Modal Logics for Nominal Transition Systems

Joachim Parrow, Johannes Borgström, Lars-Henrik Eriksson, Ramūnas Gutkovas, Tjark Weber[1]

1 **Uppsala University**
Sweden

Abstract

We define a uniform semantic substrate for a wide variety of process calculi where states and action labels can be from arbitrary nominal sets. A Hennessy-Milner logic for these systems is introduced, and proved adequate for bisimulation equivalence. A main novelty is the use of finitely supported infinite conjunctions. We show how to treat different bisimulation variants such as early, late and open in a systematic way, and make substantial comparisons with related work. The main definitions and theorems have been formalized in Nominal Isabelle.

1998 ACM Subject Classification F.1.1, F.1.2, F.3.1, F.3.2, F.4.1

1 Introduction

Transition systems Transition systems are ubiquitous in models of computing, and specifications to say what may and must happen during executions are often formulated in a modal logic. There is a plethora of different versions of both transition systems and logics, including a variety of higher-level constructs such as updatable data structures, new name generation, alias generation, dynamic topologies for parallel components etc. In this paper we formulate a general framework where such aspects can be treated uniformly, and define accompanying modal logics which are adequate for bisimulation. This is related to, but independent of, our earlier work on psi-calculi [4], which proposes a particular syntax for defining behaviours. The present paper does not depend on any such language, and provides general results for a large class of transition systems.

In any transition system there is a set of *states* $P, Q, \ldots$ representing the configurations a system can reach, and a relation telling how a computation can move between them. Many formalisms, for example all process algebras, define languages for expressing states, but in the present paper we shall make no assumptions about any such syntax.

In systems describing communicating parallel processes the transitions are labelled with *actions* α, β, representing the externally observable effect of the transition. A transition $P \xrightarrow{\alpha} P'$ thus says that in state P the execution can progress to P' while conducting the action α, which is visible to the rest of the world. For example, in CCS these actions are atomic and partitioned into output and input communications. In value-passing calculi the actions can be more complicated, consisting of a channel designation and a value from some data structure to be sent along that channel.

Scope openings With the advent of the pi-calculus [19] an important aspect of transitions was introduced: that of name generation and scope opening. The main idea is that names (i.e., atomic identifiers) can be scoped to represent local resources. They can also be transmitted in actions, to give a parallel entity access to this resource. In the monadic pi-calculus such an action is written $\overline{a}(\nu b)$, to mean that the local name b is exported along the channel a. These names can be subjected to alpha-conversion: if $P \xrightarrow{\overline{a}(\nu b)} P'$ and c is a fresh name then also $P \xrightarrow{\overline{a}(\nu c)} P'\{c/b\}$, where $P'\{c/b\}$ is P' with all bs replaced by cs. Making this idea

fully formal is not entirely trivial and many papers gloss over it. In the polyadic pi-calculus several names can be exported in one action, and in psi-calculi arbitrary data structures may contain local names. In this paper we make no assumptions about how actions are expressed, and just assume that for any action α there is a finite set of names $\mathrm{bn}(\alpha)$, the *binding names*, representing exported names. In our formalization we use nominal sets, an attractive theory to reason about objects depending on names on a high level and in a fully rigorous way.

State predicates The final general components of our transition systems are the *state predicates* ranged over by φ, representing what can be concluded in a given state. For example state predicates can be equality tests of expressions, or connectivity between communication channels. We write $P \vdash \varphi$ to mean that in state P the state predicate φ holds.

A structure with states, transitions, and state predicates as discussed above we call a *nominal transition system*.

Hennessy-Milner Logic Modal logic has been used since the 1970s to describe how facts evolve through computation. We use the popular and general branching time logic known as Hennessy-Milner Logic [15] (HML). Here the idea is that an action modality $\langle\alpha\rangle$ expresses a possibility to perform an action α. If A is a formula then $\langle\alpha\rangle A$ says that it is possible to perform α and reach a state where A holds. With conjunction and negation this gives a powerful logic shown to be *adequate* for bisimulation equivalence: two processes satisfy the same formulas exactly if they are bisimilar. In the general case, conjunction must take an infinite number of operands when the transition systems have states with an infinite number of outgoing transitions. The fully formal treatment of this requires care in ensuring that such infinite conjunctions do not exhaust all names, leaving none available for alpha-conversion. All previous works that have considered this issue properly have used uniformly bounded conjunction, i.e., the set of all names in all conjuncts is finite.

Contributions Our definition of nominal transition systems is very general since we leave open what the states, transitions and predicates are. The only requirement is that transitions satisfy alpha-conversion. A technically important point is that we do not assume the usual *name preservation principle*, that if $P \xrightarrow{\alpha} P'$ then the names occurring in P' must be a subset of those occurring in P and α. This means that the results are applicable to a wide range of calculi. For example, the pi-calculus represents a trivial instance where there are no state predicates. CCS represent an even more trivial instance where bn always returns the empty set. In the fusion calculus and the applied pi-calculus the state contains an environmental part which tells what expressions are equal to what. In the general framework of psi-calculi the states are processes with assertions describing their environments.

We define a modal logic with the $\langle\alpha\rangle$ operator that binds the names in $\mathrm{bn}(\alpha)$, and contains operators for state predicates. In this way we get a logic for an arbitrary nominal transition system such that logical equivalence coincides with bisimilarity. We also show how variants of the logic correspond to late, open and hyperbisimilarity in a uniform way. The main technical difficulty is to ensure that formulas and their alpha-equivalence classes throughout are finitely supported, i.e., only depend on a finite set of names, even in the presence of infinite conjunction. Instead of uniformly bounded conjunction we use the notion of finite support from nominal sets. This results in greater generality and expressiveness. For example, we can now define quantifiers and the next step modalities as derived operators.

Formalization Our main definitions and theorems have been formalized in Nominal Isabelle [27]. This has required significant new ideas to represent data types with infinitary constructors like infinite conjunction and their alpha-equivalence classes. As a result we corrected several details in our formulations and proofs, and now have very high confidence in their correctness. The formalization effort has been substantial, but certainly less than half of the total effort, and we consider it a very worthwhile investment.

Exposition In the following section we provide the necessary background on nominal sets. In Section 3 we present our main definitions and results on nominal transition systems and modal logics. In Section 4 we derive useful operators such as quantifiers and fixpoints, and indicate some practical uses. Section 5 shows how to treat variants of bisimilarity such as late and open in a uniform way, and in Section 6 we compare with related work and demonstrate how our framework can be applied to recover earlier results uniformly. Finally Section 7 concludes with some remarks on the formalization in Nominal Isabelle. All full proofs are contained in the appendix of the technical report [22].

