A Theory of Operational Equivalence for Interaction Nets

Maribel Fernández[1] and Ian Mackie[2]

[1] LIENS (CNRS UMR 8548), École Normale Supérieure
45 Rue d'Ulm, 75005 Paris, France. maribel@dmi.ens.fr
[2] CNRS-LIX (UMR 7650), École Polytechnique
91128 Palaiseau Cedex, France. mackie@lix.polytechnique.fr

Abstract. The notion of contextual equivalence is fundamental in the theory of programming languages. By setting up a notion of bisimilarity, and showing that it coincides with contextual equivalence, one obtains a simple coinductive proof technique for showing that two programs are equivalent in all contexts. In this paper we apply these (now standard) techniques to interactions nets, a graphical programming language characterized by local reduction. This work generalizes previous studies of operational equivalence in interaction nets since it can be applied to untyped systems, thus all systems of interaction nets are captured.

1 Introduction

Interaction nets, introduced by Lafont [7], are graph rewriting systems that generalize the multiplicative proof nets of linear logic, and can be seen both as a high-level programming language or as a low-level implementation language. A program consists of a net (a graph built from a set of agents and wires) and a set of interaction rules that describe the way in which the net will be reduced. We are interested in the problem of defining an equivalence relation between programs that compute the same results, or in other words, that behave in the same way, in all contexts. In that case, one program can be replaced by the other, for example for efficiency reasons, without altering the operational semantics of the system. To define this equivalence relation we first need to develop an operational theory of interaction nets specifying in a precise way how programs are executed (i.e. a strategy of evaluation of nets and a notion of value).

In [2] we proposed a way of adapting the coinductive techniques, used successfully for the functional and object-oriented programming paradigms, to give a notion of operational equivalence for the interaction paradigm. The language of interaction nets that was studied focussed on the notion of type, which is natural if interaction nets are seen as a programming paradigm. In particular, types allow us to distinguish values from programs. However, some applications of interaction nets do not fit into the typed framework in a natural way. For instance, systems based on the interaction combinators [8], or the systems of interaction used for the encoding of the λ-calculus [9], are untyped. Although it is possible to develop a type system for them [6], a natural approach would be to develop an operational theory of equivalence of interaction nets that does not rely on the notion of types. The same remark can be made in the case of functional languages based on the λ-calculus, where two different approaches can be found in the literature, depending on whether the calculus is typed (see for instance [10]) or untyped (see for instance [1]).

In this paper we present an operational theory for *untyped* interaction nets, including a notion of contextual equivalence and an associated bisimilarity relation which permits the use of coinductive techniques in the proofs of operational equivalence. To express these notions we use the textual calculus of interaction nets presented in [3] instead of the graphical language, since it allows us to give a concise and formal presentation. We leave the use of diagrams for the examples and intuitive explanations.

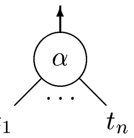
A system of interaction nets is a user-defined language, in the same spirit as systems based on term-rewriting. Our results are applicable to *any* system of interaction nets; we are not restricted to one specific set of rules. If the system is typed, the information provided by types can be used to obtain a more refined equivalence relation between nets, recovering the results of [2]. We remark that interaction nets are also used as an object language for the coding of other rewriting systems. The λ-calculus is perhaps the most studied example of this (see e.g. [4, 9]). Our results are also applicable here, so we have a proof technique for optimizations of such systems.

The paper is organized as follows. In the next section we set up the definition of interaction nets and define our evaluation strategy. Section 3 sets up to notion of bisimilarity. In Section 4 we give some examples of use of this relation. In Section 5 we formalize the notion of contextual equivalence, and Section 6 shows that this coincides with bisimilarity. Finally we conclude the paper in Section 7.

2 Background: Interaction Nets

We begin by presenting the textual calculus of interaction nets that we will use for the rest of the paper; we refer the reader to [3] for a more detailed description and examples.

Let Σ be a set of symbols, called *agents*, ranged over by $\alpha, \beta, \ldots$, each with a given *arity*, one *principal port* and a number of *auxiliary ports* equal to its arity. Σ can be partitioned into a set $\mathcal{C}$ of *constructors* and a set $\mathcal{D}$ of *destructors*, depending on the application. Let N be a disjoint set of names, ranged over by x, y, z, etc. *Terms* are defined by the grammar: $t ::= x \mid \alpha(t_1, \ldots, t_n)$, where $x \in N$, $\alpha \in \Sigma$, $arity(\alpha) = n$ and $t_1, \ldots, t_n$ are terms, with the restriction that each name may appear at most twice. $\mathcal{N}(t)$ denotes the set of names occurring in t. If a name occurs twice in a term, we say that it is *bound*, otherwise it is *free*. We write $\boldsymbol{t}$ for a list of terms $t_1, \ldots, t_n$. Graphically, a term of the form $\alpha(\boldsymbol{t})$ can be seen as a tree with connections between its leaves: the principal port of α (indicated by an arrow) is at the root, and the terms $t_1, \ldots, t_n$ are the subtrees connected to the auxiliary ports of α. A free variable represents a free port, and a bound variable represents a wire connecting two auxiliary ports.



