A Semiring-based Quantitative Analysis of Mobile Systems

Benjamin Aziz [1],

Department of Computing
Imperial College London
180 Queens Gate
London SW7 2AZ, UK

Abstract

We present in this paper, semi-π, an extension of the π-calculus that allows processes to query quantitative values of different actions and decide based on those values, whether an action is feasible or not. Our measure of quantity is based on the general notion of semirings. Furthermore, we develop a syntax-directed static analysis for the new language, which captures the properties of name substitution and semiring value retrieval. Such properties allow us to solve quantitative constraints controlling synchronisations in the analysed systems. We provide an example of a cost analysis of communications in a simple adaptive routing algorithm.

Key words: Static Analysis, Quantitative Resources, Semirings, Mobile Systems

1 Introduction

In this paper, we present a syntax-directed static analysis of an extension of the π-calculus [14] with semiring value retrieval capabilities and semiring constraints. The new language, termed *semi-π*, allows processes to query the cost of communicating actions given certain information. Based on that cost, a process can then decide through the use of semiring constraints, the kind of behaviour it should perform next. For example, in the process,

$$\textbf{let } \omega_1 = \mathcal{S}\textbf{value}(\overline{x}\langle y\rangle, t) \textbf{ in let } \omega_2 = \mathcal{S}\textbf{value}(\overline{u}\langle y\rangle, t) \textbf{ in } [\omega_1 \succcurlyeq \omega_2]\ \overline{x}\langle y\rangle \mid [\omega_2 \succcurlyeq \omega_1]\ \overline{u}\langle y\rangle$$

[1] Email: baziz@doc.ic.ac.uk

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

Then after retrieving the cost of actions, $\overline{x}\langle y\rangle$ and $\overline{u}\langle y\rangle$ at time, t, using the **let** $= \mathcal{S}$**value**() **in** and placing the results in ω_1, ω_2, we compare the two costs using some ordering relation, $\succcurlyeq$, and choose either output.

The static analysis then captures two properties of the analysed systems: The first is name substitutions, which occur as a result of synchronising actions, the second is instantiations of actions' costs, given certain context information. Using these properties, it is possible to quantify name substitutions based on the cost of actions that contributed to those substitutions. Such properties have interesting applications in the analysis of adaptive network routing, where a router calculates the output port of each packet based on the cost of the different paths of its neighbours.

Our measure of cost is based on the notion of *semirings*. A semiring is a tuple, $(\mathcal{A}, +, \times, 1, 0)$, such that $\mathcal{A}$ is a set and $0, 1 \in \mathcal{A}$. Furthermore, $+$ is the additive operation, which is commutative and associative, with 0 as its unit element, and $\times$ is the multiplicative operation, which is associative, with 1 as its unit element and 0 as its absorbing element. A semiring has the property that $\times$ distributes over $+$. Moreover, following [7], it is possible to define an ordering relation based on $+$:

$$s \succcurlyeq s' \quad \Leftrightarrow \quad s + s' = s$$

where we sometimes write $s \succ s'$ to mean $s \succcurlyeq s'$ and $s \neq s'$. This measure of cost is general: there exist many instantiations of $(\mathcal{A}, +, \times, 1, 0)$ in literature that are Boolean, probabilistic, fuzzy, set-theoretic, weighted etc.

The rest of the paper is organised as follows. In Section 2, we discuss some of the related work in the area of quantitative analysis of mobile systems. In Section 3, we introduce the syntax of the semi-π language. In Section 4, we define a domain-theoretic model as the basis of the standard semantics of semi-π. In Section 5, we extend this semantics to capture name substitutions and instantiations of semiring variables. In Section 6, we abstract the non-standard semantics in order to obtain a computable static analysis. In Section 7, we demonstrate how the results of the abstract semantics yield a solution to a constraint satisfaction problem involving the relationship between name substitutions and costs of actions. In Section 8, we apply the analysis to a simple example of an adaptive router. Finally, in Section 9, we conclude the paper and discuss future work.

2 Related Work

The work presented in this paper is a continuation of previous analyses [2,3,5,6], which were designed to detect security properties based on the name substitution property in mobile and cryptographic/PKI systems modelled in different versions of the π-calculus (e.g. the spi calculus [1] and Spiky [12]). The main novelty of the current work is that the analysis deals with quantitative prop-

erties (e.g. cost of name substitutions) rather than qualitative properties (e.g. security properties).

Other works in the area of quantitative analysis of mobile systems include [8,9,13,17], in the area of the quantitative analysis of process algebraic models have been carried out, using different approaches. Most of such works differ from our approach in that a process has no "quantitative control" over its future execution, as is the case with semi-π. Other quantitative analyses exist [15,16] but these are usually designed for simpler, KLAIM-like, languages without name-passing.

3 Semi-π

The syntax of the semi-π language is defined in Figure 1. In this syntax, names

$P, Q, R ::=$	processes
$\pi.P$	guards
$P \mid Q$	composition
$(\nu n)P$	restriction
$!P$	replication
$[x = y]\ P$	match
$[e_1 \succcurlyeq e_2]\ P$	semiring constraint
$\mathbf{0}$	null
$\mathbf{let}\ \omega = \mathcal{S}\mathbf{value}(\pi, n)\ \mathbf{in}\ P$	semiring value retrieval
$\pi ::=$	guards
$\overline{x}\langle y \rangle$	output
$x(y)$	input
$e ::=$	expressions
ω	semiring variable
s	semiring element
$e_1 + e_2$	additive operation
$e_1 \times e_2$	multiplicative operation
$E, F ::=$	systems
(θ, P)	state-process pair

Fig. 1. The syntax of the semi-π language

constitute the infinite set, $a, b, c, n, m, x, y, z \ldots \in \mathcal{N}$. Processes, $P, Q, R \in \mathcal{P}$, are defined as follows. A guarded process, $\pi.P$, proceeds as P once it fires π,

where π is either input, $x(y)$ or output, $\overline{x}\langle y\rangle$. In case of $x(y)$, the message received over x will replace y in P. Parallel composition, $P \mid Q$ runs P and Q by interleaving them. Restriction, $(\nu n)P$, creates a fresh name, n, within the scope of P. Replication, $!P$, spawns as many copies of P as required by the context. Matching, $[x = y]\ P$, proceeds as P if x is the same as y, else it blocks. Similarly, the semiring constraint, $[e_1 \succcurlyeq e_2]\ P$, proceeds as P if $e_1 \succcurlyeq e_2$, otherwise, it blocks. Expressions, e_1, e_2, are defined using semiring variables, $\omega \in \Omega$, elements $s \in \mathcal{A}$, applications of the additive operator, $e_1 + e_2$, and applications of the multiplicative operator, $e_1 \times e_2$. The null process, $\mathbf{0}$, is a process incapable of evolving any further. We usually omit trailing null residues. Finally, **let** $\omega = \mathcal{S}\mathbf{value}(\pi, n)$ **in** P retrieves the semiring value of an action, π, given some name, n, and instantiates ω with that value in P. Intuitively, n acts as a context data that may be needed in order to give more specific information about the action π (e.g. time of the action, relative position of the action, the name of the user performing the action etc.).

Based on this definition of processes, we define systems, $E, F \in \mathcal{E}$, as pairs, (θ, P), where $\theta(\pi, n) = s_{\pi,n} \in \mathcal{A}$ is a state environment mapping every communication action, π, and its context data, $n \in \mathcal{N}$, to some corresponding semiring value, $s_{\pi,n} \in \mathcal{A}$. The value, $s_{\pi,n}$, in effect represents a *quantification of action π in light of the information given by n.*

The standard notions of α-conversion as well as the free names, $fn(P)$, and bound names, $bn(P)$, of a process, P, are defined as usual, where y is bound to P in $(\nu y)P$ and $x(y).P$. A name is free if it is not bound. Furthermore, we refer to $bsemiv(P)$ as the set of bound semiring variables of P, i.e. ω in the **let** $\omega = \mathcal{S}\mathbf{value}(\pi, n)$ **in** P process, and $fsemiv(P)$ as the set of free semiring variables, i.e. variables occurring in expressions but not in $bsemiv(P)$. We usually write $semiv(P) = fsemiv(P) \cup bsemiv(P)$. From now on, we only deal with systems that have a *normal process*.

Definition 3.1 A process, P, is said to be *normal* if the following holds:

- There are no occurrences of homonymous bound names or homonymous semiring variables in P.
- $bn(P) \cap fn(P) = \{\}$.
- $fsemiv(P) = \{\}$ (no open expressions).

4 A Domain-Theoretic Model

The standard semantics of semi-π is defined in the domain-theoretic style of [4,5,18]. Assuming that $Pi_\perp$ is the semantic domain of processes, N is the predomain of names and $\mathcal{K}$ is the set underlying any (pre)domain, then concrete elements of $Pi_\perp$ and N can be defined as in Figure 2.

