INFORMATION FLOW SECURITY FOR INTERACTIVE SYSTEMS[1]

Jin Ying[1], Liu Lei[1], Zheng Xiao-juan[2]

[1] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education of P.R.China, Computer Science & Technology College, Jilin University, Changchun, 130012, P.R.China
[2]Software College, Northeast Normal University, Changchun, 130117, P.R.China
Corresponding author: Liu Lei, liulei@mail.jlu.edu.cn

Abstract. The use of the Internet raises serious behavioural issues regarding, for example, security and the interaction among agents that may travel across links. Model-building such interactive systems is one of the biggest current challenges in computer science. A general model, action calculi, has been introduced by Robin Milner to unify the various emerging disciplines of interactive behaviour. In this paper action calculi is used as an abstraction of interactive systems and information flow security properties of such systems are studied. At first an information flow analysis for static action calculi is presented to predict how data will flow both along and inside actions and its correctness is proved; Next basing on the result of the analysis information security properties of both static and dynamic action calculi are discussed; Finally a general relationship are established between the static notation of information flow security and the dynamic one.

1 Introduction

The use of the Internet raises serious behavioural issues regarding, for example, security and the interaction among agents that may travel across links. Model-building such interactive systems is one of the biggest current challenges in computer science. Action calculi have been introduced by Milner[6, 8] as a framework for representing models of interactive computation. It is indicated in [6, 8, 10, 11] that action calculi show the advantage in uniting different models of interactive systems in a common setting[6, 8, 10, 11], and such unification is necessary for studying general properties such as security properties of these systems[5].

Information flow security is concerned with controlling the flow of information within a sytem. Program analysis such as information flow analysis aims at verifying properties of a program that hold in all execution, which recently has been used for validating security and safety issues for concurrent and interactive systems[16]. In this paper, we use action calculi as an abstract of interactive systems, and study its information security properties basing on information flow analysis. The molecular forms of action calculi give normal form for the algebraic terms and suggest a modular style of programming and system description. Therefore, we propose a formal information flow analysis for the molecular forms of static action calculi, which statically shows how data will flow both along and inside molecules. In order to facilitate the formulation of the analysis, we make trivial extension to the syntax of action calculi in that giving every action and molecule a unique name. Following the idea of [1, 2], we state simple security property for static action calculi, that is: the secrecy of data is preserved if an action a never read an untrustworthy name to it or an action a never write to other actions the untrustworthy name. Finally we

[1] This paper is sponsored by Jilin Province Science and Technology Development Plan Project (No. 20050527).

introduce security property for dynamic action calculi, and establish the general relationship between the static notation and dynamic one.

The rest of the article is organized as follows. Section 2 briefly reviews the basic concepts and definition of action calculi. Section 3 presents formalization of information flow analysis of static action calculi and proof of the correctness of the analysis. Security properties basing on the analysis above and a general relationship between the static notation and dynamic one are established. Finally related work and conclusion are addressed in section 5.

2 Preliminaries

In this section we briefly review the definition of action calculi following [6,7, 8]. A trivial extension is made by adding unique identifier to every action and every molecule.

In this paper we focus on action calculi in molecular forms. An action calculus is determined by a signature K = (P, K) together with a set of control rules. K consists of a set P of basic types, called primes and denoted by p, q, …, and a set K of constants, called controls. Each control in K has an associated arity ((m1, n1), …, (mr, nr)) → (m, n), where the m's and n's are finite sequences of primes, called tensor arities; we write ε for empty sequence, ⊗ for concatenation using infix notation, and write M for the set of tensor arities.

Definition 1 (Molecules and molecular forms) Let $\mathbb{K}$ be a signature. The molecular forms over $\mathbb{K}$ are syntactic objects; they consist of the actions *a* defined as follows, in terms of molecules μ:

$a ::= A[(\vec{x})\ \mu_1 \ldots \mu_r <\vec{u}>]$ $\quad (\vec{x} : m.;\ \vec{u} : n\ ;\ a : m \to n)$

$\mu ::= M[<\vec{v}> K\ \vec{b}\ (\ \vec{y}\ )]$ $\quad (\vec{v} : k\ ;\ \vec{y} : l\ ;\ K\vec{b} : k \to l)$

where A and M are identifiers of action and molecule respectively, the sequence $\mu_1, \ldots, \mu_r$ is called the body of *a*, K is a control, $(\vec{x})$ and $(\vec{y})$ are called *imported names* (which must be distinct) denoted as *imp*(*a*/μ), $<\vec{u}>$ and $<\vec{v}>$ are called *exported names* denoted as *exp*(*a*/μ).