If t and u are terms, then the (unordered) pair $t = u$ is an *equation*. Δ, Θ, ... will be used to range over multisets of equations. The graphical representation of an equation is a pair of trees connected by their roots (principal ports).

Interaction rules are pairs of terms written as $\alpha(\boldsymbol{t}) \bowtie \beta(\boldsymbol{u})$, where $(\alpha, \beta) \in \Sigma^2$ is the *active pair* of the rule. All names occur exactly twice in a rule, and there is one rule for each pair of agents.

Definition 1 (Configurations). *A* configuration *is a pair: $c = (\mathcal{R}, \langle \boldsymbol{t} \mid \Delta \rangle)$, where $\mathcal{R}$ is a set of rules, $\boldsymbol{t}$ a list $t_1, \ldots, t_n$ of terms, and Δ a multiset of equations. Each variable occurs at most twice in c. If a name occurs once in c then it is* free*, otherwise it is* bound*. For simplicity we sometimes omit $\mathcal{R}$ when there is no ambiguity. We use c, c' to range over configurations. We call $\boldsymbol{t}$ the* head *and Δ the* body *of a configuration.*

Intuitively, $(\mathcal{R}, \langle \boldsymbol{t} \mid \Delta \rangle)$ represents a net that we evaluate using $\mathcal{R}$. To draw the net we simply draw the trees for the terms in $\langle \boldsymbol{t} \mid \Delta \rangle$, connect the common variables together, and connect the roots of the trees corresponding to the members of an equation together. The roots of the terms in the head of the configuration and the free names correspond to free ports in the *interface* of the net. Note that the head of the configuration may contain all or just some of the ports in the interface of the net, called *observable*. For this reason, the head is called the *observable interface* of the configuration.

We work modulo α-equivalence for bound names as usual, but also for free names. Configurations that differ only on the names of the free variables are equivalent, since they represent the same net.

Computation is performed by rewriting configurations using the following rewrite system, where if r is a rule, $\widehat{r}$ denotes a fresh generic *instance* of r, that is, a copy of r where we introduce a new set of names:

Indirection: If $x \in \mathcal{N}(u)$, then $x = t, u = v \longrightarrow u[t/x] = v$.
Interaction: If $r \in \mathcal{R}$ and $\widehat{r} = \alpha(t'_1, \ldots, t'_n) \bowtie \beta(u'_1, \ldots, u'_m)$, then

$$\begin{aligned} &\alpha(t_1, \ldots, t_n) = \beta(u_1, \ldots, u_m) \longrightarrow \\ &\qquad t_1 = t'_1, \ldots, t_n = t'_n, u_1 = u'_1, \ldots, u_m = u'_m \end{aligned}$$

Context: If $\Delta \longrightarrow \Delta'$, then $\langle \boldsymbol{t} \mid \Gamma, \Delta, \Gamma' \rangle \longrightarrow \langle \boldsymbol{t} \mid \Gamma, \Delta', \Gamma' \rangle$.
Collect: If $x \in \mathcal{N}(\boldsymbol{t})$, then $\langle \boldsymbol{t} \mid x = u, \Delta \rangle \longrightarrow \langle \boldsymbol{t}[u/x] \mid \Delta \rangle$.

This rewrite system generates an *equational theory*, the corresponding equivalence relation is denoted by $c \leftrightarrow^* c'$. The reduction relation $\longrightarrow$ is strongly confluent [3] since there is one rule for each pair of agents. Various strategies of evaluation are defined in [3]. The *values* that we use in this paper, called *interface normal forms*, have terms rooted by agents in the head whenever this is possible.

