On Invariants and Substitution Systems (Preliminary Report; Work in Progress)

Johan Lilius
Digital Systems Laboratory
Helsinki University of Technology
SF-02150 ESPOO
Email: Johan.Lilius@hut.fi

8 March 1994

Abstract

Invariants of algebraic nets are structures that describe invariance properties of the net. There are essentially two ways of defining invariants for algebraic nets [11, 18]. In the first method one unfolds the high-level net into its representation as a low-level net and does the calculations there [11]. The problem is thus reduced to finding invariants of this low-level net. The second method views algebraic nets as a formal system for themselves, where the semantics is defined in terms of substitutions [18]. In this approach invariants I are derived as solutions to the matrix equation $M \cdot I = 0$ where M is the incidence matrix of the net. The product operation is a complex operator that is a combination of substitution and the operations of a group. It is the aim of this work to discuss the latter method in the categorical framework of Substitution Systems as proposed by Goguen [4]. The emphasis is on the definition of a suitable substitution system for the terms labeling the net. This substitution system is constructed as a Kleisli category. The incidence matrix defines a set of arrows, one arrow for each transition, in this Kleisli category. An invariant is then a substitution that is the solution to an equation on the arrows of the Kleisli category.

1 Introduction

The following work is part of an attempt to extend the *Petri Nets are Monoids* paradigm of Meseguer and Montanari [15] to high-level nets. One possible way of extending the paradigm has been presented by Hummert et al. [1, 11]. In this approach invariants are not defined in terms of the high-level nets, but are calculated by first compiling the net into a low-level representation and then calculating the invariants. Thus invariants are not treated as high-level objects on par with the high-level net. On the other hand Reisig [18] defines algebraic high-level nets as a formal system for themselves, where the semantics is defined in terms of a suitable notion of substitution. In this approach invariants are directly calculated from the high-level net as solutions to a matrix-equation.

It is the aim of this work to give a definition of invariants as solutions to equations in a category of substitutions. Substitution systems were proposed by Goguen [4] as a category-theoretic framework for the study of a number of different problems in computer science, including: unification, polymorphic type inference and Scott Domain equations. They are also used as one of the basic paradigms in the development of a new object-oriented programming language with logical variables [7]. The motivation for the use of category theory in computer science has been extensively treated by various authors, (see eg. [5]). In the present work the advantage stems from the fact that we are able to give an abstract definition of an invariant that is independent of the specific notion of substitution used. Also, the notion of substitution is automatically obtained through a very general construction.

The present work can be seen as an attempt to apply category theoretic ideas to the study of high-level nets. The paradigm we are proposing is tentatively called *High-level Nets are monoidal substitution systems.*

The paper is structured as follows. We first define what we mean by algebraic high-level nets. This section also includes a category-theoretic formalization of the notion of a multiset of terms. We also show how algebraic high-level nets give rise to a low-level net. Then we discuss invariants, define a substitution system based on an algebraic high-level net and show how invariants arise as solutions to a set of equations on the arrows in this category of invariants. We presuppose that the reader is somewhat familiar with category theory, Net theory and the *Petri Nets are Monoids* approach.

2 Algebraic High-Level nets

So called low-level net-classes are often criticized because of their unsuitability for modelling real applications. The main problem with these net-classes is that the resulting nets are huge. This problem has been aptly named the *football field* problem. One solution is to raise the level of abstraction and model only parts of the system. An other solution is to enrich the labeling of the nets to get a more concise and expressive formalism. Several such expressive formalisms have been proposed: Pr/T-nets [2], Coloured Nets [12] and different flavors of algebraic high-level nets [17, 11, 1, 18]. Algebraic high-level nets are a combination of the algebraic specification method and Net theory. The central idea is that the algebraic specification describes the data in the system, while Net theory is used to specify the distribution and the movement of this data within the system. The aim of this section is to give a new definition of algebraic high-level nets that combines features from the previous approaches within a framework suitable to be extended to a substitution system.

