Unification and Interaction

Interaction nets and Stellar Resolution

Adrien Ragot

Abstract

We study how the Stellar Resolution due to Seiller and Eng [ES21], a model of computation based on the principle of Resolution introduced by Robinson [Rob65] can simulate in linear time the model of Interaction nets due to Lafont [Laf97]. The aim of this result is to provide an explicit and definite syntax in which can be written interaction nets, and so one can imagine this approach is also suited for the proof nets of Linear Logic.

Contents

Introduction

About a hundred years ago Jacques Herbrand discovered an important theorem concerning first order logic, that is today known as the Herbrand's theorem [Her30]. The result, much like the theorem of Rice–Shapiro, relates an object with an infinite quantity of image to a finite object, by relying on a notion of *witness*. Namely the theorem states that a formula of the form $\exists y_1, \dots, y_n F(y_1, \dots, y_n)$ is valid if and only if there exists a *finite* family of terms $(t_{i,j})_{1 \leq i \leq k}^{1 \leq j \leq n}$ called witnesses such that the disjunction $F(t_{1,1}, \dots, t_{1,n}) \vee \cdots \vee F(t_{k,1}, \dots, t_{k,n})$ is valid. This result had important implications, one of which was its impact on automated theorem–proving, which could now rely on the method of Herbrand to prove or disprove the validity of quantified formula. An automated prover cannot efficiently rely on the definitions provided by model theory: recall that $\mathcal{M} \vDash \exists x.A[x]$ holds whenever there exists an $a \in |\mathcal{M}|$ such that $A[a]$ is true. But the classical sequent calculus does not give a better way to operate since the

inference rule for the existential quantifier states that the sequent $\Gamma \vdash \exists x.A[x]$ can be obtained from a sequent $\Gamma \vdash A[t/x]$ where t is some first order term. Both the definition from model theory and the inference rule are not suited for automated theorem proving, since they do not provide any strategy for finding the element a. Furthermore a brute–force program consisting in testing each elements (or terms) cannot be bounded in reasonable time, since eventually the program would have to explore the whole space $|\mathcal{M}|$ which eventually could be infinite. Hence, if the formula $\exists x.A[x]$ was to be false, and the model was containing a infinite number of elements (say for example $\mathbb{N}$) such a brute–force program would not terminate, and so the validity (in $\mathcal{M}$) of a formula of the form $\exists x.A[x]$ would only be semi–decidable for such a program.

Thirty years later, Motivated by the contribution of Herbrand theorem to theorem–proving, J.A. Robinson introduced the concept of unification and of resolution [Rob65]. By analyzing the process of substitution and the process of truth functional analysis involved in Herbrand's method J.A. Robinson reveals that these two process could be combined in one, the *resolution* a method containing only one inference rule called the *unification* able to simulate all the axioms and inference rules from first–order predicate calculus.

Although, it was pointed out immediately – and by J.A. Robinson himself – that the procedure he presented was inefficient in time, and Carl Hewitt also showed that unification was prone to combinatorial explosion during proof search [Hew72][MM82]. From these observations, there has been several investigations and efforts towards finding an efficient algorithm for unification. The most convincing one at the time was proposed by Martelli and Montanari [MM82] in 1982 and terminates in a linear time when no critical case is encountered (To be precise, its time complexity is $O(n \times G(n))$ where G is the Ackermann function).

More recently Unification has been involved in a new set of works by Jean Yves Girard called the transcendental syntax and in the Stellar Resolution of Boris Eng and Thomas Seiller [ES21] which is a technical implementation of the transcendental syntax of Jean Yves Girard. The model of stellar resolution is a natural generalization of Robinson's resolution where the inference rule involves again the unification process. It was shown to be suited for simulating logic programming, Wang Tiles and Turing Machines (the model was shown to be Turing complete), but also to be suited for expressing the language of proof nets due to J.Y. Girard, which are known to be difficult to formalize using merely graph theory.

The goal of this note is to show show how stellar resolution can encode the model of computation of Interaction Nets due to Yves Lafont [Laf97]. In the present note we do the following:

1. First we recall the notion of unification and basic theorems regarding substitutions and renaming and most general unifiers.
2. Secondly we exhibit the notion of unification algorithm which are designed to find the most general unifier of a given unification problem. We then define the well–known Martelli–Montanari unification algorithm and give a self contained proof that it solves the unification problem.
3. The third section is a concise exposition of the paradigm of Interaction nets, the notions of cell, wire, net, redex, reduction rules and isomorphism are defined.
4. In the fourth part we introduce notions coming from Stellar Resolution. For simplicity we choose to call them unification net and exhibit three class of nets the localized, standard and elementary net. We then show that the elementary net are an accurate translation of the Interaction net of Yves Lafont [Laf97]. More precisely we exhibit two functors – which are our translations – from the category of elementary unification net to the category of interaction nets and back.
5. In the last part we introduce notions of computation on unification nets, our rewriting step corresponds to translation of the interaction rules used for interaction nets. Note that our reduction rule is not the one that occur in the stellar resolution of Boris Eng and Thomas Seiller [ES21], but since the stellar resolution is Turing complete it is safe to assume that it could simulate our reduction. We choose to not use the same rule as we don't want to study the encoding of the

interaction rules into the stellar resolution, but merely want to look at how well the Interaction nets can be captured by the language provided by unification nets. We then revisit the classical results on Interaction nets from [Laf97] into the syntax of elementary unification nets, observing how well–suited it can be to study Interaction net.

Acknowledgement. We wish to acknowledge the discussions, encounters and technical presentations with members of the "transcendalist circle" initiated by Boris Eng, this has influenced our understanding of the Stellar Resolution.

1 Unification theory

Definition 1.1 (Signature)**.** A *signature* Σ is a triple $(\mathcal{V}, \mathcal{F}, \mathrm{ar})$ where:
- $\mathcal{V}$ is a set of elements called *variables*.
- $\mathcal{F}$ is set of elements called *function symbols*.
- $\mathrm{ar} : \mathcal{F} \to \mathbb{N}$ is a total map associating to each function symbol f an integer $\mathrm{ar}(f)$ called its *arity*.

A function symbol of arity 0 is called a *constant*. The set of function of arity n is denoted $\mathcal{F}_n$.

Definition 1.2 (First order term)**.** The set of *first order terms* or simply *terms* over a signature $\Sigma = (\mathcal{V}, \mathcal{F}, \mathrm{ar})$ is denoted T^Σ, and is the language over the alphabet $\mathcal{V} \cup \mathcal{F} \cup \{(,)\}$ defined inductively as follow;
- Any variable of $\mathcal{V}$ is a term.
- Any constant is a term.
- Given a function symbol f of arity $n > 0$ and any sequence $(t_1, ..., t_n)$ of n terms, the word $f(t_1, ..., t_n)$ is a term.

If there is no ambiguity we might relax the notation T^Σ to T.

Definition 1.3 (Substitution)**.** A map $\sigma : \mathcal{V} \to T$ is said to be *cyclic* if it maps a variable x to a term $t \neq x$ containing occurrences of x. A *substitution* is an acyclic map $\sigma : \mathcal{V} \to T$.
- A variable is an *invariant* of σ if $\sigma(x) = x$.
- The *support* of a substitution σ is the set of variables that are not invariant by σ, it is denoted $supp(\sigma)$.
- A substitution is *finite* if its support is finite.
- A finite substitution σ of support $\{x_1, ..., x_n\}$ can be denoted $\{x_1 \mapsto t_1, ..., x_n \mapsto t_n\}$ where for each $1 \leq i \leq n$ we have $\sigma(x_i) = t_i$.
- A *renaming* is a substitution such that its codomain is the set of variables $\mathcal{V}$ and that it is a bijective map.
- Two substitution θ, ϕ are *equal up to a renaming* α if $\theta = \alpha\psi$.
- A substitution θ is an *instance* of a substitution ψ if there exists a substitution σ such that $\theta = \sigma\psi$

Substitution can be lifted so that they can be applied to terms and not only variables this is done by induction and correspond to simultaneously replace each variable x occuring in the term by its image $\sigma(x)$. Given a substitution σ and a term t:
- If t is a variable x then $\sigma(t) = \sigma(x)$.
- If $t = f(t_1, \ldots, t_n)$ we define $\sigma(t)$ as $f(\sigma(t_1), \ldots, \sigma(t_n))$.

To illustrate how substitution apply to terms consider the following example. Take the term $t = f(x, g(x, y))$ and the substitution $\theta = \{x \mapsto y, y \mapsto g(x)\}$ then by definition:

$$\theta t = f(\theta x, \theta g(x, y)) = f(y, g(\theta x, \theta y)) = f(y, g(y, g(x))).$$

Now note that this corresponds to the simultaneous substitution of the variable in t. indeed if we where

to consider $\theta_1 = \{x \mapsto y\}$ and $\theta_2 = \{y \mapsto g(x)\}$ note that:

$$\theta_2\theta_1 t = \theta 2 f(y, g(y, y)) = f(g(x), g(g(x), g(x)))$$

This illustrates that the application of θ_1 followed by the application of θ_2 yields out a term totally different than the one obtained by the simultaneous application of θ_1 and θ_2 (i.e.) by applying θ. Namely this is due to the fact that the image of θ_1 is contained in the domain of θ_2.

Lemma 1.1 (Eder [Ede85])**.** *Any substitution that maps variables to variables and is injective is bijective, and hence is a renaming.*

Proof. Consider θ a substitution mapping variables to variables and being injective. Given any variable x in the support of θ it must be by definition that $\theta x \neq x$. Thus, since θ is injective this inequality must be preserved by θ meaning that $\theta\theta x \neq \theta x$. Hence θx is not an invariant of θ i.e. it is an element of its support. This illustrates that θ maps its support to itself injectively: that is to say $\theta : supp(\theta) \to supp(\theta)$ is an injection. Hence since the support of θ is finite, by the pigeonhole–principle we conclude that θ is bijective on its support. Now to conclude notice that θ behaves like the identity outside of its support and therefore it is also bijective on the complementary of its support. This allows us to conclude that θ is bijective on $supp(\theta) \cup \mathcal{V} \setminus supp(\theta)$ meaning on $\mathcal{V}$.

□

Definition 1.4 (Unification)**.** Two terms t, u are *unifiable* if there exists a substitution θ such that $\theta t = \theta u$. In that case θ is also called a *unifier* of t, u. A *(most) general unifier* of two terms t and u is a unifier of t and u such that any unifier of t and u is an instance of θ.

Unification is widely used in logic programming, for instance Prolog uses pattern matching (i.e. unification) to deal with *queries*. We may declare the relation parent(Alice, Bob) to mean that Alice is a parent of Bob and then input the query ?parent(Alice, x), where x is a variable, asking to whom Alice is a parent.

```
parent(alice,bob);
?-parent(alice,X);
```

To generate the output and answer the query, Prolog tries to unify the term parent(Alice, x) with the previously declared terms. For instance in that scenario, the only previously declared term is parent(Alice, Bob), Therefore Prolog returns the answer x = Bob. It is clear that, the size of the database directly influences the query–answering time of this method.

Type inference is also involving the use of unification. Type inference consists of automated (i.e. not declared) attribution of types to terms based on their use. For instance if we encounter an expression f(f(0)) we may infer that the type of f is **int** $\to$ **int**. First f takes 0 as argument so it must take integer as input and hence be of type **int** $\to$ **A**. Since f can then be applied to f(0) this means that the output f(0) must also be an integer, therefore we can infer that **int** $\to$ **int** corresponds to the type of f.

In the context of stellar resolution, an interpretation of Girard transcendental syntax due to Boris Eng and Thomas Seiller [ES21], unification can be used to perform computation. In fact it turns out that stellar resolution –which involves unification– is a Turing complete model of computation [ES21]. We can perform unification–based computation, for instance the computation of the addition can be designed. Consider that we are given a function symbol add of arity 2 and a function symbol of arity 1 s and constant 0. Now to define the addition we pose the two equalities $\mathrm{add}(0, y) = y$ and $\mathrm{add}(\mathrm{s}(x), y) = \mathrm{add}(x, \mathrm{s}(y))$. To compute an expression of the form $\mathrm{add}(t, t')$ where t and t' are both closed term over $\{0, \mathrm{s}\}$ we unify the expression with the left hand side of either the first or the second equation, then obtaining the right hand side of the equation to which the unifier θ has been applied. Namely we use two rewriting rules, $\mathrm{add}(0, t') \to_1 \theta y$ where θ is the unifier of $\mathrm{add}(0, y)$ and $\mathrm{add}(0, t')$ and $\mathrm{add}(\mathrm{s}(t), t') \to_2$

Algorithm 1: A recursive program based on unification computing the sum of two terms.

Data: two first order terms u and v.
Result: the sum of u and v.

```
1  sum(u, v) :
2  θ ← unify(add(u, v), add(s(x), y)) ;
3  if not(θ == ⊥) then
4  |  sum(θx, s(θy));
5  else
6  |  θ ← unify(add(u, v), add(0, y));
7  |  if not(θ == ⊥) then
8  |  |  return θy ;
9  |  else
10 |  |  printf(syntax error) ;
```

$\theta\mathrm{add}(x, \mathrm{s}(y))$ where θ unifies $\mathrm{add}(\mathrm{s}(x), y)$ and $\mathrm{add}(\mathrm{s}(t), t')$. Indeed then we can yield the computation of the addition take for example the computation of the addition of 2 and 1, i.e. $\mathrm{add}(\mathrm{ss0}, \mathrm{s0})$.

$$\mathrm{add}(\mathrm{ss0}, \mathrm{s0}) \to_1 \mathrm{add}(\mathrm{s0}, \mathrm{ss0}) \to_1 \mathrm{add}(0, \mathrm{sss0}) \to_2 \mathrm{sss0}$$

The interest of involving unification is that we do not need to define the reductions $\to_1$ and $\to_2$ for each pair of terms (which indeed would be computationally impossible). A way to proceed is to exploit a unification algorithm to perform computation, for instance, the addition can be computed with a simple recursive program.