Definition 2 (Interface Normal Form). *A configuration $(\mathcal{R}, \langle \boldsymbol{t} \mid \Delta \rangle)$ is in* interface normal form *(INF) if each t_i in $\boldsymbol{t}$ is of one of the following* canonical forms*:*

- *$\alpha(\boldsymbol{s})$. E.g. $\langle S(x) \mid x = Z, \Delta \rangle$.*
- *x where either $x \in \mathcal{N}(t_j)$ for some $j \neq i$, or $x \in \mathcal{N}(u)$ for some $y = u \in \Delta$ such that $y \in N$ is free (x is in an open path). E.g. $\langle x, x \mid \Delta \rangle$*
- *x where $x \in \mathcal{N}(u)$ for some $y = u \in \Delta$ such that $y \in \mathcal{N}(u)$ (x occurs in a cycle). E.g. $\langle x \mid y = \alpha(\beta(y), x), \Delta \rangle$.*

We denote by INF_i the set of configurations where the ith port in the head is canonical.

Computing interface normal forms suggests that we do the minimum work required to bring principal ports to the interface. This strategy is defined by the inference rules:
Axiom:

$$\frac{c \in \mathit{INF}}{c \Downarrow c}$$

Collect:

$$\frac{\langle t_1, \ldots, t, \ldots, t_n \mid \Delta\rangle \Downarrow c}{\langle t_1, \ldots, x, \ldots, t_n \mid x = t, \Delta\rangle \Downarrow c}$$

Indirection: if $x \in \mathcal{N}(u)$ and $y \in \mathcal{N}(t, u = v)$

$$\frac{\langle t_1, \ldots, y, \ldots, t_n \mid u[t/x] = v, \Delta\rangle \Downarrow c}{\langle t_1, \ldots, y, \ldots, t_n \mid x = t, u = v, \Delta\rangle \Downarrow c}$$

Interaction: if $x \in \mathcal{N}(\alpha(\boldsymbol{t}) = \beta(\boldsymbol{u}))$, $r \in \mathcal{R}$, $\widehat{r} = \alpha(\boldsymbol{t'}) \bowtie \beta(\boldsymbol{u'})$

$$\frac{\langle s_1, \ldots, x, \ldots, s_n \mid \overrightarrow{t = t'}, \overrightarrow{u = u'}, \Delta\rangle \Downarrow c}{\langle s_1, \ldots, x, \ldots, s_n \mid \alpha(\boldsymbol{t}) = \beta(\boldsymbol{u}), \Delta\rangle \Downarrow c}$$

This system is deterministic [3]. If $c \Downarrow v$ can be derived with these rules we say that v is the interface normal form of c. We write $c \Downarrow_i v$ (i.e. the position i in the head of v is canonical) if the rules Indirection and Interaction are only applied at position i in the head of the configuration and the axiom is replaced by

$$\frac{c \in \mathit{INF}_i}{c \Downarrow_i c}$$

Example 1 (Combinators). The interaction combinators [8] are a universal system of interaction built from the 0-ary agent ϵ and the binary agents δ and γ, with the rules:

$$\begin{array}{ccccccc}
\delta(x,y) & \bowtie & \delta(x,y) & \quad & \delta(\epsilon,\epsilon) & \bowtie & \epsilon \\
\gamma(x,y) & \bowtie & \gamma(y,x) & & \gamma(\epsilon,\epsilon) & \bowtie & \epsilon \\
\epsilon & \bowtie & \epsilon & & \delta(\gamma(a,b),\gamma(c,d)) & \bowtie & \gamma(\delta(a,c),\delta(b,d))
\end{array}$$

The configuration $\langle x \mid \gamma(y,x) = \delta(\epsilon,y)\rangle$ gives a non-terminating sequence of reductions, but has an interface normal form: $\langle \delta(a,b) \mid \epsilon = \gamma(c,a), \delta(c,d) = \gamma(d,b)\rangle$.

3 Bisimilarity

In functional languages, we consider two functions equivalent when we can apply them to the same arguments and obtain the same results. In other words, we perform some form of experiment on the objects under test, and compare the results. For interaction nets, we take this general idea as inspiration. The way that we can make experiments with a net is to interact with it on a free principal port. Connecting nets on free principal ports is our analogue of applying a function to an argument. After evaluation, we can observe whether some principal ports are at the interface, which is analogous to observing whether a λ-term has evaluated to an abstraction. Only the *observable ports* (in the head of the configuration) are available for the experiments. Configurations with the same number of terms in the head will be said *comparable*.

Two configurations that cannot be distinguished by any experiment will be called *bisimilar*. We will show that the bisimilarity relation can be defined as the greatest postfixpoint of an operator (which allows us to use coinductive techniques to prove that two configurations are bisimilar), and more important, it coincides with the contextual equivalence, that is, two bisimilar configurations cannot be distinguished by any context and can therefore be exchanged without altering the behaviour of the system. We begin with some basic definitions to formalise these ideas.