2 Background on nominal sets

Nominal sets [25] is a general theory of objects which depend on names, and in particular formulates the notion of alpha-equivalence when names can be bound. The reader need not know nominal set theory to follow this paper, but some key definitions will make it easier to appreciate our work and we recapitulate them here.

We assume an infinitely countable multi-sorted set of atomic identifiers or *names* $\mathcal{N}$ ranged over by $a, b, \ldots$. A *permutation* is a bijection on names that leaves all but finitely many names invariant. The singleton permutation which swaps names a and b and has no other effect is written $(a\,b)$, and the identity permutation that swaps nothing is written id. Permutations are ranged over by π, π'. The effect of applying a permutation π to an object X is written $\pi \cdot X$. Formally, the permutation action $\cdot$ can be any operation that satisfies $\mathrm{id} \cdot X = X$ and $\pi \cdot (\pi' \cdot X) = (\pi \circ \pi') \cdot X$, but a reader may comfortably think of $\pi \cdot X$ as the object obtained by permuting all names in X according to π.

A set of names N *supports* an object X if for all π that leave all members of N invariant it holds $\pi \cdot X = X$. In other words, if N supports X then names outside N do not matter to X. If a finite set supports X then there is a unique minimal set supporting X, called the *support* of X, written $\mathrm{supp}(X)$, intuitively consisting of exactly the names that matter to X. As an example the set of names textually occurring in a datatype element is the support of that element, and the set of free names is the support of the alpha-equivalence class of the element. Note that in general, the support of a set is not the same as the union of the support of its members. An example is the set of all names; each element has itself as support, but the whole set has empty support since $\pi \cdot \mathcal{N} = \mathcal{N}$ for any π.

We write $a\#X$, pronounced "a is fresh for X", for $a \notin \mathrm{supp}(X)$. The intuition is that if $a\#X$ then X does not depend on a in the sense that a can be replaced with any fresh name without affecting X. If A is a set of names we write $A\#X$ for $\forall a \in A \,.\, a\#X$.

A *nominal set* S is a set with a permutation action such that $X \in S \Rightarrow \pi \cdot X \in S$, and where each member $X \in S$ has finite support. A main point is that then each member has infinitely many fresh names available for alpha-conversion. Similarly, a set of names N supports a function f on a nominal set if for all π that leave N invariant it holds $\pi \cdot f(X) = f(\pi \cdot X)$, and similarly for relations and functions of higher arity. Thus we extend the notion of support to finitely supported functions and relations as the minimal finite support, and

can derive general theorems such as $\mathrm{supp}(f(X)) \subseteq \mathrm{supp}(f) \cup \mathrm{supp}(X)$.

An object that has empty support we call *equivariant*. For example, a unary function f is equivariant if $\pi \cdot f(X) = f(\pi \cdot X)$ for all π, X. The intuition is that an equivariant object does not treat any name special.

3 Nominal transition systems and Hennessy-Milner logic

▶ **Definition 1.** A *nominal transition system* is characterized by the following

- STATES: A nominal set of *states* ranged over by P, Q.
- PRED: A nominal set of *state predicates* ranged over by φ.
- An equivariant binary relation $\vdash$ on STATES and PRED. We write $P \vdash \varphi$ to mean that in state P the state predicate φ holds.
- ACT: A nominal set of *actions* ranged over by α.
- An equivariant function bn from ACT to finite sets of names, which for each α returns a subset of $\mathrm{supp}(\alpha)$, called the *binding names*.
- An equivariant transition relation $\rightarrow$ on states and residuals. A residual is a pair of action and state. For $\rightarrow (P, (\alpha, P'))$ we write $P \xrightarrow{\alpha} P'$. The transition relation must satisfy alpha-conversion of residuals: If $a \in \mathrm{bn}(\alpha)$, $b \# \alpha, P'$ and $P \xrightarrow{\alpha} P'$ then also $P \xrightarrow{(a\,b)\cdot\alpha} (a\,b) \cdot P'$.

▶ **Definition 2.** A *bisimulation* R is a symmetric binary relation on states in a nominal transition system satisfying the following two criteria: $R(P, Q)$ implies

1. *Static implication*: $P \vdash \varphi$ implies $Q \vdash \varphi$.
2. *Simulation*: For all α, P' such that $\mathrm{bn}(\alpha) \# Q$ there exist Q' such that if $P \xrightarrow{\alpha} P'$ then $Q \xrightarrow{\alpha} Q'$ and $R(P', Q')$

We write $P \dot{\sim} Q$ to mean that there exists a bisimulation R such that $R(P, Q)$.

Static implication means that bisimilar states must satisfy the same state predicates; this is reasonable since these can be tested by an observer. The simulation requirement is familiar from the pi-calculus.

▶ Proposition 1. $\dot{\sim}$ is an equivariant equivalence relation.

The minimal HML for nominal transition systems is the following.

▶ **Definition 3.** The nominal set of formulas $\mathcal{A}$ ranged over by A is defined by induction as follows:

$$A \quad ::= \quad \bigwedge_{i \in I} A_i \quad | \quad \neg A \quad | \quad \varphi \quad | \quad \langle \alpha \rangle A$$

Support and name permutation are defined as usual (permutation distributes over all formula constructors). In $\bigwedge_{i \in I} A_i$ it is assumed that the indexing set I has bounded cardinality, by which we mean that $|I| \leq \kappa$ for some fixed infinite cardinal κ at least as large as the cardinality of STATES, ACT and PRED. It is also required that the set of conjuncts $\{A_i \mid i \in I\}$ has finite support; this is then the support of the conjunction. Alpha-equivalent formulas are identified; the only binding construct is in $\langle \alpha \rangle A$ where $\mathrm{bn}(\alpha)$ binds into A.