Definition 2 (subaction) The subactions of an action *a* comprise *a* itself and the subactions of each action in a molecule of *a*.

For simplicity, we will use identifier to represent corresponding action or molecule.

In action a the imported names $\vec{x}$ are binding, and the names $\vec{y}$ of the molecule are also binding. The scope of each binding extends to its right, to the end of the smallest subaction containing it. Molecules are binding operators. In the above molecule μ, the names $<\vec{v}>$ occur free; they are the means by which it is bound into an action. In the above action a, any name-vector in round brackets – either at the head of a or at the right end of a molecule in μ - is binding, and its scope extends rightwards to the end of a.

Definition 3 (Operations over molecular forms) The operations $\mathbf{id}_k$, ·, ⊗, ab_x, <x> and ω are defined over molecular forms. The detailed definitions are in [6, 7, 8].

Definition 4 (Action calculi: static) A static action calculi comprises a signature $\mathbb{K}$, together with a set of actions in the molecular forms over $\mathbb{K}$, and a set of operations defined as above. We call this static action calculus over $\mathbb{K}$, and denote it by $AC^s(\mathbb{K})$.

Definition 5 (Control operations) Each control *K*, is defined as a control operation upon molecular forms as follows:

$K(a) \stackrel{def}{=} A_n[(\vec{x}) <\vec{x}> Ka\ (\vec{y}) <\vec{y}>]$ $\quad (\vec{x},\ \vec{y}$ not free in $a)$

Definition 6 (Control rule) A control rule over a signature $\mathbb{K}$ takes the form

$t[\ \vec{a}\ ] \searrow t'\ [\ \vec{a}\ ]$

where t and t' are terms built from metavariables $\vec{a}$ using controls together with the operations in definition 4.

Definition 7 (Action calculi: dynamics) A dynamic action calculus comprises a signature $\mathbb{K}$ and a set of *R* of control rules over, together with the static action calculi $AC^s(\mathbb{K})$ equipped with the

smallest reaction relation $\searrow$ which satisfies the rules R (for all replacements of metavariables a by actions). We call this the dynamic action calculus over $\mathbb{K}$ and R, and denote it by AC($\mathbb{K}$, R).

3 Information Flow Analysis

3.1 Notations and definitions

A *path* is an ordered list of identifiers for actions or molecules. An *occurrence path* for an action a or a molecule m or a name x in an action a' or a molecule m' is a path consisting of all the identifiers for those actions or molecules along which the action/molecule/name could be reached, which is denoted as *Path*(a/m/x, a'/m').

We also define *path environment*, *abstract binding environment* and *abstract bound environment* as following:

- σ is the path environment that associates imported names with the occurrence path of that name in the corresponding molecular forms; Current path environment contains those names that are scope effective in current action or molecule together with corresponding paths; For a name x and its path p, $\sigma[x \rightarrow p]$ represents updating σ by modifying the path of x as p;
- ρ is the abstract binding environment that associate a given export name x and its occurrence path p with the occurrence path of the binding name. This means that $\rho(x,p)$ returns the occurrence path of the imported name x which is binding the exported name x occurring on the path p; For a name x and its path p , $\rho[(x,p) \rightarrow p']$ means adding ρ with a new binding (imported name x of path p' binds exported name of path p);
- κ is the abstract bound environment that associates a given import name x and its occurrence path p with the set of occurrence paths of the exported names bound by corresponding x. More precisely, for a given export name x and its occurrence path p, $\kappa(x,p)$ returns the paths of all the name bound by x; For a name x and its path p , $\kappa\ [(x,p) \rightarrow P']$ means adding κ with a new bound (imported name x of path p binds a set of exported names of every path in P').