Definition 3 (Visible Interface). *A configuration* $c = \langle t_1, \ldots, t_n \mid \Delta \rangle \in \mathit{INF}_i$ has a visible interface at position i *if either* t_i *is not a variable or there is an open path starting at* t_i *and finishing at some* $t_j = \alpha'(\boldsymbol{u}) \in \boldsymbol{t}$*, that is,* $t_i = \alpha(\boldsymbol{u})$ *or* $t_i = x$ *and there is an open path to* $t_j = \alpha'(\boldsymbol{u})$*. The* visible agent *at position* i *is* α *in the first case,* α' *in the second case. The rest of the net is called the* kernel*:* $\mathcal{K}_i(c) = \langle t_1, \ldots, t_{k-1}, \boldsymbol{u}, t_{k+1}, \ldots, t_n \mid \Delta \rangle$ *where* $k = i$ *or* $k = j$ *depending on whether we are in the first or second case. The set of* new observable positions *in* $\mathcal{K}_i(c)$*, denoted* $NP_{\mathcal{K}}(c, i)$*, is the set of positions of the terms* $\boldsymbol{u}$ *if* $k = i$*, otherwise it contains just the new position of* t_i*. We denote by* $\mathcal{V}_i$ *the set of all the configurations with a visible interface at position* i*.*

If v *and* v' *are comparable and have the same visible agent at position* i*, we write* $\mathit{SVA}_i(v, v')$*. If the visible agents are different, but they are not both constructors, we write* $\neg \mathit{Constr}_i(v, v')$*.*

Example 2. Let $c = \langle I(x), x \mid \rangle$. Since $t_1 = I(x)$ is not a variable, $c \in \mathcal{V}_1$. Since $t_2 = x$ and there is an open path to $t_1 = I(x)$, $c \in \mathcal{V}_2$. The visible agent is I for both positions, and $\mathcal{K}_1(c) = \mathcal{K}_2(c) = \langle x, x \mid \rangle$.

When we have different agents in the visible interfaces of the nets under test, and they are not constructors, we need to see if these agents behave in the same way for each possible agent interacting with them. For this we use closings.

Definition 4 (Closing). *A closing at position* i *of a configuration* $c = \langle \boldsymbol{t} \mid \Delta \rangle \in \mathcal{V}_i$*, denoted by* $cl_i(c)$*, is obtained from* c *by one of the following operations, where* $k = i$*, or* $k = j$ *if there is an open path starting at position* i *and finishing at position* j *in* c*:*

1. *replace* $t_k \equiv \alpha(\boldsymbol{s})$ *in* $\boldsymbol{t}$*, by a list of new variables* $z_1, \ldots, z_p \in \mathcal{N}(\boldsymbol{u})$*,* $p \geq 0$*, and add to* Δ *the equation* $t_k = \alpha'(\boldsymbol{u})$*, where* α' *is any agent, and the terms in* $\boldsymbol{u}$ *are either new variables (in which case they can appear twice in* $\alpha'(\boldsymbol{u})$ *or once in* $\alpha'(\boldsymbol{u})$ *and once in* $\boldsymbol{z}$*) or elements of* $\boldsymbol{t}$*, in which case they are erased from* $\boldsymbol{t}$*.*
   The set $NP_{cl}(c, i)$ *of* new observable positions *in* $cl_i(c)$ *contains the positions of the variables* $z_1, \ldots, z_p$ *in the new head if* $i = k$*, otherwise it contains just the new position of* t_i*.*
2. *erase* $t_k \equiv \alpha(\boldsymbol{s})$ *and another term* t_p *in* $\boldsymbol{t}$ *and add* $t_k = t_p$ *to* Δ*. In this case* $NP_{cl}(c, i) = \emptyset$ *if* $i = k$*, otherwise it contains just the new position of* t_i*.*

By abuse of notation, we will denote by $cl_i(c')$ *the result of applying to a configuration* c' *comparable with* c *the operations that define a closing at position* i *for* c*.*

Graphically, the first operation corresponds to connecting the principal port of an agent α' to the kth observable port in the interface of the net, and connecting some auxiliary ports of α' between them (if a variable appears twice in $\boldsymbol{u}$), or to other observable ports in the net (if $\boldsymbol{u}$ contains terms in $\boldsymbol{t}$). The second operation corresponds to simply adding a wire connecting the observable ports k and p.

