

Linking Interaction Nets and Post Canonical Systems

Bas van Gijzel
b.m.vangijzel@uu.nl

June 20, 2010

Abstract

This paper is meant as a technical contribution by showing the relations between Post canonical systems, Post normal systems and tag systems. An emulation of 2-tag systems in interaction nets is given followed by directions to a possible proof for the reduction of interaction nets to another model of computation.

Introduction

First I will give an introduction to Post canonical systems (PCS), Post normal systems (PNS) and tag systems in Section 1. This is followed by an introduction to interaction nets in Section 2. I will provide an emulation of tag systems in interaction nets in Section 3, and finally give directions of proof for the reduction of interaction nets to another model of computation in Section 4.

1 Post Canonical Systems

A Post canonical system, as defined by Emil Post [12], is a form of logical system, that given a set of axioms or starting *assertions*, using a set of *productions*, can arrive at new formulas of the system, called *enunciations*.

Definition 1.1. A Post canonical system is a model computation consisting of three elements, a finite set of letters or alphabet Σ , a finite set of primitive assertions, \mathcal{I} , consisting of sequences made out of Σ and a set of production rules, \mathcal{R} , of specific form that transform enunciations of the logical system.

Definition 1.2 (Production rules). A production rule can be applied when the premise(s) match, resulting into adding the conclusion as an enunciation to the logical system. In a canonical system, productions can be of the following form:

$$\begin{aligned}
& g_{11}P_{i_1'}g_{12}P_{i_2'} \dots g_{1m_1}P_{i_{m_1}'}g_{1(m_1+1)} \\
& g_{21}P_{i_1''}g_{22}P_{i_2''} \dots g_{2m_2}P_{i_{m_2}''}g_{2(m_2+1)} \\
& \quad \vdots \\
& g_{k1}P_{i_1^{(k)}}g_{k2}P_{i_2^{(k)}} \dots g_{km_k}P_{i_{m_k}^{(k)}}g_{k(m_k+1)} \\
& \Rightarrow \\
& g_1P_{i_1}g_2P_{i_2} \dots g_mP_{i_m}g_{m+1}
\end{aligned}$$

where g_{ki} are fixed sequences on the alphabet and P_{ki} are variables standing for words. So for this production rule given the k premises: $g_{11} \dots g_{1(m_1+1)}$ until $g_{k1} \dots g_{k(m_k+1)}$, we can conclude $g_1P_{i_1}g_2P_{i_2} \dots g_mP_{i_m}g_{m+1}$.

Example 1.3. Given the PCS with $\Sigma = \{a, b\}$, $\mathcal{I} = \{a\}$, $\mathcal{R} = \{P_1, a \Rightarrow_1 aP_1b; P_2aP_3 \Rightarrow_2 P_2P_3\}$ we try to derive the enunciation ab .

We start by applying the first reduction rule to the primitive assertion a :

$$a \Rightarrow_1 aab \quad (1)$$

Thus by (1), aab is now added to the set of enunciations, thereby making that word applicable to further production rules. With one more application of a production rule we arrive at the desired enunciation:

$$aab \Rightarrow_2 ab \quad (2)$$

1.1 Normal Form

A Post normal system is a special case of the Post canonical system, with certain restrictions.

Definition 1.4. A PCS is termed to be in normal form when it has but one primitive assertion in the set \mathcal{I} and production rules are of a restricted form, namely:

$$\begin{aligned}
& gP \\
& \Rightarrow \\
& Pg'
\end{aligned}$$

where g is again a fixed sequence of the alphabet and P is a variable standing for a word.

Maybe surprisingly, the two forms are *equipotent* as was proved by Post [12].

Theorem 1.5 (Normal-form theorem). *Given any Post canonical system with alphabet Σ , a system in normal form with a possibly enlarged alphabet can be set up such that the enunciations of the system in canonical form are exactly those assertions of the system in normal which involve only letters from Σ .*

1.2 Tag System

Tag systems are a computational model related to a special form of a PNS called *monogenic normal systems* [12, 1].

