Insegnamento di

Foundation of Computing

Pagina del corso: http://pages.di.unipi.it/montanari/FOC.html

Note di

History Dependent Automata

a cura di
Ugo Montanari
Dipartimento di Informatica
Università degli Studi di Pisa
ugo@di.unipi.it
Named Graphs
and HD Automata
for Network-Conscious $\pi$-Calculus

Ugo Montanari
Dipartimento di Informatica
Università di Pisa

Joint work with Matteo Sammartino (and many others)
Roadmap

- History Dependent automata
  - No symmetries
  - Spans of open maps
  - With symmetries
  - For verification
- Named sets vs. presheaves
- Network Conscious $\pi$-calculus
- Named graphs
- Conclusion
Operational Models with Resource Generation

- Generation of fresh resources is a basic operation in most distributed systems
  - Sessions, objects, keys, storage, links...

- We need models whose states are enriched with names
- We should be able to allocate, and possibly deallocate names
- Often more general kinds of resources
Roadmap

- History Dependent automata
  - No symmetries
  - Spans of open maps
  - With symmetries
  - For verification
- Named sets vs. presheaves
- Network Conscious $\pi$-calculus
- Named graphs
- Conclusion
HD-Automata: Structure of Transitions

Local names \( \#1 \) \( \#2 \) \( \vdots \) \( \#n \)

Observable names

Local names \( \#1 \) \( \#2 \) \( \vdots \) \( \#m \)

Injective name correspondence

\[ \sigma: [m] \rightarrow [n] \cup \{\ast\} \]
Definition 7 (location automaton). A location automaton is a tuple \( A = \langle Q, w, \rightarrow, q_0 \rangle \) where:

- \( Q \) is a set of states;
- \( w : Q \rightarrow 2^{Loc} \) associates to each state a finite set of locations;
- \( \rightarrow \) is a set of transitions; each transition has the form \( q \xrightarrow{a \mid l \mid \sigma} q' \) (visible transition) or the form \( q \xrightarrow{\tau \mid \sigma} q' \) (invisible transition), where:
  - \( q, q' \in Q \) are the source and target states;
  - \( l \in w(q) \) is the location of the transition;
  - \( \sigma : w(q') \hookrightarrow w(q) \cup \{ \ast \} \) (\( \sigma : w(q') \hookrightarrow w(q) \) for an invisible transition) is the injective (inverse) renaming corresponding to the transition; the newly created location is denoted with the special mark \( \ast \notin \text{Loc} \);
  - \( q_0 \in Q \) is the initial state; we require that \( w(q_0) = \{ l \} \) for some \( l \in \text{Loc} \).

- Ugo Montanari, Marco Pistore, Daniel Yankelevich: Efficient Minimization up to Location Equivalence. ESOP 1996.
Definition 8 (la-bisimulation). Two location automata $A$ and $B$ are location-automaton bisimilar, written $A \approx_{la} B$, if there is some set $\mathcal{R}$ of triples, called la-bisimulation, such that:

- if $\langle p, \delta, q \rangle \in \mathcal{R}$ then $p \in Q_A$, $q \in Q_B$ and $\delta : w_A(p) \rightarrow w_B(q)$ is a partial bijection;
- $\langle q_{0A}, \delta_0, q_{0B} \rangle \in \mathcal{R}$, where $\delta_0$ maps the location associated to $q_{0A}$ to the location associated to $q_{0B}$;
- for each $\frac{a}{l} \frac{\sigma}{\delta(l)}$ in $A$ implies $\langle p', \delta', q' \rangle \in \mathcal{R}$ and $\delta'(m) = n$ such that $\langle p', \delta', q' \rangle \in \mathcal{R}$ and $\delta'(m) = n$.

- $\sigma(m) = \ast = \rho(n)$ or $\delta(\sigma(m)) = \rho(n)$;

- Ugo Montanari, Marco Pistore, Daniel Yankelevich: Efficient Minimization up to Location Equivalence. ESOP 1996.
The Zoo of History Dependent Automata (no Symmetries)

No minimal automaton

\(\pi\)-calculus

- Ugo Montanari and Marco Pistore, Checking Bisimilarity for Finitary pi-calculus, CONCUR'95.

Causality


Causality for Petri nets. The name HD comes from sets of events as shorthands for the process of HPB containing the causes of future actions


Asynchronous \(\pi\)-calculus

- Ugo Montanari, Marco Pistore, Finite State Verification for the Asynchronous pi-Calculus. TACAS 1999

Causality for contextual Petri nets.