We consider a complete lattice $(\mathsf{Rel}, \subseteq)$ where Rel is the set of binary relations between pairs $(c, i), (c', i)$ such that c, c' are comparable configurations whose heads have at least i elements (i.e. we can talk of the ith observable port). The operators $\langle \mathcal{R} \rangle$ and $[\mathcal{R}]$ for $\mathcal{R} \in \mathsf{Rel}$ will be used to define *similarity* and *bisimilarity* respectively.

Definition 5 (Operators). *Let c, c' be comparable configurations with at least i terms in the head.*

$$
\begin{array}{lll}
(c,i)\,\langle \mathcal{R} \rangle\,(c',i) & \stackrel{\text{def}}{\Longleftrightarrow} & c \Downarrow_i v \in \mathcal{V}_i \Rightarrow \exists v',\ (c' \Downarrow_i v' \textit{ and} \\
& & \textit{either } SVA_i(v,v') \textit{ and } \forall p \in NP_{\mathcal{K}}(v,i), (\mathcal{K}_i(v),p)\ \mathcal{R}\ (\mathcal{K}_i(v'),p) \\
& & \textit{or } \neg \textit{Constr}_i(v,v'), \forall cl_i(v), \forall p \in NP_{cl}(v,i), (cl_i(v),p)\ \mathcal{R}\ (cl_i(v'),p)) \\
(c,i)[\mathcal{R}](c',i) & \stackrel{\text{def}}{\Longleftrightarrow} & (c,i)\,\langle \mathcal{R} \rangle\,(c',i) \textit{ and } (c',i)\,\langle \mathcal{R} \rangle\,(c,i)
\end{array}
$$

Property 1. $\langle \cdot \rangle$, $[\cdot]$ are monotone operators.

Definition 6 (Similarity, Bisimilarity).

- *A relation $\mathcal{S} \in \mathsf{Rel}$ such that $\mathcal{S} \subseteq \langle \mathcal{S} \rangle$ (i.e. $\mathcal{S}$ is a post-fixpoint of $\langle \cdot \rangle$) is a* simulation. *The greatest such $\mathcal{S}$ is called a* similarity, *and written as $\precsim$. If c, c' are comparable configurations with n elements in the head, then $c \precsim c'$ if $(c,i) \precsim (c',i)$, $1 \leq i \leq n$.*
- *A relation $\mathcal{B} \in \mathsf{Rel}$ such that $\mathcal{B} \subseteq [\mathcal{B}]$ (i.e. $\mathcal{B}$ is a post-fixpoint of $[\cdot]$) is a* bisimulation. *The greatest such $\mathcal{B}$ is called a* bisimilarity, *and written as $\simeq$. If c, c' are comparable configurations with n elements in the head, then $c \simeq c'$ if $(c,i) \simeq (c',i)$, $1 \leq i \leq n$.*

Note that $\langle \cdot \rangle$ and $[\cdot]$ posses a greatest post-fixpoint by the Tarski-Knaster Fixed Point Theorem. Moreover, $\precsim$ and $\simeq$ are fixed points, i.e. $\precsim = \langle \precsim \rangle$ and $\simeq = [\simeq]$.

Remark 1. The main difference with the typed approach resides in the definition of closings and the way they are used in the definition of the operators $\langle \cdot \rangle$ and $[\cdot]$. Here closings are applied "on demand" whereas they are a static notion in the typed framework. More precisely, in a typed net a closing is built just by connecting agents to *all* the free *input* ports. The Subject Reduction property ensures that reduction will not create new free input ports. Instead, here we close one principal port at a time, and since reduction might create a new free principal port, closings are applied in a dynamic way.

The relations $\precsim$, $\simeq$ can be defined by levels, as done by Abramsky for the untyped λ-calculus [1].

Proposition 1 (Coinduction Principle). *Let c, c' be comparable configurations with n observable ports. To prove $c \simeq c'$ it suffices to find a bisimulation $\mathcal{B}$ such that $(c,i)\mathcal{B}(c',i)$ for $1 \leq i \leq n$.*

By coinduction we can show that the equational theory is included in the bisimilarity relation. In Section 4 we give more examples of application of coinduction to prove bisimilarity, in particular we will show that this inclusion is strict.

Theorem 1 (Bisimilarity includes the equational theory). $c \leftrightarrow^* c' \Rightarrow c \simeq c'$.