Definition 1.6 (Monogenic normal system). A normal system is *monogenic* if the g 's of the premisses of the productions form a set g_1, g_2, \dots, g_k such that each string can be written in the form $g_i P$ for at most one i .

Definition 1.7 (Tag system). A tag system is a monogenic normal system in which the g 's of the premisses of the productions constitute all sequences of some fixed length l , while the corresponding h 's of the conclusions are identical for all g 's having the same initial symbol.

Definition 1.8 (m -tag system). An m -tag system is a tag system with positive integer m , for which the fixed length $l = m$.

Example 1.9. Given the following 2-tag system, adapted from De Mol [3], with $\Sigma = \{a, b, c\}$, $\mathcal{I} = \{aaaa\}$, $\mathcal{P} = \{a \Rightarrow_1 bc; b \Rightarrow_2 a; c \Rightarrow_3 aaa\}$. This 2-tag system can calculate the Collatz sequence for a certain number n by using a^n as the starting word, \mathcal{I} . So iff calculation of the initial word halts in this 2-tag system, the Collatz sequence should halt for that number.

Keep in mind that a 2-tag system consumes two symbols every time a rule is applied, so a formulation in terms of a monogenic normal system would actually make the production rules $\mathcal{P} = \{aa \Rightarrow_1 bc; ab \Rightarrow_1 bc; \dots; cc \Rightarrow_3 aaa\}$.

$$\begin{array}{ll}
 aaaa \Rightarrow_1 abc & \text{(Computation of 4)} \\
 abc \Rightarrow_1 bc bc & \\
 bc bc \Rightarrow_2 bca & \\
 bca \Rightarrow_2 aa & \text{(Computation of 2)} \\
 aa \Rightarrow_1 bc & \\
 bc \Rightarrow_1 a & \text{(Computation of 1)}
 \end{array}$$

Post conjectured that monogenic normal systems, similarly to PNS, were universal. This was later directly proven by Arbib [1]. Minsky [11] proved a stronger result, namely that tag systems in general were universal and this was later strengthened by the result of Wang [13] and by Cocke and Minsky [2] to apply to 2-tag systems.

Theorem 1.10. *2-tag systems in general are universal.*

Proof. 2-tag systems simulating universal Turing machines and a universal Turing machine simulating 2-tag systems in general can be constructed. \square

The following is entailed immediately because 2-tag systems are generalised by m -tag systems, which are again generalised by monogenic normal systems.

Corollary 1.11. *Tag systems in general are universal.*

Corollary 1.12. *Monogenic normal systems in general are universal.*

2 Interaction Nets

Interaction nets are a model of computation by Yves Lafont [7, 8] that generalise the proof nets as devised by Jean-Yves Girard [6]. An interaction net uses building blocks similar to logical ports and is a model of computation based on a parallel form of reduction.

Definition 2.1. An interaction net can be constructed using a finite set of symbols Σ and a finite set of interaction rules \mathcal{R} . Every symbol $\sigma \in \Sigma$ has an associated fixed *arity* represented by a natural, provided by the function $\mathbf{ar} : \Sigma \rightarrow \mathit{Nat}$.

Definition 2.2 (Cells). Occurrences of symbols from Σ are called *cells*. Cells have one *principal port*, depicted by the sharp point, and a fixed number of *auxiliary ports* given by the arity function, \mathbf{ar} . Auxiliary ports of a cell are numbered clockwise starting from the principal port so the orientation of a cell does not matter.

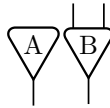


Figure 1: Two cells with arity 0 and 2 respectively

Definition 2.3 (Nets). A *net* N is represented by a graph with a finite number of *cells*, a number of *free ports* for each cell and edges or *wires* connecting the *free ports* of cells. Connections of wires are pairwise on ports, where the ports of a cell can have at most one connection.