The definition of N is trivial: N is a flat predomain[2]. Therefore, it's

[2] A domain with no bottom where every two non-identifiable elements are non-comparable.

Elements of N :		
$x \in \mathcal{N}$	$\Rightarrow$	$x \in \mathcal{K}(N)$
Elements of $Pi_\perp$:		
$\{\!\lvert \perp \rvert\!\} \in \mathcal{K}(Pi_\perp)$		
$\emptyset \in \mathcal{K}(Pi_\perp)$		
$p, q \in \mathcal{K}(Pi_\perp)$	$\Rightarrow$	$p \uplus q \in \mathcal{K}(Pi_\perp)$
$x \in \mathcal{K}(N), p \in \mathcal{K}(Pi_\perp)$	$\Rightarrow$	$new(x, p) \in \mathcal{K}(Pi_\perp)$
$p \in \mathcal{K}(Pi_\perp)$	$\Rightarrow$	$\{\!\lvert tau(p) \rvert\!\} \in \mathcal{K}(Pi_\perp)$
$x, y \in \mathcal{K}(N), p \in \mathcal{K}(Pi_\perp)$	$\Rightarrow$	$\{\!\lvert in(x, \lambda y.p) \rvert\!\} \in \mathcal{K}(Pi_\perp)$
$x, y \in \mathcal{K}(N), p \in \mathcal{K}(Pi_\perp)$	$\Rightarrow$	$\{\!\lvert out(x, y, p) \rvert\!\} \in \mathcal{K}(Pi_\perp)$
$x, y \in \mathcal{K}(N), p \in \mathcal{K}(Pi_\perp)$	$\Rightarrow$	$\{\!\lvert out(x, \lambda y.p) \rvert\!\} \in \mathcal{K}(Pi_\perp)$

Fig. 2. Elements of $Pi_\perp$ and N.

structure is quite similar to $\mathcal{N}$. On the other hand, $Pi_\perp$ is defined as a multiset of semantic elements, where $\{\!\lvert \perp \rvert\!\}$ is the bottom element representing the undefined process and $\emptyset$ is the empty multiset representing terminated or deadlocked processes[3]. Other elements are defined as follows: $p \uplus q$ is the standard multiset union of two elements, p, q. The singleton map, $\{\!\lvert\ \rvert\!\}$, takes tuples representing input, output and silent actions and creates a singleton multiset of each tuple. These tuples are $in(x, \lambda y.p)$ (input action), $out(x, y, p)$ (free output action), $out(x, \lambda y.p)$ (bound output action) and $tau(p)$ (silent action). In these tuples, x is the channel of communication, y is the message or input parameter and p is the residue process. The use of λ-abstraction to model the binding effect in input and bound output actions implies that these actions have a meaning as a function, which when instantiated with a name, the actual residue is yielded. Finally, the effects of restriction are modelled by the *new* operator defined over elements $p \in Pi_\perp$ as in Figure 3.

In general, *new* captures deadlocked situations arising from the attempt to communicate over restricted non-extruded channels. It also turns a free output into a bound output once a restricted message is directly sent over a channel (scope extrusion). In all other cases, restriction has no effect and it is simply passed to the residue or distributed over multiset union.

Using elements $p \in Pi_\perp$, it is possible to denote the meaning of systems in semi-π, with the semantic function, $\mathcal{S}(\llbracket(\theta, P)\rrbracket)\ \rho\ \phi_\mathcal{S}\ \delta_\mathcal{S} \in Pi_\perp$, defined by induction over the structure of P as in Figure 4. Since θ remains constant throughout the interpretation, it is sufficient to interpret a system in terms of elements of $Pi_\perp$. This semantics introduces the following environments.

- The multiset, ρ, containing all processes composed in parallel with the analysed process, paired with a copy of θ. The standard singleton, $\{\!\lvert\ \rvert\!\}$, and

[3] $\{\!\lvert \perp \rvert\!\} \sqsubseteq \emptyset$ and $\emptyset$ is incomparable otherwise.

$$
\begin{array}{lll}
new(x, \emptyset) & = \emptyset & \\
new(x, \{\!|\bot|\!\}) & = \{\!|\bot|\!\} & \\
new(x, \{\!|in(y, \lambda z.p)|\!\}) & = \begin{cases} \emptyset, & \text{if } x = y \\ \{\!|in(y, \lambda z.new(x, p))|\!\}, & \text{otherwise} \end{cases} \\
new(x, \{\!|out(y, z, p)|\!\}) & = \begin{cases} \emptyset, & \text{if } x = y \\ \{\!|out(y, \lambda z.p)|\!\}, & \text{if } x = z \neq y \\ \{\!|out(y, z, new(x, p))|\!\}, & \text{otherwise} \end{cases} \\
new(x, \{\!|out(y, \lambda z.p)|\!\}) & = \begin{cases} \emptyset, & \text{if } x = y \\ \{\!|out(y, \lambda z.new(x, p))|\!\}, & \text{otherwise} \end{cases} \\
new(x, \{\!|tau(p)|\!\}) & = \{\!|tau(new(x, p))|\!\} & \\
new(x, (p_1 \uplus p_2)) & = new(x, p_1) \ \uplus \ new(x, p_2) &
\end{array}
$$

Fig. 3. The definition of *new*.

multiset union, $\uplus$, operations are overloaded to deal with elements of ρ.

- The special environment, $\phi_{\mathcal{S}} : \mathcal{N} \rightarrow \mathcal{N}$, maps a name to another name that substitutes it in the semantics. Initially, $\forall n \in \mathcal{N} : \phi_{\mathcal{S}0}(n) = n$. In fact, this environment will hold substitutions of input parameters by messages received during communications. Since the standard denotational semantics is a precise semantics, an input parameter can only be mapped to, at most, one name in any possible choice of control flow (i.e. on either side of $\uplus$).
- The special environment, $\delta_{\mathcal{S}} : \Omega \rightarrow \mathcal{A}_{\bot}$, which maps semiring variables to semiring values. Initially, $\forall \omega \in \Omega : \delta_{\mathcal{S}0}(\omega) = \bot$ for any variable, ω.

The meaning of the composed systems in ρ is given by rule ($\mathcal{R}0$) as the summation of the individual meaning of each system. Rule ($\mathcal{S}1$) interprets the meaning of a null process directly as the empty multiset, $\emptyset$. Rules ($\mathcal{S}2$) and ($\mathcal{S}3$) deal with the cases of processes guarded by input and output actions after which the residues are composed with elements of ρ. Any communications between matching input/output actions are dealt with in rule ($\mathcal{S}3$), where $\phi_{\mathcal{S}}$ is updated accordingly. The interpretation is a summation of all such communications and the no-communication case. This preserves the associativity property of the parallel composition operator, $P \mid Q$. Rule ($\mathcal{S}4$) interprets conditional statements based on matching the $\phi_{\mathcal{S}}$-values of two names. Rule ($\mathcal{S}5$) interprets a semiring constraint based on the ordering relation, $\succcurlyeq$, after closing two expressions under the $\delta_{\mathcal{S}}$ environment. Rule ($\mathcal{S}6$) is straightforward allowing for two parallel processes to be composed with the rest of processes in ρ, where θ is distributed over the two processes. Rule ($\mathcal{S}7$) interprets a restriction using the *new* operation, defined earlier in Figure 3. In rule ($\mathcal{S}8$), the meaning of semiring retrieval is given by updating $\delta_{\mathcal{S}}$ with the semiring value of an action, π, given a name, n. Finally, rule ($\mathcal{S}9$) defines the least fixed point meaning of a replicated process, $!P$, as the least upper bound of the poset, $\mathcal{F}$, which contains the bottom element, $\{\!|\bot|\!\}$, and elements denoting any number

$(\mathcal{S}1)\ \ \mathcal{S}(\llbracket(\theta, \mathbf{0})\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} \quad = \emptyset$

$(\mathcal{S}2)\ \ \mathcal{S}(\llbracket(\theta, x(y).P)\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} = \{\!|in(\phi_{\mathcal{S}}(x), \lambda y.\mathcal{R}(\llbracket\{\!|(\theta, P)|\!\} \uplus \rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}})|\!\}$

