_{ij}]_{m_{ij}}, \zeta_{ij}), \zeta_i))$$

From proposition 2 follows the existence of morphisms ξ_i such that

$$L(H) \cong L(\bigotimes_{i,j}([l_{ij}]_{m_{ij}}, \xi_i \circ \zeta_{ij})) = \bigotimes_{i,j}(L'([l_{ij}]_{m_{ij}}), \xi_i \circ \zeta_{ij})$$

Again from proposition 2 it follows that

$$L(H) \cong \bigotimes_{i=1}^{n}(\bigotimes_{j=1}^{n_i}(L'([l_{ij}]_{m_{ij}}), \zeta_{ij}), \zeta_i) \cong \bigotimes_{i=1}^{n}(L(H_i), \zeta_i)$$

□

One can view the application of a linear mapping as a synchronous rewriting step, replacing every hyperedge at the same time.

Examples: we define a mapping that duplicates every edge in a hypergraph. If $H = (V, E, s, l, \chi)$ is a hypergraph, $dupl(H)$ is defined by $(V, E \cup \bar{E}, s \cup \bar{s}, l \cup \bar{l}, \chi)$ where $\bar{E} = \{\bar{e} \mid e \in E\}$, $\bar{s} : \bar{E} \to V^*$ with $\bar{s}(\bar{e}) = s(e)$ and $\bar{l} : \bar{E} \to L$ with $\bar{l}(\bar{e}) = l(e)$.
dupl is a linear mapping and can be generated by fixing the images of single hyperedges: $dupl([l]_m) = [l]_m \Box [l]_m$ (where $\Box$ is the operator defined in section 2).
Note that a mapping that duplicates all nodes, can not be linear.

Another simple example is a mapping that deletes all edges, i.e. produces a discrete graph. We define $discrete(H) = (V, \emptyset, \emptyset, \emptyset, \chi)$. It is linear and can be generated by defining $discrete([l]_m) = \mathbf{m}$ for all hyperedges.

The usefulness of linear mappings will become clear in the following section where we use a linear mapping in order to analyse mobile processes. Another important use of linear mappings is to annotate hypergraphs. We can extend the construction operation in order to concatenate hypergraphs with annotations (e.g. nodes labelled with monoid elements), as will be done in section 6. We can then define an extended notion of a linear mapping which maps hypergraph to annotated hypergraph and satisfies

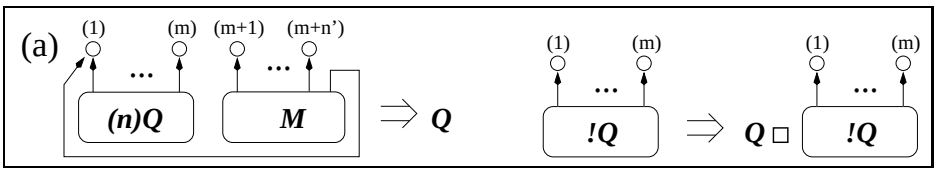
$$L(\bigotimes_{i=1}^{n}(H_i, \zeta_i)) \cong \bigotimes_{i=1}^{n}(L(H_i), \zeta_i)$$

where we use the extended construction operation on the right-hand side.

5 Typed Process Graphs

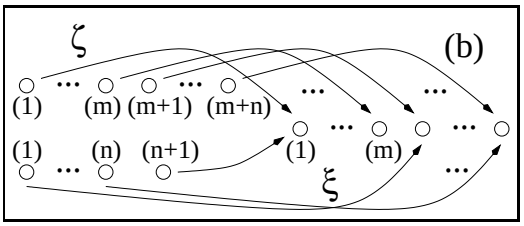
We show how to model a process calculus, closely related to the asynchronous polyadic π-calculus [14], by so-called process graphs. There is an encoding from the π-calculus into process graphs [10, 9]. On the other hand there is a straightforward encoding of process graphs into closed action calculi [7] and a close relation of our process graphs to the ones in [16]. A process graph is defined inductively in the following way.

Definition 4. *(***Process Graphs***)* *A* process graph *P is inductively defined as follows: P is a hypergraph where each edge e is either labelled with $(n)Q$ where Q is again a process graph and $1 \leq n \leq ar(Q)$ (e is a process waiting for a message with n ports arriving at its first node), with $!Q$ (e is a process which can replicate itself creating aribitrary many instances of Q) or with the constant M (e is a message sent to its last node). The reduction relation (reception of a message and its nodes by a process) is generated by the rewrite rules in figure (a) and is closed under graph construction.*



The rewrite rule in (a) is not always defined, it may fail if $ar(Q) \neq m + n'$ in the message reception rule or $ar(Q) \neq m$ in the replication rule. Furthermore we want to avoid that $n \neq n'$ in the message reception rule, that is we want to ensure that the expected number of nodes is received. We use morphisms, graph construction and a linear mapping in order to define a condition which is sufficient for avoiding this kind of runtime errors and which can be checked statically.