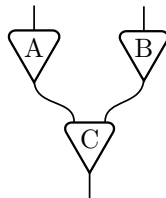


Figure 2: An example net

Definition 2.4 (Interaction rule). Two cells connected by their principal ports is called an *active pair*. Reductions on an interaction net are done by applying interaction rules from \mathcal{R} , where a reduction is only possible on an active pair. Reductions applied in this manner can be done sequentially or in parallel; a net that has no applicable interaction rules will be called *irreducible*.

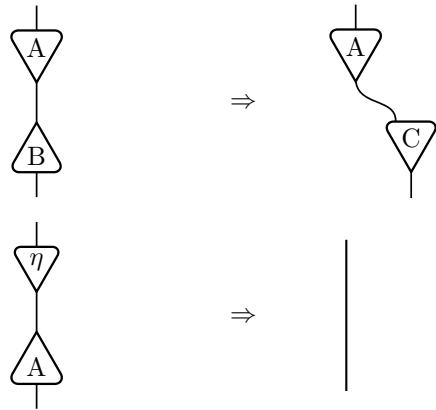


Figure 3: Two example interaction rules

For the interaction rules I will now introduce an interaction rule schema which will be used in the proof later.

Definition 2.5 (Interaction schema). An interaction schema has the form an interaction rule, where one cell has the symbol $*$, thus excluding $*$ to be in the alphabet Σ of the corresponding interaction net. Interaction rules instantiated by this schema can be obtained by substituting the $*$ for any σ from the alphabet Σ .

Example 2.6. Below is an example of a simple interaction schema representing a turning operator, η , where after turning the other cell the turning operator is deleted.

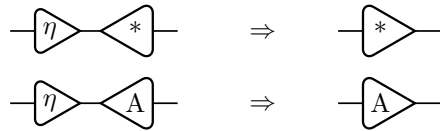


Figure 4: Interaction rule schema and one instantiation

3 2-Tag Systems to Interaction Nets

We here show that interaction nets are at least as powerful as 2-tag systems by showing that any 2-tag system can be transformed into an interaction net that halts when the 2-tag system halts. Even more strongly, the created interaction net emulates the 2-tag system, taking two steps equivalent to any step taken by the 2-tag system.

The transformed interaction nets needs an alphabet with only one more symbol, $|\Sigma|$ more interaction rules than production rules and an interaction net as large as the primitive assertion.

Outline of the proof. The alphabet stays the same, while the primitive assertion or starting word of a 2-tag system is modelled as a sequence of nodes corresponding to the symbols. The symbols are placed such that the nodes that are being reduced correspond to the two symbols reduced by the 2-tag

system. One extra step is needed in the interaction system to set up the next two symbols to be reduced. Interaction rules correspond almost directly to the production rules only adding the turning operator η to end of the generated nodes.

A primitive assertion, $\mathcal{I} = \{abcd a\}$, will thus be transformed to:

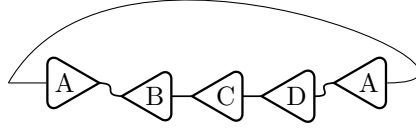


Figure 5: Interaction net for the primitive assertion \mathcal{I}

Intermediate, two step, states of the interaction net can be linked to the 2-tag system by reading the nodes' symbols starting from the active pair and continuing clockwise.

Theorem 3.1. *Every 2-tag system can be emulated by an interaction net linearly.*

Proof. Given an arbitrary 2-tag system, t , with alphabet Σ_t , production rules \mathcal{R}_t , and primitive assertion \mathcal{I}_t we need to construct an interaction net, i , with finite set symbols Σ_i , interaction rules \mathcal{R}_i and starting net \mathcal{I}_s that emulates t .