$(\mathcal{S}3)\ \ \mathcal{S}(\llbracket(\theta, \overline{x}\langle y\rangle.P)\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} =$

$$\Big(\biguplus_{(\theta, x'(z).P')\in\rho} \{\!|tau(p)|\!\}\Big) \uplus \{\!|out(\phi_{\mathcal{S}}(x), \phi_{\mathcal{S}}(y), \mathcal{R}(\llbracket\{\!|(\theta, P)|\!\} \uplus \rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}})|\!\}$$

where, $p = \mathcal{R}(\llbracket\{\!|(\theta, P)|\!\} \uplus \rho[(\theta, P')/(\theta, x'(z).P')]\rrbracket)\ \phi_{\mathcal{S}}[z \mapsto \phi_{\mathcal{S}}(y)]\ \delta_{\mathcal{S}}$

and, $\phi_{\mathcal{S}}(x) = \phi_{\mathcal{S}}(x')$

$$(\mathcal{S}4)\ \ \mathcal{S}(\llbracket(\theta, [x\ =\ y]\ P)\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} = \begin{cases} \mathcal{R}(\llbracket\{\!|(\theta, P)|\!\} \uplus \rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}}, & \text{if } \phi_{\mathcal{S}}(x) = \phi_{\mathcal{S}}(y) \\ \mathcal{R}(\llbracket\rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}}, & \text{otherwise} \end{cases}$$

$(\mathcal{S}5)\ \ \mathcal{S}(\llbracket(\theta, [e_1 \succcurlyeq e_2]\ P)\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} =$

$$\begin{cases} \mathcal{R}(\llbracket\{\!|(\theta, P)|\!\} \uplus \rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}}, \text{ if } e_1[\delta_{\mathcal{S}}(\omega_1)/\omega_1]_{\omega_1\in semiv(e_1)} \succcurlyeq e_2[\delta_{\mathcal{S}}(\omega_2)/\omega_2]_{\omega_2\in semiv(e_2)} \\ \mathcal{R}(\llbracket\rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}}, \text{ otherwise} \end{cases}$$

$(\mathcal{S}6)\ \ \mathcal{S}(\llbracket(\theta, P \mid Q)\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} \ = \mathcal{R}(\llbracket\{\!|(\theta, P)|\!\} \uplus \{\!|(\theta, Q)|\!\} \uplus \rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}}$

$(\mathcal{S}7)\ \ \mathcal{S}(\llbracket(\theta, (\nu x)P)\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} \ = new(x, \mathcal{R}(\llbracket\{\!|(\theta, P)|\!\} \uplus \rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}})$

$(\mathcal{S}8)\ \ \mathcal{S}(\llbracket(\theta, \mathbf{let}\ \omega = \mathcal{S}\mathbf{value}(\pi, n)\ \mathbf{in}\ P)\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} = \mathcal{R}(\llbracket\{\!|(\theta, P)|\!\} \uplus \rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}}[\omega \mapsto \theta(\pi, n)]$

$(\mathcal{S}9)\ \ \mathcal{S}(\llbracket(\theta, !P)\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} \quad = \bigsqcup \mathcal{F}$

where, $\mathcal{F} = \{\{\!|\bot|\!\}, \mathcal{R}(\llbracket\ (\biguplus_{i=0\ldots\infty} \{\!|(\theta, P[bn_i(P)/bn(P)][semiv_i(P)/semiv(P)])|\!\})\ \uplus\ \rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}}\}$

and, $bn_i(P) = \{x_i \mid x \in bn(P)\}, semiv_i(P) = \{\omega_i \mid \omega \in semiv(P)\}$

$(\mathcal{R}0)\ \ \mathcal{R}(\llbracket\rho\rrbracket)\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} \quad = \biguplus_{(\theta,P)\in\rho} \mathcal{S}(\llbracket(\theta, P)\rrbracket)\ (\rho\backslash\{\!|(\theta, P)|\!\})\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}}$

Fig. 4. The standard semantics of semi-π.

(up to infinity) of parallel compositions of the replicated process, P. Since the semantic domain, $Pi_{\perp}$, is infinite, $\mathcal{F}$ may contain infinite number of elements, therefore, its least upper bound may not be computable within finite limits. A labelling mechanism is also used in the rule to perform α-conversion on the spawned copies, P, by subscripting all the bound names and semiring variables of P. The renaming is necessary to maintain the normality of processes.

5 Non-standard Semantics

Our main interest is to capture instantiations of input parameters and semiring variables. Since the standard semantics of the previous section does not provide this information, we need to extend it. For this purpose, we introduce the two special environments, $\phi_{\mathcal{E}} : N \to \wp(N)$ and $\delta_{\mathcal{E}} : \Omega \to \wp(\mathcal{A})$, mapping a name to a set of names and a semiring variable to a set of semiring values, respectively. Both of these mappings represent possible instantiations that may occur during runtime. Initially, we have that $\forall n \in \mathcal{N} : \phi_{\mathcal{E}0}(n) = \{n\}$ and $\forall \omega \in \Omega : \delta_{\mathcal{E}0}(\omega) = \{\}$. Since there are no homonymous occurrences of

bound names and semiring variables (Definition 3.1), then the sets $\phi_{\mathcal{E}}(y)$ and $\delta_{\mathcal{S}}(\omega)$ will be at most singletons per choice of control flow for any name, y, and semiring variable, ω.

Using these environments, we can define the non-standard semantic domain, $D_\perp = Pi_\perp \times (N \to \wp(N)) \times (\Omega \to \wp(\mathcal{A}))_\perp$, with the following ordering:

$$\begin{array}{l}\forall (p_1, \phi_{\mathcal{E}1}, \delta_{\mathcal{E}1}), (p_2, \phi_{\mathcal{E}2}, \delta_{\mathcal{E}2}) \in D_\perp : \\ (p_1, \phi_{\mathcal{E}1}, \delta_{\mathcal{E}1}) \sqsubseteq_{D_\perp} (p_2, \phi_{\mathcal{E}2}, \delta_{\mathcal{E}2}) \quad \Leftrightarrow \quad p_1 \sqsubseteq p_2 \ \wedge \ \phi_{\mathcal{E}1} \subseteq \phi_{\mathcal{E}2} \ \wedge \ \delta_{\mathcal{E}1} \subseteq \delta_{\mathcal{E}2}\end{array}$$

where the bottom element is $\perp_{D_\perp} = (\{\!|\perp|\!\}, \phi_{\mathcal{E}0}, \delta_{\mathcal{E}0})$. We also define the unions of $\phi_{\mathcal{E}}$ and $\delta_{\mathcal{E}}$ as follows:

$$\begin{array}{rl}(\phi_{\mathcal{E}1} \cup_\phi \phi_{\mathcal{E}2})(x) = & \phi_{\mathcal{E}1}(x) \cup \phi_{\mathcal{E}2}(x) \\ (\delta_{\mathcal{E}1} \cup_\delta \delta_{\mathcal{E}2})(\omega) = & \delta_{\mathcal{E}1}(\omega) \cup \delta_{\mathcal{E}2}(\omega)\end{array}$$

Now, we can define the non-standard semantics of semi-π using the function, $\mathcal{E}(\![(\theta, P)]\!)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} \in D_\perp$, as shown in Figure 5. The semantics utilises a multiset, ρ, to hold all processes in parallel with the analysed process, sharing the same copy of θ. The contents of ρ are interpreted in rule ($\mathcal{R}0$), which uses $\uplus$ to group the choice of standard elements, $p \in Pi_\perp$, and operations, $\cup_\phi$ and $\cup_\delta$, to group the choice of non-standard elements, $\phi_{\mathcal{E}}$ and $\delta_{\mathcal{E}}$, respectively. The rest of the rules ($\mathcal{E}1$)–($\mathcal{E}9$) deal with the individual cases of P. There are a few interesting points to note. Name and semiring value instantiations, ($\phi'_{\mathcal{E}}$,$\delta'_{\mathcal{E}}$), resulting in the residue of an input action are neglected in rule ($\mathcal{E}2$). Such instantiations are instead considered in rule ($\mathcal{E}3$) during communications between matching input and output actions. In this rule, similar instantiations, ($\phi''_{\mathcal{E}}$,$\delta''_{\mathcal{E}}$), occurring under unfired output actions are also neglected.