The theory of abstract datatypes is based on universal algebra. A category-theoretic formulation is presented by Goguen and Burstall in [8]. In the following we shall follow the more concise formulation of [9].

The fundamental intuition is, that a datatype consists of a set of elements of different sorts, and a set of operations making the datatype into an algebra. Each algebra has a signature that names the sorts and the operators of the algebra. Given a set of *sorts* S an *equational signature* is a pair $\langle S, \Sigma\rangle$, where Σ is a family of sets (of *operator* names) indexed by $S^* \times S$[1]. It is customary to just write Σ instead of $\langle S, \Sigma\rangle$.

Then, given a signature Σ, a Σ-algebra A is a S-indexed family of sets $|A| = \langle A_s | s \in S\rangle$ called the carries of A together with an $S^* \times S$ indexed family α of maps $\alpha_{u,s} : \Sigma_{u,s} \to [A_u \to A_s]$ for u in S^* and s in S, where $A_{s_1 \ldots s_n} = A_{s_1} \times \ldots \times A_{s_n}$ and $[A \to B]$ denotes the set of functions from A to B.

A Σ-homomorphism is a S-indexed map $h : A \to A'$ that satisfies the equation

$$h(\sigma(x_1, \ldots, x_n) = h(\sigma)(h(x_1), \ldots, h(x_n)) \ .$$

Σ-algebras and Σ-homomorphism form a category $\underline{\Sigma\text{-Alg}}$. There exists a *forgetful* functor $U : \underline{\Sigma\text{-Alg}} \to \underline{\mathrm{Set}}_S$ which sends a Σ-algebra to its S-indexed set of carriers.

Given a S-indexed set of variables X we can form

[1] We denote by S^* the set of strings with alphabet S.

the set of all terms that are constructed with operator symbols from Σ and variables from X. This set forms a Σ-algebra in a natural way. It is called the *free* algebra over the set of variables X and denoted by $T_\Sigma(X)$. We first define a special case, defining $(T_\Sigma)_s$ to be the smallest set of strings of symbols such that:

- $\Sigma_{\lambda,s} \subseteq T_{\Sigma,s}$,
- $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and $t_i \in T_{\Sigma,s_i}$ imply that the string $\sigma(t_1, \ldots, t_n)$ is in $T_{\Sigma,s}$

If we now define α by

- for $\sigma \in \Sigma_{\lambda,s}$ let $\alpha(\sigma)$ be the string σ of length 1 in $T_{\Sigma,s}$,
- for $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and $t_i \in T_{\Sigma,s_i}$ let $\alpha(\sigma(t_1, \ldots, t_n))$ be the string $\sigma(t_1, \ldots, t_n)$ in $T_{\Sigma,s}$.

then α defines the Σ-structure on T_Σ. Let now define $\Sigma(X)$ be the signature with $(\Sigma(X))_{\lambda,s} = (\Sigma)_{\lambda,s} \cup X_s$ and $(\Sigma(X))_{u,s} = (\Sigma)_{u,s}$ if $u \neq \lambda$. Then the free algebra $T_\Sigma(X)$ is just $T_{\Sigma(X)}$ seen as a Σ-algebra.

The free algebra enjoys the following universal property. For any Σ-algebra, every map $f : X \to U(B)$ has a unique extension to a Σ-homomorphism $f^\# : T_\Sigma(X) \to B$. The map $f : X \to U(B)$ is called *variable assignment* and its extension $f^\#$ *assignment.*

A datatype is described by a signature and a set of equations. Formally a Σ-*equation* e is a triple $\langle X, t_1, t_2 \rangle$, where is a S-indexed set of variables, and t_1 and t_2 are terms in $|T_\Sigma(X)_s|$ having the same sort s. It is usually written $(\bigvee X) t_1 = t_2$. A Σ-algebra A *satisfies* a Σ-equation $(\bigvee X) t_1 = t_2$ iff $a^\#(t_1) = a^\#(t_2)$ for every variable assignment $a : X \to |A|$.