For the finite set of symbols, Σ_i , we simply take the alphabet of the tag system $\Sigma_t \cup \{\eta\}$ (if η was in Σ_i we use a different symbol in the interaction net). The starting net will be constructed on basis of the starting primitive assertion. A 2-tag system with a primitive assertion \mathcal{I}_t of length < 2 would halt immediately, thus making the transformation trivial, so we assume a length ≥ 2 . The starting net \mathcal{I}_i is then constructed as follows:

- All symbols in \mathcal{I}_t are constructed as a cell in \mathcal{I}_i with their respective symbol one on one.
- The first symbol's corresponding cell is placed with its principal port to the right. All following cells are placed to the right corresponding to the sequence and with their principal port to the left.
- All cells have an arity of 1, thus having one principal and one auxiliary port.
- All neighbouring cells are connected by their closest port. The first and last cell are connected by their remaining ports.

The interaction rules \mathcal{R}_i are then constructed by the production rules \mathcal{R}_t . For every production rule $r \in \mathcal{R}_t$ an interaction rule is constructed as follows:

- The active pair in the production rule is equivalent to the premise in \mathcal{R}_t .
- The right hand side of the production rule for \mathcal{R}_i is the sequence of cells corresponding to the produced symbols used in \mathcal{R}_t with at the end the node η .

- All nodes on the right hand side except the η node have their principal port to the left.

Additionally the interaction rules corresponding to the interaction schema for the turning operator η , as introduced in Example 2.6, are added to \mathcal{R}_i .

We can now already see that every reduction in t is emulated by a reduction in i operating on the corresponding nodes, followed by the reduction of the turning operator η , thus having i take two steps for every step in t . So any tag-system can be emulated linearly by an interaction net. \square

We will now apply the transformation, as done in the proof, to the 2-tag system simulating the Collatz sequence, as was introduced in Example 1.9.

Example 3.2. This 2-tag system, t , has $\Sigma_t = \{a, b, c, d\}$, $\mathcal{I}_t = \{aaaa\}$ and $\mathcal{P}_t = \{aa \Rightarrow_1 bc; ab \Rightarrow_1 bc; \dots; cc \Rightarrow_3 aaa\}$.

We will now construct an interaction net i that will emulate t . The alphabet of i , $\Sigma_i = \Sigma_t \cup \{\eta\}$ and the primitive assertion \mathcal{I}_i will be the net \mathcal{I}_i below:

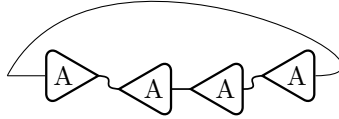


Figure 6: Interaction net for the primitive assertion \mathcal{I}_t

The interaction rules \mathcal{R}_i will contain the interaction rules as defined by the interaction schema for the turning operator from Example 2.6 and also the interaction rules corresponding to the transformation rules. The interaction rule that simulates $aa \Rightarrow_1 bc$ can be seen below.



Figure 7: Interaction rule for $aa \Rightarrow_1 bc$

The reader may want to verify the reduction done in the interaction net after applying this interaction rule.

4 Reduction of Interaction Nets

This section gives an introduction to interaction combinators; interaction nets composed of only three types of nodes which already make up a universal system. This is followed by a discussion of a non-graphical representation of interaction nets, namely an interaction calculus [10, 9, 4]. These two formalisms suggest possible starting directions of a proof for the reduction of interaction nets to another model of computation, such as tag systems.

4.1 Interaction Combinators

A simpler representation of interaction nets in general is the representation by interaction combinators. Interaction nets constructed by three combinators are also universal, as was shown by Lafont [7].