Compared to previous work there are two main novelties in Definition 3. The first is that we use conjunction of a possibly infinite and finitely supported set of conjuncts. In comparison, the earliest HML for CCS, Hennessy and Milner (1985) [15], uses finite conjunction, meaning that the logic is adequate only for finite branching transition systems. In his subsequent book (1989) [18] Milner admits arbitrary infinite conjunction, disregarding

the danger of running into paradoxes. Abramsky (1991) [3] employs a kind of uniformly bounded conjunction, with a finite set of names that supports all conjuncts, an idea that is also used in the first HML for the pi-calculus (1993) [20]. All subsequent developments follow one of these three approaches. Our main point is that both finite and uniformly bounded conjunction are expressively weak, in that the logic is not adequate for the full range of nominal transition systems, and in that quantifiers over infinite structures are not definable. In contrast, our use of finitely supported sets of conjuncts is adequate for all nominal transition systems (cf. Theorems 6 and 9 below) and admits quantifiers as derived operators (cf. Section 4 below). As a simple example, universal quantification over names $\forall x \in \mathcal{N}.A(x)$ is usually defined to mean that $A(n)$ must hold for all $n \in \mathcal{N}$. We can define this as the (infinite) conjunction of all these $A(n)$. This set of conjuncts is not uniformly bounded if $n \in \mathrm{supp}(A(n))$. But it is supported by $\mathrm{supp}(A)$ since, for any permutation π not affecting $\mathrm{supp}(A)$ we have $\pi \cdot A(n) = A(\pi(n))$ which is also a conjunct; thus the set of conjuncts is unaffected by π.

The second novelty is the use of a nominal set of actions α with binders, and the formal definition of alpha-equivalence. We define it by structural recursion over formulas. Two conjunctions $\bigwedge_{i\in I} A_i$ and $\bigwedge_{i\in I} B_i$ are alpha-equivalent if for every conjunct A_i there is an alpha-equivalent conjunct B_j, and vice versa. The other cases are standard; two formulas $\langle\alpha\rangle A$ and $\langle\beta\rangle B$ are alpha-equivalent if there exists a permutation π, renaming the binding names of α to those of β, such that $\pi \cdot A$ and B are alpha-equivalent, and $\pi \cdot \alpha = \beta$. Moreover, π must leave names that are free in A invariant. The free names in a formula are also defined by structural recursion. Most cases are standard again; a name is free in $\langle\alpha\rangle A$ if it is in $\mathrm{supp}(\alpha)$ or free in A, and not contained in $\mathrm{bn}(\alpha)$. However, the free names in a conjunction are given by the support of its alpha-equivalence class (rather than by the union of free names in all conjuncts). This is analogous to the situation for nominal sets in general, whose support is not necessarily the same as the union of the support of its members. Fortunately, our formalization proves that we need not keep the details of this construction in mind, but can simply identify alpha-equivalent formulas. The notions of free names and support then coincide.

The validity of a formula A for a state P is written $P \models A$ and is defined by recursion over A as follows.

▶ **Definition 4.**

$P \models \bigwedge_{i\in I} A_i$	if for all $i \in I$ it holds that $P \models A_i$
$P \models \neg A$	if not $P \models A$
$P \models \varphi$	if $P \vdash \varphi$
$P \models \langle\alpha\rangle A$	if there exists P' such that $P \xrightarrow{\alpha} P'$ and $P' \models A$

In the last clause we assume that $\langle\alpha\rangle A$ is a representative of its alpha-equivalence class such that $\mathrm{bn}(\alpha)\#P$. It is easy to show that $\models$ is an equivariant relation.

▶ **Definition 5.** Two states P and Q are *logically equivalent*, written $P \dot{=} Q$, if for all A it holds that $P \models A$ iff $Q \models A$

▶ **Theorem 6.** $P \dot{\sim} Q \implies P \dot{=} Q$

The proof is by induction over formulas. The converse result uses the idea of *distinguishing formulas*.

▶ **Definition 7.** A *distinguishing formula* for P and Q is a formula A such that $P \models A$ and not $Q \models A$.

The following lemma says that we can find such a formula where, a bit surprisingly, the support does not depend on Q.

▶ **Lemma 8.** *If $P \not\dot{\sim} Q$ then there exists a distinguishing formula B for P and Q such that* $\mathrm{supp}(B) \subseteq \mathrm{supp}(P)$.

The proof is by direct construction. If $P \not\dot{\sim} Q$ then there exists a distinguishing formula A for P and Q. Let Π_P be the group of finite permutations that leave names in $\mathrm{supp}(P)$ invariant, i.e., $\Pi_P = \{\pi \mid \forall n \in \mathrm{supp}(P).\ \pi(n) = n\}$. Then $\{\pi \cdot A \mid \pi \in \Pi_P\}$ is supported by $\mathrm{supp}(P)$. Since $\models$ is equivariant we have that for all $\pi \in \Pi_P$ it holds $P = \pi \cdot P \models \pi \cdot A$. Let $B = \bigwedge_{\pi \in \Pi_P} \pi \cdot A$, thus $P \models B$ but $Q \not\models B$ since the identity is in Π_P and $Q \not\models A$. Note that B here uses a conjunction which is not uniformly bounded.

▶ **Theorem 9.** $P \mathrel{\dot{=}} Q \implies P \mathrel{\dot{\sim}} Q$

The main idea of the proof is to establish that $\dot{=}$ is a bisimulation. The simulation requirement is by contradiction: Assume that $\dot{=}$ does not satisfy the simulation requirement. Then there exist P, Q, P', α such that $P \mathrel{\dot{=}} Q$ and $P \xrightarrow{\alpha} P'$ and, letting $\mathcal{Q} = \{Q' \mid Q \xrightarrow{\alpha} Q'\}$, for all $Q' \in \mathcal{Q}$ it holds $P' \not\dot{\sim} Q'$. By Lemma 8 we can find a distinguishing formula $B_{Q'}$ for P' and Q' with $\mathrm{supp}(B_{Q'}) \subseteq \mathrm{supp}(P')$. Therefore the formula $B = \bigwedge_{Q' \in \mathcal{Q}} B_{Q'}$ is well-formed with support included in $\mathrm{supp}(P')$. We thus get that $P \models \langle\alpha\rangle B$ but not $Q \models \langle\alpha\rangle B$, contradicting $P \mathrel{\dot{=}} Q$.

This proof of the simulation property is different from other such proofs in the literature. For finite branching transition systems, $\mathcal{Q}$ is finite so finite conjunction is enough to define B. For transition systems with the name preservation property, i.e., that if $P \xrightarrow{\alpha} P'$ then $\mathrm{supp}(P') \subseteq \mathrm{supp}(P) \cup \mathrm{supp}(\alpha)$, uniformly bounded conjunction suffices with common support $\mathrm{supp}(P) \cup \mathrm{supp}(Q) \cup \mathrm{supp}(\alpha)$. Without the name preservation property, we here use a not uniformly bounded conjunction in Lemma 8.