A Σ-*presentation* is a pair $\langle \Sigma, E \rangle$, where Σ is a signature and E is a collection of Σ-equations. A Σ-algebra A satisfies a Σ-presentation $\langle \Sigma, E \rangle$ if it satisfies every equation in E. Σ-presentations are abstract datatype specifications. They are a syntactic representation of some Σ-algebra, which is seen as an implementation of the specification.

An Algebraic Net Specification will consist of a net and Σ-presentation. But before we give the precise definition we need to take a look at one more detail. What we want to do is to be able to label the arcs of the net with multisets of terms from $T_\Sigma(X)$. Because we work in the Petri Nets are monoids paradigm it seems natural to try to extend the free construction to work with an algebra where the carriers are monoids. The notion of a signature is the same as previously, but some of the other definitions need to be changed.

Definition 1
Let $\langle S, \Sigma \rangle$ be a signature. Then a Σ-*Mon-algebra* A is a S-indexed family of monoids $|A| = \langle A_s | s \in S \rangle$ called the carries of A together with an $S^* \times S$ indexed family α of maps $\alpha_{u,s} : \Sigma_{u,s} \to [A_u \to A_s]$ for u in S^* and s in S, where $A_{s_1 \ldots s_n} = A_{s_1} \times \ldots \times A_{s_n}$ and $[A \to B]$ denotes the set of monoid homomorphisms from A to B. For $u = s_1 \ldots s_n$, for $\sigma \in \Sigma_{u,s}$ and for $\langle a_1, \ldots, a_n \rangle \in A_u$ we will write $\sigma(a_1, \ldots, a_n)$ for $\alpha_{u,s}(\sigma)(a_1, \ldots, a_n)$. $\square$

Definition 2
A Σ-Mon-algebra-morphism is a Σ-algebra-morphism that is also a monoid-morphism.
Σ-Mon-algebras and their homomorphisms form a category $\underline{\Sigma\text{-MAlg}}$. $\square$

A Σ-algebra morphism satisfies the equation

$$h(\sigma(x_1, \ldots, x_n) = h(\sigma)(h(x_1), \ldots, h(x_n)) .$$

A Σ-Mon-algebra-morphism satisfies the additional requirement that

$$h(t_1 \otimes \ldots \otimes t_n) = h(t_1) \otimes \ldots \otimes h(t_n) .$$

The construction of initial and free algebras is complicated slightly because we have to take into account the monoidal structure.

Definition 3
Given a Mon-Signature we can construct the initial Σ-Mon-algebra $T_\Sigma^\otimes$ as follows:

- $e_s \in T_{\Sigma,s}^\otimes$, where e_s is the unit of sort s.
- $\Sigma_{\lambda,s} \subseteq T_{\Sigma,s}^\otimes$,
- $\sigma \in \Sigma_{s_1 \ldots s_n,s}$ and $t_i \in T_{\Sigma,s_i}^\otimes$ imply that the string $\sigma(t_1,\ldots,t_n)$ is in $T_{\Sigma,s}^\otimes$
- $t_1 \in \Sigma_{s_1 \ldots s_n,s}$ and $t_2 \in \Sigma_{s'_1 \ldots s'_n,s'}$ imply that $t_1 \otimes t_2 \in T_{s_1 \ldots s_n s'_1 \ldots s'_n, ss'}^\otimes$.

The map α is defined as for T_Σ.

Now, given a S-index set of variables $X = \cup\{X_{s\in S}\}$ form the monoid $X^\otimes = \{X_{s\in S}\}^\otimes$. The free Σ-Mon-algebra over the set of variables X, denoted by $T_\Sigma^\otimes(X)$, is defined as $T_{\Sigma\cup\{X^\otimes\}}^\otimes$ □

Again we have a forgetful functor $U : \underline{\Sigma\text{-MAlg}} \to \underline{\text{Mon}}$ and the free property.