Proposition 5. *Let L be a linear mapping which is defined on the hyperedges as follows: $L([M]_n) = [t]_n$ (t is a new edge label), $L([!Q]_m) = L(Q)$ if $m = ar(Q)$ and $L([(n)Q]_m) = (L(Q), \zeta) \otimes ([t]_{n+1}, \xi)$ if $n + m = ar(Q)$ (undefined otherwise). ζ, ξ are defined in figure (b).*



Let P be a process graph. If there exists a strong morphism $\psi : L(P) \twoheadrightarrow H$ into a hypergraph H which satisfies

$$e_1, e_2 \in E_H, \ \lfloor s_H(e_1) \rfloor_{ar(e_1)} = \lfloor s_H(e_2) \rfloor_{ar(e_2)} \ \Rightarrow \ e_1 = e_2 \tag{2}$$

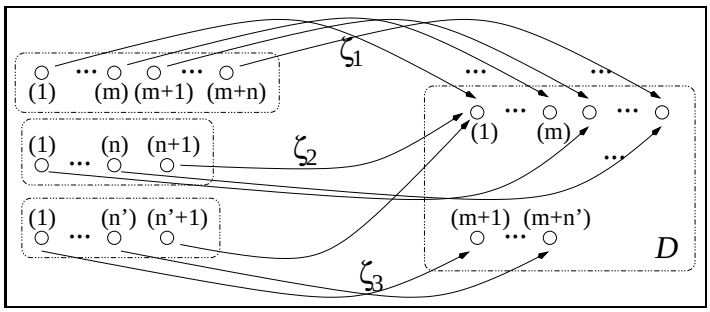
(i.e. all messages that share the last node are already the same) then P will never encounter a runtime error during reduction.

Proof. Let P be a process graph with $\psi : L(P) \twoheadrightarrow H$ where H satisfies condition 2.

We now show that P does not encounter a runtime error in its next reduction step, along with the fact that the subject reduction property ist satisfied. The subject reduction property says that if $P \Rightarrow P'$, then there is also a strong morphism $\psi' : L(P') \twoheadrightarrow H$. These two properties together ensure absence of runtime errors for the entire reduction.

We proceed by induction on the reduction rules:

– Let P be the left-hand side of the message reception rule in figure (a). From proposition 2 it follows that $L(P) \cong (L(Q), \zeta_1) \otimes ([t]_{n+1}, \zeta_2) \otimes ([t]_{n'+1}, \zeta_3) =: R$ where $\zeta_1, \zeta_2, \zeta_3$ are defined as follows:



The condition in the definition of L tells us that $m + n = ar(Q)$ and since we will later show that $n = n'$ we have thus eliminated the first sort of runtime error.

Now let $\eta_1 : L(Q) \to R$, $\eta_2 : [t]_{n+1} \to R$, $\eta_3 : [t]_{n'+1} \to R$ be the embeddings into R generated by the colimit. Furthermore we know that there exists a strong morphism $\psi : R \twoheadrightarrow H$. Our aim is to show that $\psi \circ \eta_1 : L(Q) \to H$ is the strong morphism we are looking for. We proceed in two steps:

- We first show that $\psi(\eta_1(\lfloor\chi_{l(Q)}\rfloor_{1\ldots m})) = \lfloor\chi_H\rfloor_{1\ldots m}$:

$$\begin{aligned}&\psi(\eta_1(\lfloor\chi_{L(Q)}\rfloor_{1\ldots m})) = \psi(\eta_1(\phi_1(\lfloor\chi_{\mathbf{m+n}}\rfloor_{1\ldots m})))\\ &= \psi(\phi(\zeta_1(\lfloor\chi_{\mathbf{m+n}}\rfloor_{1\ldots m}))) = \psi(\phi(\lfloor\chi_D\rfloor_{1\ldots m})) = \psi(\lfloor\chi_R\rfloor_{1\ldots m})\\ &= \lfloor\chi_H\rfloor_{1\ldots m}\end{aligned}$$

  (ϕ_1 ist the canonical strong morphism from $\mathbf{m}+\mathbf{n}$ into $L(Q)$ and $\phi : D \twoheadrightarrow R$ is the strong morphism generated by the colimit.)
- In the next step we show that $\psi(\eta_1(\lfloor\chi_{L(Q)}\rfloor_{m+1\ldots m+n})) = \lfloor\chi_H\rfloor_{m+1\ldots m+n'}$ and thus $n = n'$ which avoids the second sort of runtime errors.
  Let e be the one edge in $[t]_{n+1} = T$ and let e' be the one edge in $[t]_{n'+1} = T'$. It follows that

$$\begin{aligned}&\lfloor s_H(\psi(\eta_2(e)))\rfloor_{ar(e)} = \lfloor\psi(\eta_2(s_T(e)))\rfloor_{n+1} = \lfloor\psi(\eta_2(\chi_T))\rfloor_{n+1}\\ &= \lfloor\psi(\eta_2(\phi_2(\chi_{\mathbf{n+1}})))\rfloor_{n+1} = \lfloor\psi(\phi(\zeta_2(\chi_{\mathbf{n+1}})))\rfloor_{n+1}\\ &= \lfloor\psi(\phi(\zeta_3(\chi_{\mathbf{n'+1}})))\rfloor_{n'+1} = \lfloor\psi(\eta_3(\phi_3(\chi_{\mathbf{n'+1}})))\rfloor_{n'+1} = \lfloor\psi(\eta_3(\chi_{T'}))\rfloor_{n'+1}\\ &= \lfloor\psi(\eta_3(s_{T'}(e)))\rfloor_{n'+1} = \lfloor s_H(\psi(\eta_3(e)))\rfloor_{ar(e')}\end{aligned}$$