4 Derived formulas

Dual connectives We define logical disjunction $\bigvee_{i \in I} A_i$ in the usual way as $\neg \bigwedge_{i \in I} \neg A_i$, when the indexing set I has bounded cardinality and $\{A_i \mid i \in I\}$ has finite support. A special case is $I = \{1, 2\}$: we then write $A_1 \wedge A_2$ instead of $\bigwedge_{i \in I} A_i$, and dually for $A_1 \vee A_2$. We write $\top$ for the empty conjunction $\bigwedge_{i \in \emptyset}$, and $\bot$ for $\neg\top$. The must modality $[\alpha]A$ is defined as $\neg\langle\alpha\rangle\neg A$, and requires A to hold after every possible α-labelled transition from the current state. For example, $[\alpha](A \wedge B)$ is equivalent to $[\alpha]A \wedge [\alpha]B$, and dually $\langle\alpha\rangle(A \vee B)$ is equivalent to $\langle\alpha\rangle A \vee \langle\alpha\rangle B$.

Quantifiers Let S be any finitely supported set of bounded cardinality and use v to range over members of S. Write $A\{^v/_x\}$ for the substitution of v for x in A, and assume this substitution function is equivariant. Then we define $\forall x \in S\,.\,A$ as $\bigwedge_{v \in S} A\{^v/_x\}$. There is not necessarily a common finite support for the formulas $A\{^v/_x\}$, for example if S is some term algebra over names, but the set $\{A\{^v/_x\} \mid v \in S\}$ has finite support bounded by $\{x\} \cup \mathrm{supp}(S) \cup \mathrm{supp}(A)$. In our examples in Section 6, substitution is defined inductively on the structure of formulas, based on primitive substitution functions for actions and state predicates, avoiding capture and preserving the binding names of actions.

Existential quantification $\exists x \in S\,.\,A$ is defined as the dual $\neg\forall x \in S\,.\,\neg A$. When X is a metavariable used to range over a nominal set $\mathcal{X}$, we simply write X for "$X \in \mathcal{X}$". As an example, $\forall a\,.\,A$ means that the formula $A\{^n/_a\}$ holds for all names $n \in \mathcal{N}$.

New name quantifier The new name quantifier $\text{И}x.A$ intuitively states that $P \models A\{^n/_x\}$ holds where n is a fresh name for P. For example, suppose we have actions of the form $a\,b$ for input, and $\overline{a}\,b$ for output where a and b are free names, then the formula $\text{И}x.[a\,x]\langle\overline{b}\,x\rangle\top$ expresses that whenever a process inputs a fresh name x on channel a, it has to be able to output that name on channel b. If the name received is not fresh (i.e., already present in P) then P is not required to do anything. Therefore this formula is weaker than $\forall x\,.\,[a\,x]\langle\overline{b}\,x\rangle\top$.

To define this formally we use name permutation rather than substitution. Since A and P have finite support, if $P \models (x\,n)\cdot A$ holds for some n fresh for P, by equivariance it also holds for almost all n, i.e., all but finitely many n. Conversely, if it holds for almost all n, it must hold for some $n\#\,\mathrm{supp}(P)$. Therefore $\text{И}x$ is often pronounced "for almost all x". In other words, $P \models \text{И}x.A$ holds if $\{x \mid P \models A(x)\}$ is a cofinite set of names [25, Definition 3.8]. Letting $\textsc{cof} = \{S \subseteq \mathcal{N} \mid \mathcal{N} \setminus S \text{ is finite}\}$ we thus encode $\text{И}x.A$ as $\bigvee_{S\in\textsc{cof}} \bigwedge_{n\in S}(x\,n)\cdot A$. This formula states there is a cofinite set of names such that for all of them A holds. The support of $\bigwedge_{n\in S}(x\,n)\cdot A$ is bounded by $(\mathcal{N}\setminus S)\cup\mathrm{supp}(A)$ where $S \in \textsc{cof}$, and the support of the encoding $\bigvee_{S\in\textsc{cof}} \bigwedge_{n\in S}(x\,n)\cdot A$ is bounded by $\mathrm{supp}(A)$.

Next step We generalise the action modality to sets of actions in the following way. If T is a finitely supported set of actions such that $\mathrm{bn}(\alpha)\#A$ for all $\alpha \in T$, we write $\langle T\rangle A$ for $\bigvee_{\alpha\in T}\langle\alpha\rangle A$. The support of the set $\{\langle\alpha\rangle A \mid \alpha \in T\}$ is bounded by $\mathrm{supp}(T)\cup\mathrm{supp}(A)$ and thus finite. Dually, we write $[T]A$ for $\neg\langle T\rangle\neg A$, denoting that A holds after all transitions with actions in T.

To encode the next-step modality, let $\textsc{act}_A = \{\alpha \mid \mathrm{bn}(\alpha)\#A\}$. Note that $\mathrm{supp}(\textsc{act}_A) \subseteq \mathrm{supp}(A)$ is finite. We write $\langle\,\rangle A$ for $\langle\textsc{act}_A\rangle A$, meaning that we can make some (non-capturing) transition to a state where A holds. As an example, $\langle\,\rangle\top$ means that the current state is not deadlocked. The dual modality $[\,]A = \neg\langle\,\rangle\neg A$ means that A holds after every transition from the current state. Larsen [17] uses the same approach to define next-step operators in HML, though his version is less expressive since he uses a finite action set to define the next-step modality.

Fixpoints Fixpoint operators are a way to introduce recursion into a logic. For example, they can be used to concisely express safety and liveness properties of a transition system, where by safety we mean that some invariant holds for all reachable states, and by liveness that some property will eventually hold. Kozen (1983) [16] introduced the least ($\mu X.A$) and the greatest ($\nu X.A$) fixpoints in modal logic. Intuitively, the least fixpoint states a property that holds for states of a finite path, while the greatest holds for states of an infinite path.

▶ **Theorem 10.** *The least and greatest fixpoint operators are expressible in our HML.*

For the full proofs and definitions, see the appendix of [22]. The idea is to start with an extended language with the forms $\mu X.A$ and X, where X ranges over a countable set of variables and all occurrences of X in A are in the scope of an even number of negations. Write $A(B)$ for the capture-avoiding substitution of B for X in A, and let $A^0(B) = B$ and $A^{i+1}(B) = A(A^i(B))$. Then the encoding of a least fixpoint $\mu X.A$ is $\bigvee_{i\in\mathbb{N}} A^i(\bot)$, given that fixpoints have been recursively expanded in A. The disjunction has finite support $\mathrm{supp}(A)$, since substitution is equivariant. When interpreting formulas as elements of the power-set lattice of $\textsc{states}$, this encoding yields a fixpoint of $A(\cdot)$: the sequence of formulas $A^i(\bot)$ yields an approximation from below. We define the greatest fixpoint operator $\nu X.A$ in terms of the least as $\neg\mu X.\neg A(\neg X)$.