Proposition 1.1 (uniqueness of the most general unifier up to renaming)**.** *Given two most general unifier θ, ψ of some term t and u, θ and ψ are equal up to renaming.*

Proof. Since θ is a general unifier of t, u and ψ is a unifier of t, u we conclude that ψ is an instance of θ. In the same way we can deduce that θ is an instance of ψ. This means there exists two substitutions σ_1 and σ_2 such that $\theta = \sigma_1\psi$ while $\psi = \sigma_2\theta$. This implies in particular that $\theta = \sigma_1\sigma_2\theta$. Now we want to show that σ_1 is a renaming in order to conclude. Assume σ_1 is not a renaming this can be for two cases:

- It can be because σ_1 does not map variables to variables. Consider y a variable and assume $\sigma_1 y$ is not a variable, meaning $\sigma_1 y = f(t_1, \ldots, t_n)$. Consider then x a variable, such that y occurs in $t = \theta x$. In that case $\sigma_1\theta x$ is longer that θx and therefore $\sigma_2\sigma_1\theta x$ is longer that θx this is indeed a contradiction since these two terms are the same.
- It could be because σ_1 is not one–to–one but maps variables to variables. First it can be because we do not have the property of injectivity. Consider two variables x and y such that $w = \sigma_1 x = \sigma_1 y$ Consider z a variable such that x and y occurs in θz then in $\sigma_1\theta z$ each occurrence of x and y are replaced by occurrence of w. σ_2 maps w to some variable v and since $\sigma_2\sigma_1\theta z = \theta z$ noting that $x \neq y$ it cannot be that $\sigma_2\sigma_1\theta z = \theta z$ (even if $v = y$ or $v = x$).
  The second possibility is that the substitution σ_1 is not surjective. In that case it means in particular that σ_1 is not a bijection. Thanks to Eder's lemma this implies that σ_1 is not injective, and so we fall back in the previous case.

□

Definition 1.5 (Potential equation, Unification problem)**.** A *potential equation* is a pair of term (t, u) denoted $t \doteq u$. A *solution* of a potential equation $t \doteq u$ is a unifier of t and u. A *unification problem* is a set of potential equation $\{t_1 \doteq u_1, \ldots, t_n \doteq u_n\}$ or the special element $\bot$, whereas a the solution of unification problem is a substitution that is the solution of each of its potential equations. A unification

DELETE	$P \cup \{t \doteq t\} \to P$
DECOMPOSITION	$P \cup \{f(t_1, ..., t_n) \doteq f(t'_1, ..., t'_n)\} \to P \cup \{t_1 \doteq t'_1, \ldots, t_n \doteq t'_n\}$
CONFLICT	$P \cup \{f(t_1, ..., t_n) \doteq g(t'_1, ..., t'_k)\} \to \bot$ if $g \neq f$ or $k \neq n$.
SWAP	$P \cup \{f(t_1, ..., t_n) \doteq x\} \to P \cup \{x \doteq f(t_1, ..., t_n)\}$
ELIMINATION	$P \cup \{x \doteq t\} \to P\{x \mapsto t\} \cup \{x \doteq t\}$ if $x \notin var(t)$ and $x \in var(P)$.
CYCLE	$P \cup \{x \doteq f(t_1, \ldots, t_n)\} \to \bot$ if $x \in var(f(t_1, \ldots, t_n))$.

Figure 1: Martelli–Montanari rewriting system computing the unification problem [MM82]. In the elimination step x must occur in the set of equations P in order to ensure termination of the rewriting system.

problem is *unsolvable* if it has no unifier or is $\bot$. A unification problem $\{t_1 \doteq u_1, \ldots, t_n \doteq u_n\}$ is *solved* or in *in solved* form if each $t_i = x_i$ is a variable. And for any index $1 \leq i, j \leq n$ $t_i = x_i$ does not occur in any u_j. A solved unification problem $P = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ natural defines a substitution $\theta_P = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ The set of unification problems is denoted $\mathcal{U}$ and corresponds to $\bigcup_{n\in\mathbb{N}}(T \times T)^n$. Given a unification problem P we denote the set of its unifiers as $unifier(P)$

Definition 1.6 (occurring variables, rank, degree, redundancy)**.** The set of variables occurring in a term t, denoted $var(t)$ is defined inductively by $var(x) = \{x\}$ and $var(f(t_1, \ldots, t_n)) = \bigcup_{1\leq i\leq n} var(t_i)$. The *rank* of a term is the number of the function symbol occurring in it. It is defined by induction:

$$r(x) = 0 \qquad r(f(t_1, \ldots, t_n)) = 1 + r(t_1) + \ldots r(t_n).$$

The set of variables occurring in a unification problem is the union of the set of variables of its terms. The *left–hand* terms of a unification problem $\{u_1 \doteq t_1, \ldots u_n \doteq t_n\}$ is the set of terms $\{u_1, \ldots, u_n\}$ The *rank* of a unification problem, is the sum of the rank of its *left–hand* terms. The *degree* of unification problem is its number of equations of the form $x \doteq x$ or $t \doteq x$ where t is not a variable. A variable x is *fixed* by a unification problem P if P contains exactly one occurence of the form x. The *redundancy* of a unification problem P is the number of variable occurring more than once in P i.e. that are not fixed. We define the degree, rank and redundancy of $\bot$ to have the value 0.

2 Martelli-Montari unification Algorithm

Definition 2.1 (Rewriting system)**.** A rewriting system is pair $(A, \to)$ made of a set A of elements called *terms* and a *binary relation* over A. A term x *(directly) reduces to* a term y if $x \to y$. A term x *reduces to* a term y if $x \to^* y$. A term is *normal* or *in normal form* if it does not reduce to any other term. A term is *normalizable* if it reduces to a normal form. A *computation* is a potentially infinite sequence of terms $(t_i)_{i\in I}$ such that for each i there is a direct reduction $t_i \to t_j$. The *current value* of a finite computation $(t_1, \ldots, t_n)$ the ultimate term of the sequence i.e. t_n. If the current value is a normal form we might call it the *output*. A computation is *terminated* if its current value is a normal form. The computation of a term t is a computation with t as first element. A term is *strongly normalizable* if any computation of t is finite. A rewriting system is *terminating* if any of its term is strongly normalizing.

Definition 2.2 (Unification Algorithm)**.** A *Unification rewriting system* or *unification algorithm* is a rewriting system on the set $\mathcal{U}$ of unification problem. A unification algorithm is *sound* if its normal forms are either $\bot$ or a solved unification problem. A unification is *complete* if any solved unification problem and $\bot$ is a normal form. We say a unification algorithm is *correct* if for any unification problems P and P' such that $P \to P'$ we have $unifier(P) = unifier(P')$.

Definition 2.3 (Martelli–Montanari unification algorithm)**.** The *Martelli Montanari unification algorithm* is the rewriting system on the set $\mathcal{U}$ depicted in figure 1.

Proposition 2.1 (Martelli–Montanari completeness and soundness)**.** *The Martelli Montanari unification algorithm enjoys soundness and completeness.*

Proof. Let us show that the algorithm enjoys completeness. It is clear that $\bot$ is a normal form since no rule can be applied. In the case of a solved unification problem P by definition it must be of the form $P = \{x_1 \doteq t_1, \dots, x_n \doteq t_n\}$ then only two rule may be potentially applied the CYCLE rule or the ELIMINATION rule, but none of these rule may be applied since no x_i occur in any of the term t_j by definition of a solved form.

Now let us show the algorithm enjoys soundness, meaning we must show any normal form of the rewriting system is $\bot$ or a solved form. Consider a normal form P, in the case where $P = \bot$ the property is immediately true. Now assume $P \neq \bot$, and fix $P = \{u_1 \doteq t_1, \dots, u_n \doteq t_n\}$. Assume that one of the u_i is of the form $f(v_1, \dots, v_k)$ then P is of the form $P' \cup \{f(v_1, \dots, v_k) \doteq t_i\}$. If t_i is a variable then we can apply a SWAP. If $t_i = g(w_1, \dots, w_l)$ we can either apply a CONFLICT or a DECOMPOSITION rule. In any of these cases a rule can be applied. Hence, all u_i need to be variables, and we assume so from now on.

Now assume the variable u_i occurs in some term t_j. If $t_j = t_i$ then two scenario can occur: either t_i is a variable in which case we can apply the DELETE rule, or t_i is a proper term and so we can apply the CYCLE rule. On the other hand if $t_j \neq t_i$, this means that P can be written as $P' \cup \{u_i \doteq t_i\}$ where t_j occurs in P', therefore, since by assumption u_i occurs in t_j then u_i occurs in P' and we can further assume u_i does not occur in t_i (otherwise we fall back in the previous case). In this situation we can then apply an ELIMINATION rule. We saw that none of the variables u_i can occur in terms t_j, this conclude our proof to show that P is in solved form.

□

Lemma 2.1 (Invariance lemma)**.** *Given x a variable and t a term. For any unifier θ of x and t, and any term u, θ unifies u and $u\{x \mapsto t\}$.*

Proof. We proceed by induction on u. If u is a variable and $u = x$ then $\theta u = \theta x$ and $\theta u\{x \mapsto t\} = \theta t$ since θ unifies x and t we can deduce $\theta u = \theta u\{x \mapsto t\}$. Otherwise if $u = y$ where $y \neq x$, we have $u\{x \mapsto t\} = u$ and so $\theta u\{x \mapsto t\} = \theta u$.
For the induction step, consider some term $t = f(t_1, ..., t_n)$, and note that we have the following.

$$\theta t = \theta f(t_1, ..., t_n) \overset{\text{def}}{=} f(\theta t_1, ..., \theta t_n) \overset{\text{ind}}{=} f(\theta t_1\{x \mapsto t\}, ..., \theta t_n\{x \mapsto t\}) \overset{\text{def}}{=} \theta f(t_1, ..., t_n)\{x \mapsto t\}$$

□

Proposition 2.2 (Correctness of the Martelli–Montanari algorithm)**.** *The Martelli Montanari unification algorithm is correct.*

Proof. Given two unification problems P, Q such that $P \to Q$ let us show that P and Q have the same unifier. To do so we analyze each possible reduction.

- If the reduction is the application of a DELETE rule, i.e. $P \cup \{t \doteq t\} \to P$. Clearly if θ is a unifier for $P \cup \{t \doteq t\}$, then it is a unifier for P. On the other hand if θ is a unifier for P it is also a unifier for $\{t \doteq t\}$ since obviously $\theta t = \theta t$. Hence, θ is a unifier for $P \cup \{t \doteq t\}$.
- If the reduction is the application of a DECOMPOSITION rule, i.e. $P \cup \{f(t_1, \dots, t_n) \doteq f(t'_1, \dots, t'_n)\} \to P \cup \{t_1 \doteq t'_1, \dots, t_n \doteq t'_n\}$. If θ unifies $P \cup \{f(t_1, \dots, t_n) \doteq f(t'_1, \dots, t'_n)\}$ then $\theta f(t_1, \dots, t_n) = \theta f(t'_1, \dots, t'_n)$ and so by definition $f(\theta t_1, \dots, \theta t_n) = f(\theta t'_1, \dots, \theta t'_n)$ therefore necessarily, for each $1 \leq i \leq n$ θ unifies t_i and t'_i $\theta t_i = \theta t'_i$. Hence, θ unifies $P \cup \{t_1 \doteq t'_1, \dots, t_n \doteq t'_n\}$. The other direction is based on the same argument.
- If the reduction is the application of a CONFLICT rule, i.e. $P \cup \{f(t_1, \dots, t_n) \doteq g(t'_1, \dots, t'_k)\} \to \bot$ where either $f \neq g$ or $k \neq n$. Since $\bot$ has no unifier, it suffice to show that $P \cup \{f(t_1, \dots, t_n) \doteq g(t'_1, \dots, t'_k)\}$ has no unifier to conclude. Assume otherwise that there exists such a unifier and

denote it θ, then in particular θ unifies $f(t_1, \dots, t_n) \doteq g(t'_1, \dots, t'_k)$ meaning $\theta f(t_1, \dots, t_n) = \theta g(t'_1, \dots, t'_k)$ and so $f(\theta t_1, \dots, \theta t_n) = g(\theta t'_1, \dots, \theta t'_k)$. If $f \neq g$ it is clear that these two terms cannot be equal. If otherwise $n \neq k$ the number of virgula's in both terms cannot be the same hence the term cannot be equal. This shows that the unification has no solution, just as $\bot$.