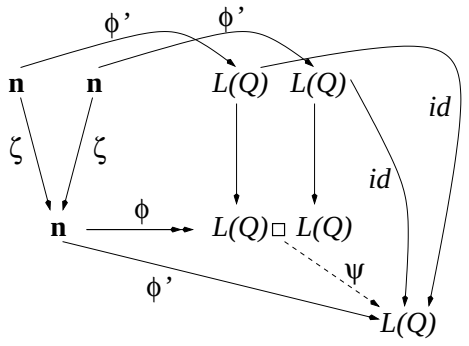
  (ϕ_2 respectively ϕ_2 are the canonical strong morphisms from $\mathbf{n}+\mathbf{1}$ into $[t]_{n+1}$ respectively from $\mathbf{n'}+\mathbf{1}$ into $[t]_{n'+1}$.) Condition (2) implies that $\psi(\eta_2(e)) = \psi(\eta_3(e'))$. It follows that $n+1 = ar(e) = ar(\psi(\eta_2(e))) = ar(\psi(\eta_3(e'))) = ar(e') = n'+1$ and thus $n = n'$. Now

$$\begin{aligned}&\psi(\eta_1(\lfloor\chi_{L(Q)}\rfloor_{m+1\ldots m+n})) = \psi(\eta_1(\phi_1(\lfloor\chi_{\mathbf{m+n}}\rfloor_{m+1\ldots m+n})))\\ &= \psi(\phi(\zeta_1(\lfloor\chi_{\mathbf{m+n}}\rfloor_{m+1\ldots m+n}))) = \psi(\phi(\zeta_2(\lfloor\chi_{\mathbf{n+1}}\rfloor_{1\ldots n})))\\ &= \psi(\eta_2(\phi_2(\lfloor\chi_{\mathbf{n+1}}\rfloor_{1\ldots n}))) = \psi(\eta_2(\lfloor s_T(e)\rfloor_{1\ldots n})) = \lfloor s_H(\psi(\eta_2(e)))\rfloor_{1\ldots n}\\ &= \lfloor s_H(\psi(\eta_3(e')))\rfloor_{1\ldots n} = \psi(\eta_3(\lfloor s_{T'}(e')\rfloor_{1\ldots n})) = \psi(\eta_3(\psi_3(\lfloor\chi_{\mathbf{n'+1}}\rfloor_{1\ldots n})))\\ &= \psi(\phi(\zeta_3(\lfloor\chi_{\mathbf{n'+1}}\rfloor_{1\ldots n}))) = \psi(\phi(\lfloor\chi_D\rfloor_{m+1\ldots m+n'})) = \lfloor\chi_H\rfloor_{m+1\ldots m+n'}\end{aligned}$$

  Together we conclude that $\psi(\eta_1(\chi_{L(Q)})) = \chi_H$ and $\psi \circ \eta_1$ is thus the desired strong morphism.

– Let $P = [!Q]_n$ be the left-hand side of the replication rule in figure (a). Then $P \Rightarrow P'$ and $P' = Q\square[!Q]_n$. $L(P) = L(Q)$ and $L(P') = L(Q)\square L(Q)$. Since $L(P) = L(Q)$ is defined, it follows that $ar(Q) = m$ and therefore $L(P')$ is also defined. We have to show that there exists a strong morphism $\psi : L(Q)\square L(Q) \twoheadrightarrow L(Q)$ ($L(Q)\square L(Q) = (L(Q),\zeta) \otimes (L(Q),\zeta)$ where $\zeta : \mathbf{n} \twoheadrightarrow \mathbf{n}$).

  Obviously the identity morphism $id : L(Q) \twoheadrightarrow L(Q)$ is a strong morphism. Furthermore there is a unique strong morphism $\phi' : \mathbf{n} \twoheadrightarrow L(Q)$ and it holds that $\phi' \circ \zeta = id \circ \phi'$. Now let $\phi : \mathbf{n} \twoheadrightarrow L(Q)\square L(Q)$ be the morphism generated by the colimit. The properties of a colimit imply the existence of a morphism $\psi : L(Q)\square L(Q) \to L(Q)$ such that $\psi \circ \phi = \phi'$. Since ϕ and ϕ' are both strong, it follows that ψ is also a strong morphism.