Path environment, binding environment and bound environment together are called abstract environment.

We define an operation $\vee$ for unifying abstract binding and bound environment:

$$\rho_1 \vee \rho_2 (x, p) = \begin{cases} \rho_1(x, p), & \text{if } \rho_1(x, p) \neq () \\ \rho_2(x, p), & \text{if } \rho_2(x, p) \neq () \\ () , & \text{otherwise} \end{cases}$$

$$\kappa_1 \vee \kappa_2 (x, p) = \kappa_1(x, p) \cup \kappa_2 (x, p)$$

3.2 Strategy of Information Flow Analysis

Information flow analysis for action calculi is presented and its correctness is proved.

The aim of information flow analysis for static action calculi is to determine which imported names in a certain action or molecule will bind which exported names. This result will be used to check the information flow along internal actions so as to prove whether such information flow is secure under security policy. The result of static analysis of the syntactic molecular forms include:

a) the binding imported name for any exported name;

b) all bounded exported names for any binding imported name x;

A function performing information flow analysis is defined. For given molecular form of an action or a molecule with its path, the function accepts current abstract environment and computes new abstract environment.

The definition of the function is as follows:

[m]pσρκ= (σ', ρ', κ'), where m is the molecular form of an action or a molecule, p is the path for m, (σ', ρ', κ') is the current abstract environment, and (σ', ρ', κ') is the new abstract environment.

The function is defined on an action in the form of id[($\bar{x}$)< $\bar{y}$ >] and id[($\bar{x}$) [μ1 …μr]< $\bar{y}$ >], a molecule in the form of id[< $\bar{u}$ > K ($\bar{v}$)]and id[< $\bar{u}$ > Ka ($\bar{v}$)]. The analyzing process is based on nesting scope rules for static action calculi. The detailed function definition and explanation are as follows.

$[\![$id[($\bar{x}$)< $\bar{y}$ >]$]\!]$ pσρκ=

let σ' = σ[x_i→p] (i = 1, …, m)

in *let* $\rho_0 = \Phi$, $\kappa_0 = \Phi$

in *let* for i = 1 to n do

{ if $y_i \in \bar{x}$ then { $\rho_i = \rho_{i-1}[(y_i,p)\rightarrow p]$; $\kappa_i=\kappa_{i-1}[(y_i,p)\rightarrow\{p\}];$}

else if $\sigma'(y_i)\neq\{\}$ then {$\rho_i= \rho_{i-1}[(y_i,p)\rightarrow\sigma'(y_i)]$;

$\kappa_i=\kappa_{i-1}[(y_i,p)\rightarrow\kappa_{i-1}(y_i,p)\cup\{p\}];$}

else { $\rho_i = \rho_{i-1}[(y_i,p)\rightarrow()]$; $\kappa_i = \kappa_{i-1}$}

in (σ', $\rho \vee \rho_n$, $\kappa \vee \kappa_n$)

For simple actions with empty body, first we would update path environment with every imported name and its path as current path p, then for every exported name, we will check if there is corresponding name in binding environment, if so a new binding should be added to binding environment and put the path of this exported name into the set of bound path of the binding name, otherwise the exported name is a free name, so we would add an empty path as the path of its binding and make bound environment unchanged.

$[\![$id[($\bar{x}$) [μ_1 …μ_r]< $\bar{y}$ >]$]\!]$pσρκ =

let σ_0= σ[x_i→p] (i = 1, …, m)

in *let* $\rho_0 = \Phi$, $\kappa_0 = \Phi$

in *let* for i = 1 to r do

$(p_i, \sigma_i, \rho_i , \kappa_i) = [\![\mu_i]\!](cons(p,id(\mu_i)), \sigma_{i-1}, \rho_{i-1} , \kappa_{i-1})$

in *let* $\rho_0 = \rho_r$, $\kappa_0 = \kappa_r$

for j = 1 to n do

if $\sigma_r(y_j) \neq\{\}$ then {$\rho_j= \rho_{j-1}[(y_i,p)\rightarrow\sigma'(y_j)]$;