- If the reduction is the application of a SWAP rule, i.e. $P \cup \{f(t_1, \dots, t_n) \doteq x\} \to P \cup \{x \doteq f(t_1, \dots, t_n)\}$. It is straightforward to see that a unifier of one unification problem is a unifier of the other and vice versa.
- If the reduction is the application of a ELIMINATION rule, i.e. $P \cup \{x \doteq t\} \to P\{x \mapsto t\} \cup \{x \doteq t\}$ where x does not occur in t but occur in P. Consider θ a unifier of $P \cup \{x \doteq t\}$ obviously it unifies in particular $\{x \doteq t\}$, let us show it unifies $P\{x \mapsto t\}$. Consider $u \doteq v$ a potential equation of P, since θ unifies x and t by the invariance lemma we know that θ unifies u and $u\{x \mapsto t\}$, as well as v and $v\{x \mapsto t\}$. Therefore since θ unifies u and v we can conclude it unifies $u\{x \mapsto t\}$ and $v\{x \mapsto t\}$.

  $$\theta u\{x \mapsto t\} = \theta u = \theta v = \theta v\{x \mapsto t\}$$

  Now consider θ a unifier of $P\{x \mapsto t\} \cup \{x \doteq t\}$. Let $u \doteq v$ be a potential equation of P in that case we apply the same argument using the invariance lemma, and using the fact that θ unifies $u\{x \mapsto t\}$ and $v\{x \mapsto t\}$ to conclude.

  $$\theta u = \theta u\{x \mapsto t\} = \theta v\{x \mapsto t\} = \theta v$$

- If the reduction is the application of a CYCLE rule, i.e. $P \cup \{x \doteq f(t_1, \dots, t_n)\} \to \bot$, where x occurs $f(t_1, \dots, t_n)$. Let us show $P \cup \{x \doteq f(t_1, \dots, t_n)\}$ has no unifier. Assume that θ is a unifier of $P \cup \{x \doteq f(t_1, \dots, t_n)\}$ then in particular it unifies x and $f(t_1, \dots, t_n)$, meaning $\theta x = \theta f(t_1, \dots, t_n)$ but since x occurs in $f(t_1, \dots, t_n) \neq x$ it means θ is cyclic and therefor it is not a substitution.

□

Proposition 2.3 (Termination of the Martelli–Montanari algorithm)**.** *The Martelli Montanari unification algorithm is terminating.*

Proof. Consider P some unification problem and consider the triplet $\mu(P) = (c(P), r(P), d(P))$ which respectively correspond to the redundancy the rank and the degree of the unification problem. Now to conclude we will show that each step of the rewriting system decreases the measure of P in the lexicographic order for $\mathbb{N}^3$. Since no infinite decreasing chain exists in $(\mathbb{N}^3, <)$ this will allow us to conclude termination. Consider two unification problem such that $P_1 \to P_2$ and show $\mu(P_1) > \mu(P_2)$ we do so by treating all the possible case of reductions.

- If the reduction is the application of a DELETE rule, i.e. $P \cup \{t \doteq t\} \to P$. Then indeed the number of equation has decreased meaning, $\mu(P_2) = (c(P_2), r(P_2), d(P_1) - 1) < \mu(P_1)$. Also if a variable occurs in t it was necessarily occurring more than once in P then if this variable occurs only once in P the redundancy of the unification problem could decrease. Also if t was not a variable the rank of the system has decreased.
- If the reduction is the application of a DECOMPOSITION rule, i.e. $P \cup \{f(t_1, \dots, t_n) \doteq f(t'_1, \dots, t'_n)\} \to P \cup \{t_1 \doteq t'_1, \dots, t_n \doteq t'_n\}$. In that case the redundancy of the unification problem is unchanged. But P_2 contains one less function symbol than P_1 on the left hand side, therefore the measure has decreased.
- If the reduction is the application of a SWAP rule, i.e. $P \cup \{f(t_1, \dots, t_n) \doteq x\} \to P \cup \{x \doteq f(t_1, \dots, t_n)\}$. Indeed here the redundancy of the unification problem, i.e. the number of variable occurring more than once in the unification system is unchanged. But note that the number of function symbols on the left hand side is decreased by one. And so the measure of the problem has decreased.

- If the reduction is the application of a elimination rule, i.e. $P \cup \{x \doteq t\} \to P\{x \mapsto t\} \cup \{x \doteq t\}$ where x does not occur in t but occur in P. Indeed x occurs only once in P_2 while x occurs at least two times in P_1, since by hypothesis it occurs in P but also in $x \doteq t$. Hence the redundancy of the unification problem has decreased and so –with respect to the lexicographic order– the measure has decreased.
- If the reduction is the application of a conflict rule, i.e. $P \cup \{f(t_1, \dots, t_n) \doteq g(t'_1, \dots, t'_k)\} \to \bot$ where either $f \neq g$ or $k \neq n$. In that case the measure has necessarily decreased to $(0, 0, 0)$ and the decreasing was strict since the rank and degree of the unification problem were at least of 1.
- If the reduction is the application of a cycle rule, i.e. $P \cup \{x \doteq f(t_1, \dots, t_n)\} \to \bot$, where x occurs $f(t_1, \dots, t_n)$. In that case the measure has necessarily decreased to $(0, 0, 0)$ and the decreasing was strict since the degree of the unification problem was at least of 1.

□

3 Interaction nets

In 1989 Yves Lafont introduced the language of interaction nets [Laf89], a model of computation based on cells interacting through their principal ports, inspired by the proof nets proposed by Jean Yves Girard for Linear Logic. The interaction nets were shown to be a valid model of computation: it was shown to be able to simulate Turing machines and so to be Turing–complete, but also, to be able to simulate cellular automata aswell as combinators which are closely related to functional programming first in an asymmetric way proposed by Lafont [Laf97], and later in a symmetric way proposed by Mazza [Maz07].

As of now , we assume that we are given a countable set $\mathcal{P}$ of elements called *ports*. This section is entirely dedicated to the definition of the paradigm of Interaction nets.

Definition 3.1 (Interaction net , [Laf89])**.** An interaction alphabet is a set of symbols Σ such that each symbol has a given arity $n_\alpha \in \mathbb{N}$. An *interaction cell* of symbol α is a tuple (X_c, δ_c, α) where X_c is a finite non–empty set of elements called *ports*, and $\delta_c : \{0, \dots, n_\alpha\} \to X_c$ is a bijection. The natural number $\delta_c^{-1}(p)$ is the *relative address* of the port p. The element $\delta_c(0)$ is the *principal port* of the cell c, whereas any other port is called an *auxiliary port.* An *unwired interaction net* is a family of cells $(c_i)_{i \in I}$ where I is countable. Equivalently it is a tuple (X, C, δ, l) where X is a set of ports, C is a set of non–empty and finite sets of ports, $\delta : C \times \mathbb{N} \to X$ such that for any cell $c \in C$ the map $\delta(c, \cdot)$ is a bijection from $\{1, \dots, card(c)\} \to c$. A labelling function $f : C \to S$ associating to each cell an interaction symbol $l(c)$.

A *wire* is a pair of ports. A wired interaction net or simply *interaction net* is a tuple (X, C, δ, l, W) where W is set of pairwise disjoint wires. An *axiom* is a wire between two auxiliary doors, a *cut* is a wire between two principal doors.

An $\alpha\beta$*–redex* is a cut wire connecting the two principal ports of two cells of symbols α and β, such a configuration is denoted $\alpha \bowtie \beta$. An *interaction rule* is a rewriting rule r, that rewrites an $\alpha\beta$–redex $\alpha \bowtie \beta$ into an interaction net $r[\alpha, \beta]$ such that the free ports $\{p_1, \dots, p_n\}$ of the redex are exactly the free ports of the net $r[\alpha, \beta]$, i.e. the rule conserve free ports. An *interaction system* is a set of interaction rules $r_1, \dots, r_n$ such that if a rule r_i is given for the $\alpha\beta$–redexes then no other rule is given for $\alpha\beta$–redexes or $\beta\alpha$–redexes (i.e. no ambiguity). Given an interaction system $\mathcal{S}$ its *computation* or *reduction* is denoted $\to_{\mathcal{S}}$ is the relation containing the interaction rules and stable by sum. Equivalently we can present the definition using inference rules:

$$\frac{}{\alpha \bowtie \beta \to_{\mathcal{S}} r[\alpha, \beta]}\ r \in \mathcal{S} \qquad \frac{\mathcal{N}_1 \to \mathcal{N}_2}{\mathcal{N} + \mathcal{N}_1 \to \mathcal{N} + \mathcal{N}_2}$$

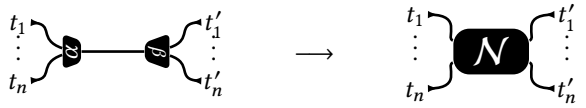

Figure 2: An interaction rule rewriting an $\alpha\beta$–redex, also called an *active pair*, into a net $\mathcal{N}$. Note that the free ports are conserved.

Definition 3.2 (Wired, tied and bounded port)**.** A port p is *wired* if there exists a wire w containing the port p. A port p is *tied* if there exists a cell c containing the port p. A port is *bounded* if it is tied and wired, otherwise we say it is a *free port*.

Definition 3.3 (Morphism, Isomorphism and Equivalence of interaction nets)**.** Two interaction cells are *equivalent* if their have the same symbol α. In an interaction nets for any symbol α, the *α–proximity* is the binary relation on ports, denoted $\frown^\alpha$, defined such that $p \frown^\alpha q$ holds whenever p and q belong to the same cell c and c is a cell of symbol α. The *proximity* relation is the binary relation denoted $\frown$ where two ports are such that $p \frown q$ if they belong to the same cell. The *role* of a port is word on $\Sigma \times \mathbb{N}$ such that if p belong to a cell of symbol α and $\delta_c(p) = i$ then the role of p denoted $\overline{p}$ is αi. If p does not belong to a cell then $\overline{p} = \epsilon$. Two ports are *equivalent* if they have the same role.

A *morphism* between two unwired interaction nets $\mathcal{N}_1$ and $\mathcal{N}_2$ is a function $f : P_1 \to P_2$ preserving the relation of proximity and of equivalence. An *isomorphism* f between two unwired interaction $\mathcal{N}_1$ and $\mathcal{N}_2$ nets is a morphism between $\mathcal{N}_1$ and $\mathcal{N}_2$ that is also a bijection.

Given a function $f : P_1 \to P_2$ its *lift* which we denote $\widehat{f}$, is the function mapping W_1 to W_2 associating to a wire $\{p, q\}$ the image $\{f(p), f(q)\}$. A *morphism* f between two interaction nets is a morphism between the two underlying unwired interaction nets such that $\widehat{f}$ is defined for all wires in W_1. An *isomorphism* f between two interaction nets is an isomorphism for the two underlying unwired interaction nets such that $\widehat{f} : W_1 \to W_2$ is a bijection.

Two interaction nets are *equivalent* or *isomorphic* if there exists an isomorphism between them.

4 Unification Nets

4.1 Locative signature

Definition 4.1 (Resolution signature)**.** A *resolution signature* is a signature $\Sigma = (\mathcal{V}, \mathcal{F}, \mathrm{ar})$ containing a symbol of function "$\cdot$" $\in \mathcal{F}$ of arity 2 called the *concatenation*, that is associative. Meaning that for any terms t, u, v the equality $t \cdot (u \cdot v) = (t \cdot u) \cdot v$ holds, hence these term can be written $t \cdot u \cdot v$. Furthermore the notation of the concatenation can be made implicit if there are no ambiguities meaning tuv denotes the term $t \cdot u \cdot v$.

Definition 4.2 (polarized term, opposites)**.** A *polarized term* is an expression αt where t is first order term, and $\alpha \in \{+, -\}$. The polarities + and − are *opposites* we denote $\overline{+} = -$ and $\overline{-} = +$. Two polarized term $+t_1$ and $-t_2$ are of *opposite polarities*. A substitution θ *unifies two polarized terms* αt_1 and βt_2 if θ unifies t_1 and t_2 and α and β are opposite polarities. The opposite of a polarized term αt is $\overline{\alpha} t$, this term is also denoted $\overline{\alpha t}$. A term s is a *prefix* (resp. *suffix*) of some first order term t if $t = s \cdot t_0$ (resp. $t = t_0 \cdot s$).

Definition 4.3 (Locative signature)**.** A *locative signature* is a resolution signature $\Sigma = (\mathcal{V}_L \uplus \mathcal{V}_C \uplus \mathcal{V}_A, \mathcal{F}, \mathrm{ar})$ containing two types of variables:

- The set $\mathcal{V}_C \uplus \mathcal{V}_A$ of *address variables* which are divided in two disjoint set: $\mathcal{V}_C$ the set of cell–location variables and $\mathcal{V}_A$ the set of absolute location variables.

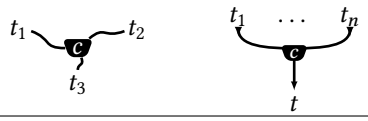

Figure 3: The representation of a unification cell with unpolarized term $[t_1, t_2, t_3]$, and of an oriented cell $[-t_1, \ldots, -t_n, +t]$. If the source of the arrow is the term this illustrates a negative polarity, and it represents a positive polarity otherwise.

- The set $\mathcal{V}_L$ of *logical variables*, denoted as usual as $x, y, z, \ldots$.

And also containing two special types of constant symbols *types*, and *biases* [1] denoted as integers $0, 1, 2 \ldots$.

A *relative address* is a term $si_1 \cdots i_n$ where s is a cell–location variable and $i_1, \ldots, i_n$ are biases. An *address* is either an absolute position variable or a relative address. A term t is in *standard form* if it is of the form $U\tau x$ where U is an address, τ is a type constant and x is a logical variable.

Definition 4.4 (Substitution types in a locative Signature)**.** A *logical substitution* is a substitution which supports is contained on logical variables. A *location substitution* is a substitution of which the support is contained in address variables. A *relocation* is a renaming involving only address variables.

4.2 Nets

As of now we assume that we are given a locative signature $\Sigma = (\mathcal{V}_L \uplus \mathcal{V}_C \uplus \mathcal{V}_A, \mathcal{F}, \mathrm{ar})$ and that terms may be polarized. Also we assume that there exists a bijection $\lambda : \mathcal{P} \mapsto \mathcal{V}_A$ from the set of ports – given to define interaction nets – to the set $\mathcal{V}_A$ of absolute addresses i.e. absolute position variables. We also assume that this bijection extends to $\lambda : \mathcal{C} \mapsto \mathcal{V}_C$ from the set of interaction cells to the set of cell location variables. The elements $\lambda(p)$ will be denoted $\check{p}$ similarly an element $\lambda(c)$ will be denoted $\check{c}$. Hence we make the absolute position variables range over $\check{p}, \check{q}, \ldots$ and the cell location variables range over $\check{c}, \check{d}, \ldots$. Furthermore we assume that the set of logical variables is made of the interaction symbols plus one extra symbol called the wire symbol and denoted w.