Proposition 1
For any Σ-Mon-algebra, every map $f : X \to U(B)$ has a unique extension to a Σ-Mon-homomorphism $f^\# : T_\Sigma^\otimes(X) \to B$. □

An *Algebraic Net Specification* is now defined as a Σ-presentation together with a $T_\Sigma^\otimes(X)$-labelled net.

Definition 4
An *Algebraic Net Specification* is a tuple $NS = \langle \Sigma, E, X, T, P, \text{pre}, \text{post}, sort\rangle$ where

- $\langle \Sigma, E\rangle$ is a Σ-presentation,
- X is a set of S-sorted variables,
- T, P are sets of transitions and places respectively,
- $\text{pre}, \text{post} : T \to [P \to T_\Sigma^\otimes(X)]$, and
- $sort : P \to S$.

$Var(t) \subseteq X$ is the union of the variables in $\text{pre}(t), \text{post}(t)$. □

`sorts`	:	$phil, fork$	
`opns`	:	ph_1, ph_2, ph_3	$:\to phil$
		f_1, f_2, f_3	$:\to fork$
		l	$: phil \to fork$
		r	$: phil \to fork$
`eqns`	:	$l(ph_i) = f_i$	
		$r(ph_i) = f_i$	for $i = 1, 2$
		$r(ph_3) = f_1$	

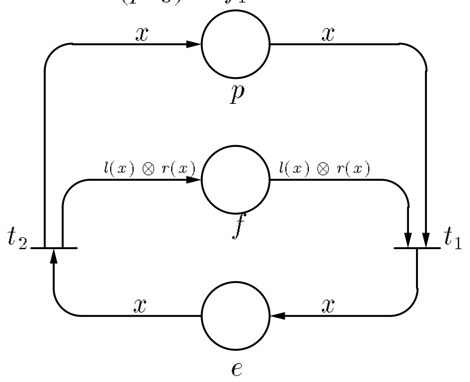

Figure 1: The specification NS of the dining philosophers problem.

An algebraic net specification can be seen as a schema that has several instances depending on the implementation of the equational signature. In figure 1 we have a specification for the dining philosophers problem, as presented for example in [18]. The specification is divided into two parts, the first consisting of the Σ-presentation, or the algebraic specification, of the net-inscriptions and the second consisting of the inscribed net. This particular specification contains 3 philosophers.

If we combine an Algebraic High-level Net with an implementation (ie. an algebra) we have refined the specification to one particular interpretation.

Definition 5
Let $A \in |\underline{\Sigma\text{-Alg}}|$ and NS be an algebraic net specification. Then an *Algebraic High-level Net* is a pair $AN = \langle NS, A\rangle$. □

Because we are now talking of one particular net we can define its unfolding into a low-level net.

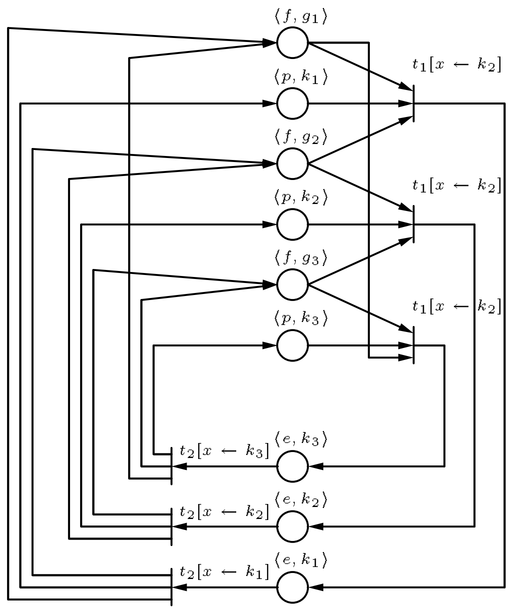

Figure 2: The interpretation of NS in A.

Definition 6
Given an algebraic high-level net $AN = \langle NS, A\rangle$ we can define a Petri net $F_{NS}(A) = \langle T_F, P_F, \iota_F, o_F\rangle$ as follows:

- $P_F = \bigcup_{p \in P} A_{sort(p)} \times p$,
- $T_F = \{\langle t, \mathrm{ass}_A^{\#}\rangle | \, \mathrm{ass}_A \in [Var(t) \rightarrow A], t \in T\}$,
- $\iota_F(\langle t, \mathrm{ass}_A^{\#}\rangle) = \mathrm{ass}_A^{\otimes}(\mathrm{pre}(t))$,
- $o_F(\langle t, \mathrm{ass}_A^{\#}\rangle) = \mathrm{ass}_A^{\otimes}(\mathrm{post}(t))$,

with $\mathrm{ass}_A^{\otimes} : T_\Sigma(X)^{\otimes} \rightarrow (P \times A)^{\otimes}$. □

For a place p in the high-level net we create a number of places $p', p'', p''', \ldots$ in the low-level net such that to every possible token x that may reside in place p there exists a place p' (ie. the pair $\langle p, x\rangle$) in the low-level net that represents this fact. Analogously for a transition t in the high-level net we have to create transitions in the low-level net that record the specific assignment on the variables that the transition fires with.

An example will clarify this. Figure 2 gives an interpretation in the algebra:

$$A = (A, \alpha),$$

with

$$\begin{aligned} A &= \{k_1, k_2, k_3\}_{phil} \uplus \{g_1, g_2, g_3\}_{fork} \text{ , and} \\ \alpha &= \{r \mapsto r, l \mapsto l, \\ & \quad ph_1 \mapsto k_1, ph_2 \mapsto k_2, ph_3 \mapsto k_3, \\ & \quad f_1 \mapsto g_1, f_2 \mapsto g_2, f_3 \mapsto g_3\} \; . \end{aligned}$$

We have omitted the obvious type declarations from the set of functions. The functions r, l have the following definition:

$$r = \{k_1 \mapsto g_1, k_2 \mapsto g_2, k_3 \mapsto g_3\}$$

$$l = \{k_1 \mapsto g_2, k_2 \mapsto g_3, k_3 \mapsto g_1\}$$

while the functions $k_1, k_2, k_3, g_1, g_2, g_3$ are the constant functions. The interpretation of NS in A, the net $F_{NS}(A)$ is given in figure 2. Usually one is interested in the interpretation in the free algebra. This corresponds to the natural intuition about the semantics of the net. In [14] we aim to give a precise definition of this semantics in terms of substitution systems. Let us now turn to the discussion of invariants.

3 Invariants of High-Level Nets

In this section we will show how invariants can be seen as solutions to equations in a category of substitutions. Let us first recall what invariants are. An invariant is a solution to a set of equations that can be derived from the static structure of the net. There are two kinds of invariants:

S-invariants, that describe invariance properties of the distribution of tokens in the net, and

T-invariants, that describe sets of transitions whose firing will keep the marking of the net invariant.

Given a net, we can use its ι and o functions to define a canonical $P \times T$ matrix whose entries are defined by $N[p,t] = -\iota(t,p) + o(t,p)$ with $+$ and $-$ the operations of a group. The entries encode the amount of tokens that are produced or consumed in place p by the firing of transition t. An S-invariant i is now defined as a solution to the matrix equation $N^T \cdot i = 0$ for a suitable notion of product. For low-level P/T-nets the product is simple product of matrices over integers. For high-level nets the definition of a product is more complicated and depends on the formalism used in the labeling of the arcs in the net. The invariance property of an S-invariant can be alternatively stated as follows. Given an initial marking M_0 of the net and any other reachable marking M, then i is an S-invariant iff

$$M_0 \cdot i = M \cdot i \, .$$

Take the net in figure 1. The matrix equation that is generated by it is

$$\begin{bmatrix} -x & x \\ -(r(x)+l(x)) & r(x)+l(x) \\ x & -x \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

which is equivalent to the system of equations

$$\begin{array}{rrrcl} -x \cdot z_1 + & -(r(x)+l(x)) \cdot z_2 + & x \cdot z_3 & = & 0 \\ x \cdot z_1 + & (r(x)+l(x)) \cdot z_2 + & -x \cdot z_3 & = & 0 \end{array} .$$

The question now is, how do we define this product operation? First we need to extend the construction of the monoidal term-algebra in the previous section to take into account the abelian group structure of the invariants. Below we shall demonstrate a general construction that allows us to construct the correct notion of product. The construction is modelled along the lines of the previous section. We extend the construction of the free algebra to account for the abelian group structure on the terms. This free algebra construction has an associated free functor F and a forgetful functor U. By forming the composition of the the functor F and U we obtain an endofunctor T over the category of groups. This endofunctor is then used to construct the Kleisli category $\underline{\mathrm{Grp}_T}$. The Kleisli category is our wanted category of substitutions.

We first define the notion of an algebra over a group.

Definition 7
Let $\langle S, \Sigma \rangle$ be a signature. Then a Σ-*Grp-algebra* A is a S-indexed family of abelian groups $|A| = \langle A_s^{\oplus} | s \in S \rangle$ called the carries of A together with an $S^* \times S$ indexed family α of maps $\alpha_{u,s} : \Sigma_{u,s} \to [A_u \to A_s]$ for u in S^* and s in S, where $A_{s_1 \ldots s_n} = A_{s_1} \times \ldots \times A_{s_n}$ and $[A \to B]$ denotes the set of group homomorphisms from A to B. For $u = s_1 \ldots s_n$, for $\sigma \in \Sigma_{u,s}$ and for $\langle a_1, \ldots, a_n \rangle \in A_u$ we will write $\sigma(a_1, \ldots, a_n)$ for $\alpha_{u,s}(\sigma)(a_1, \ldots, a_n)$. □

Definition 8
A Σ-Grp-algebra-morphism is a Σ-algebra-morphism that is also group-morphism.
Σ-Grp-algebras and their homomorphisms form a category $\underline{\Sigma\text{-GAlg}}$. □

A Σ-algebra morphism satisfies the equation

$$h(\sigma(x_1, \ldots, x_n) = h(\sigma)(h(x_1), \ldots, h(x_n)) \, .$$

A Σ-Grp-algebra-morphism satisfies the additional requirements that

$$h(t_1 \oplus \ldots \oplus t_n) = h(t_1) \oplus \ldots \oplus h(t_n),$$

and

$$h(-t_1) = -h(t_1),$$

that is they preserve the group structure.

Now the construction of initial and free algebras is again complicated slightly because we have to take into account the group structure.

Definition 9
Given a Grp-Signature we can construct the initial Σ-Grp-algebra $T_{\Sigma}^{\oplus}$ as follows:

- $\Sigma_{\lambda,s} \subseteq T^{\oplus}_{\Sigma,s}$,
- $\sigma \in \Sigma_{s_1 \oplus \ldots \oplus s_n, s}$ and $t_i \in T^{\oplus}_{\Sigma,s_i}$ imply that the string $\sigma(t_1, \ldots, t_n)$ is in $T^{\oplus}_{\Sigma,s}$
- $t_1 \in \Sigma_{s_1 \ldots s_n, s}$ and $t_2 \in \Sigma_{s'_1 \ldots s'_n, s'}$ imply that $t_1 \oplus t_2 \in T^{\oplus}_{s_1 \ldots s_n s'_1 \ldots s'_n, ss'}$.
- $t \in T^{\oplus}_{\Sigma,s}$ then $-t \in T^{\oplus}_{\Sigma,s}$

The map α is defined as for T_Σ.

Given a S-indexed set of variables $X = \cup\{X_{s \in S}\}$ form the free abelian group $X^{\oplus} = \{X_{s \in S_b}\}^{\oplus}$. The free Σ-Grp-algebra over the set of variables X, denoted by $T^{\oplus}_{\Sigma}(X)$, is defined as $T^{\oplus}_{\Sigma \cup \{X^{\oplus}\}}$ □

Again we have a forgetful functor.