Theorem 4.1. *Any interaction system can be translated into the system of interaction combinators.*

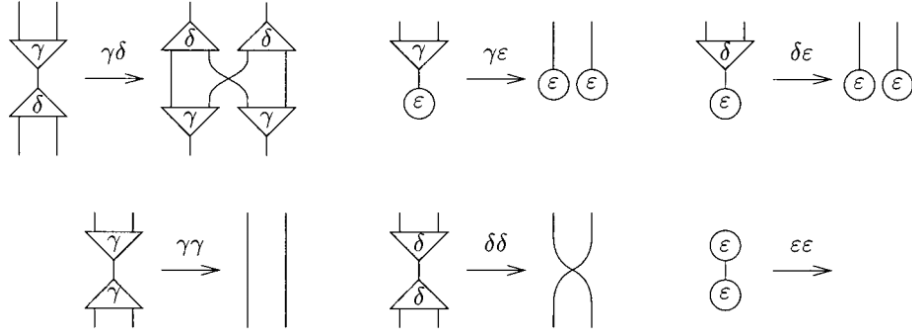


Figure 8: Interaction rules for the combinators as by Lafont [7]

Here the γ combinator is a *constructor*, the ε combinator an *eraser* and the δ a *duplicator*. Note that the δ combinator is the combinator switching ports, not the other way around (this can be seen when considering the numbering of ports).

To represent interaction net by interaction combinators would simplify a possible reduction by only needing to reduce a predefined alphabet and set of interaction rules.

4.2 Interaction Calculus

The interaction calculus as described by Mackie [10] is a non-graphical representation of interaction nets. The textual calculus, which tries to stay close to the graphical representation, captures the formalism of interaction nets by terms such as $add(T, Z) \perp S(Z)$; (here \perp depicts a wire between two principal ports of a node). The interaction calculus is defined by first reducing interaction nets to a *configuration*, restricting nets to be deadlock free. These configurations are then reduced to an interaction calculus and proved to be equivalent to the graphical representation.

Mackie [10] reduces this interaction calculus to a term rewriting system, linear chemical abstract machine (CHAM). This might also be a viable step to be taken should a proof of reduction to a Post canonical system or alike be attempted.

Conclusion

This paper demonstrates that interaction nets can emulate 2-tag systems in a linear fashion, taking only two steps for every step taken in that tag system. The ease of emulation shows the power of interaction nets as a model of computation. That interesting result raises the question if it is feasible to emulate interaction nets by other models of computations, such as Post canonical systems. Starting directions for a reduction/emulation proof for the other direction have been given, by means of a discussion of interaction combinators and interaction calculi, but the possibility still remains unclear.

References

- [1] Michael A. Arbib. Monogenic normal systems are universal. *Journal of the Australian Mathematical Society*, 3(03):301–306, 1963.
- [2] John Cocke and Marvin Minsky. Universality of tag systems with $p = 2$. *J. ACM*, 11(1):15–20, 1964.
- [3] Liesbeth De Mol. Tag systems and collatz-like functions. *Theor. Comput. Sci.*, 390(1):92–101, 2008.
- [4] Maribel Fernández and Ian Mackie. A calculus for interaction nets. In *PPDP '99: Proceedings of the International Conference PPDP'99 on Principles and Practice of Declarative Programming*, pages 170–187, London, UK, 1999. Springer-Verlag.
- [5] Maribel Fernández. *Models of Computation: An Introduction to Computability Theory*. Springer, 2009.
- [6] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.
- [7] Yves Lafont. Interaction combinators. *Information and Computation*, 137:69–101, 1995.
- [8] Yves Lafont. Introduction to interaction nets. 1998.
- [9] Ian Mackie. Towards a programming language for interaction nets. *Electron. Notes Theor. Comput. Sci.*, 127(5):133–151, 2005.
- [10] Ian Mackie and Shinya Sato. A calculus for interaction nets based on the linear chemical abstract machine. *Electron. Notes Theor. Comput. Sci.*, 192(3):59–70, 2008.
- [11] Marvin L. Minsky. Recursive unsolvability of post's problem of 'tag' and other topics in theory of turing machines. *Annal. Math*, 74:437–455, 1961.
- [12] Emil Leon Post. Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics*, 65:197–215, 1943.
- [13] Hao Wang. Tag systems and lag systems. *Mathematische Annalen*, 152:65–74, 1963.