$\kappa_j=\kappa_{j-1}[(y_i,p)\rightarrow\kappa_{j-1}(y_j,p) \cup\{p\}];$}

else { $\rho_j = \rho_{j-1}[(y_j,p)\rightarrow()]$; $\kappa_j = \kappa_{j-1};$}

in (σ_r, $\rho \vee \rho_n$, $\kappa \vee \kappa_n$)

For actions with its body as a nonempty sequence of molecules, first we would also update path environment with every imported name and its path as current path p; then in order to apply for the binding mechanism of molecules, for every molecule we will sequentially compute their result abstract environment with former result abstract environment as input. Finally we will compute on the exported names of the action. Here we use cons(p, id(μi)) represent the concatenating current path with current molecule identifier to get the path for this molecule.

Analyzing molecule also includes two computation forms:

$[\![$ id[< $\bar{u}$ > K ($\bar{v}$)]$]\!]$pσρκ=

let $\rho_0 = \Phi, \kappa_0 = \Phi$

for j = 1 to n do

if $\sigma(u_j) \neq \{\}$ then $\{\rho_j = \rho_{j-1}[(u_i,p) \rightarrow \sigma'(u_j)]; \kappa_j = \kappa_{j-1}[(u_i,p) \rightarrow \{p\}];\}$

else $\{ \rho_j = \rho_{j-1}[(u_j,p) \rightarrow ()]; \kappa_j = \kappa_{j-1};\}$ $\sigma' = \sigma[v_i \rightarrow p]$ (i = 1, …, m)

in $(\sigma', \rho \vee \rho_n, \kappa \vee \kappa_n)$

For molecules with constant control, first we would also update binding environment and bound environment by computing on their exported names with respect to input abstract environment, then update path environment with every imported name and its path as current path p.

$[\![id[<\bar{u} > Ka(\bar{v})]]\!]p\sigma\rho\kappa =$

let $\rho_0 = \Phi, \kappa_0 = \Phi$

for j = 1 to k do

if $\sigma(y_j) \neq \{\}$ then $\{\rho_j = \rho_{j-1}[(u_i,p) \rightarrow \sigma'(y_j)]; \kappa_j = \kappa_{j-1}[(u_i,p) \rightarrow \{p\}];\}$

else $\{ \rho_j = \rho_{j-1}[(u_j,p) \rightarrow ()]; \kappa_j = \kappa_{j-1};\}$

in *let* $(\sigma_1, \rho_1, \kappa_1) = [\![a_1]\!](cons(p,id(a_1)), \sigma', \Phi, \Phi)$

……

$(\sigma_n, \rho_n, \kappa_n) = [\![a_n]\!](cons(p, id(a_n)), \sigma', \Phi, \Phi)$

in *let* $\sigma' = \sigma[v_i \rightarrow p]$ (i = 1, …, *l*)

in $(\sigma', \rho_k \vee \rho_1 \vee \ldots \vee \rho_n, \kappa_k \vee \kappa_1 \vee \ldots \vee \kappa_n)$

For molecules with nonzero rank control, first we would also update binding environment and bound environment by computing on their exported names with respect to input abstract environment. Then since the scopes of the action parameters of a molecule are independent, we will update their result abstraction environment in parallel with the same input abstract environment. Next path environment will be obtained by updating original input path environment with every imported name and its path as current path p because the imported names in action parameters will not be effective outside the molecule. Finally the resulting binding environment and bound environment are the combination of all the binding environment and bound environment obtained from the computations of those action parameters.

3.3 Correctness

The correctness of the function is demonstrated by showing that the function can compute correct abstract environment for any molecular forms.

Here with four prepositions we get one lemma which will show that with the above function we can get correct abstract environment for any molecular forms.

Proposition 1 Assume that $id[(\bar{x})<\bar{y}>]$ is a subaction of an action *a*, p is its path in *a*, σ is the right set of names and their paths that is effective for id, if $[\![id[(\bar{x})<\bar{y}>]]\!]p\sigma\rho\kappa = (\sigma', \rho', \kappa')$ then we have :

(1) yi is bound by an imported name of path p' in a iff $\rho'(p, yi) = p'$;

(2) xi is binding an exported name of path p' in a iff $p' \in \kappa'(p, xi)$.