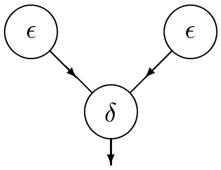
4 Examples

The Identity agent and a wire. Let I be the identity agent defined by rules

$$I(\alpha(x_1,\ldots,x_n)) \bowtie \alpha(I(x_1),\ldots,I(x_n))$$

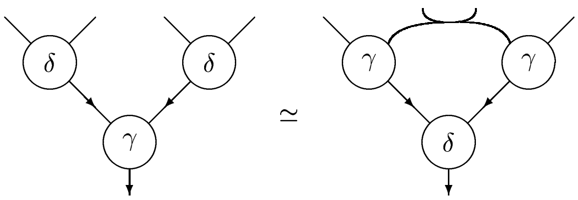
for any $\alpha \in \Sigma$. We can prove $\langle I(x), x \mid \rangle \simeq \langle x, x \mid \rangle$ by coinduction. Take a symmetric R containing the pairs $((c,i),(c',i))$ such that c' is obtained from c by erasing the I agents at the root of a term in the head, or at the root of a member of an equation. We show that R is a bisimulation: if $c \Downarrow_i v \in \mathcal{V}_i$, then $c' \Downarrow_i v'$, and either they have the same visible agent α at position i, in which case the kernels are in the relation for all the new observable positions, or if they differ, then one is rooted by I and the other is just a variable. In that case the closings are in the relation, which is sufficient since I is not a constructor.

Copying before erasing or just erasing. In the system of the interaction combinators, replacing a net of the form



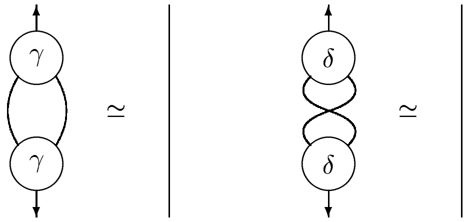
by the agent ϵ seems an intuitive optimization. We can prove that they are bisimilar by coinduction. The Main Theorem 2 tells us then that these configurations are contextually equivalent— the optimization is correct.

Agents γ and δ. The following nets are bisimilar:



To show it using the coinduction principle we consider a relation containing $\simeq$ and these pairs, for any closing of the free principal port. The interesting closings are built by adding an agent ϵ, γ, or δ (the closings using just wires do not reduce to a value with a visible interface). The case of ϵ is trivial. For the other cases, by reducing to interface normal form we obtain configurations that have the same visible agents and whose kernels are easily shown to be bisimilar, hence contained in our relation.

η-rules for γ and δ. The following nets are bisimilar:



Note that these last two equivalences are neither included in the equational theory (since the nets are different normal forms) nor provable using the path semantics developed by Lafont [8].

5 Contextual Equivalence

We define a set of operations that build a context for a configuration, in the same way that closings were defined by operations. But there are more operations in the case of contexts, and we can have a sequence of operations instead of just one operation.

Definition 7 (Context). *A context at position i for a configuration $c = \langle \boldsymbol{t} \mid \Delta \rangle$ is defined by a (possibly empty) sequence of operations, where $k = i$, or $k = j$ if there is an open path starting at position i and finishing at position j in c. Non-empty sequences (i.e. contexts) are defined inductively, there are three cases according to the first operation used.*

1. *Addition of agent by principal port: This operation replaces t_k in $\boldsymbol{t}$ by a list of new variables $z_1, \dots, z_p \in \mathcal{N}(\boldsymbol{u})$, and adds to Δ the equation $t_k = \alpha(\boldsymbol{u})$, where α is any agent, and the terms in $\boldsymbol{u}$ are either new variables (in which case they can occur twice in $\alpha(\boldsymbol{u})$ or once in $\alpha(\boldsymbol{u})$ and once in $\boldsymbol{z}$) or elements of $\boldsymbol{t}$, in which case they are erased from $\boldsymbol{t}$.*
   In this case the rest of the sequence is the concatenation of contexts at the positions of the variables $z_1, \dots, z_p$ in the new head and at the new position of t_i if $i \neq k$.
2. *Addition of agent by auxiliary port: This operation replaces t_k in $\boldsymbol{t}$ by a list of new variables $z_1, \dots, z_p$ occurring free in $y = \alpha(\boldsymbol{u})$ and adds this equation to Δ, where α is any agent, and the terms in $\boldsymbol{u}$ are either new variables (in which case they can occur twice in $\alpha(\boldsymbol{u})$ or once in $\alpha(\boldsymbol{u})$ and once in $\boldsymbol{z}$) or elements of $\boldsymbol{t}$, in which case they are erased from $\boldsymbol{t}$. The term t_k must occur in $\boldsymbol{u}$.*
   Also in this case the rest of the sequence is composed of contexts at the positions of the variables $z_1, \dots, z_p$ in the new head and at the new position of t_i if $i \neq k$.
3. *Addition of a wire: erase t_k and another term t_p in $\boldsymbol{t}$ and add $t_k = t_p$ to Δ. In this case the rest of the sequence is empty if $i = k$, otherwise it is a context at the new position of t_i.*