Definition 4.5 (Unification Cell)**.** A *unification cell* is a finite multiset of polarized first order terms. A unification cell is represented by a sequence that is denoted $[t_1, \ldots, t_n]$. is an indexed family of first order term, i.e. a pair $c = (I_c, \pi_c)$ where I_c is a set of elements called *positions* and $\pi_c : I_c \to T$ is a total map which associates to each position a polarized term. Two unification cells are *superposed* if they share a common position, otherwise they are *disjoint.* Given a unification cell $c = [t_1, \ldots, t_n]$ the *content* of c is the set $\{t_1, \ldots, t_n\}$. An *input (resp. output)* of a cell c is a position $p \in P_c$ such that $\pi(p)$ is a term of negative (resp. positive) polarity. A cell is *oriented* if all its positions are either inputs or outputs.

Definition 4.6 (Unification Net)**.** A *unification net* or *unification system* is a countable set of unification cells that are pairwise disjoint. A unification net is denoted as a sum i.e. the unification net composed of the cells $(c_j)_{j\in J}$ where j is numerable is denoted $\sum_{j\in J} c_j$. The *positions* of unification net $\sum_{j\in J} c_j$ is the set $\bigcup_{j\in J} I_{cj}$. Equivalently a unification net is tuple (P, C, π) where P is a set of positions C is a set of sets of positions, and $\pi : P \to T$ associates to each position a first order term.

Two position are *ajdacent* if they belong to the same cell c i.e. $p, q \in c$. Given a position p in a disjoint unification net the *support* of p is the only cell c containing p we denote it C_p. Similarly given a set of positions P_0 its *support* is the set of cells $\bigcup_{p\in P_0} C_p$. A unification net is *oriented* if all its cells are oriented.

[1] This terminology comes from Ludics [Gir01].

Definition 4.7 (Bind)**.** Two distinct positions p, q of a unification net are *bindable* if their associated first order term $\pi(p), \pi(q)$ are unifiable. A position p is *bindable* if there exists a position q such that the pair of positions p, q is bindable. A position is said to be *free* if it is not bindable. A *bind* is a pair of bindable positions. A *binding* is a set of disjoint binds. A wiring is *unifiable* if its induced unification problem is unifiable. A unification net is *deterministic* if all its position are bindable with *at-most* one other position (potentially none).

Definition 4.8 (Prefix action and extensions)**.** Given a cell $c = [t_1, \dots, t_n]$ and a first order term t we define the *prefix action* (or merely action if there is no confusion) of t upon c that we denote $t \cdot c$ as the cell $[t \cdot t_1, \dots, t \cdot t_n]$. A cell d is an *extension* of a cell c, denoted $c \prec d$ if it results from a prefix action on c i.e. there exists a term t such that $d = t \cdot c$.

Definition 4.9 (Hypergraph, multigraph)**.** An *hypergraph* is a tuple (V, E) where V is a set of vertices and E is a finite family of non empty subsets of V. A *multigraph* is a couple (V, E) where V is a set of vertices and E is a multiset of doubleton set of vertices. A *multidigraph* or *quiver* is a couple (V, E) where V is a set of vertices and E is a multiset of pairs of vertices.

From unification nets to hypergraphs and multigraphs. To each unification net (P, C, π) we can canonically associate an hypergraph merely taking the couple (P, C) of positions and cells whose definition exactly corresponds to an hypergraph. On the other hand to each hypergraph (V, E) we can associate canonical unification net (V, E, π) where the assignment associates to each vertex v a term made of only one symbol s_v, given that the set of constants contains a symbol s_v for each vertex v.

Similarly from a unification net (P, C, π) we can obtain a multigraph (C, E) which vertices are the cells and the multiset of edges E is constructed such that the multiplicity of the doubleton set $\{c, d\}$ is defined to be $card\{\{p, q\} \mid p \text{ and } q \text{ are unifiable} \wedge p \in c \wedge q \in d\}$. Eventually we can also define $\mu(\{p, q\})$ to be $card\{\{p, q\} \mid \pi(p) = \pi(q) \wedge p \in c \wedge q \in d\}$. Involving the polarities we can also obtain a canonical multidigraph, defining it to be (C, E) where $E(p, q) = card\{\{p, q\} \mid p \text{ and } q \text{ are unifiable} \wedge p \in c^- \wedge q \in d^+\}$.

This point is important since it means that the language of unification nets is suited to study graph rewriting. Graph rewriting methods can be hard to implement, yet unification nets are suited to simulate logic programming, Turing machines or lambda–calculus, hence unification nets could be used as a tool to implement algorithm coming from graph rewriting methods.

Definition 4.10 (Propagation of substitutions)**.** A substitution θ can be applied to a unification cell $[t_1, \dots, t_n]$ such that $\theta[t_1, \dots, t_n] = [\theta t_1, \dots, \theta t_n]$. Equivalently, given a cell $c = (P_c, \pi_c)$ then its image θc is $(P_c, \theta \circ \pi_c)$.

Given a unification net $\mathcal{N} = \sum_{c \in C} c$, a substitution θ, the image of $\mathcal{N}$ under θ is denoted $\theta\mathcal{N}$ and corresponds to $\sum_{c \in C} \theta c$

Proposition 4.1 (Linearity of substitutions)**.** *Given two unification nets $\mathcal{N}_1$ and $\mathcal{N}_2$ and θ some substitution, the net $\theta(\mathcal{N}_1 + \mathcal{N}_2)$ corresponds to $\theta\mathcal{N}_1 + \theta\mathcal{N}_2$.*

Proof. consider two unification nets $\mathcal{N}_1 = (P, C, \pi)$ and $\mathcal{N}_2 = (P', C', \pi')$ without loss of generality assume P and P' to be disjoint. The sum of the nets corresponds to $(P \uplus P', C \uplus C', \pi \uplus \pi')$ and applying the substitution θ we obtain the net $(P \uplus P', C \uplus C', \theta \circ (\pi \uplus \pi'))$.

On the other hand $\theta\mathcal{N}_1 = (P, C, \theta\pi)$ and $\theta\mathcal{N}_2 = (P', C', \theta\pi')$ and their sum correspond to $(P \uplus P', C \uplus C', (\theta \circ \pi) \uplus (\theta \circ \pi'))$.

To conclude note that $\theta \circ (\pi \uplus \pi')$ and $(\theta \circ \pi) \uplus (\theta \circ \pi')$ are equal. First, this is because the two substitutions have the same domain being $P \uplus P'$. And secondly because for any position p in P both map associates $\theta\pi(p)$ while for any position p' of P' both maps associate the term $\theta\pi'(p')$.

□

Definition 4.11 (Net Equivalence)**.** Two nets $\mathcal{N}_1$ and $\mathcal{N}_2$ are equivalent if there exists a relocation θ such that $\theta\mathcal{N}_1 = \mathcal{N}_2$.

Proposition 4.2 (Linearity of net–equivalence)**.** *Given two nets $\mathcal{N}_1$ and $\mathcal{N}_2$ if $\mathcal{N}_1$ is equivalent to $\mathcal{N}_1'$ and $\mathcal{N}_2$ is equivalent to $\mathcal{N}_2'$ then $\mathcal{N}_1 + \mathcal{N}_2$ is equivalent to $\mathcal{N}_1' + \mathcal{N}_2'$.*

Proof. By assumption $\mathcal{N}_1' = \theta\mathcal{N}_1$ while $\mathcal{N}_2' = \rho\mathcal{N}_2$ where θ and ρ are location substitution with a support made of cell locations. Since $\mathcal{N}_1$ and $\mathcal{N}_2$ have no cell location in common (they are summable) θ and ρ don't have overlapping codomains hence we can construct the substitution $\theta \uplus \rho$. Indeed $\theta \uplus \rho\mathcal{N}_1 + \theta \uplus \rho\mathcal{N}_2$ will correspond to $\theta\mathcal{N}_1 + \rho\mathcal{N}_2$ i.e. $\mathcal{N}_1' + \mathcal{N}_2'$. Note that since $\mathcal{N}_1'$ and $\mathcal{N}_2'$ are summable they must have no location in common, hence the image of the support of θ and ρ are disjoint which ensure we can sum $\theta\mathcal{N}_1$ and $\rho\mathcal{N}_2$.

□

Definition 4.12 (Wires)**.** A cell is *local* if its terms do not contain any logical variable that is not the wire symbol w. A *wire* is an oriented local cell containing exactly two positions. A *cut* is a wire with two negative positions. An *axiom* is a wire with two positive positions. A *composition* is a wire with one negative and one positive position. A *detour* is a pair of wires that can be unified An *interaction cell* is an oriented cell with exactly one output.

Definition 4.13 (Unification and computation of unification nets)**.** A unification net $\mathcal{N} = (P, C, \pi)$ *unifies* a unifiable wiring $B = \{\{p_1, q_1\} \dots \{p_n, q_n\}\}$ into a unification system $\mathsf{U}_B(\mathcal{N}, \theta)$ where θ is the unifier of B. We denote this relation $\mathcal{N} \xrightarrow{B,\theta} \mathcal{N}'$ we denote by $\mathcal{N} \to \mathcal{N}'$ the transitive closure of its existential quantification.

A unification net $\mathcal{N}$ *computes to* or *reduces in* a sum of nets $\mathcal{N}_1 + \mathcal{N}_2$ if $\mathcal{N}$ unifies in $\mathcal{N}_1$. Then the net $\mathcal{N}_1$ is called the *unification* of the computation, and $\mathcal{N}_2$ is called the *product* of the computation. A relation on nets R is a *computation* if for any two net $\mathcal{N}_1 R \mathcal{N}_2$ implies that $\mathcal{N}_1$ reduces in $\mathcal{N}_2$ The *type* of computation is the function symbols introduced by the unifier θ involved in the reduction. An *identification* is a unification involving the identity as most general unifier. A rewriting rule $\mathcal{N} \to \mathcal{N}'$ is an *interaction rule* w.r.t. a substitution θ if the there exists a bijection $\Phi : free(\mathcal{N}) \to free(\mathcal{N}')$ such that for any free port p the image $\theta\pi(p)$ corresponds to $\pi'(\Phi(p))$.

Definition 4.14 (correct assignment)**.** Given a unification net (P, C, π) the assignment π is *correct* if any two positions p and q that do not belong to the same cell don't share any variable i.e. $FV(p) \cap FV(q) = \varnothing$. This condition is necessary for unification to occur without problems i.e. to avoid *cycles* in the Montelli–Montanari algorithm.

4.3 Classes of unification nets

In this subsection we define three classes of unification nets that will be our tool for simulating interaction nets.

Definition 4.15 (Localized and standard unification net)**.** A *localized term* is a term of the form $\check{u}t$ where $\check{u}$ is either a cell variable or and absolute position variable and such that t does not contain any cell variable or absolute position variable. A *localized unification net* is a unification net such that each of its position is labelled by a localized term. The class of localized unification net will be denoted **LUN**. A *standard unification net* is a unification net defined by the following induction:

- A cell $\check{c}[1\alpha x, \dots n\alpha x, 0\alpha x]$ is a standard unification net.
- A wire $[Uwx_1, Vwx_2]$ is a standard unification net. Where U and V are absolute position variables or relative adresses.
- The sum of standard unification net is a standard unification net.

The class of standard unification net will be denoted **SUN**.

Definition 4.16 (Matching substitution)**.** An *atomic match* is a substitution of the form $\{\check{p} \mapsto \check{c}i\}$ where $\check{p}$ is a position variable, $\check{c}$ is a cell variable and i is a relative address. Two atomic match are *summable* if they have distinct support and codomain. A *matching subsitution* is a the union of summable atomic match. Given $f : \mathcal{N}_1 \to \mathcal{N}_2$ a relocation and a match $\theta = \{\check{p} \mapsto \check{c}i\}$ the analog of θ under f is denoted θ^f and corresponds to $\{f\check{p} \mapsto f\check{c}i\}$. We lift this definition to matching substitution.

A matching is *coherent* with a standard unification net $\mathcal{N}$ if it maps to relative position $\check{c}i$ such that $\check{c}$ in a cell of $\mathcal{N}$ and $\check{c}i$ does not occur in any wire of $\mathcal{N}$.

Proposition 4.3 (Commutation of Analogues)**.** *Given $f : \mathcal{N}_1 \to \mathcal{N}_2$ a relocation and θ a matching of $\mathcal{N}_1$ then $f(\theta(\mathcal{N}_1)) = \theta^f f(\mathcal{N}_1)$.*

Proof. We assume $\theta = \{\check{p} \mapsto \check{c}i\}$ without loss of generality since induction follows immediately. Consider a term of the form $s = \check{u}t$ then if $\check{u}$ is an absolute location variable we consider two cases: if $\check{u} = \check{p}$ then $\theta s = \check{c}it$. Then applying f results in $f(\check{c})it$. In the other first applying f results in $f(s) = f(\check{p})t$ then since is by definition $\theta^f = \{f\check{p} \mapsto f\check{c}i\}$, applying θ^f to $f(s)$ results in $f(\check{c})it$ so the two images coincide.

If on the other hand $\check{u} \neq \check{p}$ then $f(\check{u}) \neq f(\check{p})$ since f is injective. Then $\theta s = s$ and $f(s) = f(\check{p})t$ and so $f(s)$ is invariant under θ^f. Hence again the two images coincide.