Proof: From the definition of $[\![id[(\bar{x})<\bar{y}>]]\!]$we can see that:

(1) σ' is just obtained by updating σ with current new effective imported names x; since σ is right, therefore σ' is right too;

(2) with new computed σ' we can compute (ρn, κn) by adding exported names y; since σ' is right, therefore from the computation process of (ρn, κn) we can say that (ρn, κn) is correct

for id, therefore adding (ρn, κn) to original (ρ, κ) will produce a correct one (ρ' , κ')so far including id;

On the other hand, if there is ρ'(p, yi) = p' and p' ∈κ'(p, xi), since the path is unique for id, therefore they can only be computed by [id[($\bar{x}$)< $\bar{y}$ >]], and since σ' is effective, therefore, yi is bound by an imported name of path p' in a and xi is binding an exported name of path p' in a.

Proposition 2 Assume that id[($\bar{x}$) [μ1 …μr] < $\bar{y}$ >] is one subaction of an action a, p is its path in a, σ is the right set of names and their paths that is effective for id, if [id[($\bar{x}$) [μ1 …μr] < $\bar{y}$ >]]pσρκ= (σ', ρ' , κ') , and if [μi](cons(p,id(μi)), σi-1, ρi-1 , κi-1) are correct, then we have :

(1) y_i is bound by an imported name of path p' in *a* iff ρ'(p, y_i) = p' ;

(2) xi is binding an exported name of path p' in a iff p' ∈ κ'(p, xi).

Proof: From the definition of ⟦ id[($\bar{x}$) [μ_1 …μ_r]< $\bar{y}$ >]⟧ we can see that:

(1) σ' is just obtained by updating σ with current new effective imported names x; since σ is right, therefore σ' is right too;

(2) we can sequentially compute binding and bound environment for every molecule, and we know that [μi](cons(p,id(μi)), σi-1, ρi-1 , κi-1) are correct; then for exported names of id, the computation is processed with the result of all the molecules, which is also correct. Therefore, adding the correct result of molecules and that of the exported names of id to original (ρ, κ) will produce a correct one so far including id; On the other hand, if there is ρ'(p, yi) = p' and p' ∈κ'(p, xi), since the path is unique for id, therefore they can only be computed by [id[($\bar{x}$) [μ1 …μr] < $\bar{y}$ >]], and since path environment is effective, therefore, yi is bound by an imported name of path p' in a and xi is binding an exported name of path p' in a.

Proposition 3 Assume that id[< $\bar{u}$ > K ($\bar{v}$)] is a molecule of one subaction of an action *a*, p is its path in *a*, σ is the right set of names and their paths that is effective for id, if ⟦id[< $\bar{u}$ > K ($\bar{v}$)]⟧pσρκ= (σ', ρ' , κ') then we have :

(1) u_i is bound by an imported name of path p' in *a* iff ρ'(p, u_i) = p' ;

(2) vi is binding an exported name of path p' in a iff p' ∈κ'(p, vi).

Proof: The proof is similar to preposition 1.

Proposition 4 Assume that id[< $\bar{u}$ > Ka ($\bar{v}$)] is a molecule of one subaction of an action *a*, p is its path in *a*, σ is the right set of names and their paths that is effective for id, if ⟦id[< $\bar{u}$ > Ka ($\bar{v}$)]⟧pσρκ= (σ', ρ' , κ') , and if ⟦a_i⟧(cons(p,id(a_1)), σ', Φ, Φ) are correct, then we have :

(1) u_i is bound by an imported name of path p' in *a* iff ρ'(p, u_i) = p' ;

(2) vi is binding an exported name of path p' in a iff p' ∈κ'(p, vi).

Proof: The proof is similar to preposition 2.