Proposition 2
There exists a forgetful functor $U : \underline{\Sigma\text{-GAlg}} \rightarrow \underline{\text{Grp}}$ that maps a Σ-Grp-algebra to its group of elements. □

Proposition 3
For any Σ-Grp-algebra, every map $f : X \rightarrow U(B)$ has a unique extension to a Σ-Grp-homomorphism $f^{\#} : T_\Sigma(X) \rightarrow B$. □

The forgetful functor has a left adjoint.

Proposition 4
There exist a free functor $F : \underline{\text{Grp}} \rightarrow \underline{\Sigma\text{-GAlg}}$ that maps a group X to $T^{\oplus}_{\Sigma}(X)$. □

Composing F with U we obtain a functor $T : \underline{\text{Grp}} \rightarrow \underline{\text{Grp}}$. Because T is constructed as the composition of two adjoints it forms a *monad* over $\underline{\text{Grp}}$. A pair of adjoint functor also gives rise to a Kleisli category as follows.

Definition 10
The *Kleisli* category of the adjoint pair T is denoted by $\underline{\text{Grp}_T}$ has as elements the elements of $\underline{\text{Grp}}$. Its arrows are formed as follows. Let $f : a \rightarrow b$ be an arrow in $\underline{\text{Grp}}$, then the corresponding arrow in $\underline{\text{Grp}_T}$ is $f' : a \rightarrow U(F(b))$. □

The arrows in the category $\underline{\Sigma\text{-GAlg}_T}$ are substitutions. So a substitution $y \mapsto f(x) \oplus g(x) - k(z)$ is an arrow $\{y\}^{\oplus} \rightarrow \{x, z\}^{\oplus}$ from the free group over the set $\{y\}$ to the free group on the set $\{x, z\}$.

The crucial idea is now that each entry $N[p, t]$ in the incidence matrix M can be seen as a substitution $\sigma_{p,t} : p \mapsto N[p, t]$. Thus each matrix is a t-indexed set of arrows $\{p\}^{\oplus} \rightarrow X^{\oplus}$. The product operation is now reduced to composition in $\underline{\Sigma\text{-GAlg}_T}$. If we take co-products as representation for vectors the matrix definition of invariants can be cast into the following form.

Definition 11
Given an algebraic high-level net N, a set of P-indexed substitutions $i_p : \{X\}^{\oplus} \rightarrow T^{\oplus}_{\Sigma}(X)$ is an invariant iff the following equation holds:

$$\coprod_{t \in T} (\coprod_{p \in P} \sigma_{p,t} \doteqdot i_p) = 0 \; .$$

□

The inner co-product gives us for each t a set of P-indexed substitutions. The outer co-product the forces these sets to be equal.

Example 1 ([18])
The following set of substitutions is an invariant for the net in figure 1:

$$\begin{array}{cccc} i_p : & x & \mapsto & l(x) \oplus r(x) \\ i_f : & x & \mapsto & -x \\ i_e : & x & \mapsto & 0 \end{array}$$

Above we have only shown the generators of the maps. Thus the result of applying the substitution i_f to the substitution σ_{f,t_2} is $\sigma_{f,t_2} \doteqdot i_f : f \mapsto r(x) \oplus l(x)$. Intuitively this invariant says that *if a fork is available, then both of its potential users are thinking.* If we, given an initial marking, substitute into 3, and then solve for $M(f)$ we get that $M(f) = l(M(p)) + r(M(p)) - M_0(f)$, which implies that if $f_i \in M(f)$ then both $f_i \in l(M(p))$ and $f_i \in r(M(p))$ which implies that $p_i, p_{i+1} \in M(p)$. □

4 Conclusions

In this work we have shown how invariants can be defined as an equation on arrows in a suitable category of substitutions. The advantage of this definition is that it separates the definition of the substitution system from the definition of the invariants. Thus the same constructions and definitions will work for other kinds of formal systems like order sorted algebra [10], hidden sorted algebra [6] or unified algebra [16].