– Let

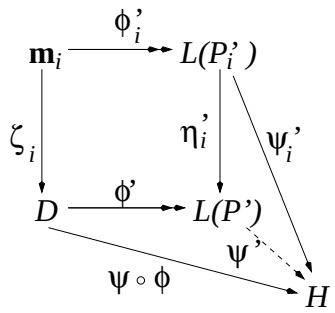
$$\frac{P_1 \to P_1',\ P_2 \cong P_2'}{P = (P_1,\zeta_1) \otimes (P_2,\zeta_2) \Rightarrow (P_1',\zeta_1) \otimes (P_2',\zeta_2) = P'}$$

  $L(P) \cong (L(P_1),\zeta_1) \otimes (L(P_2),\zeta_2)$. Let $\eta_i : L(P_i) \to L(P)$, $i \in [2]$ be the embeddings and let $\phi : D \to L(P)$ be the strong morphism generated by the colimit. It follows that[5]

$$\psi \circ \eta_i : L(P_i) \twoheadrightarrow H[\psi(\eta_i(\chi_{L(P_i)}))]$$

[5] Let H be a hypergraph. $H[\chi']$ is the hypergraph we obtain from H by replacing χ_H with χ'.

The induction hypothesis implies that there is a strong morphism $\psi_1' : L(P_1') \twoheadrightarrow H[\psi(\eta_i(\chi_{L(P_i)}))]$. Furthermore we set $\psi_2' = \psi \circ \eta_2$.
Let $m_i = ar(P_i) = ar(P_i')$. Now let $\phi_i' : \mathbf{m_i} \twoheadrightarrow L(P_i')$ be the canonical strong morphisms, and let $\eta_i' : L(P_i') \to L(P')$ and $\phi' : D \twoheadrightarrow L(P')$ be the morphisms generated by the colimit $L(P') \cong (L(P_1'), \zeta_1) \otimes (L(P_2'), \zeta_2)$.
We know that $\psi_i' : L(P_i') \to H$ and $\psi \circ \phi' : D \to H$. If we can show that $\psi_i' \circ \phi_i' = (\psi \circ \phi) \circ \zeta_i$ the properties of the colimit guarantee the existence of a morphism $\psi' : L(P') \to H$. And since $\psi' \circ \phi' = \psi \circ \phi$ and ψ, ϕ, ϕ' are strong, it follows that ψ' is also a strong morphism.



It is left to show that $\zeta_i \circ \psi_i' = (\psi \circ \phi) \circ \zeta_i$:

$$\psi_i'(\phi_i'(\chi_{\mathbf{m_i}})) = \psi_i'(\chi_{L(P_i')})$$
$$= \psi(\eta_i(\chi_{L(P_i)})) = \psi(\phi(\zeta_i(\chi_{\mathbf{m_i}})))$$

And thus $\psi_i' \circ \phi_i' = (\psi \circ \phi) \circ \zeta_i$.

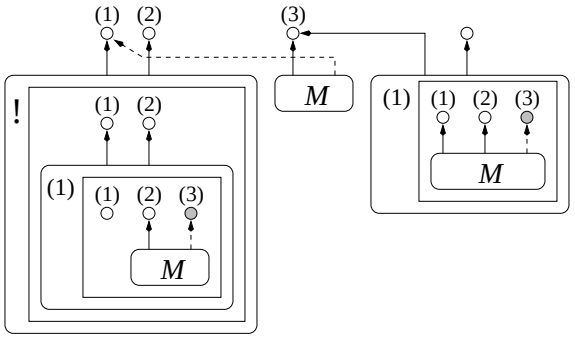
□

L extracts pure communication structure from a process graph, i.e. an edge of the form $[t]_n$ indicates that its nodes (except the last) might be sent or received via its last node. Condition (2) makes sure that the arity of the arriving message matches the expected arity and that nodes that might get fused during reduction are already fused in H. It thus guarantees absence of undefined rewrites for the entire reduction.

H can be regarded as a type of P and we can easily unfold H into well-known type trees of π-calculus processes [15]. The method presented above corresponds to a type system for the π-calculus with recursive types and simple polymorphism. The type of a process can also be specified by inductive typing rules rather than by a linear mapping.

More expressive (generic) type systems for the analysis of processes, which can be nicely integrated into the graph-based setting, can be found in [10, 9].

As an example we regard the following process graph:



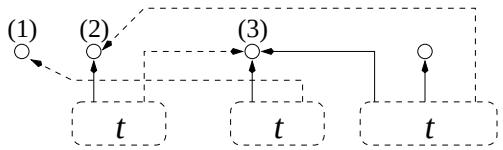
There is a server (the process on the left) which receives messages on its first port (before the server can receive a message it must create a copy of itself) and sends back its second port to the address that was attached to the message. At this address another process (the process on the right) is waiting, it receives the message with the second port of the server and sends its own message there.

We use the following syntactic sugar: the last node of a message (i.e. the node or port to which the message is sent) is connected with the message by a dashed line. Nodes of hypergraph in inner level which will be merged with nodes attached to a message are shaded grey.