Lemma 1 For an action *a* whose identifier is id_a, we assume that the initial abstract environment is { σ_0 = [x→()]; ρ_0 = [(p, x)→()]; κ_0 = [(p, x)→{}] }, for any name x occurring in *a*, and the initial path p is (id_a). If ⟦*a*⟧p$\sigma_0\rho_0\kappa_0$= (σ, ρ, κ), then we have:

(1) an exported name x of path p_1 is bound by the imported name of path p_2 in *a* iff ρ(p_1, x) = p_2 ;

(2) an imported name x of path p1 is binding an exported name of path p2 in a iff p2∈κ(p1, x).

Proof:

(1) for any action, the initial abstract given here as input is no doubt correct;

(2) since the number of subactions of an action is finite number, therefore, the [a]pσ0ρ0κ0 will surely terminate;

(3) for any action a, from the definition of the analyzing function, computation of [a]pσ0ρ0κ0 will be processed recursively until meeting the constant action or a molecule with constant control, from the preposition 1, 2, 3, 4 we can prove that since in every step the path and the path environment are correct, therefore in every step we result in exact binding environment and bound environment, so in the termination we have the exact binding environment and bound environment for that action, which satisfy (1) and (2).

4 Security Properties

In this section we will present the security policies for both static and dynamic action calculi in terms of corresponding security properties, and then establish the relationship between the static notation and dynamic one.

We use the result of the above formal flow analysis to establish security properties mainly integrating the idea of [1, 3], which is to ensure that an action preserve the secrecy of data.

Similar to [3], we partition names with its paths defined in an action a into trustworthy T(a) and untrustworthy U(a). We say that the secrecy of data is preserved if an action a never read an untrustworthy name to it and an action a never write to other actions the untrustworthy name, which can be represented by following formal definition.

Definition 8 A subaction *a* in action *b* with occurrence path p is *defended*, iff $(\sigma, \rho, \kappa)=[\![b]\!]p_0\sigma_0\rho_0\kappa_0$, and there is no such $(x, p_x)\in U(a)$ that $p\in\kappa(x, p_x)$, and $\forall x\in exp(a)$, $(x, \rho(x,p))\in T(a)$.

Definition 9 A subaction *a* in action *b* with occurrence path p has *no leakage* iff $(\sigma, \rho, \kappa)=[\![b]\!]p_0\sigma_0\rho_0\kappa_0$, and there is no such *c* of path p' in *b*, $Path(c, a)\cap Path(b, a) \neq \Phi$, that $\forall x\in(U(c) \cap names(c))$, $\sigma(x, p') = p$, and $\forall x\in imp(a)$, $p''\in\kappa(x, p)$, $(x, p'')\in T(d)$, where $Path(d, a)=p''$.

Definition 10 An action *a* preserves secrecy of data iff all the subactions of *a* are defended and have no leakage.

The definitions above are static security notations for action calculi. Next we would like to present dynamic secrecy notation for action calculi. Since the dynamics need to be defined when you are defining a concrete action calculus, the concrete form of the reactive rule are undefined, therefore, we can only give a general result.

Definition 11 An action *a* is *protected* iff whenever $a \searrow b$, and *c* is communicating y to *d* via x in this reaction, we have $x\in T(c)$ and $x\in T(d)$, $y\in T(d)$, where x is a name, *c* and *d* are subaction of *a*,

Lemma 2 If an action *a* preserves secrecy of data then it is protected.

Proof: From the definition 8, 9 and 10, if an action *a* preserves secrecy of data, then we have that every subaction *b* of *a* are defended and have no leakage, which means:

$\forall x\in exp(c)$, $(x, \rho(x, p))\in T(c)$.
$\forall x\in exp(d)$, $(x, \rho(x, p))\in T(d)$.
$\forall x\in imp(a)$, $p''\in\kappa(x, p)$, $(x, p'')\in T(d)$, where $Path(d, a)=p''$
where $(\sigma, \rho, \kappa)=[\![b]\!]p_0\sigma_0\rho_0\kappa_0$.

x is a name, if $a\searrow b$, *c*, *d* are subaction of *a*, and *c* is communicating y to *d* via x in this reaction, then we have:

(1) x must be exported names of *c* and *d*;
(2) y must be imported name of *c* and exported name of *d* satisfying that y in *c* is binding y in *d*;

therefore, we can conclude that $x\in T(c)$ and $x\in T(d)$, $y\in T(d)$. So according to definition 11 we can say that *a* is protected.