In rule ($\mathcal{E}4$), the matching process is resolved based on whether there exist $\phi_{\mathcal{E}}$-values of the matched names that are equal, or not. Since the semantics is precise, these values can only be singleton sets per choice of control flow. A similar argument is true in rule ($\mathcal{E}5$), where either expression may assume only a single value when closing it under $\delta_{\mathcal{E}}$. Hence, the semiring constraint is either satisfied, or not. In this rule, we define the following operation:

$$fold\ f\ e\ \{x_1, \ldots, x_n\} = f(x_n, \ldots, f(x_1, e)\ldots)$$

applied to function, $Y_{\delta_{\mathcal{E}}}$, which instantiates each variable in a set of expressions to generate a new set of children expressions, as follows:

$$Y_{\delta_{\mathcal{E}}}(\omega, \{e_1, \ldots, e_n\}) = (\bigcup_{s \in \delta_{\mathcal{E}}(\omega)} \{e_1[s/\omega]\}) \ \cup \ \ldots \ \cup \ (\bigcup_{s \in \delta_{\mathcal{E}}(\omega)} \{e_n[s/\omega]\})$$

The rule for replication, ($\mathcal{E}9$), defines the least fixed point meaning of a replicated process as the least upper bound of the (possibly infinite) poset, $\mathcal{F}$. The computation of this least fixed point may not terminate within finite limits due to the infinite size of $D_\perp$. Also, any spawned copies of $!P$ are α-

$$
\begin{array}{lll}
(\mathcal{E}1) & \mathcal{E}(\llbracket(\theta,\mathbf{0})\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} & = (\emptyset,\phi_{\mathcal{E}},\delta_{\mathcal{E}}) \\
(\mathcal{E}2) & \mathcal{E}(\llbracket(\theta,x(y).P)\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} & = (\{\!|in(x',\lambda y.p')|\!\},\phi_{\mathcal{E}},\delta_{\mathcal{E}}) \\
 & & \quad \text{where, } (p',\phi'_{\mathcal{E}},\delta_{\mathcal{E}}{}') = \mathcal{R}(\llbracket\{\!|(\theta,P)|\!\}\uplus\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} \\
 & & \quad \text{and, } \phi_{\mathcal{E}}(x) = \{x'\} \\
(\mathcal{E}3) & \mathcal{E}(\llbracket(\theta,\overline{x}\langle y\rangle.P)\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} & = ((\biguplus_{(\theta,x'(z).P')\in\rho}\{\!|tau(p')|\!\})\ \uplus\ \{\!|out(x'',y'',p'')|\!\}, \\
 & & \qquad (\bigcup^{\phi}_{(\theta,x'(z).P')\in\rho}\phi'_{\mathcal{E}})\ \cup_{\phi}\ \phi_{\mathcal{E}}, (\bigcup^{\delta}_{(\theta,x'(z).P')\in\rho}\delta'_{\mathcal{E}})\ \cup_{\delta}\ \delta_{\mathcal{E}}) \\
\multicolumn{3}{l}{\quad \text{where, } (p',\phi'_{\mathcal{E}},\delta'_{\mathcal{E}}) = \mathcal{R}(\llbracket\{\!|(\theta,P)|\!\}\uplus\rho[(\theta,P')/(\theta,x'(z).P')]\rrbracket)\ \phi_{\mathcal{E}}[z\mapsto\ \{y\}]\ \delta_{\mathcal{E}},} \\
\multicolumn{3}{l}{\qquad (p'',\phi''_{\mathcal{E}},\delta''_{\mathcal{E}}) = \mathcal{R}(\llbracket\{\!|(\theta,P)|\!\}\uplus\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}}} \\
\multicolumn{3}{l}{\quad \text{and, } \phi_{\mathcal{E}}(x) = \phi_{\mathcal{E}}(x') = \{x''\}, \phi_{\mathcal{E}}(y) = \{y''\}} \\
(\mathcal{E}4) & \mathcal{E}(\llbracket(\theta,[x\ =\ y]\ P)\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} & = \begin{cases} \mathcal{R}(\llbracket\{\!|(\theta,P)|\!\}\uplus\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}}, \text{ if } \exists z\in\phi_{\mathcal{E}}(x), z'\in\phi_{\mathcal{E}}(y): \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad z=z' \\ \mathcal{R}(\llbracket\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}}, \text{ otherwise} \end{cases} \\
(\mathcal{E}5) & \mathcal{E}(\llbracket(\theta,[e_1\succcurlyeq e_2]\ P)\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} = & \\
\multicolumn{3}{l}{\quad \begin{cases} \mathcal{R}(\llbracket\{\!|(\theta,P)|\!\}\uplus\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}}, \text{ if } \exists e\in(\mathit{fold}\ Y_{\delta_{\mathcal{E}}}\ \{e_1\}\ \mathit{semiv}(e_1)), \\ \qquad\qquad\qquad\qquad e'\in(\mathit{fold}\ Y_{\delta_{\mathcal{E}}}\ \{e_2\}\ \mathit{semiv}(e_2)) : e\succcurlyeq e' \\ \mathcal{R}(\llbracket\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}}, \text{ otherwise} \end{cases}} \\
(\mathcal{E}6) & \mathcal{E}(\llbracket(\theta,P\mid Q)\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} & = \mathcal{R}(\llbracket\{\!|(\theta,P)|\!\}\uplus\{\!|(\theta,Q)|\!\}\uplus\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} \\
(\mathcal{E}7) & \mathcal{E}(\llbracket(\theta,(\nu x)P)\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} & = (\mathit{new}(x,p'),\phi'_{\mathcal{E}},\delta_{\mathcal{E}}{}') \\
 & & \quad \text{where, } (p',\phi'_{\mathcal{E}},\delta_{\mathcal{E}}{}') = \mathcal{R}(\llbracket\{\!|(\theta,P)|\!\}\uplus\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} \\
(\mathcal{E}8) & \mathcal{E}(\llbracket(\theta,\mathbf{let}\ \omega=\mathcal{S}\mathbf{value}(\pi,n)\ \mathbf{in}\ P)\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} & = \mathcal{R}(\llbracket\{\!|(\theta,P)|\!\}\uplus\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}}[\omega\mapsto\{\theta(\pi,n)\}] \\
(\mathcal{E}9) & \mathcal{E}(\llbracket(\theta,!P)\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} & = \bigsqcup\mathcal{F} \\
\multicolumn{3}{l}{\quad \text{where, } \mathcal{F} = \{(\{\!|\bot|\!\},\phi_{\mathcal{E}0},\delta_{\mathcal{E}0}),} \\
\multicolumn{3}{l}{\qquad\qquad \mathcal{R}(\llbracket\ (\biguplus_{i=0\ldots\infty}\{\!|(\theta,P[bn_i(P)/bn(P)][\mathit{semiv}_i(P)/\mathit{semiv}(P)])|\!\})\ \uplus\ \rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}}\}} \\
\multicolumn{3}{l}{\quad \text{and, } bn_i(P) = \{x_i\mid x\in bn(P)\}, \mathit{semiv}_i(P) = \{\omega_i\mid\omega\in\mathit{semiv}(P)\}} \\
(\mathcal{R}0) & \mathcal{R}(\llbracket\rho\rrbracket)\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} & = (\biguplus_{(\theta,P)\in\rho}p',\ \bigcup^{\phi}_{(\theta,P)\in\rho}\phi'_{\mathcal{E}},\ \bigcup^{\delta}_{(\theta,P)\in\rho}\delta'_{\mathcal{E}}) \\
 & & \quad \text{where, } (p',\phi'_{\mathcal{E}},\delta'_{\mathcal{E}}) = \mathcal{E}(\llbracket(\theta,P)\rrbracket)\ (\rho\backslash\{\!|(\theta,P)|\!\})\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}}
\end{array}
$$

Fig. 5. The non-standard semantics of semi-π.

converted using the labelling mechanism, introduced in the previous section to maintain the process normality requirement.

The following theorem states that the non-standard semantics of semi-π is *correct* with respect to the standard semantics.

Theorem 5.1 (Correctness of the Non-Standard Semantics)
$\forall P,\theta,\rho,\phi_{\mathcal{S}},\phi_{\mathcal{E}},\delta_{\mathcal{S}},\delta_{\mathcal{E}} : (\mathcal{S}(\llbracket(\theta,P)\rrbracket)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}} = p)\ \wedge\ (\mathcal{E}(\llbracket(\theta,P)\rrbracket)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}} = (p',\phi'_{\mathcal{E}},\delta'_{\mathcal{E}}))\ \Rightarrow\ p = p'$

Intuitively, the theorem states that for any system, (θ, P), it is possible to extract its standard meaning, as interpreted by $\mathcal{S}(\![(\theta, P)]\!)\ \rho\ \phi_{\mathcal{S}}\ \delta_{\mathcal{S}}$, from its non-standard meaning, as interpreted by $\mathcal{E}(\![(\theta, P)]\!)\ \rho\ \phi_{\mathcal{E}}\ \delta_{\mathcal{E}}$. In other words, the former is equal to the first element of the triple generated by the latter.

6 Abstract Semantics

As we mentioned earlier in the previous section, the non-standard semantics of semi-π contains least fixed point calculations that may not be computable due to the infinite nature of the concrete semantic domain. Therefore, in this section, we introduce a couple of abstractions that help limit the size of the semantic domain to a finite level. First, we need to introduce the definitions of *abstract semiring*, *abstract expression*, β and $\succcurlyeq^\sharp$.

Definition 6.1 An abstract semiring, $(\mathcal{A}^\sharp, +^\sharp, \times^\sharp, 1^\sharp, 0^\sharp)$, is a semiring such that $|\mathcal{A}^\sharp| < \infty$.