If we represent the three external ports by a, b, c respectively and denote the internal node by d, the process graph above corresponds to the following process in the π-calculus [14]:

$$!a(x).\bar{x}\langle b\rangle \mid \bar{a}\langle c\rangle \mid (\nu d)(c(y).\bar{y}\langle c, d\rangle)$$

Typing the process graph above intuitively means flattening the hypergraph until it consists of only one hierarchy level. Hyperedges representing replicating processes are discarded, messages and processes waiting for a message are appropriately replaced by edges labelled t. After folding the hypergraph (i.e. merging the hyperedges according to condition (2)) we obtain the following type graph T:



We can now reconstruct the type assignment of the corresponding π-calculus process (a type assignment consists of type trees for every free name) by unfolding the type graph starting at one of the external nodes. The nodes of the graph T are transformed into nodes of the tree: if there is an edge e with $s_T(e) = v_1 \dots v_{n-1} v_n$ then $v_1, \dots, v_{n-1}$ are the children of the parent v_n (in that order). In this case the unfolding of the type graph leads to infinite trees which can be represented by using a recursion operator μ. Thus a valid type assignment for the π-calculus process above is

$$a : [\mu\alpha[[\alpha, \beta]]],\ b : \mu\alpha[[\alpha], \beta],\ c : \mu\alpha[[\alpha, \beta]]$$

$[t_1, \dots, t_n]$ is a tree with subtrees $t_1, \dots, t_n$, α and β are tree-valued variables. There is some polymorphism in the type of the process: β is not bound and represents an arbitrary type tree.

6 A Compositional Semantics for Petri Nets

We will now show another application of graph construction by giving a compositional semantics for Petri nets. A Petri net can easily be represented by a hypergraph H (compare with [12]): nodes are places and edges are transitions. Since we do not distinguish source and target nodes a priori, we partition the nodes of an edge into sources and targets with the labelling function $l : E \to \mathbb{N} \times \mathbb{N}$. If $l(e) = (s, t)$ then $s + t = ar(e)$, s is the number of sources (the first s nodes) and t (the last t nodes) is the number of targets. We also need an additional labelling $z : V_H \to \mathit{Mon}$ mapping each node to an element of a cancellative commutative monoid, in order to represent the tokens present at each node. In our example we will set $\mathit{Mon} = \mathbb{N}$, but we could also represent high-level Petri nets by assuming that Mon is the set of all multi-sets over certain elements.

A Petri net is now a pair $[H, z]$ where H is a hypergraph with labels taken from $\mathbb{N} \times \mathbb{N}$ and $z : V_H \to \mathit{Mon}$ is a mapping. Before we can define the semantics we first have to extend our notion of hypergraph construction to hypergraphs with tokens. But this is easy since our construction operation yields morphisms of the subgraphs into the constructed graph. Let $[H_i, z_i]$ be Petri nets. We define:

$$\bigotimes_{i=1}^{n}([H_i, z_i], \zeta_i) \cong [\bigotimes_{i=1}^{n}(H_i, \zeta_i), z] \text{ where } z(v) = \sum_{i=1}^{n} \sum_{\eta_i(w)=v} z_i(w)$$

The η_i are the morphisms of H_i into $\bigotimes_{i=1}^{n}(H_i, \zeta_i)$ yielded by the colimit. $\sum$ is the commutative operation of Mon.

Inductive Definition of Petri Nets: a Petri net N is either of the form $[[s, t]_{s+t}, z]$ where $z : V_{[s,t]_{s+t}} \to \mathit{Mon}$ or $\bigotimes_{i=1}^{n}(N_i, \zeta_i)$ with adequate discrete morphisms ζ_i, where the N_i are again Petri nets.

Semantics of Petri Nets: we now assume that $\mathit{Mon} = \mathbb{N}$. A single transition fires if all its source nodes are labelled with tokens. And if one transition fires, the entire net is changed accordingly.

$$[T,z] \Longrightarrow [T,z'] \qquad \frac{N_1 \Longrightarrow N_1'}{(N_1,\zeta_1)\otimes(N_2,\zeta_2) \Longrightarrow (N_1',\zeta_1)\otimes(N_2,\zeta_2)}$$

where $T = [s,t]_{s+t}$ is a transition, $z, z' : V_T \to \{0,1\}$ and $z(\lfloor \chi_T \rfloor_i) = 1 \iff i \in [s] \iff z'(\lfloor \chi_T \rfloor_i) = 0$.

This approach works also when a place is allowed to hold more than one token. In this case the extra tokens which are not needed to fire a transition are not attached to the transition (or edge) itself but to the external nodes of the surrounding hypergraphs.

7 Conclusion

We have presented a method of hypergraph construction which allows us to build hypergraphs out of smaller ones. The basic units are single edges. This method can be used to give an operational semantics to concurrent systems whose states can often be represented by hypergraphs in a natural way.

Another approach concerning inductive graph representation was given in [5] with a different categorical representation of graphs (in this paper graphs are the *morphisms* of a category) and different operators on graphs.

Future work will consist in designing further techniques for the analysis of concurrent systems with a hypergraph-based semantics. One promising direction is to continue the work on generic type systems for process graphs [10] and to extend it to more general graph rewrite systems. Furthermore we plan to investigate the connection between our approach to model Petri nets and existing approaches as in [13, 12].

Remark: this technical report is the extended version of [11].

Acknowledgements: I would like to thank my colleagues and my former advisor Jürgen Eickel. Furthermore I want to express my gratitude to the anonymous referees for their valuable comments.