We denote by $op_{i,\boldsymbol{j}}(c)$ the result of applying an operation as above to the configuration c at position i, using the positions $\boldsymbol{j}$ in $\boldsymbol{t}$. We denote by $C_i[c]$ the configuration resulting of applying the context C, defined by a sequence of operations as above, to the configuration c at position i, and by $C(c, i)$ a generic context for c at position i. We will also denote by $C_i[c']$ the result of applying to a configuration c' comparable with c the operations that define a context at position i for c.

The set $NP_C(c, i)$ of new observable positions *of $C_i[c]$ is computed as follows: we start with the set $\{i\}$, and compute a new set each time we perform an operation. The first and second operations add the positions of the variables $z_1, \dots, z_p$ in the new head, and if $i = k$ they erase i, otherwise they replace i by the new position of t_i in the head. The third operation simply erases the position i from the set if $i = k$, otherwise replaces i by the new position of t_i.*

Graphically, the first two operations correspond to connecting an agent α to an observable port of the net (using the principal port of α in the first one, and an auxiliary port in the second one). The third operation corresponds to adding a wire connecting the observable ports k and p. Closings are particular cases of contexts defined by one operation of the first or third class.

Definition 8 (Contextual Preorder and Contextual Equivalence). *Let c, c' be comparable configurations with n elements in the head.*

$$\begin{array}{rcl}
c \;\leq\; c' & \stackrel{\text{def}}{\Longleftrightarrow} & \forall i \in [1 \ldots n], (c,i) \leq (c',i) \\
(c,i) \leq (c',i) & \stackrel{\text{def}}{\Longleftrightarrow} & \forall C(c,i), \forall p \in NP_C(c,i), C_i[c] \Downarrow_p v \in \mathcal{V}_p \Rightarrow \exists v',\ (C_i[c'] \Downarrow_p v' \\
& & \quad \textit{and either } \mathit{SVA}_p(v,v') \textit{ or } \neg \mathit{Constr}_p(v,v')) \\
(c,i) = (c',i) & \stackrel{\text{def}}{\Longleftrightarrow} & (c,i) \leq (c',i) \textit{ and } (c',i) \leq (c,i) \\
c \;=\; c' & \stackrel{\text{def}}{\Longleftrightarrow} & \forall i \in [1 \ldots n], (c,i) = (c',i)
\end{array}$$

6 Main Result

We will show that the notions of contextual equivalence and bisimilarity coincide, if the interaction net system has "enough contexts" to extract the kernels of all values.

Definition 9. *A system of interaction is* complete *if for any $v \in \mathcal{V}_i$ with visible agent α at position i, there exists a context C^α such that $\forall p \in NP_{\mathcal{K}}(v,i), (\mathcal{K}_i(v),p) \simeq (C_i^\alpha[v],p)$ and $p \in NP_{C^\alpha}(v,i)$.*

Theorem 2. *If the interaction net system is complete, $\leq$ (resp. $=$) coincides with $\precsim$ (resp. $\simeq$). Otherwise $\precsim$ (resp. $\simeq$) is included in $\leq$ (resp. $=$).*

Proof. To prove $\precsim\ \subseteq\ \leq$ it is sufficient to show that $\precsim$ is preserved by context. Following Howe [5] we prove that $\precsim$ is a precongruence (a preorder preserved by context) using an auxiliary relation $\precsim^*$, the *precongruence candidate*, defined as follows.

Let c, c' be comparable configurations with n elements in their heads.

$$\begin{array}{rcll}
c \precsim^* c' & \stackrel{\text{def}}{\Longleftrightarrow} & \forall i \in [1 \ldots n],\ (c,i) \precsim^* (c',i) & \\
(c,i) \precsim^* (c',i) & \stackrel{\text{def}}{\Longleftrightarrow} & \text{either} & (c,i) \precsim (c',i), \\
& & \text{or} & c = op_{p,\boldsymbol{j}}(d), i \in NP_{op}(d,p), \\
& & & (d,q) \precsim^* (d',q), \forall q \in \boldsymbol{j} \text{ and} \\
& & & (op_{p,\boldsymbol{j}}(d'),i) \precsim (c',i).
\end{array}$$

The precongruence candidate enjoys the following properties.