Definition 6.2 An abstract semiring expression, $e^\sharp$, is obtained from a standard expression, e, by replacing every occurrence of s, $+$, $\times$, 1 and 0, by their abstract forms, $s^\sharp$, $+^\sharp$, $\times^\sharp$, $1^\sharp$ and $0^\sharp$, respectively, and keeping occurrences of ω in e.

Definition 6.3 define the abstraction, $\beta : \mathcal{A} \to \mathcal{A}^\sharp$, to return an abstract semiring value, $\beta(s) = s^\sharp$, corresponding to the concrete value, s. We leave out the definition of β here, since this is application-dependent.

Definition 6.4 Define $s^\sharp \succcurlyeq^\sharp s'^\sharp$ as $s^\sharp +^\sharp s'^\sharp = s^\sharp$. We may write $s^\sharp \succ^\sharp s'^\sharp$ to indicate that $s^\sharp \succcurlyeq^\sharp s'^\sharp$ and $s^\sharp \neq s'^\sharp$.

Next, we introduce a finite predomain of tags, *Tag*, ranged over by t, t' etc. Given a process, P, we place distinct tags on all messages of output actions of P, except those occurring as π in **let** $\omega = \mathcal{S}$**value**(π) **in** n. For example, tagging $!(\nu\, x)\overline{y}\langle x\rangle.\overline{y}\langle y\rangle.y(z).\overline{y}\langle z\rangle$ results in $!(\nu\, x)\overline{y}\langle x^t\rangle.\overline{y}\langle y^{t'}\rangle.y(z).\overline{y}\langle z^{t''}\rangle$. Copies of tags can be renamed by subscripting them with the number of their copy. Hence, the replication above when spawning two copies, becomes:

$$!(\nu\, x)\overline{y}\langle x^t\rangle.\overline{y}\langle y^{t'}\rangle.y(z).\overline{y}\langle z^{t''}\rangle \mid$$
$$(\nu\, x_1)\overline{y}\langle x_1^{t_1}\rangle.\overline{y}\langle y^{t'_1}\rangle.y(z_1).\overline{y}\langle z_1^{t''_1}\rangle \mid (\nu\, x_2)\overline{y}\langle x_2^{t_2}\rangle.\overline{y}\langle y^{t'_2}\rangle.y(z_2).\overline{y}\langle z_2^{t''_2}\rangle$$

We also define the following two functions:

- $\mathit{value\_of}(\{t_1, \ldots, t_n\}) = \{x_1, \ldots, x_m\}$, which when applied to a set of tags, it returns the corresponding messages. For example,

$$\mathit{value\_of}(\{t, t_1, t_2, t', t'_1, t'_2, t'', t''_1, t''_2\}) = \{x, x_1, x_2, y, z, z_1, z_2\}$$

- $\mathit{tags\_of}(P) = \{t_1, \ldots, t_n\}$, which when applied to a process, P, it returns

the set of tags in P. For example,

$$tags\_of\,(!(\nu\, x)\overline{y}\langle x^t\rangle.\overline{y}\langle y^{t'}\rangle.y(z).\overline{y}\langle z^{t''}\rangle) = \{t, t', t''\}$$

Now, we define the following abstraction function for names, tags and semiring variables, which places an upper limit, k, on the total number of copies of names, tags and semiring variables that can be captured during the analysis. In general, selecting k is non-decidable and relies, to a great extent, on user's experience and the specific program being analysed.

Definition 6.5 Define the abstraction function, $\alpha_k : \mathbb{N} \times (N + Tag + \Omega) \rightarrow (N^\sharp + Tag^\sharp + \Omega^\sharp)$, as follows:

$$\forall z \in (N + Tag + \Omega) : \alpha_k(z) = \begin{cases} z_k, \; \textit{if}\; z = z_i \;\wedge\; i > k \\ z, \quad \textit{otherwise} \end{cases}$$

Note that $N^\sharp = N \backslash \{x_j \mid j > k\}$, $Tag^\sharp = Tag \backslash \{t_j \mid j > k\}$ and $\Omega^\sharp = \Omega \backslash \{\omega_j \mid j > k\}$. Using the α_k abstraction function and the abstract semiring, $(\mathcal{A}^\sharp, +^\sharp, \times^\sharp, 1^\sharp, 0^\sharp)$, we can define the abstract environments, $\phi_\mathcal{A} : N^\sharp \rightarrow \wp(Tag^\sharp)$ and $\delta_\mathcal{A} : \Omega^\sharp \rightarrow \wp(\mathcal{A}^\sharp)$. Due to the imprecise nature of the abstract semantics, both $\phi_\mathcal{A}(x)$ and $\delta_\mathcal{A}(\omega)$ may be larger than singleton sets. This imprecision results from the inability to distinguish between the different copies of input parameters, tags and semiring variables beyond the k^{th} copy, and the use of the abstract semiring, $(\mathcal{A}^\sharp, +^\sharp, \times^\sharp, 1^\sharp, 0^\sharp)$.

The abstract semantic domain, $D^\sharp_\perp = (N^\sharp \rightarrow \wp(Tag^\sharp)) \times (\omega^\sharp \rightarrow \wp(\mathcal{A}^\sharp))$, has the following ordering based on subset inclusion (with $\cup_\phi$, $\cup_\delta$ defined as in the previous section, but over abstract rather than concrete environments):

$$\forall (\phi_{\mathcal{A}1}, \delta_{\mathcal{A}1}), (\phi_{\mathcal{A}2}, \delta_{\mathcal{A}2}) \in D^\sharp_\perp : \; (\phi_{\mathcal{A}1}, \delta_{\mathcal{A}1}) \sqsubseteq_{D^\sharp_\perp} (\phi_{\mathcal{A}2}, \delta_{\mathcal{A}2}) \quad \Leftrightarrow \\ \phi_{\mathcal{A}1} \subseteq \phi_{\mathcal{A}2} \;\wedge\; \delta_{\mathcal{A}1} \subseteq \delta_{\mathcal{A}2}$$

The bottom element, $\perp_{D^\sharp_\perp}$, is the pair, $(\phi_{\mathcal{A}0}, \delta_{\mathcal{A}0})$, mapping every abstract name to the empty set and every abstract semiring variable to the empty set, respectively. Using $D^\sharp_\perp$, the abstract semantics of semi-π is defined by a function, $\mathcal{A}(\![(\theta, P)]\!)\; \rho\; \phi_\mathcal{A}\; \delta_\mathcal{A} \in D^\sharp_\perp$, as shown in Figure 6.

The multiset, ρ, holds as usual all the processes composed in parallel with the interpreted process, along with a copy of θ. The rules of the abstract semantics are described informally as follows. Rules, ($\mathcal{A}1$) and ($\mathcal{A}2$), for null processes and input actions do not change the values of $\phi_\mathcal{A}$ and $\delta_\mathcal{A}$, since no communications take place in these rules. In rule ($\mathcal{A}3$), the meaning of an output action is composed from the two cases of no-communications and communications with matching input actions in ρ. A communication is fired whenever the sets of values of two channels, as given by $\phi_\mathcal{A}$, have a non-empty intersection. The effect of the communication is reflected by adding the tag

$(\mathcal{A}1)\ \mathcal{A}([\![(\theta, \mathbf{0})]\!])\ \rho\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} = (\phi_{\mathcal{A}}, \delta_{\mathcal{A}})$

$(\mathcal{A}2)\ \mathcal{A}([\![(\theta, x(y).P)]\!])\ \rho\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} = (\phi_{\mathcal{A}}, \delta_{\mathcal{A}})$

$(\mathcal{A}3)\ \mathcal{A}([\![(\theta, \overline{x}\langle y^t\rangle.P)]\!])\ \rho\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} = ((\bigcup_{\phi\ (\theta,x'(z).P')\in\rho} \phi'_{\mathcal{A}})\ \cup_{\phi}\ \phi_{\mathcal{A}}, (\bigcup_{\delta\ (\theta,x'(z).P')\in\rho} \delta'_{\mathcal{A}})\ \cup_{\delta}\ \delta_{\mathcal{A}})$

where, $(\phi'_{\mathcal{A}}, \delta'_{\mathcal{A}}) = \mathcal{R}([\![\{\!|(\theta, P)|\!\} \uplus \rho[(\theta, P')/(\theta, x'(z).P')]]\!])\ \phi_{\mathcal{A}}[z \mapsto\ \phi_{\mathcal{A}}(z) \cup \{t\}]\ \delta_{\mathcal{A}}$

and, $\exists t \in \phi_{\mathcal{A}}(x), t' \in \phi_{\mathcal{A}}(x') : \mathit{value\_of}(t) = \mathit{value\_of}(t')$