5 Related Work and Conclusion

Applying static analysis of formal models in studying security properties of concurrent systems is an active and interesting research topic in formal methods in recent years[1,2].There is a large body of research on information flow control aiming at specifying, verifying and analyzing security.

Secure information flow in programming language received its recent reincarnation. Many researchers have investigated the problem including [12,13,14,15,16], which all use concrete calculi or programming languages as research target, while this paper use action calculi, an abstract model for a class of calculi.

Recently, the use of type systems for information flow is also developed[3, 9, 16].

In [1, 2, 3, 4] data flow or control flow analysis for pi calculus, safe ambients, CML have been presented, and some encouraging results in proving security property are established.

The idea of using information flow analysis techniques for studying the security properties of action calculi arises from that of pi calculus and ambient[1, 2, 3]. We propose information flow analysis for static action calculi, and use it for the validation of rather simple security properties of interactive systems. More complicated security properties will be studied in the future. Because action calculi is a framework for a class of models for interactive system, these work can open a way to study the generality of these properties by comparing the results with those of each concrete model for interactive system.

References

1. C. Bodel, P. Deggan, F. Nielson, and H. Riis Nielson. Static Analysis for the π-calculus with Applications to Security. Available at http://www.di.unipi.it/~chiars/publ-40/BDNN100.ps.
2. C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Control flow analysis for the π-calculus. In Proceedings of CONCUR'98, LNCS 1466, pp84-89. Springer-Verlag, 1998.
3. P. Degano, F. Levi and C. Bodei. Safe Ambients: Control Flow Analysis and Security. In Proceedings of ASIAN '00. LNCS 1961, 199-214. Springer-Verlag.
4. Gasser, K.L.S., Nielson, F., Nielson H.R. Systematic realization of control flow analysis for CML. In Proceedings of ICFP'97, P38-51. ACM press, 1997.
5. Gardner, P.. Closed Action Calculi. Theoretical Computer Science, vol. 228, 1999.
6. Milner, R.. "Action calculi, or concrete action structures", Proc. MFCS Conference, Gdansk, Poland, LNCS, Vol. 711, Springer-Verlag, pp105-121, 1993.
7. Milner R.. Action Calculi IV: Molecular forms, Computer Sciences Department, University of Edinburgh. Draft(1993).
8. Milner R.. " Calculi for Interaction", Acta Inform. 33(8), pp 707-737, 1996.
9. Silvia Crafa, Michele Bugliesi, Giuseppe Castagna. Information Flow Security in Boxed Ambients. Electronic Notes in Theoretical Computer Science 66, No.3(2003.
10. Ying JIN, Chengzhi JIN. Encoding Gamma Calculus in Action Calculus. Journal of Software, Vol. 14, No. 1, January 2003.
11. Ying JIN, Chengzhi JIN. Representing Imperative Language in Higher-order Action Calculus. Journal of Computer Research and Development, Vol. 39, No.10, Oct. 2002.
12. R. Focardi. Analysis and Automatic Detection of Information Flows in Systems and Networks. PhD thesis, Unviersity of Bologna(Italy), 1999.
13. R. Focardi, R. Gorrieri, and F. Martinelli. Information Flow Analysis in a Discrtete Time Process Algebra. In Proceedings of 13th IEEE Computer Security Foundations Workshop(CSFW13), P. Syverson ed., Page 170-184. IEEE CS Press, July 2000.

14. R. Focardi, R. Gorrieri, and R. Segala. A New Definition of Security Properties. In Proceedings of Workshop on Issues in the Theory of Security(WITS'00), University of Geneva, July 2000.
15. Heiko Mantel, Andrei Sabelfeld. A Unifying Approach to Security of Distributed and Multi-Threaded Programs. In Proceedings of the 14th IEEE Computer Security Foundations Workshop, Cape Breton, Nova Scotia, Canada, June 2001.
16. Andrei Sabelfeld, Andrew C. Myers. Language-Based Information-Flow Security. IEEE Journal of Selected Areas in Commnications, Vol. 21, No. 1, January 2003.