- Paolo Baldan, Andrea Corradini, Ugo Montanari, History Preserving Bisimulation for Contextual Nets. WADT 1999

Causality for graph grammars


\[\text{Dagstuhl Seminar Number 13422, October 13-16, 2013 - Nominal Computation Theory}
\text{Ugo Montanari, Named Graphs and HD Automata for Network-Conscious }\pi\text{-Calculus}\]
Roadmap

- History Dependent automata
  - No symmetries
  - Spans of open maps
  - With symmetries
  - For verification
- Named sets vs. presheaves
- Network Conscious $\pi$-calculus
- Named graphs
- Conclusion
Bisimilarity Via Spans of Open Maps, I

- An *agent category* with agents \(M, N\) as objects and arrows \(m, x, y\)
- An *observation subcategory* with observations \(X, Y\) as objects and arrows \(f\)
- An arrow \(m\) is an open map if for every commuting square (a) there is a commuting diagonal \(y'\) (b).
- Agents \(M_1\) and \(M_2\) are bisimilar if there is a span (c) of open maps.
Definition 3.1 (named sets) A named set $E$ is a set denoted by $E$, and a family of name sets indexed by $E$, namely $\{E[e] \in \text{Set}\}_{e \in E}$ (i.e., $E[\_]$ is a map from $E$ to Set).

Given two named sets $E$ and $E'$, a named function $m : E \rightarrow E'$ is a function on the sets $m : E \rightarrow E'$ and a family of name embeddings (i.e., of injective functions) indexed by $m$, namely $\{m[e, e'] : E'[e'] \hookrightarrow E[e]\}_{(e, e') \in m}$.

The agent category is a category of algebras of labelled multigraphs on the category of named sets.

Agents are span of open maps bisimilar iff they are HD bisimilar.


Roadmap

- History Dependent automata
  - No symmetries
  - Spans of open maps
  - With symmetries
  - For verification
- Named sets vs. presheaves
- Network Conscious $\pi$-calculus
- Named graphs
- Conclusion
**Definition 33 (HD-automata).** A HD-automaton with symmetries (or simply **HD-automaton**) \( \mathcal{A} \) is a tuple \( \langle S, \text{sym}, L, \rightarrow \rangle \), where:

- \( S \) is the set of **states**;
- \( \text{sym} : S \to \text{Sym} \) associates to each state a finite-support **symmetry**;
- \( L \) is the set of **labels**;
- \( \rightarrow \subseteq \{(Q, l, \zeta, Q')|Q, Q' \in S, l \in L, \zeta \text{ is a finite-kernel permutation}\} \) is the **transition relation**, where:
  - \( Q \) and \( Q' \) are, respectively, the source and the target states;
  - \( l \) is the label of the transition, and
  - \( \zeta \) is a permutation that describes how the names of the target state \( Q' \) correspond, along this transition, to the names of the source state \( Q \).

Moreover, we assume that \( L = L_0 \cup L_1 \), with \( L_0 \cap L_1 = \emptyset \), and that \( l \in L_i \) iff \( \rho(l) \in L_i \) for every permutation \( \rho \). Labels in \( L_0 \) correspond to transitions that do not generate any new name, while labels in \( L_1 \) correspond to transitions that generate one new name.\(^6\)

Finally, we do not allow distinct isomorphic transitions between the same states to be present in a HD-automaton, where two transitions \( Q \xrightarrow{\zeta_1} Q' \) and \( Q \xrightarrow{\zeta_2} Q' \) are isomorphic if there exists some \( \rho \in \text{sym}(Q) \) such that

- \( l_2 = \rho(l_1) \);
- \( \zeta_2^{-1} \circ \rho \circ \zeta_1 \in \text{sym}(Q') \) if \( l_1 \in L_0 \) and \( \zeta_2^{-1} \circ \rho_+ \circ \zeta_1 \in \text{sym}(Q') \) if \( l_1 \in L_1 \).
Definition 36 (HD-bisimulation). Let $A$ be a HD-automaton. A HD-simulation for $A$ is a set of triples

$$\mathcal{R} \subseteq \{ \langle Q_1, \delta, Q_2 \rangle | Q_1, Q_2 \in S, \delta \text{ is a finite-kernel permutation} \}$$

such that, whenever $\langle Q_1, \delta, Q_2 \rangle \in \mathcal{R}$ then

- for each $\rho_1 \in \text{sym}(Q_1)$ and each $Q_1 \overset{l_1}{\to}_{\zeta_1} Q_1'$, there exist some $\rho_2 \in \text{sym}(Q_2)$ and some $Q_2 \overset{l_2}{\to}_{\zeta_2} Q_2'$ such that

  - $l_2 = \gamma(l_1)$, where $\gamma = \rho_2^{-1} \circ \delta \circ \rho_1$;