$(\mathcal{A}4)\ \mathcal{A}([\![(\theta, [x\ =\ y]\ P)]\!])\ \rho\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} =$

$$\begin{cases} \mathcal{R}([\![\{\!|(\theta, P)|\!\} \uplus \rho]\!])\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}}, \text{ if } \exists t \in \phi_{\mathcal{A}}(x), t' \in \phi_{\mathcal{A}}(y) : \mathit{value\_of}(t) = \mathit{value\_of}(t') \\ \mathcal{R}([\![\rho]\!])\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}}, \text{ otherwise} \end{cases}$$

$(\mathcal{A}5)\ \mathcal{A}([\![(\theta, [e_1 \succcurlyeq e_2]\ P)]\!])\ \rho\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} =$

$$\begin{cases} \mathcal{R}([\![\{\!|(\theta, P)|\!\} \uplus \rho]\!])\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}}, \text{ if } \exists e^\sharp \in (\mathit{fold}\ Y_{\delta_{\mathcal{A}}}\ \{e_1^\sharp\}\ \mathit{semiv}(e_1^\sharp)), \\ \qquad\qquad\qquad\qquad e'^\sharp \in (\mathit{fold}\ Y_{\delta_{\mathcal{A}}}\ \{e_2^\sharp\}\ \mathit{semiv}(e_2^\sharp)) : e^\sharp \succcurlyeq^\sharp e'^\sharp \\ \mathcal{R}([\![\rho]\!])\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}}, \text{ otherwise} \end{cases}$$

$(\mathcal{A}6)\ \mathcal{A}([\![(\theta, P \mid Q)]\!])\ \rho\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} = \mathcal{R}([\![\{\!|(\theta, P)|\!\} \uplus \{\!|(\theta, Q)|\!\} \uplus \rho]\!])\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}}$

$(\mathcal{A}7)\ \mathcal{A}([\![(\theta, (\nu x)P)]\!])\ \rho\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} = \mathcal{R}([\![\{\!|(\theta, P)|\!\} \uplus \rho]\!])\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}}$

$(\mathcal{A}8)\ \mathcal{A}([\![(\theta, \mathbf{let}\ \omega = \mathcal{S}\mathbf{value}(\pi, n)\ \mathbf{in}\ P)]\!])\ \rho\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} =$

$$\mathcal{R}([\![\{\!|(\theta, P)|\!\} \uplus \rho]\!])\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}}[\omega \mapsto \delta_{\mathcal{A}}(\omega) \cup \{\beta(\theta(\pi, n))\}]$$

$(\mathcal{A}9)\ \mathcal{A}([\![(\theta, !P)]\!])\ \rho\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} = \bigsqcup \mathcal{F}$

where, $\mathcal{F} = \{(\phi_{\mathcal{A}0}, \delta_{\mathcal{A}0}), \mathcal{R}([\![\biguplus_{i=0\ldots\infty} \{\!|(\theta, \mathit{ren}(P, i, \alpha_k))|\!\}\ \uplus\ \rho]\!])\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}}\}$

and, $\forall x \in \mathit{bn}(P), t \in \mathit{tags\_of}(P), \omega \in \mathit{semiv}(P)$:

$$\mathit{ren}(P, i, \alpha_k) = P[\alpha_k(x_i)/x][\alpha_k(t_i)/t][\alpha_k(\omega_i)/\omega]$$

$(\mathcal{R}0)\ \mathcal{R}([\![\rho]\!])\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}} = (\bigcup_{\phi\ (\theta,P)\in\rho} \phi'_{\mathcal{A}}, \bigcup_{\delta\ (\theta,P)\in\rho} \delta'_{\mathcal{A}})$

where, $(\phi'_{\mathcal{A}}, \delta'_{\mathcal{A}}) = \mathcal{A}([\![(\theta, P)]\!])\ (\rho \backslash \{\!|(\theta, P)|\!\})\ \phi_{\mathcal{A}}\ \delta_{\mathcal{A}}$

Fig. 6. The abstract semantics of semi-π.

of the output message to the $\phi_{\mathcal{A}}$-value of the input parameter.

In rule ($\mathcal{A}4$), the matching of two names leads to either choosing process, P, or not, based on the presence of at least one $\phi_{\mathcal{A}}$-value for each of the matched names that are equal. A similar argument follows in rule ($\mathcal{A}5$), when resolving semiring constraints. The definition of *fold* is similar to that of the previous section, however, we redefine $Y_{\delta_{\mathcal{A}}}$ as follows:

$$Y_{\delta_{\mathcal{A}}}(\omega^\sharp, \{e_1^\sharp, \ldots, e_n^\sharp\}) = (\bigcup_{s^\sharp \in \delta_{\mathcal{A}}(\omega^\sharp)} \{e_1^\sharp[s^\sharp/\omega^\sharp]\})\ \cup \ldots \cup\ (\bigcup_{s^\sharp \in \delta_{\mathcal{A}}(\omega^\sharp)} \{e_n^\sharp[s^\sharp/\omega^\sharp]\})$$

In rule ($\mathcal{A}8$), we use the abstraction, β, to abstract a concrete semiring value as returned by $\theta(\pi, n)$ to an element of the abstract set, $\mathcal{A}^\sharp$. The rule for replicated processes, ($\mathcal{A}9$), attaches abstract subscripts to bound names, tags and semiring variables of the spawned processes according to the copy number

of each process. This will only maintain the distinction requirement up to the kth copy, since after that, copies will be identified. Therefore, this semantics will necessarily be approximate and the size of $D_\perp^\sharp$ will necessarily be finite. The rule also defines a least fixed point meaning of the replication as the least upper bound of a poset, $\mathcal{F}$, containing the bottom element, $\perp_{D_\perp^\sharp}$. Since the semantic domain, $D_\perp^\sharp$, is finite in nature, $\mathcal{F}$ may only contain a finite number of $\phi_\mathcal{A}$, $\delta_\mathcal{A}$ elements. As a result, the calculation of the least fixed point is guaranteed to terminate. This is stated more formally as follows.

Theorem 6.6 (Termination of the least fixed point calculation) *The calculation of rule* ($\mathcal{A}9$) *terminates.*

Proof Sketch. We give a sketch of the proof of the termination property. Two requirements must be satisfied. First, the semantic domain must be finite. This is satisfied by the definition of $D_\perp^\sharp$ and the fact that the number of names, tags and semiring values remains finite. The second requirement is to prove that $\mathcal{R}(\![\biguplus_i \{\!|(\theta, ren(P,i,\alpha_k))|\!\}]\!)\ \phi_\mathcal{A}\ \delta_\mathcal{A} \sqsubseteq_{D_\perp^\sharp} \mathcal{R}(\![\biguplus_{i+1} \{\!|(\theta, ren(P,i+1,\alpha_k))|\!\}]\!)\ \phi_\mathcal{A}\ \delta_\mathcal{A}$ (i.e., proving that the meaning is monotonic with respect to an increment in the number of copies of (θ, P)). To prove this, we simplify the inequality into $\mathcal{R}(\![E]\!)\ \phi_\mathcal{A}\ \delta_\mathcal{A} \sqsubseteq \mathcal{R}(\![E \uplus \{\!|(\theta, ren(P,i+1,\alpha_k))|\!\}]\!)\ \phi_\mathcal{A}\ \delta_\mathcal{A}$, where $E = \biguplus_i \{\!|(\theta, ren(P,i,\alpha_k))|\!\}$. This result can be proven by induction over the rules of $\mathcal{A}$. In particular, the most interesting rules are those of ($\mathcal{A}3$) and ($\mathcal{A}8$), where the values of $\phi_\mathcal{A}$ and $\delta_\mathcal{A}$ change. The proof relies on the fact that any communications and semiring value retrievals taking place in $\mathcal{R}(\![E]\!)\ \phi_\mathcal{A}\ \delta_\mathcal{A}$ will necessarily take place in $\mathcal{R}(\![E \uplus \{\!|(\theta, ren(P,i,\alpha_k))|\!\}]\!)\ \phi_\mathcal{A}\ \delta_\mathcal{A}$, since the latter is a larger system than the former, and larger systems always induce at least as much communications and semiring value retrievals as the smaller ones. □

The safety of the abstract semantics can now be established formally in the following theorem.