References

1. Michel Bauderon and Bruno Courcelle. Graph expressions and graph rewritings. *Mathematical Systems Theory*, 20:83–127, 1987.
2. Roy L. Crole. *Categories for Types*. Cambridge University Press, 1993.
3. H. Ehrig. Introduction to the algebraic theory of graphs. In *Proc. 1st International Workshop on Graph Grammars*, pages 1–69. Springer-Verlag, 1979. LNCS 73.
4. H. Ehrig, M. Pfender, and H. Schneider. Graph grammars: An algebraic approach. In *Proc. 14th IEEE Symp. on Switching and Automata Theory*, pages 167–180, 1973.
5. F. Gadducci and R. Heckel. An inductive view of graph transformation. In *Recent Trends in Algebraic Development Techniques, 12th International Workshop, WADT '97*, volume 1376, pages 223–237, 1997. LNCS 1376.
6. F. Gadducci, R. Heckel, and M. Llabres. A bicategorical axiomatization of concurrent graph rewriting. In *Proceedings of CTCS'99*, 1999. ENTCS 29.
7. Philippa Gardner. Closed action calculi. *Theoretical Computer Science (in association with the conference on Mathematical Foundations in Programming Semantics)*, 1998.
8. Annegret Habel. *Hyperedge Replacement: Grammars and Languages*. Springer-Verlag, 1992. LNCS 643.
9. Barbara König. *Description and Verification of Mobile Processes with Graph Rewriting Techniques*. PhD thesis, Technische Universität München, 1999.
10. Barbara König. Generating type systems for process graphs. In *Proc. of CONCUR '99*, pages 352–367. Springer-Verlag, 1999. LNCS 1664.
11. Barbara König. Hypergraph construction and its application to the compositional modelling of concurrency. In *GRATRA '2000: Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation Systems*, 2000. to appear.

12. H.-J. Kreowski. A comparison between petri-nets and graph grammars. In H. Noltemeier, editor, *Graphtheoretic Concepts in Computer Science*, pages 306–317. Springer-Verlag, 1980. LNCS 100.
13. José Meseguer and Ugo Montanari. Petri nets are monoids. *Information and Computation*, 88(2):105–155, 1990.
14. Robin Milner. The polyadic π-calculus: a tutorial. In F. L. Hamer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*. Springer-Verlag, Heidelberg, 1993.
15. David Turner. *The Polymorphic Pi-Calculus: Theory and Implementation*. PhD thesis, University of Edinburgh, 1995. ECS-LFCS-96-345.
16. Nobuko Yoshida. Graph notation for concurrent combinators. In *Proc. of TPPP '94*. Springer-Verlag, 1994. LNCS 907.

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

Liste aller erschienenen Berichte von 1990-1994
auf besondere Anforderung

342/01/95 A Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids
342/02/95 A Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law
342/03/95 A Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space
342/04/95 A Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes
342/05/95 A Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems
342/06/95 A Maximilian Fuchs: Formal Design of a Model-N Counter
342/07/95 A Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology
342/08/95 A Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies
342/09/95 A Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness
342/10/95 A Ketil Stølen, Max Fuchs: A Formal Method for Hardware/Software Co-Design
342/11/95 A Thomas Schnekenburger: The ALDY Load Distribution System
342/12/95 A Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm
342/13/95 A Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming
342/14/95 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks
342/15/95 A Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs
342/16/95 A Bernhard Schätz, Katharina Spies: Formale Syntax zur logischen Kernsprache der Focus-Entwicklungsmethodik
342/17/95 A Georg Stellner: Using CoCheck on a Network of Workstations
342/18/95 A Arndt Bode, Thomas Ludwig, Vaidy Sunderam, Roland Wismüller: Workshop on PVM, MPI, Tools and Applications
342/19/95 A Thomas Schnekenburger: Integration of Load Distribution into ParMod-C
342/20/95 A Ketil Stølen: Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication
342/21/95 A Andreas Listl, Giannis Bozas: Performance Gains Using Subpages for Cache Coherency Control
342/22/95 A Volker Heun, Ernst W. Mayr: Embedding Graphs with Bounded Treewidth into Optimal Hypercubes
342/23/95 A Petr Jančar, Javier Esparza: Deciding Finiteness of Petri Nets up to Bisimulation
342/24/95 A M. Jung, U. Rüde: Implicit Extrapolation Methods for Variable Coefficient Problems
342/01/96 A Michael Griebel, Tilman Neunhoeffer, Hans Regler: Algebraic Multigrid Methods for the Solution of the Navier-Stokes Equations in Complicated Geometries
342/02/96 A Thomas Grauschopf, Michael Griebel, Hans Regler: Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic-Multigrid Coarsening for Second Order Elliptic PDEs
342/03/96 A Volker Heun, Ernst W. Mayr: Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes

Reihe A

342/04/96 A Thomas Huckle: Efficient Computation of Sparse Approximate Inverses
342/05/96 A Thomas Ludwig, Roland Wismüller, Vaidy Sunderam, Arndt Bode: OMIS — On-line Monitoring Interface Specification
342/06/96 A Ekkart Kindler: A Compositional Partial Order Semantics for Petri Net Components
342/07/96 A Richard Mayr: Some Results on Basic Parallel Processes
342/08/96 A Ralph Radermacher, Frank Weimer: INSEL Syntax-Bericht
342/09/96 A P.P. Spies, C. Eckert, M. Lange, D. Marek, R. Radermacher, F. Weimer, H.-M. Windisch: Sprachkonzepte zur Konstruktion verteilter Systeme
342/10/96 A Stefan Lamberts, Thomas Ludwig, Christian Röder, Arndt Bode: PFSLib – A File System for Parallel Programming Environments
342/11/96 A Manfred Broy, Gheorghe Ştefănescu: The Algebra of Stream Processing Functions
342/12/96 A Javier Esparza: Reachability in Live and Safe Free-Choice Petri Nets is NP-complete
342/13/96 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Many-to-Many Data-flow Networks
342/14/96 A Giannis Bozas, Michael Jaedicke, Andreas Listl, Bernhard Mitschang, Angelika Reiser, Stephan Zimmermann: On Transforming a Sequential SQL-DBMS into a Parallel One: First Results and Experiences of the MIDAS Project
342/15/96 A Richard Mayr: A Tableau System for Model Checking Petri Nets with a Fragment of the Linear Time μ -Calculus
342/16/96 A Ursula Hinkel, Katharina Spies: Anleitung zur Spezifikation von mobilen, dynamischen Focus-Netzen
342/17/96 A Richard Mayr: Model Checking PA-Processes
342/18/96 A Michaela Huhn, Peter Niebert, Frank Wallner: Put your Model Checker on Diet: Verification on Local States
342/01/97 A Tobias Müller, Stefan Lamberts, Ursula Maier, Georg Stellner: Evaluierung der Leistungsfähigkeit eines ATM-Netzes mit parallelen Programmierbibliotheken
342/02/97 A Hans-Joachim Bungartz and Thomas Dornseifer: Sparse Grids: Recent Developments for Elliptic Partial Differential Equations
342/03/97 A Bernhard Mitschang: Technologie für Parallele Datenbanken - Bericht zum Workshop
342/04/97 A nicht erschienen
342/05/97 A Hans-Joachim Bungartz, Ralf Ebner, Stefan Schulte: Hierarchische Basen zur effizienten Kopplung substrukturierter Probleme der Strukturmechanik
342/06/97 A Hans-Joachim Bungartz, Anton Frank, Florian Meier, Tilman Neunhoeffer, Stefan Schulte: Fluid Structure Interaction: 3D Numerical Simulation and Visualization of a Micropump
342/07/97 A Javier Esparza, Stephan Melzer: Model Checking LTL using Constraint Programming
342/08/97 A Niels Reimer: Untersuchung von Strategien für verteiltes Last- und Ressourcenmanagement
342/09/97 A Markus Pizka: Design and Implementation of the GNU INSEL-Compiler gic
342/10/97 A Manfred Broy, Franz Regensburger, Bernhard Schätz, Katharina Spies: The Steamboiler Specification - A Case Study in Focus
342/11/97 A Christine Röckl: How to Make Substitution Preserve Strong Bisimilarity
342/12/97 A Christian B. Czech: Architektur und Konzept des Dycos-Kerns
342/13/97 A Jan Philipps, Alexander Schmidt: Traffic Flow by Data Flow
342/14/97 A Norbert Fröhlich, Rolf Schlagenhaft, Josef Fleischmann: Partitioning VLSI-Circuits for Parallel Simulation on Transistor Level
342/15/97 A Frank Weimer: DaViT: Ein System zur interaktiven Ausführung und zur Visualisierung von INSEL-Programmen

Reihe A

342/16/97 A Niels Reimer, Jürgen Rudolph, Katharina Spies: Von FOCUS nach INSEL - Eine Aufzugssteuerung

342/17/97 A Radu Grosu, Ketil Stølen, Manfred Broy: A Denotational Model for Mobile Point-to-Point Data-flow Networks with Channel Sharing

342/18/97 A Christian Röder, Georg Stellner: Design of Load Management for Parallel Applications in Networks of Heterogenous Workstations

342/19/97 A Frank Wallner: Model Checking LTL Using Net Unfoldings

342/20/97 A Andreas Wolf, Andreas Kmoch: Einsatz eines automatischen Theorembeweisers in einer taktikgesteuerten Beweisumgebung zur Lösung eines Beispiels aus der Hardware-Verifikation – Fallstudie –

342/21/97 A Andreas Wolf, Marc Fuchs: Cooperative Parallel Automated Theorem Proving

342/22/97 A T. Ludwig, R. Wismüller, V. Sunderam, A. Bode: OMIS - On-line Monitoring Interface Specification (Version 2.0)

342/23/97 A Stephan Merkel: Verification of Fault Tolerant Algorithms Using PEP

342/24/97 A Manfred Broy, Max Breitling, Bernhard Schätz, Katharina Spies: Summary of Case Studies in Focus - Part II