Then we can conclude for any unification net made of localized term. Since both functions are linear for the sum of nets and also with respect to the cell–structure. □

Definition 4.17 (Elementary Net)**.** An *elementary net* is a unification net $\mathcal{N}$ defined by the following induction:

- A wire $[\check{p}wx_1, \check{q}wx_2]$ with $\check{p} \neq \check{q}$, and a cell $[\check{c}1\alpha x, \ldots, \check{c}n\alpha x, \check{c}0\alpha x]$ are elementary net.
- The sum of elementary net is elementary.
- Given $\mathcal{N}$ an elementary net and M a coherent matching of $\mathcal{N}$, then $M(\mathcal{N})$ is an elementary net.

The class of elementary unification net will be denoted **EUN**.

Inclusion of classes. We can immediatly note that the classes of unification net we introduced are included in one another namely **EUN** ⊂ **SUN** ⊂ **LUN**. Depending on the situation, it might be easier to prove that a property holds in a class than in another, or worse, it could also be that a property holds in a class and not its upper class. For these reason we must be careful in which class we state our propositions to hold.

Proposition 4.4 (Characterization)**.** *Given a standard unification net $\mathcal{N}$, the two statement are equivalent:*

- *$\mathcal{N}$ is an elementary net.*
- *Each absolute position variable occurs at most once in $\mathcal{N}$. And each cellule variable $\check{c}$ occurs at most once adjacently. Each relative position occurs at most once in a wire. Each relative position that occurs in a wire occurs in a cell.*

Proof. ($\Rightarrow$). We proceed by induction. Consider that the net is a wire $[\check{p}wx_1, \check{q}wx_2]$ such that $\check{p} \neq \check{q}$, then indeed each absolute position variable occurs at most once. If the net is a cell indeed each cellule variable occurs adjacently.

For the induction consider first the sum of disjoint elementary net $\mathcal{N}_1 + \mathcal{N}_2$. Since by induction absolute position symbols occur at most once in $\mathcal{N}_1$ and $\mathcal{N}_2$ and that the two nets share no symbols necessarily it remains true for their sum. The property of adjacent cell symbol is also preserved as location variables are not shared.

The second case to consider for the induction is a net of the form $\theta\mathcal{N}$ where $\mathcal{N}$ an elementary net and θ is a matching substitution over $\mathcal{N}$. θ substitutes absolute position variables of the form $\check{p}$

by relative positions $\check{c}i$ such that θ is injective and cannot map to wired relative positions. Since $\theta\mathcal{N}$ contains less absolute position variables indeed the property of uniqueness is preserved. On the other hand, since $\mathcal{N}$ is elementary $\check{c}$ occurs adjacently at most once in $\mathcal{N}$. This property is invariant under θ since θ is invariant on each cell. Then θ may map position variables to a relative position $\check{c}i$ only if this relative position is not in use, hence in $\theta\mathcal{N}$ each relative position occurs at most once in a wire. Finally if in $\theta\mathcal{N}$ a relative position $\check{c}i$ occurs in a wire it means by definition that $\check{c}$ occurs adjacently in a cell, which is the property we seek.

($\Leftarrow$). Let us treat the other direction of the implication. Consider a standard net $\mathcal{N}$. If it is a wire or a cell then indeed it is an elementary net. Consider a net W that is only made of wires and is standard. This net is a sum of wires and hence a sum of elementary nets therefore it is a elementary net.
To conclude we will proceed by induction on the number of cells in the net. Meaning we consider a net of the form $W + C$ where W is a set of wires and C is a set of cells. In the case where the net contain only one cell and so is of the form $W + c$ the wires in W either contain absolute position variables or relative positions of the form $\check{c}i$ that may occur at most once in W. Hence W is of the form $\theta(W_0)$ where θ is a matching substitution of W_0 on c. Indeed then since W_0 and c are elementary nets, by definition their sum $M(W_0) + c$ is a elementary net.
Now let us proceed with the induction assuming the assumption to be true for a when the set of cells is of size n and let us proceed with a set of size $n + 1$ that we will denote $C + c$, hence the net is of the form $W + C + c$. By induction $W + C$ is an elementary net, the set of wires will be of the form $\theta(W_0)$ where θ is a matching substitution of W_0 on $C + c$. which can be decomposed in a matching θ_1 of W_0 onto C and another θ_2 of W_0 onto c. Hence $\theta_1(W_0) + C$ is an elementary net and therefor $\theta_0(\theta_1(W_0) + C) + c$ is a elementary net by inductive definition.

□

4.4 From Interaction nets to elementary unification nets

Definition 4.18 (Implicit translation)**.** The implicit translation of an interaction net $\mathcal{N}$ into a unification net $\Phi(\mathcal{N})$ is given by the following induction:

- If the net is made of on cell c of arity n and of symbol α given that the set of ports is $\{p_0, \dots, p_n\}$ then $\Phi(c) = [-\check{p_1}\alpha x, \dots, -\check{p_n}\alpha x, +\check{p_0}\alpha x]$.
- If the net is made of one wire w $\{p, q\}$ then $\Phi(w) = [\check{p}wx_1, \check{q}wx_2]$.
- If the net if the sum of two subnets $\mathcal{N}_1 + \mathcal{N}_2$ then its image is defined as $\Phi(\mathcal{N}_1) + \Phi(\mathcal{N}_2)$.

Given $f : \mathcal{N} \to \mathcal{N}'$ a morphism on interaction nets we define its translation Φ_f as the location–substitution $\{\check{p} \mapsto f(\check{p}), p \in P\}$ where P is the set of ports of $\mathcal{N}$.

Proposition 4.5 (Implicit Translation as functor)**.** *Given $f : \mathcal{N}_1 \to \mathcal{N}_2$ an isomorphism then Φ_f is an isomorphism from $\Phi\mathcal{N}_1$ to $\Phi\mathcal{N}_2$.*

Proof. We reason by induction, in the case where the nets are of size one, since f is an isomorphism between the two nets it must preserve the role of the ports, and hence the two nets are either both cells or both wires. If both of them are cells $c_1 = (P_1, \delta_{c_1}, \alpha)$ and $c_2 = (P_2, \delta_{c_2}, \alpha)$ since f is an isomorphism then note that $P_2 = \{f(p_0), \dots, f(p_n)\}$, to conclude note that we have the following series of equalities.

$$\begin{aligned}
\Phi_f\Phi(c_1) &= \Phi_f[-\check{p_1}\alpha x, \dots, -\check{p_n}\alpha x, +\check{p_0}\alpha x] \\
&= [-\Phi_f\check{p_1}\alpha x, \dots, -\Phi_f\check{p_n}\alpha x, +\Phi_f\check{p_0}\alpha x] \\
&= [-f(\check{p_1})\alpha x, \dots, -f(\check{p_n})\alpha x, +f(\check{p_0})\alpha x] \\
&= \Phi(c_2)
\end{aligned}$$

If both nets are wires $w = \{p, q\}$ and $f(w) = \{f(p), f(q)\}$ then the associated cells are $[\check{p}wx_1, \check{q}wx_2]$ and $[f(\check{p})wx_1, f(\check{q})wx_2]$ to conclude note that applying Φ_f to $\Phi(w)$ results in $\Phi(f(w))$.

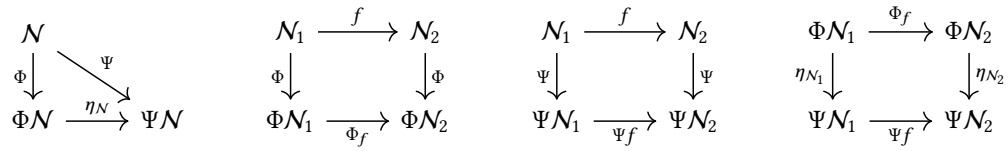

Figure 4: By providing the translations Φ and Ψ we provide from a categorical point of view two functors from the category of interaction nets to the category of unification nets. Also the proposition 4.6 can be expressed in the means of a diagram, furthermore we exhibit that the substitutions $\{\check{p} \mapsto pos_{\mathcal{N}}(p)\}$ yield a natural transformation η from the functor Φ to the functor Ψ.

To conclude we proceed with the induction step. Assume the nets are of size $n+1$ and $\mathcal{N}_1 = \mathcal{N}_1' + c$ where c is either a wire or a cell, now since f is an isomorphism it means that $\mathcal{N}_2 = f(\mathcal{N}_1' + c)$. To conclude we note first the linearity of the function involved: Φ is linear by definition, Φ_f is a substitution and hence is linear (see section 4.2), f is a morphism and hence is linear w.r.t. to the union of interaction nets. Together with the induction hypothesis we can conclude.

$$\begin{aligned}
\Phi_f\Phi(\mathcal{N}_1) &= \Phi_f\Phi(\mathcal{N}_1' + c) \\
&= \Phi_f(\Phi(\mathcal{N}_1') + \Phi(c)) \\
&= \Phi_f\Phi(\mathcal{N}_1') + \Phi_f\Phi(c)) \\
&= \Phi(f(\mathcal{N}_1')) + \Phi(f(c)) \\
&= \Phi(f(\mathcal{N}_1') + f(c)) \\
&= \Phi(f(\mathcal{N}_1' + c)) \\
&= \Phi(\mathcal{N}_2)
\end{aligned}$$

□

Definition 4.19 (Position prefix, locating system)**.** Given a port p in an interaction net $\mathcal{N}$ we define its position prefix denoted $pos_{\mathcal{N}}(p)$ inductively such that, if the port p is in use in a cell c then $\text{pos}(p) = \check{c}i$ if p is an auxiliary port of the cell such that $\delta_c(p) = i$. If p is the principal door of c then $\text{pos}(p) = \check{c}0$. If on the other hand p is not in use then $\text{pos}(p) = \check{p}$.

The *locating system* of an interaction net $\mathcal{N}$ is the substitution $\{\check{p} \mapsto pos_{\mathcal{N}}(p), p \in P\}$ which associates to each absolute position variable $\check{p}$ a the term $pos_{\mathcal{N}}(p)$, leaving invariant other variables.

Definition 4.20 (Explicit Translation)**.** We define the translation map Ψ from interaction nets to unification nets by induction:

- To a cell c of symbol α of arity n, the map Ψ associates $\check{c} \cdot [-1x, \ldots, -nx, +\alpha x]$.
- To a wire $\{p, q\}$ we associate $[\text{pos}(p)wx_1, \text{pos}(q)wx_2]$.
- Given $\mathcal{N}$ to which is added a cell c hence to $\mathcal{N} + c$, the map Ψ associates $\Psi(\mathcal{N})[\check{p} \mapsto pos(p)] + \Psi(c)[\check{p} \mapsto pos(p)]$.
- To a net to which is added a wire $\mathcal{N} + w$ it associates the image $\Psi(\mathcal{N})[\check{p} \mapsto pos(p)] + \Psi(w)[\check{p} \mapsto pos(p)]$.

Proposition 4.6 (Decomposition of the explicit translation)**.** *Given an interaction net $\mathcal{N}$, then* $\Psi(\mathcal{N}) = \Phi(\mathcal{N})\{\check{p} \mapsto pos_{\mathcal{N}}(p)\}$.

Proof. We do again the proof by induction on the size of the net $\mathcal{N}$. In the case where the net is a cell c of symbol α of arity n. Let us denote $P = \{p_0, \ldots, p_n\}$ where p_0 is the principal port and for each i we have $\delta_c(p_i) = i$. Thus, the image of c under Φ is $[-\check{p_1}\alpha x, \ldots, -\check{p_n}\alpha x, +\check{p_0}\alpha x]$. Hence $\Phi(\mathcal{N})\{\check{p} \mapsto pos_{\mathcal{N}}(p)\}$ will correspond to $[-\check{c}1\alpha x, \ldots, -\check{c}n\alpha x, +\check{c}0\alpha x]$ which is exactly the image of c under Ψ.

On the other hand is the net is made of only one wire $w = \{p, q\}$ the associated net by Φ and Ψ is $[\check{p}wx_1, \check{q}wx_2]$ then remark that this net is unchanged by the substitution $\{\check{p} \mapsto pos_{\mathcal{N}}(p)\}$ to conclude.

Let us treat the induction step writing the net as $\mathcal{N}+c$ where c is either a wire or a cell. By definition then $\Psi(\mathcal{N}+c) = \Psi(\mathcal{N})[\check{p} \mapsto pos_{\mathcal{N}}(p)] + \Psi(c)[\check{p} \mapsto pos_{\mathcal{N}}(p)]$ applying the induction hypothesis this sum corresponds to $\Phi(\mathcal{N})\{\check{p} \mapsto pos_{\mathcal{N}}(p)\}[\check{p} \mapsto pos_{\mathcal{N}}(p)] + \Phi(c)\{\check{p} \mapsto pos_{\mathcal{N}}(p)\}[\check{p} \mapsto pos_{\mathcal{N}}(p)]$ since $\{\check{p} \mapsto pos_{\mathcal{N}}(p)\}$ is idempotent we can then conclude the identity. □

Proposition 4.7 (Commutation for the position prefix)**.** *Given $\mathcal{N}_1$ and $\mathcal{N}_2$ two interaction nets and $f : \mathcal{N}_1 \to \mathcal{N}_2$ an isomorphism, for any port $p \in P_1$ we have the following equality $\check{f} pos_{\mathcal{N}_1}(p) = pos_{\mathcal{N}_2}(f(p))$.*

Proof. Consider some port p say p has no role in $\mathcal{N}_1$ then $pos_{\mathcal{N}_1}(p) = \check{p}$ and by definition $\check{f}\check{p}$ corresponds to $\check{f(p)}$. Since f is an isomorphism f preserve the role of the port and so $f(p)$ has no role aswell hence $pos_{\mathcal{N}_2}(f(p)) = \check{f(p)}$ this show the two image are equal.