  - $\langle Q_1', \delta', Q_2' \rangle \in \mathcal{R}$, where: $\delta' = \begin{cases} \zeta_2^{-1} \circ \gamma \circ \zeta_1 & \text{if } l_1 \in L_0, \\ \zeta_2^{-1} \circ \gamma_{+1} \circ \zeta_1 & \text{if } l_1 \in L_1. \end{cases}$

A HD-bisimulation for $A$ is a set of triples $\mathcal{R}$ such that both $\mathcal{R}$ and $\mathcal{R}^{-1} = \{ \langle Q_2, \delta^{-1}, Q_1 \rangle | \langle Q_1, \delta, Q_2 \rangle \in \mathcal{R} \}$ are HD-simulations for $A$. 
HD Automata with Symmetries, III

- Given any HD automata with symmetries, the maximal bisimilarity on its states corresponds to a minimal automaton.
- Coalgebras for the $\pi$-calculus on the category of permutation algebras are isomorphic to HD automata with symmetries.
- Each state of an automaton is a concise representation of an orbit of the permutation algebra, and transitions between pairs of HD-states represent all the transitions between the corresponding orbits.
- Finite HD automata with symmetries can be minimized using the list partitioning/final sequence algorithm.

- Ugo Montanari, Marco Pistore: Pi-Calculus, Structured Coalgebras, and Minimal HD-Automata. MFCS 2000
- Ugo Montanari, Marco Pistore: Structured coalgebras and minimal HD-automata for the pi-calculus. TCS 2005
Roadmap

- History Dependent automata
  - No symmetries
  - Spans of open maps
  - With symmetries
  - For verification
- Named sets vs. presheaves
- Network Conscious $\pi$-calculus
- Named graphs
- Conclusion
HD Automata and Verification of Mobile Processes

The HAL environment

- Gian Luigi Ferrari, Gianluigi Ferro, Stefania Gnesi, Ugo Montanari, Marco Pistore, Gioia Ristori: An Automated Based Verification Environment for Mobile Processes. TACAS 1997

Named sets and their coalgebras defined type-theoretically

MiHDa minimization algorithm for pi-calculus proved in the finite case and implemented

The MIHDA Minimization Tool

- Written in OCAML
- Implements the Kanellakis-Smolka list partitioning algorithm
- Parametric wrt the underlying category
- Most time spent in handling name permutations
- Reasonably efficient execution:
  - handover protocol for GSM Mobile Network
  - the $\pi$-calculus specification has 506 states, 745 transitions
  - minimization in 9 seconds on a one-core processor
  - resulting HD-automaton with 105 states and 197 transitions
Roadmap

- History Dependent automata
  - No symmetries
  - Spans of open maps
  - With symmetries
  - For verification
- **Named sets vs. presheaves**
- Network Conscious $\pi$-calculus
- Named graphs
- Conclusion
Three Equivalent Structures

Categorical equivalence between

- the nominal sets of Gabbay and Pitts/permutation algebras
- the Schanuel topos
- the named sets of Montanari and Pistore (whose coalgebras are HD-automata)

Equivalence for Coalgebras

- Marcelo Fiore, Sam Staton, Information and Computation, 2006
The Category of Named Sets, Revisited, Generalized

- We represent the wide pullback preserving full subcategory of \( \text{Set}^C \) as the category \( \text{Fam}(\text{Sym}(C)^{op}) \)

- \( \text{Sym}(C) \) is the category of groups of automorphisms of \( C \), representing the \text{support} and \text{symmetry} of an element of a presheaf

- Symmetries are the essential information that is needed to reconstruct each represented presheaf: first one reconstructs the presheaf “freely” using representables, then a quotient is made using the symmetry.

For named sets, \( C \) is \( I \), the category of finite subsets of natural numbers and injections
Families

Given a small category $\mathbf{C}$, we define the category $\text{Fam}(\mathbf{C})$

- Objects are coproducts in $\text{Set}$: indexed collections of objects of $\mathbf{C}$
  $$\bigsqcup_{i \in I} \{c_i\}$$

- Each $i \in I$ is considered an element whose local interface is $c_i$

- An arrow from $\bigsqcup_{i \in I} \{c_i\}$ to $\bigsqcup_{j \in J} \{d_j\}$ is a function $h : i \to j$ and a family of $\mathbf{C}$ arrows
  $$\bigsqcup_{i \in I} \{\mathcal{H}_i : c_i \to d_{h(i)}\}$$
Category Symset = Sym(Set)

Objects:
permutation groups over finite subsets of $\omega$

Morphisms:
$\text{Symset} [\Phi_1, \Phi_2] := \{ i \circ \Phi_1 \mid i : \text{dom}(\Phi_1) \hookrightarrow \text{dom}(\Phi_2) \land \Phi_2 \circ i \subseteq i \circ \Phi_1 \}$

Composition:
$id_\Phi := id_{\text{dom}(\Phi)} \circ \Phi = \Phi$

$G \circ F = \{ g \circ f \mid g \in G \land f \in F \}$

Can be generalized to automorphism groups of any category
Families vs. Presheaves

**Q:** what categories of presheaves can be represented as families?

1. Our answer: **small index categories of monos**, all automorphisms are iso, and **(weak) wide pullback preservation** give rise to an equivalence of categories.