Theorem 6.7 (Safety of the abstract semantics of semi-π)
$\forall \theta, P, \rho, \phi_\mathcal{E}, \phi_\mathcal{A}, \delta_\mathcal{E}, \delta_\mathcal{A}, x, \omega, \alpha_k, \beta,$
$\mathcal{E}(\![(\theta, P)]\!)\ \rho\ \phi_\mathcal{E}\ \delta_\mathcal{E} = (p, \phi'_\mathcal{E}, \delta'_\mathcal{E}), \mathcal{A}(\![(\theta, P)]\!)\ \rho\ \phi_\mathcal{A}\ \delta_\mathcal{A} = (\phi'_\mathcal{A}, \delta'_\mathcal{A}) :$
$(\exists y \in \phi_\mathcal{E}(x)\ \Rightarrow\ \exists t \in \phi_\mathcal{A}(\alpha_k(x)) : value\_of(\{t\}) = \alpha_k(y)) \wedge$
$(\exists s \in \delta_\mathcal{E}(\omega)\ \Rightarrow\ \exists s^\sharp \in \delta_\mathcal{A}(\alpha_k(\omega)) : s^\sharp = \beta(s))$
$\Rightarrow$
$(\exists y \in \phi'_\mathcal{E}(x)\ \Rightarrow\ \exists t \in \phi'_\mathcal{A}(\alpha_k(x)) : value\_of(\{t\}) = \alpha_k(y)) \wedge$
$(\exists s \in \delta'_\mathcal{E}(\omega)\ \Rightarrow\ \exists s^\sharp \in \delta'_\mathcal{A}(\alpha_k(\omega)) : s^\sharp = \beta(s))$

The safety theorem above states that values present in the final environments resulting from the concrete non-standard semantics will always be present, as abstract tags and elements of the abstract semiring, in the environments resulting from the abstract semantics.

7 Analysing the Satisfaction of Semiring Constraints

Semiring constraints in fact allow us to define a Constraint Satisfaction Problem (CSP) [10, §2.1.1], as follows:

$$\mathcal{C} = (\{\omega_1, \ldots, \omega_n\}, \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}, \{e_1 \succcurlyeq e_2, \ldots, e_k \succcurlyeq e_{k+1}\})$$

where $\omega_{i=1\ldots n} \in \bigcup_{j=1}^{k+1} semiv(e_j)$. The solution to this problem is obtained by taking "good" instantiations of the semiring variables, $\omega_1 \ldots \omega_n$, i.e. instantiations that satisfy the set of constraints, $\{e_1 \succcurlyeq e_2, \ldots, e_k \succcurlyeq e_{k+1}\}$. Hence, a solution is defined as a set, $Sol(\mathcal{C}) = \{\omega_1 \mapsto s_1 \in \mathcal{A}_1, \ldots, \omega_n \mapsto s_n \in \mathcal{A}_n\}$. Furthermore, we refer to the set of all solutions as $SOL(\mathcal{C}) = \{Sol_1(\mathcal{C}), \ldots, Sol_m(\mathcal{C})\}$.

From the set, $SOL(\mathcal{C})$, it is possible to relate a solution to the property of name substitutions as follows. Assume that C is a context, i.e. it is a process with a hole, $P[.]$. This allows us to write a process with a string of semiring constraints as $C[[e_1 \succcurlyeq e_2] \ \ldots [e_k \succcurlyeq e_{k+1}] \ P]$. Then, we have the following two environments:

$$\phi_{\mathcal{A}1} = fst(\mathcal{A}[\![(\theta, C[P])]\!] \ \rho \ \phi_{\mathcal{A}} \ \delta_{\mathcal{A}})$$
$$\phi_{\mathcal{A}2} = fst(\mathcal{A}[\![(\theta, C[\mathbf{0}])]\!] \ \rho \ \phi_{\mathcal{A}} \ \delta_{\mathcal{A}})$$

Which result from replacing $[e_1 \succcurlyeq e_2] \ \ldots [e_k \succcurlyeq e_{k+1}] \ P$ once by P and once by $\mathbf{0}$. Now, we define the *difference* between the two environments as follows:

$$\phi^P_{\mathcal{A}} = \phi_{\mathcal{A}1} \backslash \phi_{\mathcal{A}2}$$

The $\phi^P_{\mathcal{A}}$ environment represents the effect reflected in name substitutions of satisfying the set of constraints, $\{e_1 \succcurlyeq e_2, \ldots, e_k \succcurlyeq e_{k+1}\}$.

The following definition formalised *name substitution/CSP dependency*.

Definition 7.1 Given a system, $(\theta, C[[e_1 \succcurlyeq e_2] \ \ldots [e_k \succcurlyeq e_{k+1}] \ P])$, an abstract interpretation, $\mathcal{A}[\![(\theta, C[[e_1 \succcurlyeq e_2] \ \ldots [e_k \succcurlyeq e_{k+1}] \ P])]\!] \ \rho \ \phi_{\mathcal{A}} \ \delta_{\mathcal{A}} = (\phi'_{\mathcal{A}}, \delta'_{\mathcal{A}})$, and a difference environment, $\phi^P_{\mathcal{A}}$, then every substitution, $[x^\sharp \mapsto \{a^\sharp_1, \ldots, a^\sharp_n\}] \in \phi^P_{\mathcal{A}}$, is dependent on reaching one or more of the solutions, $SOL(\mathcal{C}^\sharp)$, such that:

$$\mathcal{C}^\sharp =$$
$$(\{\omega^\sharp \mid \omega^\sharp \in \bigcup_{j=1}^{k+1} semiv(e^\sharp_j)\}, \{set^\sharp \mid set^\sharp = \delta'_{\mathcal{A}}(\omega^\sharp)\}, \{e^\sharp_1 \succcurlyeq^\sharp e^\sharp_2, \ldots, e^\sharp_k \succcurlyeq^\sharp e^\sharp_{k+1}\})$$

The property states that a name substitution can only occur subject to the satisfaction of the CSP over the abstract constraints. Since we are dealing with normal processes only, with no occurrences of homonymous semiring variables or bound names, it is possible to relate every $\omega^\sharp$ to some abstract action, $\pi^\sharp$, and name, $z^\sharp$, by observing a subprocess, **let** $\omega = \mathcal{S}\mathbf{value}(\pi, z)$ **in** P, in the syntax of the analysed process. This will allows us to relate name substitutions to the cost of actions under certain circumstances.

8 Example: Adaptive Router

We consider in this section an example of a simplified adaptive router. The system consists mainly of a process, *Router*, which is connected to an n number of other routers through channels, out_i, for $i = 1 \ldots n$. The router repeatedly accepts a message (the destination) after which it queries the state θ for the cost of routing the message over each of its output connections. One may think of θ as implementing some algorithm for the calculation of the shortest path, for example Dijkstra's [11]. Using these costs, the router decides which connection is best to send the message over. The specification of the system is shown in Figure 7, in which it runs the router process for the case of five connections.

$$
\begin{array}{lcl}
\mathit{Router}(n) & \stackrel{\text{def}}{=} & !\mathit{route}(\mathit{dest}). \\
 & & \textbf{let } \omega_1 = \mathcal{S}\textbf{value}(\overline{\mathit{out}_1}\langle \mathit{dest}\rangle, \mathit{dest}) \textbf{ in} \\
 & & \vdots \\
 & & \textbf{let } \omega_n = \mathcal{S}\textbf{value}(\overline{\mathit{out}_n}\langle \mathit{dest}\rangle, \mathit{dest}) \textbf{ in} \\
 & & \textbf{let } \omega_{1+n} = \mathcal{S}\textbf{value}(\mathit{out}_1(x_1), \mathit{dest}) \textbf{ in} \\
 & & \vdots \\
 & & \textbf{let } \omega_{n+n} = \mathcal{S}\textbf{value}(\mathit{out}_n(x_n), \mathit{dest}) \textbf{ in} \\
 & & \prod_{i=1}^{n} \left(\bigodot_{j=min(\{1,\ldots,n\}\setminus i)}^{max(\{1,\ldots,n\}\setminus i)} [(\omega_i \times \omega_{n+i}) \succcurlyeq (\omega_j \times \omega_{n+j})] \right) \overline{\mathit{out}_i}\langle \mathit{dest}^t \rangle \\
 & & \\
\mathit{System} & \stackrel{\text{def}}{=} & (\theta, \mathit{Router}(5) \mid \overline{\mathit{route}}\langle \mathit{our\_dest}^{t'} \rangle \mid \prod_{i=1}^{5} \mathit{out}_i(x_i))
\end{array}
$$

Fig. 7. The Specification of the Routing Protocol.