Using the greatest fixpoint operator we can state global invariants: $\nu X.[\alpha]X\wedge A$ expresses that A holds along all paths labelled with α. Temporal operators such as eventually can also

be encoded using the least fixpoint operator: the formula $\mu X.\langle\alpha\rangle X \vee A$ states that eventually A holds along some path labelled with α. We can freely mix the fixpoint operators to obtain formulas like $\nu X.[\alpha]X \wedge (\mu Y.\langle\beta\rangle Y \vee A)$ which means that for each state along any path labelled with α, a state where A holds is reachable along a path labelled with β. Formulas with mixed fixpoint combinators are very expressive, and with the next operator they can encode the branching-time logic CTL* [11].

5 Logics for variants of bisimilarity

The bisimilarity of Section 3 is of the early kind: any substitutive effect of an input (typically replacing a variable with the value received) must have manifested already in the action corresponding to the input, since we apply no substitution to the target state. Alternative treatments of substitutions include late-, open- and hyperbisimilarity, where the input action instead contains the variable to be replaced, and there are different ways to make sure that bisimulations are preserved by relevant substitutions.

In our definition of nominal transition systems there are no particular input variables in the states or in the actions, and thus no a priori concept of "substitution". We therefore choose to formulate the alternatives using so called effect functions. An *effect* is simply a finitely supported function from states to states. For example, in the monadic pi-calculus the effects would be the functions replacing one name by another. In a value-passing calculus the effects would be substitutions of values for variables. In the psi-calculi framework the effects would be sequences of parallel substitutions. Our definitions and results are applicable to any of these; our only requirement is that the effects form a nominal set which we designate by $\mathcal{F}$. Variants of bisimilarity then correspond to requiring continuation after various effects. For example, if the action contains an input variable x then the effects appropriate for late bisimilarity would be substitutions for x.

We will formulate these variants as F/L-*bisimilarity*, where F (for *first*) represents the set of effects that must be observed before following a transition, and L (for *later*) is a function that represents how this set F changes depending on the action of a transition, i.e., $L(\alpha, F)$ is the set of effects that must follow the action α if the previous effect set was F. In the following let $\mathcal{P}_{\text{fs}}(\mathcal{F})$ ranged over by F be the finitely supported subsets of $\mathcal{F}$, and L range over equivariant functions from actions and $\mathcal{P}_{\text{fs}}(\mathcal{F})$ to $\mathcal{P}_{\text{fs}}(\mathcal{F})$.

▶ **Definition 11.** An L-*bisimulation* where $L : \text{ACT} \times \mathcal{P}_{\text{fs}}(\mathcal{F}) \to \mathcal{P}_{\text{fs}}(\mathcal{F})$ is a $\mathcal{P}_{\text{fs}}(\mathcal{F})$-indexed family of symmetric binary relations on states satisfying the following:

If $R_F(P, Q)$ then:

1. *Static implication*: for all $f \in F$ it holds that $f(P) \vdash \varphi$ implies $f(Q) \vdash \varphi$.
2. *Simulation*: For all $f \in F$ and α, P' such that $\text{bn}(\alpha)\#f(Q)$ there exist Q' such that

$$\text{if } f(P) \xrightarrow{\alpha} P' \text{ then } f(Q) \xrightarrow{\alpha} Q' \text{ and } R_{L(\alpha,F)}(P', Q')$$

We write $P \overset{F/L}{\sim} Q$, called F/L-bisimilarity, to mean that there exists an L-bisimulation R such that $R_F(P, Q)$.

Most strong bisimulation varieties can be formulated as F/L-bismilarity. Write $\text{id}_{\text{STATES}}$ for the identity function on states, ID for the singleton set $\{\text{id}_{\text{STATES}}\}$ and all_{ID} for the constant function $\lambda(\alpha, F).\text{ID}$.

- *Early bisimilarity*, precisely as defined in Definition 2, is ID / all_{ID}-bisimilarity.
- *Early equivalence*, i.e., early bisimilarity for all possible effects, is $\mathcal{F}$ / all_{ID}-bisimilarity.

- *Late bisimilarity* is ID / L-bisimilarity, where $L(\alpha, F)$ yields the effects that represent substitutions for variables in input actions α (and ID for other actions).
- *Late equivalence* is similarly $\mathcal{F}$ / L-bisimilarity.
- *Open bisimilarity* is $\mathcal{F}$ / L-bisimilarity where $L(\alpha, F)$ is the set F minus all effects that change bound output names in α.
- *Hyperbisimilarity* is $\mathcal{F}$ / $\lambda(\alpha, F).\mathcal{F}$-bisimilarity.

All of the above are generalizations of known and well-studied definitions. The original value-passing variant of CCS [18] uses early bisimilarity. The original bisimilarity for the pi-calculus is of the late kind [19], where it also was noted that late equivalence is the corresponding congruence. Early bisimilarity and equivalence and open bisimilarity for the pi-calculus were introduced in 1993 [20, 26], and hyperbisimilarity for the fusion calculus in 1998 [23].

In view of this we only need to provide a modal logic adequate for F/L-bisimilarity; it can then immediately be specialized to all of the above variants. For this we introduce a new kind of logical operator as follows.

▶ **Definition 12.** For each $f \in \mathcal{F}$ the logical unary *effect consequence* operator $\langle f \rangle$ has the definition

$$P \models \langle f \rangle A \quad \text{if} \quad f(P) \models A$$

Thus the formula $\langle f \rangle A$ means that A holds if the effect f is applied to the state. Note that by definition this distributes over conjunction and negation, e.g. $P \models \neg\langle f \rangle A$ iff $P \models \langle f \rangle \neg A$ iff not $f(P) \models A$ etc. The effect consequence operator is similar in spirit to the action modalities: both $\langle f \rangle A$ and $\langle \alpha \rangle A$ assert that something (an effect or action) must be possible and that A holds afterwards. Indeed, effects can be viewed as a special case of transitions (as formalised in Definition 16 below) which is why we give the operators a common syntactic appearance.