Now assume p has a role αi and belong to a cell c, in that case $pos_{\mathcal{N}_1}(p) = \check{c}i$ then the image under f of this term becomes $f\check{(c)}i$. On the other hand, since f is an isomorphism $f(p)$ is also of role αi and belong to some cell d which is the image of c under f (again because f is an isomorphism), hence $pos_{\mathcal{N}_2}(f(p)) = \check{d}i = f\check{(c)}i$. This concludes the proof and show that the two maps are the same. □

Proposition 4.8 (Locating systems as Natural transformations)**.** *Given an isomorphism $f : \mathcal{N}_1 \to \mathcal{N}_2$ then $\Psi_f\Phi(\mathcal{N}_1)\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\} = \Phi(\mathcal{N}_2)\{\check{p} \mapsto pos_{\mathcal{N}_2}(p)\}$. Note that $\Phi(\mathcal{N}_2)\{\check{p} \mapsto pos_{\mathcal{N}_2}(p)\}$ corresponds to $[\Phi_f\Phi(\mathcal{N}_1)]\{\check{p} \mapsto pos_{\mathcal{N}_2}(p)\}$.*

Proof. We proceed by induction. Consider a cell c_1 and c_2 two isomorphic cell under f. The cell $\Phi(c_1)$ is of the form $[p_1\alpha x, \dots, p_n\alpha x, p_0\alpha x]$, whereas the cell $\Phi(c_2)$ the image of its isomorphic cell c_2 will be of the form $[f(p_1)\alpha x, \dots, f(p_n)\alpha x, f(p_0)\alpha x]$. Then applying to the two functions to $\Phi(c_1)$ we obtain the following:

$$\begin{aligned}
\Psi_f\Phi(c_1)\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\} &= \Psi_f[p_1\alpha x, \dots, p_n\alpha x, p_0\alpha x]\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\} \\
&= \Psi_f[\check{c}1\alpha x, \dots, \check{c}n\alpha x, \check{c}0\alpha x] \\
&= [f\check{(c)}1\alpha x, \dots, f\check{(c)}n\alpha x, f\check{(c)}0\alpha x] \\
&= [f(\check{p_1})\alpha x, \dots, f(\check{p_n})\alpha x, f(\check{p_0})\alpha x]\{\check{p} \mapsto pos_{\mathcal{N}_2}(p)\} \\
&= \Phi(c_2)\{f(p_i) \mapsto pos_{\mathcal{N}_2}(f(p_i))\}
\end{aligned}$$

If we were to consider that both nets are wires $w_1 = \{p, q\}$ and $w_2 = \{f(p), f(q)\}$ then we follow the same reasoning:

$$\begin{aligned}
\Psi_f\Phi(w_1)\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\} &= \Psi_f[pwx_1, qwx_2]\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\} \\
&= \Psi_f[pos_{\mathcal{N}_1}(p)wx_1, pos_{\mathcal{N}_1}(q)wx_2] \\
&= [pos_{\mathcal{N}_2}(f(p))wx_1, pos_{\mathcal{N}_2}(f(q))wx_2] \\
&= [f(p)wx_1, f(q)wx_2]\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\} \\
&= \Phi(w_2)\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\}
\end{aligned}$$

Now let us proceed with the induction, consider that $\mathcal{N}_1$ is of the form $\mathcal{N}_1' + c$ where c is either a cell or a wire. Noting that all the function involved – i.e. Ψ_f, Φ and the location system induced by $\mathcal{N}_1$ and $\mathcal{N}_2$

– are linear, we can conclude by calling the induction hypothesis:

$$\begin{aligned}\Psi_f\Phi(\mathcal{N}_1' + c)\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\} &= \Psi_f\Phi(\mathcal{N}_1')\{\check{p} \mapsto pos_{\mathcal{N}_1'}(p)\} + \Psi_f\Phi(c)\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\}\\ &= \Phi(\mathcal{N}_2')\{\check{p} \mapsto pos_{\mathcal{N}_2'}(p)\} + \Phi(f(c))\{\check{p} \mapsto pos_{\mathcal{N}_2}(p)\}\\ &= \Phi(\mathcal{N}_2' + f(c))\{\check{p} \mapsto pos_{\mathcal{N}_2}(p)\}\\ &= \Phi(\mathcal{N}_2)\{\check{p} \mapsto pos_{\mathcal{N}_2}(p)\}\end{aligned}$$

□

Proposition 4.9 (Explicit translation as Functor)**.** *Given $f : \mathcal{N}_1 \to \mathcal{N}_2$ an isomorphism between two interaction nets, then $\check{f}$ is an isomorphism between $\Psi(\mathcal{N}_1)$ and $\Psi(\mathcal{N}_2)$.*

Proof. Using the decomposition of proposition 4.6 $\Psi(\mathcal{N}_1) = \Phi(\mathcal{N}_1)\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\}$ therefore by applying Ψ_f, we obtain $\Psi_f\Phi(\mathcal{N}_1)\{\check{p} \mapsto pos_{\mathcal{N}_1}(p)\}$. Using the proposition 4.8 we know this corresponds to $\Phi(\mathcal{N}_2)\{\check{p} \mapsto pos_{\mathcal{N}_2}(p)\}$ which itself finally corresponds to $\Psi(\mathcal{N}_2)$ using the proposition 4.6. □

4.5 From elementary unification nets to Interaction nets

Definition 4.21 (Matching)**.** A *matching* of an interaction net $\mathcal{N}$ is an injective function M mapping untied port of $\mathcal{N}$ to tied port of $\mathcal{N}$ i.e. port of the form $\delta_c(i)$. Given an isomorphism $f : \mathcal{N}_1 \to \mathcal{N}_2$ and a matching M of $\mathcal{N}_1$ the *matching associated to M under f* is denoted $f(M)$ and corresponds to the function matching each port $f(p)$ to the port $\delta_{f(M_1(p))}(M_2(p))$ whenever p is in the support of M and is the identity otherwise.

Proposition 4.10 (Matching commutation)**.** *Given an isomorphism $f : \mathcal{N}_1 \to \mathcal{N}_2$ between two interaction nets, for any matching M of $\mathcal{N}_1$ the function f is an isomorphism from $M(\mathcal{N}_1)$ to $M(\mathcal{N}_2)$. Furthermore, we have the identity $f(M)(f(\mathcal{N}_1)) = f(M(\mathcal{N}_1))$.*

Proof. We proceed by induction. If both nets are cells of symbol α there can be no matching since no port is untied in $\mathcal{N}_1$, hence the proposition is true.

If the two nets are wires $w = \{p_1, p_2\}$ and $f(w) = \{q_1, q_2\}$ applying M to w results in $\{M(p_1), M(p_2)\}$, then if we apply f we obtain $\{f(M(p_1)), f(M(p_2))\}$. On the other applying f follow by $f(M)$ results in $\{f(M)(f(p_1)), f(M)(f(p_2))\}$. Now given p a port that is either p_1 or p_2, if p is invariant under M then $M(p) = p$ and hence by definition $f(M)(f(p)) = f(p)$. If p is in the support of M then it corresponds to $\delta_{M_1(p)}(M_2(i))$. Hence since f is an isomorphism, $f(M(p))$ is $\delta_{f(M_1(p))}(M_2(p))$. On the other hand, $f(M)f(p)$ is by definition $\delta_{f(M_1(p))}(M_2(p))$, noting that both image are the same we conclude. So we conclude $f(M(w))$ corresponds to $f(M)(f(w))$

For the induction step consider a net $\mathcal{N}+c$ (where c is either a wire or a cell) then by linearity of both functions applying M followed by f corresponds to $f(M(\mathcal{N})) + f(M(c))$ now calling the induction hypothesis we claim this sum is equal to $f(M)(f(\mathcal{N})) + f(M)(f(c))$ which again by linearity corresponds to $f(M)(f(\mathcal{N} + c))$, which concludes the induction.

□

Definition 4.22 (Reverse Translation)**.** Given a relocation f we construct an isomorphism in the class of interaction nets as Λ_f which maps the cells $\lambda^{-1}\check{c}$ tp $\lambda^{-1}\check{f(c)}$.

Given a matching substitution θ made of atomic substitutions of the form $\{\check{p} \mapsto \check{c}i\}$, its *Lafont translation* is denoted Λ_θ, and is the function that operates on interaction nets such that it associates interaction net (P, C, W, ρ) the net (P, C, W, ρ') such that given a port q of $\mathcal{N}$:

- For any port $q \neq p$ of $\mathcal{N}$ we have $\rho_{\mathcal{N}'}(q) = \rho_{\mathcal{N}}(q)$.
- If p belong to $\mathcal{N}$ we define $\rho_{\mathcal{N}'}(p) = (c, i)$.

Given an unification net we give a way to construct a interaction net by induction:

- To a cell $\check{c}[1\alpha x, \dots, n\alpha x, 0\alpha x]$ we associate the cell c of symbol α and arity n.
- To a wire $[\check{p}wx_1, \check{q}wx_2]$ we associate the wire $\{p, q\}$ or the wire $\{\delta_c^{-1}(i), \delta_c^{-1}(j)\}$.
- To a sum $\mathcal{N}_1 + \mathcal{N}_2$ we associate the union $\Lambda(\mathcal{N}_1) + \Lambda(\mathcal{N}_2)$.
- Given an elementary unification net $\mathcal{N}$ and a matching θ we associate $\Lambda_\theta \Lambda\mathcal{N}$.

Proposition 4.11 (Λ–matching commutation)**.** *Given $f : \mathcal{N}_1 \to \mathcal{N}_2$ a relocation and θ a matching substitution for any standard unifiction net $\mathcal{N}$ we have $\Lambda_f(\Lambda_\theta)\Lambda(\mathcal{N}) = \Lambda_{\theta^f}\Lambda(f\mathcal{N})$*

Proof. Again we proceed by induction. Consider that the net $\mathcal{N}$ is made of one cell. θ contains substitutions of the form $\{\check{p} \mapsto \check{c}i\}$ and without loss of generality we can assume it is made of only one. Λ_θ maps the wire–port p to ports $\delta_c(i)$. Hence in the case of a cell this map has no effect since the wire port p does not occur in c. Applying then Λ_f results in $\Lambda_f(c)$ which is $f(c)$. On the other hand $\Lambda(f(c))$ corresponds to $f(c)$ and Λ_{θ^f} maps the positions $f(p)$ to $\delta_{f\theta_1(p)}(\theta_2(p))$ again this map is invariant and so this results in $f(c)$.

Consider that the two nets are wires w_1 and w_2. then $\Lambda w = \{p_1, p_2\}$ if it is unchanged by Λ_θ applying then Λf results in $\Lambda(f(w))$. Then note that $\Lambda(f(w))$ is also invariant under Λ_{θ^f} to conclude.

If on the contrary the wire is in the support of Λ_θ consider p one of the two p_1 or p_2 then $\Lambda_\theta p = \lambda^{-1}\theta(\check{p})$ then, applying Λf we obtain $\lambda^{-1}f\theta(\check{p})$. On the other hand $\lambda f p$ corresponds to $\lambda^{-1}(f(\check{p}))$ which then applying $\Lambda f\theta$ maps this element to $\delta_{f(\theta_1(p))}(\theta_2(p))$ which correspond to $f\delta_{\theta_1(p)}(\theta_2(p))$ i.e. $\lambda^{-1}f\theta(\check{p})$. Hence we obtain the equality we seek.

$$\begin{aligned}
\Lambda_f\Lambda_\theta p &= \Lambda_f\lambda^{-1}\theta\check{p} \\
&= \lambda^{-1}f\theta\check{p} \\
&= \lambda^{-1}\theta^f f\check{p} \\
&= \Lambda_{\theta^f}\lambda^{-1}f\check{p} \\
&= \Lambda_{\theta^f}\Lambda_f\lambda^{-1}\check{p} \\
&= \Lambda_{\theta^f}\Lambda_f p
\end{aligned}$$

For the induction step, Assume the net is of the form $\mathcal{N} + c$ where c is either a cell or a wire, then by linearity $\Lambda_f(\Lambda_\theta)(\mathcal{N}+c) = \Lambda_f(\Lambda_\theta)\mathcal{N} + \Lambda_f(\Lambda_\theta)c$ by applying the induction hypothesis we conclude that this sum corresponds to $(\Lambda_{\theta^f})\Lambda(f\mathcal{N}) + (\Lambda_{\theta^f})\Lambda(fc)$ which again by linearity of the maps corresponds to $(\Lambda_{\theta^f})\Lambda(f(\mathcal{N}+c))$

□

Proposition 4.12 (Translation as functor)**.** *Given a relocation $f : \mathcal{N}_1 \to \mathcal{N}_2$ then $\Lambda f : \Lambda(\mathcal{N}_1) \to \Lambda(\mathcal{N}_2)$ is an isomorphism of interaction nets.*

Proof. We reason by induction. Assume the two nets are cells $\check{c}[t_0, \dots, t_n]$ and $\check{d}[t_0, \dots, t_n]$ then f rename $\check{c}$ in $\check{d}$. Then applying Λ we obtain the cell c and d but Λf associates $\lambda^{-1}(\check{c})$ to $\lambda^{-1}(\check{d})$ and hence c to d. This shows $\Lambda(f\mathcal{N}_1) = \Lambda f\Lambda(\mathcal{N}_2)$.

$$\begin{aligned}
\Lambda_f\Lambda\check{c}[t_0, \dots, t_n] &= \Lambda_f c \\
&= \lambda^{-1}f(\check{c}) \\
&= \lambda^{-1}\lambda d \\
&= d \\
&= \Lambda\check{d}[t_0, \dots, t_n] \\
&= \Lambda f\check{c}[t_0, \dots, t_n]
\end{aligned}$$

Assume both nets are wires $[\check{p}_1 w x_1, \check{p}_2 w x_2]$ and $[\check{q}_1 w x_1, \check{q}_2 w x_2]$ then f corresponds to $\{\check{p}_1 \mapsto \check{q}_1, \check{p}_2 \mapsto \check{q}_2\}$. Applying Λ to the wires we respectively obtain $\{p_1, p_2\}$ and $\{q_1, q_2\}$. Then $\Lambda_f\{p_1, p_2\} = \{\Lambda_f p_1, \Lambda_f p_2\} = \{\lambda^{-1}\check{q}_1, \lambda^{-1}\check{q}_2\}$ which finally corresponds to $\{q_1, q_2\}$ as wanted.