Property 2. 1. $\precsim\ \subseteq\ \precsim^*$.
2. $\precsim^*$ is reflexive.
3. $c \precsim^* c', c' \precsim c'' \Rightarrow c \precsim^* c'$
4. $\precsim^*$ is preserved by context: $c \precsim^* c' \Rightarrow \forall i, \forall C(c,i), C_i[c] \precsim^* C_i[c']$.

To show that $\precsim$ is a precongruence it is sufficient to prove that it coincides with $\precsim^*$, for which it remains to prove $\precsim^*\ \subseteq\ \precsim$. This follows, by coinduction, from:

Proposition 2. *1.* $v \in \mathcal{V}_i, (v,i) \precsim^* (c',i) \Rightarrow (v,i)\ \langle\precsim^*\rangle\ (c',i)$
2. $c \precsim^* c', c \Downarrow_i v \in \mathcal{V}_i \Rightarrow v \precsim^* c'$

This concludes the proof of the first inclusion: $\precsim\ \subseteq\ \leq$. Now we prove $\leq\subseteq\ \precsim$ by coinduction, showing $\leq\ \subseteq\ \langle\leq\rangle$. Assume $(c,i) \leq (c',i)$. By definition of $\leq$, using an empty context, $c \Downarrow_i v \in \mathcal{V}_i \Rightarrow \exists v',\ c' \Downarrow_i v'$ and either $\mathit{SVA}_i(v,v')$ or $\neg \mathit{Constr}_i(v,v')$. In the latter case we are done, since closings are particular cases of contexts. In the first case, we know by completeness that $(\mathcal{K}_i(v),p) \simeq (C_i^{\alpha}[v],p), \forall p \in NP_{\mathcal{K}}(v,i)$. Moreover, since bisimilarity includes the equational theory (Theorem 1), and $(c,i) \leq (c',i)$: $(C_i^{\alpha}[v],p) \simeq (C_i^{\alpha}[c],p) \leq (C_i^{\alpha}[c'],p) \simeq (C_i^{\alpha}[v'],p)$. Again by completeness (since $\mathit{SVA}_i(v,v')$), $(C_i^{\alpha}[v'],p) \simeq (\mathcal{K}_i(v'),p)$. Since we have already proved $\simeq\ \subseteq =$, we get $(\mathcal{K}_i(v),p) \leq (\mathcal{K}_i(v'),p), \forall p \in NP_{\mathcal{K}}(v,i)$ as required. □

7 Conclusion

In this paper we have presented a notion of bisimilarity for (untyped) interaction nets. This notion has been shown to coincide with the contextual equivalence, thus we have a simple proof technique for showing when two nets are equivalent in all contexts. One of the main applications that we see for this work are general correctness proofs for optimizations in interaction net implementations of various systems, such as the λ-calculus or term rewriting systems.

References

1. Samson Abramsky. The lazy λ-calculus. In David A. Turner, editor, *Research Topics in Functional Programming*, chapter 4, pages 65–117. Addison Wesley, 1990.
2. Maribel Fernández and Ian Mackie. Coinductive techniques for operational equivalence of interaction nets. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS'98)*, pages 321–332. IEEE Computer Society Press, June 1998.
3. Maribel Fernández and Ian Mackie. A calculus for interaction nets. In *Proceedings of the first International Conference on Principles and Practice of Declarative Programming (PPDP'99)*, Lecture Notes in Computer Science, Springer-Verlag, September 1999.
4. Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In *Proceedings of the 19th ACM Symposium on Principles of Programming Languages (POPL'92)*, pages 15–26. ACM Press, January 1992.
5. Douglas J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.
6. Lionel Khalil. Mémoire de DEA SPP, 1999. Available at http://www.dmi.ens.fr/∼khalil.
7. Yves Lafont. Interaction nets. In *Proceedings of the 17th ACM Symposium on Principles of Programming Languages (POPL'90)*, pages 95–108. ACM Press, January 1990.
8. Yves Lafont. Interaction combinators. *Information and Computation*, 137(1):69–101, 1997.
9. Ian Mackie. YALE: Yet another lambda evaluator based on interaction nets. In *Proceedings of the 3rd ACM SIGPLAN International Conference on Functional Programming (ICFP'98)*, pages 117–128. ACM Press, September 1998.
10. Andrew M. Pitts. Operationally-based theories of program equivalence. In P. Dybjer and A. M. Pitts, editors, *Semantics and Logics of Computation*, Publications of the Newton Institute, pages 241–298. Cambridge University Press, 1997.