Now define the formulas that can directly use effects from F and after actions use effects according to L, ranged over by $A^{F/L}$, in the following way:

▶ **Definition 13.** Given L as in Definition 11, for all $F \in \mathcal{P}_{\text{fs}}(\mathcal{F})$ define $\mathcal{A}^{F/L}$ as the set of formulas given by the mutually recursive definitions:

$$A^{F/L} \quad ::= \quad \bigwedge_{i \in I} A_i^{F/L} \quad | \quad \neg A^{F/L} \quad | \quad \langle f \rangle \varphi \quad | \quad \langle f \rangle \langle \alpha \rangle A^{L(\alpha,F)/L}$$

where we require $f \in F$ and that the conjunction has bounded cardinality and finite support.

Let $P \overset{F/L}{=} Q$ mean that P and Q satisfy the same formulas in $\mathcal{A}^{F/L}$.

▶ **Theorem 14.** $P \overset{F/L}{\sim} Q \quad \Leftrightarrow \quad P \overset{F/L}{=} Q$

Proof: The direction $\Rightarrow$ is a generalization of Theorem 6. The other direction is a generalization of Theorem 9: we prove that $\overset{F/L}{=}$ is an F/L-bisimulation. It needs a variant of Lemma 8:

▶ **Lemma 15.** *If $A \in \mathcal{A}^{F/L}$ is a distinguishing formula for P and Q, then there exists a distinguishing formula $B \in \mathcal{A}^{F/L}$ for P and Q such that* $\text{supp}(B) \subseteq \text{supp}(P, F)$.

The proof is an easy generalisation of Lemma 8.

An alternative to the effect consequence operators is to transform the transition system such that standard (early) bisimulation on the transforms coincides with F/L-bisimilarity. The idea is to let the effect function be part of the transition relation, thus $f(P) = P'$ becomes $P \xrightarrow{f} P'$.

▶ **Definition 16.** Assume $\mathcal{F}$ and L as above. The *L-transform* of a nominal transition system $\mathbf{T}$ is a nominal transition system where:

- The states are of the form $\textsc{ac}(F, f(P))$ and $\textsc{ef}(F, P)$, for $f \in F \in \mathcal{P}_{\text{fs}}(\mathcal{F})$ and states P of $\mathbf{T}$. The intuition is that states of kind AC can perform ordinary actions, and states of kind EF can commit effects.
- The state predicates are those of $\mathbf{T}$.
- $\textsc{ac}(F, P) \vdash \varphi$ if in $\mathbf{T}$ it holds $P \vdash \varphi$, and $\textsc{ef}(F, P) \vdash \varphi$ never holds.
- The actions are the actions of $\mathbf{T}$ and the effects in $\mathcal{F}$.
- bn is as in $\mathbf{T}$, and additionally $\text{bn}(f) = \emptyset$ for $f \in \mathcal{F}$.
- The transitions are of two kinds. If in $\mathbf{T}$ it holds $P \xrightarrow{\alpha} P'$, then there is a transition $\textsc{ac}(F, P) \xrightarrow{\alpha} \textsc{ef}(L(\alpha, F), P')$. And for each $f \in F$ it holds $\textsc{ef}(F, P) \xrightarrow{f} \textsc{ac}(F, f(P))$.

▶ **Theorem 17.** *$P \overset{F/L}{\sim} Q$ in $\mathbf{T}$ if and only if $\textsc{ef}(F, P) \dot{\sim} \textsc{ef}(F, Q)$ in the L-transform of $\mathbf{T}$.*

The proof idea is that from an F/L-bisimulation in $\mathbf{T}$ it is easy to construct an (ordinary) bisimulation in the L-transform of $\mathbf{T}$, and vice versa. A direct consequence is that $P \overset{F/L}{\sim} Q$ iff $\textsc{ef}(F, P) \doteq \textsc{ef}(F, Q)$ in the L-transform of $\mathbf{T}$. Here the actions in the logic would include effects $f \in \mathcal{F}$.

6 Related work and examples

In this first part of this section we discuss other modal logics for process calculi, with a focus on how their constructors can be captured by finitely supported conjunction in our HML. This comparison is by necessity somewhat informal; a fully formal correspondence would fail to hold in many cases due to differences in the conjunction operator of the logic (finite, uniformly bounded or unbounded vs. bounded support). In the later part of this section, we obtain novel, adequate HMLs for more recent process calculi.

HML for CCS The first published HML is Hennessy and Milner (1985) [15]. They use finite (binary) conjunction with the assumption of image-finiteness for ordinary CCS. The same goes for the value-passing calculus and logic by Hennessy and Liu (1995) [14], where image-finiteness is due to a late semantics and the logic contains quantification over data values. A similar idea and argument is in a logic for LOTOS by Calder et al. (2002) [8], though that only considers stratified bisimilarity up to ω.

Hennessy and Liu's value-passing calculus is based on abstractions $(x)P$ and concretions (v, P) where v is drawn from a set of values. To encode the modalities of their logic in ours, we add effects $\text{id}_{\text{STATES}}$ and $?v$, with $?v((x)P) = P\{{}^{v}/_{x}\}$, and transitions $(v, P) \xrightarrow{!v} P$. Letting $L(a?, \_) = \{?v \mid v \in \text{values}\}$ and $L(\alpha, \_) = \{\text{id}_{\text{STATES}}\}$ otherwise, late bisimilarity is $\{\text{id}_{\text{STATES}}\}/L$-bisimilarity as defined in Section 5. We can then encode their universal quantifier $\forall x.A$ as $\bigwedge_v \langle ?v\rangle A\{{}^{v}/_{x}\}$, which has support $\text{supp}(A)\backslash\{x\}$, and their output modality $\langle c!x\rangle A$ as $\langle c!\rangle \bigvee_v \langle !v\rangle A\{{}^{v}/_{x}\}$, with support $\{c\} \cup (\text{supp}(A) \setminus \{x\})$.

An infinitary HML for CCS is discussed in Milner's book (1989) [18], where also the process syntax contains infinite summation. There are no restrictions on the indexing sets

and no discussion about how this can exhaust all names. The adequacy theorem is proved by stratifying bisimilarity and using transfinite induction over all ordinals, where the successor step basically is the contraposition of the argument in Theorem 9, though without any consideration of finite support. A more rigorous treatment of the same ideas is by Abramsky (1991) [3] where uniformly bounded conjunction is used throughout.

Pi-calculus The first HML for the pi-calculus is by Milner et al. (1993) [20], where infinite conjunction is used in the early semantics and conjunctions are restricted to use a finite set of free names. The adequacy proof is of the same structure as in this paper. The logic defined in this paper, applied to the pi-calculus transition system omitting bound input actions $x(y)$, contains the logic $\mathcal{F}$ of Milner et al., or the equipotent logic $\mathcal{FM}$ if we take the set of name matchings $[a = b]$ as state predicates.