For the induction step consider a net first consider the sum of two nets $\mathcal{N}_1 = \mathcal{N}^1 + \mathcal{N}^2$ in that case we can conclude immediately by induction since f, Λ and Λ_f are linear with respect to the sums. The second case to consider is when $\mathcal{N}_1$ is of the form $\theta(\mathcal{N}_1 *)$ then we conclude thanks to the following series of equation:

$$\begin{aligned}
\Lambda_f(\theta(\mathcal{N}_1^*)) &= \Lambda_f \Lambda_\theta \Lambda(\mathcal{N}_1^*) && \text{(definition 4.22)}\\
&= \Lambda_{\theta^f} \Lambda(f\mathcal{N}_1^*) && \text{(proposition 4.11)}\\
&= \Lambda(\theta^f f\mathcal{N}_1^*) && \text{(definition 4.22)}\\
&= \Lambda(f(\theta\mathcal{N}_1^*)) && \text{(proposition 4.3)}
\end{aligned}$$

□

5 Interaction nets in Stellar Resolution

5.1 General interaction nets

In this section we show how the class of unification nets we introduced is suited to describe the computation occuring in interaction nets. First we treat the case of general interaction nets, showing it is well–suited by proving general results due to Lafont occuring in [Laf97]. We follow the same arguments but in this new setting obtaining more syntactical proofs. We then in the next section do the same analysis for interaction combinators, which are known to simulate any other class of interaction nets [Laf97].

Definition 5.1 (Translation of an interaction rule)**.** Given an interaction rule $\rightarrow_s$ for the α, β redex the *translation* of $\rightarrow_s$ is denoted $\rightarrow_{\bar{s}}$ and corresponds to the rewriting rule on unification nets define such that for two unification nets $\mathcal{N}_1$ and $\mathcal{N}_2$ $\mathcal{N}_1 \rightarrow_{\bar{s}}$ if and only if $\Lambda\mathcal{N}_1 \rightarrow_s \Lambda\mathcal{N}_2$.

Redex in unification nets. Consider an interaction rule s that rewrites the redex $\alpha \bowtie \beta$ into $s[\alpha, \beta]$. Note that the translation of this rule in unification nets is given by the following redex

$$[-11\alpha x, \dots, -1n\alpha x, +10\alpha x] + [-21\beta y, \dots, -2k\beta y, +20\beta y] + [-1z_1, -2z_2] \rightarrow_{\bar{s}} \Phi(s[\alpha, \beta])$$

together with the fact that $\rightarrow_{\bar{s}}$ is linear with respect to the sum. Note that the left hand side of the rewriting rule corresponds to the translation of $\alpha \bowtie \beta$.

Unification and computation of Interaction nets. In order to fully work with elementary interaction net we must consider a new rule that does not occur in interaction net that we will call the wiring reduction given two wires $[Uwx_1, Vwx_2]$ and $[U'wy_2, V'wy_2]$ assuming that U and U' can be unified then this reduces to $[Vwx_2, V'wy_2]$ note that this is a particular case of the only reduction used in Stellar resolution [ES21].

On the other hand note that the reduction used in the previous paragraph do not necessarily look like the reduction of stellar resolution, yet since both model are turing complete we can safely assume that the resolution–rule can simulate any rules on interaction nets. We choose however to not look for such an encoding here since the given translation conserve reduction *in the same number of step* plus the wiring reductions which are at most the number of wires. In other words the computational steps

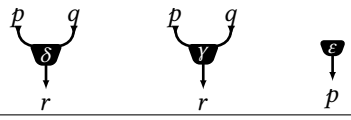

Figure 5: Interaction combinators from the interaction net paradigm proposed by Lafont [Laf89],[Laf97]. The δ–combinator is called *duplicator*, the γ–combinator is called *generator* while the ε–combinator is called the *eraser*.

(and so the computation) in the version of unification nets we presented is atmost $P(n) = n + k$ where n is the number of reduction steps of the interaction net and k is its number of wires. This shows both models are closely related since simulation of the execution occurs in polynomial time.

Proposition 5.1 (Strong confluence of interaction nets)**.** *Given an interaction net $\mathcal{N}$ such that $\mathcal{N}$ reduces to $\mathcal{N}_1$ using an interaction rule between two cells and to $\mathcal{N}_2$ using an interaction rule between two other cells. Then $\mathcal{N}_1$ and $\mathcal{N}_2$ reduce to a same net $\mathcal{N}'$ in one step.*

Proof. An interaction net reduction is a rewriting rule of the form $c + d + [-cz, -dz'] \to \mathcal{N}(c, d)$ that conserve free ports. Hence the two distinct redex in $\mathcal{N}$ are of the form $R_1 = c + d + [-cz, -dz]$ and $R_2 = e + f + [-ez, -fz]$. The symbols c and d (resp. e and f) cannot be the same since a single wire cannot involve the same position twice. Furthermore, if c (resp. d) is equal to e or f, since a position may only be involved in at must one wiring the two wiring would be equal, hence the redex are the same which would indeed ensure strong confluency. So let's now assume c (and d) are equal to neither e or f and hence that all the positions c, d, e, f are distinct.

To conclude strong confluency, note that we can write the net $\mathcal{N}$ as a sum of modules $\mathcal{N} = R_1 + R_2 + \mathcal{N}_0$. Since reduction is stable by the sum $\mathcal{N} \to c \bowtie d + R_2 + \mathcal{N}_0$ while $\mathcal{N} \to R_1 + e \bowtie f + \mathcal{N}_0$ it is then clear that we can close the diagram reducing the remaining redexes R_2 (resp. R_1), obtaining the same net $\mathcal{N}_1 + \mathcal{N}_2 + \mathcal{N}_0$.

On the other hand it could occur that the two rules are wiring reduction in that case again we can easily verify confluence is preserved.

The last case is when one reduction is a wiring and the other an interaction rule again we can show confluence in the same fashion.

□

Given a net $\mathcal{N}$ with free ports $p_1, \ldots, p_n$ we also denote $\mathcal{N}$ as $\mathcal{N}[\pi p_1, \ldots, \pi p_n]$.

Definition 5.2 (Principal net)**.** A *principal net* is a net containing no cut or axiom and having exactly one free positive port. A net is a *tree* if it is defined by the following induction:

- It is a cell of the form $[-1\alpha x, \ldots, -n\alpha x, +0\alpha x]$.
- given $\mathcal{N}_1, \ldots, \mathcal{N}_n$ some trees and an oriented cell $[-1x, \ldots, -nx, +\alpha x]$ then the unification net $[-1x, \ldots, -nx, +\alpha x] + 1\mathcal{N}_1 + \cdots + n\mathcal{N}_n$ is a tree.

Definition 5.3 (Tracing)**.** The *tracing* of a cell c in a net $\mathcal{N}$ is the net made of the cells c' such that the principal port of c' is connected to an auxiliary door of c.

Proposition 5.2 (Trees and principal nets)**.** *Given a net $\mathcal{N}$ made of polarized cells the proposition are equivalent:*

- *$\mathcal{N}$ is a principal.*
- *$\mathcal{N}$ is isomorphic to a tree.*

Proof. Let us show the first implication, and assume that $\mathcal{N}$ is principal. We reason by induction on the number of cells. If $\mathcal{N}$ contains one cell then indeed, since its only cell must be oriented i.e. it must contain one positive port exactly, indeed such a cell is a tree.

If on the other hand $\mathcal{N}$ is made of several cells $c_1 + \cdots + c_n$ (at least two). Let us denote c the cell containing the only free positive port of the cell. We aim to show that not all negative port of c can be free. Assume all negative ports of c are free. and take a second cell c_0 distinct from c, we know such a cell exists since our net contains at least two cells. The principal port of c_0 must be unifiable with some other negative port (otherwise the net is not principal) this negative port belongs to another cell c_1. The cell c_1 cannot be c otherwise one of the negative port of c is not free, furthermore it cannot be c_0 otherwise the path c_0, c_1 contains a *vicious cycle*. In the same way we can generate a cell c_2, distinct from c and the previous cells c_0, c_1. This procedure generates an infinite quantity of distinct cells $c_0, \ldots, c_l, \ldots$ which is absurd since the net is supposed to be made of a finite number of cells.

Consider then the non free negative ports of c denote them $q_1, \ldots, q_l$ each of these is unifiable with a positive port $\overline{q}_1, \ldots, \overline{q}_l$ and each of these port belong to a distinct cell $c_1, \ldots, c_l$. Consider some cell c' that is not $c, c_1, \ldots, c_l$. The principal port of c' must be unified since otherwise the net is not principal. Let us show that the principal door of c' is not connected to an auxiliary port of c. We claim that it must be connected to an auxiliary door of some c_i. Assume otherwise, then the principal port of c' is connected to some cell c'_1 that is not any of the cells $c_1, \ldots, c_l$, but also c'_1 may not be c since otherwise c' would be one of the cells $c_1, \ldots, c_l$. Furthermore the positive port of c'_1 must be unifiable hence, there exists a cell c'_2 such that c'_1 connects its principal door to an auxiliary door of c'_2. Again, c'_2 may not be a c_i by hypothesis, and it may not be c since otherwise it means c'_1 is one of the cell c_i. In this way we can generate an infinite number of cells which leads to a contradiction of the finiteness of the net.

So we know that a cell c' of $\mathcal{N}$ is either c or belongs to the tracing of one of the cells $c_1, \ldots, c_l$. Hence $\mathcal{N} = c + tracing(c_1) + \cdots + tracing(c_l)$ since each tracing is a proper subnet of $\mathcal{N}$ and is principal we can call the induction hypothesis and claim they are equivalent to trees $\mathcal{T}_1, \ldots, \mathcal{T}_l$. Then $\mathcal{N}$ is equivalent to $[-1x, \ldots, -nx, +\alpha x] + i_1\mathcal{T}_1 \cdots + i_l\mathcal{T}_l$. Indeed note that $i_j x$ can be unified with $i_j \beta y$ which is the form of the principal door of $\mathcal{T}_j$.

Let us treat the other implication showing that any tree is a principal net. We also reason by induction. Consider a tree containing one cell, indeed such a tree is principal. Inductively consider the tree $[-1x, \ldots, -nx, +\alpha x] + 1\mathcal{T}_1 + n\mathcal{T}_n$. Each net $\mathcal{T}_i$ is a tree of smaller size than $\mathcal{T}$, hence we can call the induction hypothesis and claim that they are principal nets. Indeed since no $\mathcal{T}_i$ contains a cut or axiom and that the cell $c = [-1x, \ldots, -nx, +\alpha x]$ is not one the net contains no such cells. Furthermore given a positive free port if it belongs to a cell d of the tree $\mathcal{T}_i$ if it belongs to the root that port as for prefix $+i$ and can therefore be unified with the port of the form $-ix$ in c. If it is not the root of $\mathcal{T}_i$ d must be unifiable with some d' otherwise $\mathcal{T}_i$ is not a tree. Let us show no vicious cycle is occuring in the net. Consider a vicious cycle $c_1, \ldots, c_n$. Then each principal door of the cell must be unifiable, hence c cannot be contained in the cycle. Furthermore if two cells d and d' are connected and are distinct from c then there binds involves a negative position p and a positive one q that must have the same prefix. If the cells belong to two distinct tree there prefix are different hence the position cannot be unified. Hence each $c_1, \ldots, c_n$ are distinct from c and must belong to the same tree $\mathcal{T}_i$ since the tree contains no vicious cycle then $c_1, \ldots, c_n$ is not a vicious cycle. Since the net contains no cut or axiom, exactly one free positive ports, and has no vicious cycle we conclude that – by definition – it is a principal net. □

Proposition 5.3. *Given two nets $\mathcal{N}$ and $\mathcal{N}'$ and their free ports are respectively $p_1, \ldots, p_n$ and $q_1, \ldots, q_k$.*

If the two sets of free ports are equivalent and the two anchored sub–nets are equivalent then the two nets are equivalent.

Proof. If the nets contains no wire then by hypothesis they are anchored and so by hypothesis they are equivalent. If $\mathcal{N}$ contains one wire (p, q)

□

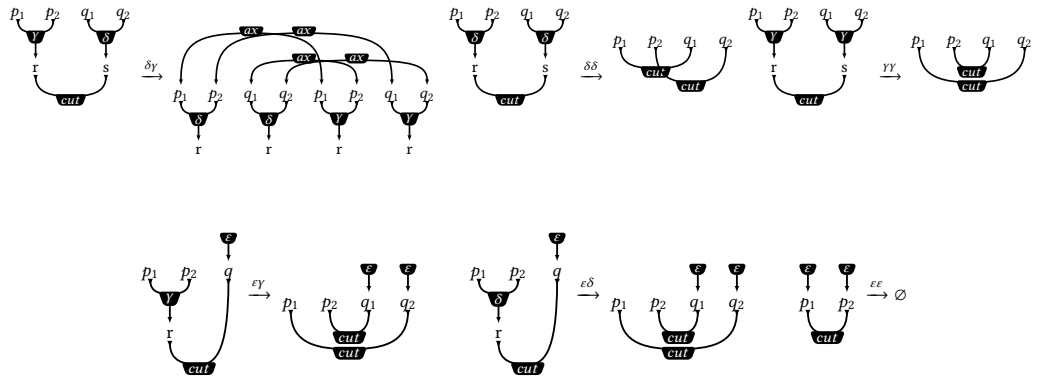

Figure 6: Rules for the interaction combinators [Laf97] written in the style of proof net (cut corresponds to interaction). We distinguish two types of reductions, the *communications* $(\varepsilon\delta)$, $(\varepsilon\gamma)$ and $(\delta\gamma)$, and the *annihilations* $(\delta\delta)$, $(\gamma\gamma)$ and $(\varepsilon\varepsilon)$

.