2. [Adamek, Velebil - TAC 2008]: **locally presentable index categories** and **weak wide pullback preservation** represent presheaves - natural transformations are not encoded. Generalises Joyal’s species as representations of analytic functors.

- The two conditions are different: (1) includes coproducts of categories, (2) includes Set. They obviously overlap (e.g. finite sets and injections).

Vincenzo Ciancia, Alexander Kurz, Ugo Montanari, Families of Symmetries as Efficient Models of Resource Binding. CMCS 2010
Roadmap

- History Dependent automata
  - No symmetries
  - Spans of open maps
  - With symmetries
  - For verification
- Named sets vs. presheaves
- **Network Conscious** $\pi$-calculus
- Named graphs
- Conclusion
Network Conscious $\pi$-calculus: Motivating Application

Software Defined Networks
- network administrators can program network services via high level constructs
Network Conscious π-calculus: An Example

Network-aware extension of the π-calculus
Two kinds of names:
• sites \(a, b, c\), i.e. network nodes
• links \(l_{ab}\), connecting two nodes

\[
M = m(x).m(y).(l_{xy})(\overline{m}x.l_{xy}.M)
\]
\[
p = \overline{a}ma.\overline{a}mb.a(l_{(xy)}).(L(l_{xy})|\overline{abc}.p')
\]
\[
q = b(x).q'
\]
\[
L(l_{xy}) = l_{xy}.L(l_{xy})
\]
Roadmap

- History Dependent automata
  - No symmetries
  - Spans of open maps
  - With symmetries
  - For verification

- Named sets vs. presheaves

- Network Conscious π-calculus

- Named graphs

- Conclusion
Network Conscious π-calculus: Resources, II

\(G_I\): Directed multigraphs up to isomorphism with injective graph homomorphisms

- **Category** \(\text{Sym}(G_I)\)
  - Objects: for all \(g \in |G_I|\), we take all subgroups of \(G_I[g,g]\) (group operation is composition in \(G_I\))
  - Morphisms and composition as before

- **Category** \(\text{Gset}\)
  - Objects are \(N = (Q_N, S_N)\), where \(S_N : Q_N \rightarrow |\text{Sym}(G_I)|\)
    \[|q| \in |G_I|\] is the local graph of \(q\)
  - Morphisms are suitable graph homomorphisms
  - Products \(N \times M\) are sets of pairs \(n_{g_1} \in N, n_{g_2} \in M\) equipped with an embedding of \(g_1, g_2\) in a bigger graph
  - More abstractly: \(\text{Fam}(\text{Sym}(G_I)^{op})\)
Network Conscious $\pi$-calculus: Resources, III

Endofunctors for resource allocation
Q: what categories of presheaves can be represented as families?

1. Our answer: small index categories of monos, all automorphisms are iso, and (weak) wide pullback preservation give rise to an equivalence of categories.

2. [Adamek, Velebil - TAC 2008]: locally presentable index categories and weak wide pullback preservation represent presheaves - natural transformations are not encoded. Generalises Joyal’s species as representations of analytic functors.

$G_i$ satisfies condition 1: thus resource conscious $\pi$-calculus has HD automata


Ugo Montanari, Matteo Sammartino, A Network-Conscious Pi-Calculus and Its Coalgebraic Semantics, TCS.

Roadmap

- History Dependent automata
  - No symmetries
  - Spans of open maps
  - With symmetries
  - For verification
- Named sets vs. presheaves
- Network Conscious $\pi$-calculus
- Named graphs
- Conclusion
What’s Next: Software Architectures

Synchronized hyperedge replacement (CCS-like, i.e. without mobility) to describe

Software architecture

Detailed design

Synchronization rules

Productions for single hyperedges
Architectures as Resources

Model of resources
- Architectures as a category of hypergraphs
- Δs add new components (hyperedges) and connections (nodes)
- Presheaves index systems by their architecture

Two levels of behavior
1. *In the large*
   - Algebra of parallel composition of components
   - Coalgebra of component synchronization
2. *in the small*
   - Syntax of sequential programs/processes
   - Coalgebra of process actions

(1) + (2) = Composition of bialgebras to define the whole system

Similar structure for BI(P): Behavior of atomic components + Interactions among them