Spi Calculus Frendrup et al. (2002) [12] provide three Hennessy-Milner logics for the spi calculus [2]. The action modalities in Frendrup's logic only use parts of the labels: on process output, the modality $\langle\overline{a}\rangle$ tests only the channel used. On process input, the modality $\langle a\xi\rangle$ describes how the observer σ computed the received message $M = \mathbf{e}(\xi\sigma)$, where ξ is an expression that may contain decryptions and projections, and $\mathrm{supp}(\xi) \setminus \mathrm{dom}(\sigma)$ is fresh for P and σ. Simplifying the labels of the transition system to τ and the aforementioned $\overline{a}$ and $a\xi$ labels, our minimal HML applied to the particular nominal transition system of the spi calculus has the same modalities as the logic $\mathcal{F}$ of Frendrup et al., although the latter uses infinite conjunction without any mechanism to prevent formulas from exhausting all names, leaving none available for alpha-conversion. Thus their notion of substitution is not formally well defined.

Their logic $\mathcal{EM}$ replaces the simple input modality by an early input modality $\langle\underline{a}(x)\rangle^E A$, which (after a minor manipulation of the input labels) can be encoded as the conjunction $\bigwedge_\xi \langle a\xi\rangle A\{^\xi/_x\}$, which has support $\mathrm{supp}(A) \setminus \{x\}$. We do not consider their logic $\mathcal{LM}$ that uses a late input modality, since its application relies on sets that do not have finite support [12, Theorem 6.12], which are not meaningful in nominal logic.

Applied Pi-calculus A more recent work is a logic by Pedersen (2006) [24] for the applied pi-calculus [1], where the adequacy theorem uses image-finiteness of the semantics in the contradiction argument. The logic contains atomic formulae for equality in the frame of a process, corresponding to our state predicates. The main difference to our logic is an early input modality and a quantifier $\exists x$.

Their early input modality $\langle\underline{a}(x)\rangle A$ can be straightforwardly encoded as the conjunction $\bigwedge_M \langle\underline{a}M\rangle A\{^M/_x\}$, with support $\{a\} \cup (\mathrm{supp}(A) \setminus \{x\})$. For the existential quantifier, there is a requirement that the received term M can be computed from the current knowledge available to an observer of the process, which we here write $M \in \mathcal{S}(P)$. We add actions M/x with $\mathrm{bn}(M/x) = x$ and transitions $P \xrightarrow{M/x} P \mid \{^M/_x\}$ if $M \in \mathcal{S}(P)$ and $x\#P$. We can then encode $\exists x.A$ as $\bigvee_M \langle M/x\rangle A$, which has support $\mathrm{supp}(A) \setminus \{x\}$.

Fusion calculus In an HML for the fusion calculus by Haugstad et al. (2006) [13] the fusions (i.e., equality relations on names) are action labels φ. The corresponding modal operator $\langle\varphi\rangle A$ has the semantics that the formula A must be satisfied for all substitutive effects of φ (intuitively, substitutions that map each name to a fixed representative for its equivalence class). By making the substitutive effects of fusion actions visible in the transition system, we can encode this modal operator. Their adequacy theorem uses the contradiction argument

with infinite conjunction, with no argument about finiteness of names for the distinguishing formula.

Nominal transition systems De Nicola and Loreti (2008) [10] define a general format for nominal transition systems and an associated modal logic, that is adequate for image-finite transition systems only and uses several different modalities for name revelation and resource consumption. In contrast, we seek a small and expressive HML for general nominal transition systems. Indeed, the logic of De Nicola and Loreto can be seen as a special case of ours: their different transition systems can be merged into a single one, and we can encode their quantifiers and fixpoint operator as described in Section 4. Nominal SOS of Cimini et al. (2012) [9] is also a special case of nominal transition systems.

In each of the final two examples below, no HML has to our knowledge yet been proposed, and we immediately obtain one by instantiating the logic in the present paper.

Concurrent constraint pi calculus The concurrent constraint pi calculus (CC-pi) by Buscemi and Montanari (2007) [6] extends the explicit fusion calculus [28] with a more general notion of constraint stores c. The reference equivalence for CC-pi is open bisimulation [7] (closely corresponding to hyperbisimulation in the fusion calculus [23]), which differs from labelled bisimulation in two ways: First, two equivalent processes must be equivalent under all store extensions. To encode this, we let the effects $\mathcal{F}$ be the set of constraint stores c different from 0, and let $c(P) = c \mid P$. Second, when simulating a labelled transition $P \xrightarrow{\alpha} P'$, the simulating process Q can use any transition $Q \xrightarrow{\beta} Q'$ with an equivalent label, as given by a state predicate $\alpha = \beta$. As an example, if $\alpha = \overline{a}\langle x\rangle$ is a free output label then $P \vdash \alpha = \beta$ iff $\beta = \overline{b}\langle y\rangle$ where $P \vdash a = b$ and $P \vdash x = y$. To encode this, we transform the labels of the transition system by replacing them with their equivalence classes, i.e., $P \xrightarrow{\alpha} P'$ becomes $P \xrightarrow{[\alpha]_P} P'$ where $\beta \in [\alpha]_P$ iff $P \vdash \beta = \alpha$. Hyperbisimilarity (Definition 11) on this transition system then corresponds to open bisimilarity, and the modal logic defined in Section 5 is adequate.

Psi-calculi In psi-calculi by Bengtson et al. (2011) [4], the labelled transitions take the form $\Psi \triangleright P \xrightarrow{\alpha} P'$, where the assertion environment Ψ is unchanged after the step. We model this as a nominal transition system by letting the set of states be pairs (Ψ, P) of assertion environments and processes, and define the transition relation by $(\Psi, P) \xrightarrow{\alpha} (\Psi, P')$ if $\Psi \triangleright P \xrightarrow{\alpha} P'$. The notion of bisimulation used with psi-calculi also uses an assertion environment and is required to be closed under environment extension, i.e., if $\Psi \triangleright P \sim Q$, then $\Psi \otimes \Psi' \triangleright P \sim Q$ for all Ψ'. We let the effects $\mathcal{F}$ be the set of assertions, and define $\Psi((\Psi', P)) = (\Psi \otimes \Psi', P)$. Hyperbisimilarity on this transition system then subsumes the standard psi-calculi bisimilarity, and the modal logic defined in Section 5 is adequate.

7 Conclusion

We have given a general account of transition systems and Hennessy-Milner Logic using nominal sets. The advantage of our approach is that it is more expressive than previous work. We allow infinite conjunctions that are not uniformly bounded, meaning that we can encode e.g. quantifiers and the next-step operator. We have given ample examples of how the definition captures different variants of bisimilarity and how it relates to many different versions of HML in the literature.