5.2 Interaction Combinators

Definition 5.4 (Interaction Combinator)**.** The interaction combinators are the following: $[+0\varepsilon x]$, $[-1\gamma x, -2\gamma x, +0\gamma x]$ and $[-1\delta x, -2\delta x, +0\delta x]$. The nets we consider are elementary nets build with this set of cells.

Definition 5.5 (Reductions)**.** We give the following reductions, each of them being an interaction rule:

- $[-1\mathsf{l}\delta x, -1\mathsf{r}\delta x, +10\delta x]+[-2\mathsf{l}\delta y, -2\mathsf{r}\delta y, +20\delta y]+[-1z, -2z] \to [-\theta 1\mathsf{l}\delta x, -\theta 2\mathsf{l}\delta x]+[-\theta 1\mathsf{r}\delta x, -\theta 2\mathsf{r}\delta x]$
- $[-1\mathsf{l}\gamma x, -1\mathsf{r}\gamma x, +10\gamma x]+[-2\mathsf{l}\gamma y, -2\mathsf{r}\gamma y, +20\gamma y]+[-1z, -2z] \to [-\theta 1\mathsf{l}\gamma x, -\theta 2\mathsf{r}\gamma x]+[-\theta 1\mathsf{r}\gamma x, -\theta 2\mathsf{l}\gamma x]$
- $[+1\varepsilon x] + [+2\varepsilon y] + [-1z, -2z] \to []$
- $[-1\mathsf{l}\delta x, -1\mathsf{r}\delta x, +1\delta x] + [+2\varepsilon y] + [-1z, -2z] \to [-\theta 1\mathsf{l}\delta x, -\theta 2\varepsilon x] + [+2\varepsilon y] [-\theta 1\mathsf{r}\delta x, -\theta 3\varepsilon x] + [+3\varepsilon w]$
- $[-1\mathsf{l}\gamma x, -1\mathsf{r}\gamma x, +1\delta x] + [+2\varepsilon y] + [-1z, -2z] \to [-\theta 1\mathsf{l}\gamma x, -\theta 2\varepsilon x] + [+2\varepsilon y] [-\theta 1\mathsf{r}\gamma x, -\theta 3\varepsilon x] + [+3\varepsilon w]$
- $[-1\mathsf{l}\delta x, -1\mathsf{r}\delta x, +1\delta x]+[-2\mathsf{l}\gamma y, -2\mathsf{r}\gamma y, +2\delta y]+[-1z, -2z] \to \sum \theta[-ij\mathsf{l}\alpha x, -ij\mathsf{r}\alpha x, +ij\alpha x]+\sum_k [+1ki, +2jk]$
- $[\alpha u, -t] + [+t, \beta v] \to [\alpha u, \beta v]$ where α and β are polarities.

Definition 5.6 (Multiplexor and Transpositor)**.** A right (resp. left) multiplexor of size n, is denoted M_n (resp. M_n^*) and is defined inductively as follow:

- $M_0 = M_0^* = [+\epsilon x]$ and $M_1 = M_1^* = []$ and $M_2 = M_2^* = [-\mathsf{l}x, -\mathsf{r}x, +\gamma x]$
- $M_{n+1} = [-\mathsf{l}x, -\mathsf{r}x, +\gamma x] + \mathsf{r} \cdot M_n$ and $M_{n+1}^* = [-\mathsf{l}x, -\mathsf{r}x, +\gamma x] + \mathsf{l} \cdot M_n$.

The autodual multiplexor of size n, is denoted T_n and is defined inductively as follow:

- $T_0 = [+\epsilon x]$ and $T_1 = []$ and $T_2 = [-\mathsf{l}x, -\mathsf{r}x, +\delta x]$
- $T_{n+1} = [-\mathsf{l}x, -\mathsf{r}x, +\delta x] + \mathsf{r} \cdot T_n$

Definition 5.7 (dual of a word with respect to an alphabet involution)**.** The *dual* of a word $w = (w_1, \ldots, w_n)$ constructed on $\{1, 2\mathsf{l}, \mathsf{r}\}$ is defined as $\overline{w} = (\overline{w_1}, \ldots, \overline{w_n})$ where $x \mapsto \overline{x}$ is involutive and $\overline{1} = 2$ and $\overline{\mathsf{l}} = \overline{\mathsf{r}}$.

The *prefix of a net* $\mathcal{N}$ denoted $pre(\mathcal{N})$ is the set of prefix of the terms $\pi(p)$.

Proposition 5.4 (Prefix of the transpositor and multiplexor)**.** *Given any integer n* $pre(M_n) = pre(T_n) = \{\mathsf{r}^k\mathsf{l} \mid k < n-1\} \cup \{\mathsf{r}^{n-1}\}$.

Proof. Again we proceed by induction. If $n = 0$ or 1 both sets are empty which is corresponds and the nets have no prefix so this is coherent. By induction now consider $T_{n+1} = [-\mathsf{l}x, -\mathsf{r}x, +\delta x] + \mathsf{r} \cdot T_n$ The

prefix of T_{n+1} is the set $\{\mathsf{l}\} \cup \mathsf{r}pre(T_n)$ and by induction $pre(T_n) = \{\mathsf{r}^k\mathsf{l} \mid k < n-1\} \cup \{\mathsf{r}^{n-1}\}$ Let's show now that the two sets are the same by showing the reciprocal inclusion.

Consider some prefix $\mathsf{r}^k\mathsf{l}$ where $k < n$ and show it belong to $\{\mathsf{l}, \mathsf{r}\} \cup \mathsf{r}pre(T_n)$. First, if $k = 0$ then the prefix belong to $\{\mathsf{l}\}$ If $0 < k < n$ then $k - 1 < n - 1$, and $\mathsf{r}^k\mathsf{l} = \mathsf{r}\mathsf{r}^{k-1}\mathsf{l}$, this element belong to $\mathsf{r}pre(T_n)$ and so to the prefix of T_{n+1}. The word r^n corresponds to $\mathsf{r}\mathsf{r}^{n-1}$ and hence belong to $\mathsf{r}pre(T_n)$.

For the other inclusion consider an element of $\{\mathsf{l}\} \cup \mathsf{r}pre(T_n)$ indeed the elements of $\{\mathsf{l}\}$ is of the form $\mathsf{r}^k\mathsf{l}$ (resp. $\mathsf{r}^k\mathsf{r}$) for $k = 0$. Now take some element of $\mathsf{r}pre(T_n)$ it can be in two form, the first case is if it corresponds to $\mathsf{r}\mathsf{r}^k\mathsf{l}$ (where l is either the left or the right symbol). with $0 < k < n - 1$ and hence $1 < k + 1 < n$, hence it is an element of the form $\mathsf{r}^k\mathsf{l}$ with $k < n$. In the second case the element can corresponds to $\mathsf{r}\mathsf{r}^{n-1}$ which is equal to r^n and so belong to $pre(T_{n+1})$.

□

Proposition 5.5 (Multiplexor and Transpositor behavior)**.** *Given some integer n we have the following reductions:*

- $[1z, 2z'] + 1T_n + 2T_n \to \sum_{p \in Prefix(T_n)} [-1px, -2px']$
- $[1z, 2z'] + 1M_n + 2M_n^* \to \sum_{p \in Prefix(T_n)} [-1px, -2\overline{p}x']$

Proof. We reason by induction on n if $n = 0$ then each net T_0, M_0, M_0^*, corresponds (up to renaming) to $[+\varepsilon x]$. Furthermore, in the net $[1z, 2z'] + 1[+\varepsilon x] + 2[+\varepsilon y]$ the cut $[1z, 2z']$ can be saturated by the substitution $\{z, z' \mapsto \epsilon x\}$, the reduction of this saturated cut then produces the empty net $[]$. On the other hand $[+\varepsilon x]$ has no position prefix hence a sum indexed by this sum is an empty sum, which corresponds to $[]$. Let us proceed with the induction assuming the proposition to be true for T_n lets show it holds for T_{n+1}. The net $[1z, 2z'] + 1T_n + 2T_n$ corresponds to

$$[1z, 2z'] + 1([\frac{-\mathsf{l}x, -\mathsf{r}x}{+\delta x}] + \mathsf{r} \cdot T_n) + ([\frac{-\mathsf{l}x, -\mathsf{r}x}{+\delta x}] + \mathsf{r} \cdot T_n) = [1z, 2z'] + [\frac{-1\mathsf{l}x, -1\mathsf{r}x}{+1\delta x}] + 1\mathsf{r} \cdot T_n + [\frac{-2\mathsf{l}x, -2\mathsf{r}x}{+2\delta x}] + 2\mathsf{r} \cdot T_n$$

In that net $[1z, 2z']$ can be saturated by the substitution $\{z, z' \mapsto \delta\}$, this producing the reduction (1). We can then apply the induction hypothesis in the step (2). Using the previous lemma $pre(T_{n+1}) = \{\mathsf{r}^k\mathsf{l} \mid k < n\} \cup \{\mathsf{r}^n\}$ and $pre(T_n) = \{\mathsf{r}^k\mathsf{l} \mid k < n-1\} \cup \{\mathsf{r}^{n-1}\}$ hence $pre(\mathsf{r}T_n) = \{\mathsf{r}^k\mathsf{l} \mid 0 < k < n\} \cup \{\mathsf{r}^n\}$. Note that $pre(T_{n+1}) = pre(\mathsf{r}T_n) \cup \{\mathsf{l}\}$, since the only element that is a prefix of T_{n+1} and not of $\mathsf{r}T_n$ correspond to $\mathsf{r}^0\mathsf{l}$ i.e. l. Hence we can conlude and obtain the equality (3).

$$\begin{aligned}
[1z, 2z'] + [-1\mathsf{l}x, -1\mathsf{r}x, +1\delta x] + 1\mathsf{r} \cdot T_n + \\
+2[-2\mathsf{l}x, -2\mathsf{r}x, +2\delta x] + 2\mathsf{r} \cdot T_n &\to [1\mathsf{l}z, 2\mathsf{l}z'] + [1\mathsf{r}z, 2\mathsf{r}z'] + 1\mathsf{r} \cdot T_n + 2\mathsf{r} \cdot T_n && (1)\\
&\to [1\mathsf{l}z, 2\mathsf{l}z'] + \sum_{p \in pre(\mathsf{r}T_n)} [-1px, -2px] && (2)\\
&= \sum_{p \in pre(T_n)} [-1px, -2px] && (3)
\end{aligned}$$

For the reduction of the interaction of the two transpositor nets M_n and M_n^* we also proceed by induction, the reasoning is then similar.

□

Bibliography

[Her30] Jacques Herbrand. *Recherches sur la théorie de la démonstration.* fr. Thèses de l'entre-deux-guerres 110. 1930. URL: http://www.numdam.org/item/THESE_1930__110__1_0/.

[Rob65] J. A. Robinson. "A Machine-Oriented Logic Based on the Resolution Principle". In: *J. ACM* 12.1 (Jan. 1965), pp. 23–41. ISSN: 0004-5411. DOI: 10.1145/321250.321253. URL: https://doi.org/10.1145/321250.321253.

[Hew72] Carl Hewitt. *Description and Theoretical Analysis (Using Schemata) of Planner: A Language for Proving Theorems and Manipulating Models in a Robot*. en. AI Technical Reports (1964 - 2004) AITR-258. 1972. URL: http://hdl.handle.net/1721.1/6916.

[MM82] Alberto Martelli and Ugo Montanari. "An Efficient Unification Algorithm". In: *Transactions on programming languages and systems (TOPLAS)* 4.2 (1982), pp. 258–282.

[Ede85] Elmar Eder. "Properties of substitutions and unifications". In: *Journal of Symbolic Computation* 1.1 (1985), pp. 31–46. ISSN: 0747-7171. DOI: https://doi.org/10.1016/S0747-7171(85)80027-4. URL: https://www.sciencedirect.com/science/article/pii/S0747717185800274.

[Laf89] Yves Lafont. "Interaction Nets". In: *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '90. San Francisco, California, USA: Association for Computing Machinery, 1989, pp. 95–108. ISBN: 0897913434. DOI: 10.1145/96709.96718. URL: https://doi.org/10.1145/96709.96718.

[Laf97] Yves Lafont. "Interaction Combinators". In: *Inf. Comput.* 137 (1997), pp. 69–101.

[Gir01] Jean-Yves Girard. "Locus Solum: From the Rules of Logic to the Logic of Rules". In: *Mathematical. Structures in Comp. Sci.* 11.3 (June 2001), pp. 301–506. ISSN: 0960-1295. DOI: 10.1017/S096012950100336X. URL: https://doi.org/10.1017/S096012950100336X.

[Maz07] Damiano Mazza. "A denotational semantics for the symmetric interaction combinators". In: *Mathematical Structures in Computer Science* 17.3 (2007), pp. 527–562. DOI: 10.1017/S0960129507006135.

[ES21] Boris Eng and Thomas Seiller. "Multiplicative Linear Logic from Logic Programs and Tilings". working paper or preprint. Jan. 2021. URL: https://hal.archives-ouvertes.fr/hal-02895111.