Here, we use the special symbol,

$$\left(\bigodot_{i=1}^{k} [e_i \succcurlyeq e_i']\right) P$$

as a shorthand for,

$$[e_1 \succcurlyeq e_1'] \ \ldots [e_k \succcurlyeq e_k'] \ P$$

We start the analysis of this system by first adopting the assumptions:

- The concrete semiring is the weighted semiring, $(\mathbb{R}^+, min, +, +\infty, 0)$.
- θ holds values for the different actions, as follows:
  $\theta(\overline{\mathit{out}_1}\langle \mathit{our\_msg}, \mathit{our\_dest}\rangle, \mathit{our\_dest}) = 30$
  $\theta(\overline{\mathit{out}_2}\langle \mathit{our\_msg}, \mathit{our\_dest}\rangle, \mathit{our\_dest}) = 3$
  $\theta(\overline{\mathit{out}_3}\langle \mathit{our\_msg}, \mathit{our\_dest}\rangle, \mathit{our\_dest}) = 3$
  $\theta(\overline{\mathit{out}_4}\langle \mathit{our\_msg}, \mathit{our\_dest}\rangle, \mathit{our\_dest}) = 1010$
  $\theta(\overline{\mathit{out}_5}\langle \mathit{our\_msg}, \mathit{our\_dest}\rangle, \mathit{our\_dest}) = 60$

$\theta(out_1(x_1, y_1), our\_dest) = 100$
$\theta(out_2(x_2, y_2), our\_dest) = 5$
$\theta(out_3(x_3, y_3), our\_dest) = 10000$
$\theta(out_4(x_4, y_4), our\_dest) = 50000$
$\theta(out_5(x_5, y_5), our\_dest) = 5$

- The abstract semiring is $(\{low, medium, high\}, min, max, high, low)$. Additionally, we define β as follows:

$$\forall r \in \mathbb{R}^+ : \beta(r) = \begin{cases} low, & \text{if } r < 10 \\ medium, & \text{if } 10 \leq r \leq 100 \\ high, & \text{if } r > 100 \end{cases}$$

- We assume a non-uniform analysis with the abstraction function is α_5.
- We allow ourselves to abuse the notation by using parameterised non-recursive definitions, like Router(n).

Next, we run the abstract interpretation for the following systems:

$\mathcal{A}(\![\ System\ ]\!)\ \{\!|\}\ \phi_{\mathcal{A}0}\ \delta_{\mathcal{A}0} = (\phi_{\mathcal{A}}, \delta_{\mathcal{A}})$
$\mathcal{A}(\![\ System[\mathbf{0}/string]\ ]\!)\ \{\!|\}\ \phi_{\mathcal{A}0}\ \delta_{\mathcal{A}0} = (\phi'_{\mathcal{A}i}, \delta'_{\mathcal{A}i})$
$\mathcal{A}(\![\ System[\overline{out_i}\langle msg, dest\rangle/string]\ ]\!)\ \{\!|\}\ \phi_{\mathcal{A}0}\ \delta_{\mathcal{A}0} = (\phi''_{\mathcal{A}i}, \delta''_{\mathcal{A}i})$

where,

$$string \stackrel{\texttt{def}}{=} \Big(\bigodot_{j=min(\{1,\ldots,n\}\setminus i)}^{max(\{1,\ldots,n\}\setminus i)} [\omega_i \times \omega_{n+i} \succcurlyeq \omega_j + \omega_{n+j}]\Big)\ \overline{out_i}\langle msg, dest\rangle$$

for all the cases of $i \in \{1, \ldots, 5\}$. As a result, it is now possible to construct the following difference environments:

$\phi^P_{\mathcal{A}i} = \phi''_{\mathcal{A}i} \backslash \phi'_{\mathcal{A}i}$ for each value of $i \in \{1, \ldots, 5\}$

Examining these difference environments, we find that the only environment which is not empty is $\phi^P_{\mathcal{A}2} = \{(x_2, \{t_2\})\}$, where $value\_of(\{t_2\}) = \{dest_2\}$. This implies that only the second set of constraints and the second output are capable of influencing the routing of the message to its destination. The reason is because $\beta(\omega_{2_1}) \times \beta(\omega_{7_1})$ constitutes in fact the best abstract semiring value among all the other possible values (based on a *min* ordering). Consequently, our CSP-based Property 7.1 as defined in the previous section allows us to state that the name substitution, $\{(x_2, \{t_2\})\}$, is dependent reaching the solution that $\beta(\omega_{2_2}) = \beta(\omega_{7_2}) = low$ in every constraint, and that for the each of the remaining semiring variables, that at least one variable occurring in each constraint will reach one of the abstract values of *medium* or *high*. Put more

precisely, the costs of $\overline{out_2}\langle our\_dest\rangle$ and $out_2(x_2)$ are the best.

9 Conclusion and Future Work

We have presented a static analysis of semi-π: an extension of the π-calculus that incorporates capabilities for the retrieval of semiring values of communication actions and then allows these values to be reasoned about in semiring constraints. The analysis captures the property of name substitutions and instantiations of semiring variables using semiring values. The results of the analysis allow us to relate name substitutions to the satisfaction of semiring constraints, and ultimately, to the cost of communication actions. We applied the analysis to a simple adaptive routing example.

In the future, we are planning to implement the analysis in a functional programming language, like SML. Such languages are quite suitable to the syntax-directed approach adopted in the definition of our analysis. Moreover, we are planning to apply the analysis to more interesting adaptive network routing algorithms and adaptive power-saving in small devices. Another interesting extension would be to model the cost of cryptographic operations, in languages like the spi calculus.

References

[1] Martín Abadi and Andrew Gordon. A calculus for cryptographic protocols: The spi calculus. Technical Report 414, University of Cambridge Computer Laboratory, January 1997.

[2] B. Aziz, G.W. Hamilton, and D. Gray. A denotational approach to the static analysis of cryptographic processes. In *Proceedings of International Workshop on Software Verification and Validation*, volume 118, pages 19–36, Mumbai, India, December 2003. Electronic Notes in Theoretical Computer Science.

[3] Benjamin Aziz, David Gray, and Geoff Hamilton. A static analysis of pki-based systems. In *Proceedings of the* 9^{th} *Italian Conference on Theoretical Computer Science*, volume 3701 of *Lecture Notes in Computer Science*, pages 51–65, Siena, Italy, October 2005. Springer Verlag.

[4] Benjamin Aziz and Geoff Hamilton. A denotational semantics for the π-calculus. In *Proceedings of the* 5^{th} *Irish Workshop in Formal Methods*, Electronic Workshops in Computing, Dublin, Ireland, July 2001. British Computing Society Publishing.

[5] Benjamin Aziz and Geoff Hamilton. A privacy analysis for the π-calculus: The denotational approach. In *Proceedings of the* 2^{nd} *Workshop on the Specification, Analysis and Validation for Emerging Technologies*, number 94 in Datalogiske Skrifter, Copenhagen, Denmark, July 2002. Roskilde University.

[6] Benjamin Aziz, Geoff Hamilton, and David Gray. A static analysis of cryptographic processes: The denotational approach. *Journal of Logic and Algebraic Programming*, 64(2):285–320, August 2005.

[7] Stefano Bistarelli. *Semirings for Soft Constraint Solving and Programming*, volume 2962 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[8] Linda Brodo, Pierpaolo Degano, and Corrado Priami. A tool for quantitative analysis of calculus processes. In José D. P. Rolim, Andrei Z. Broder, Andrea Corradini, Roberto Gorrieri, Reiko Heckel, Juraj Hromkovic, Ugo Vaccaro, and J. B. Wells, editors, *Proceedings of the* 27th *International Colloquium on Automata, Languages and Programming Satellite Workshops*, pages 535–550, Geneva, Switzerland, 2000. Carleton Scientific.

[9] Peter Buchholz and Peter Kemper. Quantifying the dynamic behavior of process algebras. In *Proceedings of the Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, volume 2165 of *Lecture Notes in Computer Science*, pages 184–199, Aachen, Germany, September 2001. Springer Verlag.

[10] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.

[11] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerical Mathematics*, 1:269–271, October 1959.

[12] David Gray, Benjamin Aziz, and Geoff Hamilton. Spiky: A nominal calculus for modelling protocols that use pkis. In *Proceedings of the International Workshop on Security Analysis of Systems: Formalism and Tools*, Orléans, France, June 2004.

[13] Dan Hirsch and Emilio Tuosto. Shreq: Coordinating application level qos. In *Proceedings of the* 3rd *IEEE International Conference on Software Engineering and Formal Methods*, Koblenz, Germany, September 2005. IEEE Computer Society Press (to appear).

[14] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes (parts I & II). *Information and Computation*, 100(1):1–77, September 1992.

[15] Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Measuring the confinement of probabilistic systems. *Theoretical Computer Science*, 340(1):3–56, 2005.

[16] Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Quantitative static analysis of distributed systems. *Journal of Functional Programming*, 15(5):703–749, 2005.

[17] Corrado Priami. Integrating behavioural and performance analysis with topology information. In *Proceedings of the* 29th *Annual Hawaii International Conference on System Sciences*, volume 1, pages 508–516, Maui, Hawaii, USA, January 1996.

[18] Ian Stark. A fully abstract domain model for the π-calculus. In *Proceedings of the* 11^{th} *Annual IEEE Symposium on Logic in Computer Science*, pages 36–42, New Brunswick, New Jersey, USA, July 1996. IEEE Computer Society.