We have formalized the results of Section 3, including Theorems 6 and 9, using Nominal Isabelle [27].[1] Nominal Isabelle is an implementation of nominal logic in Isabelle/HOL [21], a popular interactive proof assistant for higher-order logic. It adds convenient specification mechanisms for, and automation to reason about, datatypes with binders.

However, Nominal Isabelle does not directly support infinitely branching datatypes. Therefore, the mechanization of formulas (Definition 3) was challenging. We construct formulas from first principles in higher-order logic, by defining an inductive datatype of *raw* formulas (where alpha-equivalent raw formulas are *not* identified). The datatype constructor for conjunction recurses through sets of raw formulas of bounded cardinality, a feature made possible only by a recent re-implementation of Isabelle/HOL's datatype package [5].

We then define alpha-equivalence of raw formulas. For finitely branching datatypes, alpha-equivalence is based on a notion of free variables. Here, to obtain the correct notion of free variables of a conjunction, we define alpha-equivalence and free variables via mutual recursion. This necessitates a fairly involved termination proof. (All recursive functions in Isabelle/HOL must be terminating.) To obtain formulas, we quotient raw formulas by alpha-equivalence, and finally carve out the subtype of all terms that can be constructed from finitely supported ones. We then prove important lemmas; for instance, a strong induction principle for formulas that allows the bound names in $\langle\alpha\rangle A$ to be chosen fresh for any finitely supported context.

Our development, which in total consists of about 2700 lines of Isabelle definitions and proofs, generalizes the constructions that Nominal Isabelle performs for finitely branching datatypes to a type with infinite branching. To our knowledge, this is the first mechanization of an infinitely branching nominal datatype in a proof assistant.

Acknowledgements

We thank Andrew Pitts for enlightening discussions on nominal datatypes with infinitary constructors, and Dmitriy Traytel for providing a formalization of cardinality-bounded sets.

References

1 Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of POPL '01*, pages 104–115. ACM, January 2001.

2 Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The Spi calculus. *Journal of Information and Computation*, 148(1):1–70, 1999.

3 Samson Abramsky. A domain equation for bisimulation. *Journal of Information and Computation*, 92(2):161–218, 1991.

4 Jesper Bengtson, Magnus Johansson, Joachim Parrow, and Björn Victor. Psi-calculi: a framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science*, 7(1), 2011.

5 Jasmin Christian Blanchette, Johannes Hölzl, Andreas Lochbihler, Lorenz Panny, Andrei Popescu, and Dmitriy Traytel. Truly modular (co)datatypes for Isabelle/HOL. In Gerwin Klein and Ruben Gamboa, editors, *Proceedings of ITP 2014*, volume 8558 of *LNCS*, pages 93–110. Springer, 2014.

6 Maria Grazia Buscemi and Ugo Montanari. CC-Pi: A constraint-based language for specifying service level agreements. In Rocco De Nicola, editor, *Proceedings of ESOP 2007*, volume 4421 of *LNCS*, pages 18–32. Springer, 2007.

[1] Our Isabelle theories are available at `https://github.com/tjark/ML-for-NTS`.

7 Maria Grazia Buscemi and Ugo Montanari. Open bisimulation for the concurrent constraint pi-calculus. In Sophia Drossopoulou, editor, *Proceedings of ESOP 2008*, volume 4960 of *LNCS*, pages 254–268. Springer, 2008.

8 Muffy Calder, Savi Maharaj, and Carron Shankland. A modal logic for full LOTOS based on symbolic transition systems. *The Computer Journal*, 45(1):55–61, 2002.

9 Matteo Cimini, Mohammad Reza Mousavi, Michel A. Reniers, and Murdoch J. Gabbay. Nominal SOS. *Electron. Notes Theor. Comput. Sci.*, 286:103–116, September 2012.

10 Rocco De Nicola and Michele Loreti. Multiple-labelled transition systems for nominal calculi and their logics. *Mathematical Structures in Computer Science*, 18(1):107–143, 2008.

11 E. Allen Emerson. Model checking and the mu-calculus. In *DIMACS Series in Discrete Mathematics*, pages 185–214. American Mathematical Society, 1997.

12 Ulrik Frendrup, Hans Hüttel, and Jesper Nyholm Jensen. Modal logics for cryptographic processes. *Electr. Notes Theor. Comput. Sci.*, 68(2):124–141, 2002.

13 Arild Martin Møller Haugstad, Anders Franz Terkelsen, and Thomas Vindum. A modal logic for the fusion calculus. Unpublished, University of Aalborg, http://vbn.aau.dk/ws/files/61067487/1149104946.pdf, 2006.

14 Matthew Hennessy and Xinxin Liu. A modal logic for message passing processes. *Acta Informatica*, 32(4):375–393, 1995.

15 Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

16 Dexter Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27(3):333 – 354, 1983. Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer 1982.

17 Kim G. Larsen. Proof systems for Hennessy-Milner logic with recursion. In M. Dauchet and M. Nivat, editors, *Proceedings of CAAP '88*, volume 299 of *LNCS*, pages 215–230. Springer, 1988.

18 Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

19 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Inf. Comput.*, 100(1):1–40, 1992.

20 Robin Milner, Joachim Parrow, and David Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114(1):149 – 171, 1993.

21 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

22 Joachim Parrow, Johannes Borgström, Lars-Henrik Eriksson, Ramūnas Gutkovas, and Tjark Weber. Modal logics for nominal transition systems. Technical Report 2015-021, Department of Information Technology, Uppsala University, June 2015.

23 Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proceedings of LICS 1998*, pages 176–185. IEEE Computer Society Press, 1998.

24 Michael Pedersen. Logics for the applied pi calculus. Master's thesis, Aalborg University, 2006. BRICS RS-06-19.

25 Andrew M. Pitts. *Nominal Sets*. Cambridge University Press, 2013. Cambridge Books Online.

26 Davide Sangiorgi. A theory of bisimulation for the π-calculus. In Eike Best, editor, *Proceedings of CONCUR '93*, volume 715 of *LNCS*, pages 127–142. Springer, 1993.

27 Christian Urban and Cezary Kaliszyk. General bindings and alpha-equivalence in Nominal Isabelle. *Logical Methods in Computer Science*, 8(2), 2012.

28 Lucian Wischik and Philippa Gardner. Explicit fusions. *Theoretical Computer Science*, 304(3):606–630, 2005.