This work is ongoing research and, apart from the obvious lose ends and the rather terse exposition of the present paper, many questions still remain open. One area for future research is the development of an algorithm for calculating invariants based on the algorithm for calculation unifiers proposed by Rydehard and Burstall [19]. Also as can be seen from the example the resulting invariants are sometimes not so obvious to interpret. One possibility is to interpret invariants as formula of linear logic [3] along the lines of [13].

Acknowledgements I would like to thank Prof. Joseph Goguen for suggestions and encouragement.

References

[1] Dimitrovici, C., Hummert, U., and Pétrucci, L. *The properties of algebraic net schemes in some semantics.* L.R.I Rap. de Recherce 539. Univeriste de Paris-Sud, 1990.

[2] Genrich, H. *Predicate/transition nets.* Petri Nets: Central Models and Their Properties. Lecture Notes in Computer Science 254. Springer Verlag, 1986, pp. 207–247.

[3] Girard, J.-Y. *Linear logic.* Theoretical Computer Science, 50(1987), pp. 1–102.

[4] Goguen, J. *What is unification? A categorial view of substitution, equation and solution.* Research Report SRI-CSL-88-2R. Computer Science Laboratory, SRI International, April 1988.

[5] Goguen, J. A. *A categorical manifesto.* Research Report PRG-72. Programming Research Group, Oxford University Computing Laboratory, England, 1989.

[6] Goguen, J. A. *Types as theories.* Category Theory in Computer Science. Oxford University Press, Oxford, England, 1991. pp. 357–390.

[7] Goguen, J. A. *Algebraic semantics for the object paradigm.* 9th WADT - 4th Compass Workshop. Caldes de Malavella, October 1992.

[8] Goguen, J. A. and Burstall, R. *Some fundamental algebraic tools for the semantics of computation. Part1: Comma categories, colimits, signatures and theories.* Theoretical Computer Science, 31(1984), pp. 175–209.

[9] Goguen, J. A. and Burstall, R. *Institutions: Abstract model theory for specification and programming.* Research Report ECS-LFCS-90-106. Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland, 1990.

[10] Goguen, J. A. and Meseguer, J. *Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations.* Research Report PRG-80. Programming Research Group, Oxford University Computing Laboratory, England, 1989.

[11] Hummert, U. *Algebraische theorie von high-level netzen.* PhD thesis, Technische Universität Berlin, 1989.

[12] Jensen, K. *Coloured Petri nets.* Petri Nets: Central Models and Their Properties. Lecture Notes in Computer Science 254. Springer Verlag, 1986, pp. 247–299.

[13] Lilius, J. *High-level nets and linear logic.* Application and Theory of Petri Nets. Lecture Notes in Computer Science 616. Springer Verlag, Berlin, 1992, pp. 310–319.

[14] Lilius, J. *On high-level nets and substitution systems.* Manuscript, October 1992.

[15] Meseguer, J. and Montanari, U. *Petri nets are monoids.* Information and Computation, 88(1990), pp. 105–155.

[16] Mosses, P. *Unified algebras and institutions.* Technical Report DAIMI PB-274. Computer Science Department, Aarhus University, 1989.

[17] Reisig, W. and Vautherin, J. *An algebraic approach to high level Petri nets.* 8th Workshop on Applications and Theory of Petri Nets. Zaragoza, Spain, 1987, pp. 51–72.

[18] Reisig, W. *Petri nets and algebraic specifications.* Theoretical Computer Science, 80(1991), pp. 1–34.

[19] Rydehard, D. and Burstall, R. *A categorical unification algorithm.* Category Theory and Computer Programming. Lecture Notes in Computer Science 240. Springer Verlag, Berlin, 1985, pp. 493–505.