342/25/97 A Michael Jaedicke, Bernhard Mitschang: A Framework for Parallel Processing of Aggregat and Scalar Functions in Object-Relational DBMS

342/26/97 A Marc Fuchs: Similarity-Based Lemma Generation with Lemma-Delaying Tableau Enumeration

342/27/97 A Max Breitling: Formalizing and Verifying TimeWarp with FOCUS

342/28/97 A Peter Jakobi, Andreas Wolf: DBFW: A Simple DataBase FrameWork for the Evaluation and Maintenance of Automated Theorem Prover Data (incl. Documentation)

342/29/97 A Radu Grosu, Ketil Stølen: Compositional Specification of Mobile Systems

342/01/98 A A. Bode, A. Ganz, C. Gold, S. Petri, N. Reimer, B. Schiemann, T. Schnekenburger (Herausgeber): "'Anwendungsbezogene Lastverteilung"', ALV'98

342/02/98 A Ursula Hinkel: Home Shopping - Die Spezifikation einer Kommunikationsanwendung in FOCUS

342/03/98 A Katharina Spies: Eine Methode zur formalen Modellierung von Betriebssystemkonzepten

342/04/98 A Stefan Bischof, Ernst-W. Mayr: On-Line Scheduling of Parallel Jobs with Runtime Restrictions

342/05/98 A St. Bischof, R. Ebner, Th. Erlebach: Load Balancing for Problems with Good Bisectors and Applications in Finite Element Simulations: Worst-case Analysis and Practical Results

342/06/98 A Giannis Bozas, Susanne Kober: Logging and Crash Recovery in Shared-Disk Database Systems

342/07/98 A Markus Pizka: Distributed Virtual Address Space Management in the MoDiS-OS

342/08/98 A Niels Reimer: Strategien für ein verteiltes Last- und Ressourcenmanagement

342/09/98 A Javier Esparza, Editor: Proceedings of INFINITY'98

342/10/98 A Richard Mayr: Lossy Counter Machines

342/11/98 A Thomas Huckle: Matrix Multilevel Methods and Preconditioning

342/12/98 A Thomas Huckle: Approximate Sparsity Patterns for the Inverse of a Matrix and Preconditioning

342/13/98 A Antonin Kucera, Richard Mayr: Weak Bisimilarity with Infinite-State Systems can be Decided in Polynomial Time

342/01/99 A Antonin Kucera, Richard Mayr: Simulation Preorder on Simple Process Algebras

342/02/99 A Johann Schumann, Max Breitling: Formalisierung und Beweis einer Verfeinerung aus FOCUS mit automatischen Theorembeweisern – Fallstudie –

342/03/99 A M. Bader, M. Schimper, Chr. Zenger: Hierarchical Bases for the Indefinite Helmholtz Equation

Reihe A

342/04/99 A Frank Strobl, Alexander Wisspeintner: Specification of an Elevator Control System

342/05/99 A Ralf Ebner, Thomas Erlebach, Andreas Ganz, Claudia Gold, Clemens Harlfinger, Roland Wismüller: A Framework for Recording and Visualizing Event Traces in Parallel Systems with Load Balancing

342/06/99 A Michael Jaedicke, Bernhard Mitschang: The Multi-Operator Method: Integrating Algorithms for the Efficient and Parallel Evaluation of User-Defined Predicates into ORDBMS

342/07/99 A Max Breitling, Jan Philipps: Black Box Views of State Machines

342/08/99 A Clara Nippl, Stephan Zimmermann, Bernhard Mitschang: Design, Implementation and Evaluation of Data Rivers for Efficient Intra-Query Parallelism

342/09/99 A Robert Sandner, Michael Mauderer: Integrierte Beschreibung automatisierter Produktionsanlagen - eine Evaluierung praxisnaher Beschreibungstechniken

342/10/99 A Alexander Sabbah, Robert Sandner: Evaluation of Petri Net and Automata Based Description Techniques: An Industrial Case Study

342/01/00 A Javier Esparza, David Hansel, Peter Rossmanith, Stefan Schwoon: Efficient Algorithm for Model Checking Pushdown Systems

342/02/00 A Barbara König: Hypergraph Construction and Its Application to the Compositional Modelling of Concurrency

SFB 342 : Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

342/1/90 B Wolfgang Reisig: Petri Nets and Algebraic Specifications
342/2/90 B Jörg Desel: On Abstraction of Nets
342/3/90 B Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems
342/4/90 B Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme
342/1/91 B Barbara Paech: Concurrency as a Modality
342/2/91 B Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier- Toolbox - Anwenderbeschreibung
342/3/91 B Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen
342/4/91 B Werner Pohlmann: A Limitation of Distributed Simulation Methods
342/5/91 B Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis
342/6/91 B Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme
342/7/91 B W. Reisig: Concurrent Temporal Logic
342/1/92 B Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support
342/2/92 B Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen
342/1/93 B Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung
342/2/93 B Ekkart Kindler: Sicherheits- und Lebendigkeitseigenschaften: Ein Literaturüberblick
342/1/94 B Andreas Listl; Thomas Schnekenburger; Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS