

*Imt Graduation And Honorary Doctorate Award Ceremony for Ugo Montanari  
Lucca, March 26, 2014*

---



# Why Sometimes Abstract Models Are Better in Practice

**Ugo Montanari**  
Dipartimento di Informatica  
University of Pisa

# From Practice To Theory (and Back?)

---

- Personal experience
- From practical research areas
- Pictures, Graphics and Heuristic Search

# A Quasi-Euclidean Skeleton

```

X X X X X X X X X X
X X   X X X   X
X     X X     X
X       X       X
X X           X
X X X         X X
X X X         X X
X X X           X
X           X   X
X           X   X
X X       X X   X X
X X X X X X X X X X
    
```

(a)

```

0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 1 1 0
0 1   1 0 0 1   1 0
0 1   1 0 1     1 0
0 0 1   1       1 0
0 0 0 1         1 0 0
0 0 0 1         1 0 0
0 0 0 1   1     1 0
0 1 1   1 0 1   1 0
0 1     1 0 1   1 0
0 0 1 1 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0
    
```

(b)

```

0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 1 1 0
0 1 2 1 0 0 1 2 1 0
0 1 2 1 0 1 2 2 1 0
0 0 1 2 1 2     2 1 0
0 0 0 1 2     2 1 0 0
0 0 0 1 2 2 2 1 0 0 0
0 0 0 1 2 1 2 2 1 0
0 1 1 2 1 0 1 2 1 0
0 1 2 2 1 0 1 2 1 0
0 0 1 1 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0
    
```

(c)

```

0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 1 1 0
0 1 2 1 0 0 1 2 1 0
0 1 2 1 0 1 2 2 1 0
0 0 1 2 1 2 3 2 1 0
0 0 0 1 2 3 2 1 0 0
0 0 0 1 2 2 2 1 0 0
0 0 0 1 2 1 2 2 1 0
0 1 1 2 1 0 1 2 1 0
0 1 2 2 1 0 1 2 1 0
0 0 1 1 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0
    
```

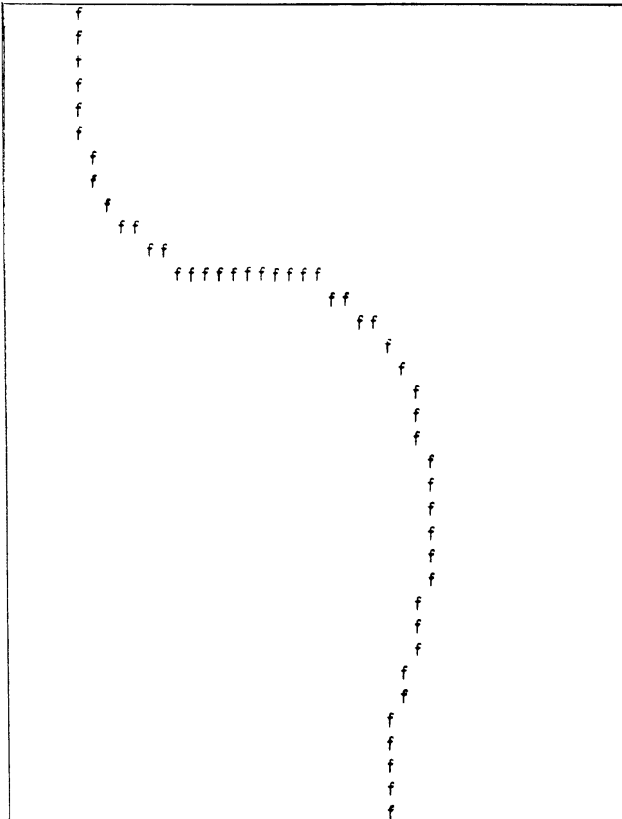
(d)

```

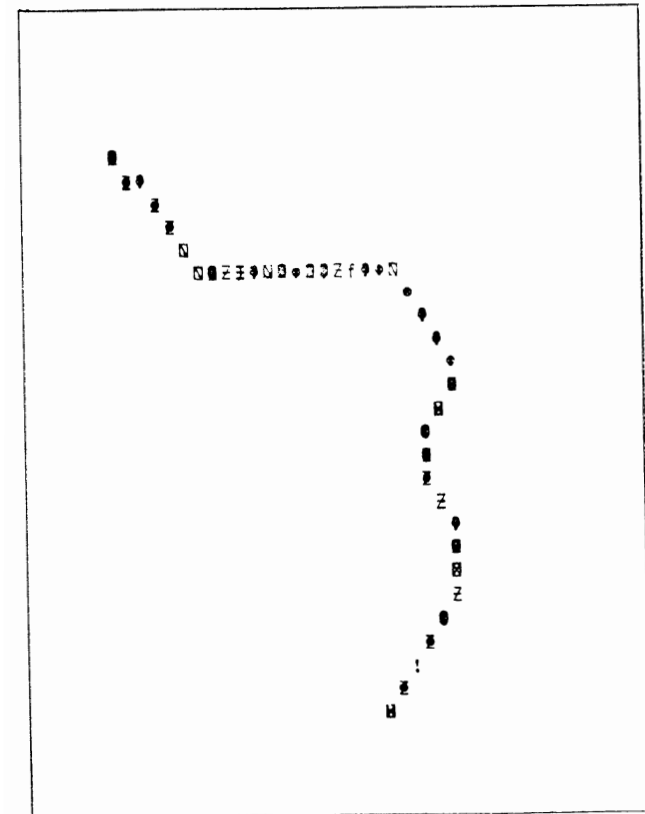
      1           1
     2           2
    2           2
      2         3
         3
        2       2
       2       2 2
      1 2     2
     2 2     2
           1
    
```

(e)

# Recognizing Curves in (very) Noisy Pictures



! 0? f Z N ! 0?? 0? Z M Z 0 0 , \* f / N ) \* 0 \* Z , \* 0 : f , \* Z  
( Z 0 0 0 0 0 0 0 0 ) Z 0 0 0 0 0 0 0 0 ! ) 0 0 0 0 , 0 ) 0 0 0 0 \* Z 0 f 0  
i 0 0 0 0 0 0 0 0 ? f ! , f ) f \* 0 0 ! f 0 0 f \* f , \* f , ? Z 0 0 ) , 0 0 0 0 )  
0 ? ! \* 0 0 ) ! ) \* 0 0 ! 0 0 0 0 , \* 0 f ) Z 0 0 0 0 ! ! 0 0 0 0 0 0 0 0 ? 0 0  
f f \* , 0 ? 0 0 0 0 0 0 ! 0 0 ) \* , ) Z 0 0 0 ! 0 f ! \* , ) , ! Z 0 0 0 0 Z  
? 0 Z f ! f 0 ) Z 0 0 0 0 0 0 ) 0 0 Z , ? 0 0 0 ! ! 0 ? 0 ! \*  
! ? f \* f ) 0 , Z \* ! 0 0 0 0 ) \* 0 ! 0 0 ) 0 0 0 ! Z ) ? ! f 0 ? f Z ? 0 0 )  
Z 0 ? \* 0 0 0 0 0 0 0 0 ) \* 0 \* Z 0 f Z , 0 0 0 0 ? \* ! ? 0 0 0 0 ! ? 0 0 0 0  
! \* f ! 0 ? 0 0 0 f 0 0 0 0 f ? 0 0 ! 0 \* ? ? ! 0 ? 0 0 0 0 0 0 0 0 0 0 0 0  
0 ! \* ? Z 0 0 0 0 Z Z , \* 0 ) \* , ! , Z ) ) , 0 0 ) 0 0 0 0 ? 0 0 ? f ! ? 0  
\* Z ? 0  
0 0 0 0 ? f 0  
0 0 ) \* 0 0 f Z ) 0 f 0  
0 ! 0 ? ) ! ? 0 f 0 0 0 0 0 0 ) \* 0  
f ! 0 ! 0 f \* f 0  
Z 0 ! Z \* ! ? ! , \* , 0 , Z Z ? f , , \* , 0 0 0 0 ? \* Z 0 ! ) \* 0 0 0 0 f Z  
0 0 , 0 ) 0 0 ! 0 ! ! ) \* ) \* ! \* , \* ! f \* 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
\* ) f 0 0 f \* f \* \* f , ) \* 0 0 f ? , 0 0 f 0 0 , ) Z 0 0 0 ! \* 0 ) f  
! \* Z ) 0 0 f \* , \* \* 0 0 0 0 f Z ? \* \* 0 0 0 0 ! \* ! f Z ) Z 0 0 , 0 Z \* !  
! , \* 0 0 Z f 0 ! ) 0 0 ? , , 0 ) 0 0 ? \* , ! 0 \* f 0 0 0 0 ? , ? \* ) 0 f 0 \* ?  
Z \* Z Z , \* 0 0 0 0 ) \* 0 f 0 ! f 0 Z 0 0 0 0 0 0 ? ? ) \* 0 0 0 , )  
\* , 0 0 0 0 ! ) ? ) , 0 \* , , f 0 0 0 , , ) f Z \* 0 0 f 0 0 ) f 0 0 0 ) \* , 0  
0 0 Z 0 \* Z 0 f , \* 0 f Z , \* 0 ) ? 0 0 ? ! ! 0 0 ! \* ! 0 f ! ! ) \* 0 f 0 0 0  
0 0 0 0 0 0 ! ) ) 0 ? \* 0 0 ? 0 0 0 ) 0 f 0 Z ? 0 0 ! ) ? \* 0 0 0 0 )  
0 0 0 ? ? 0 0 0 0 ! 0 ! 0 ! ) \* 0 ? 0 0 0 0 ? 0 0 f ? 0 0 Z , 0 0 0 .  
\* 0 ! Z ? 0 f 0 0 f Z 0 f \* 0 z f \* 0 0 0 ! 0 0 0 0 f 0 0 0 0 0 ) ) ? \* 0 0 )  
\* 0 ? 0 0 \* , f 0 ? 0 \* 0 Z ) ! 0 \* 0 0 ) ) 0 Z f 0 , ? ? 0 0 0 \* , , 0  
! f ? , \* f 0 0 Z 0 0 0 0 0 f 0 0 ) 0 0 0 0 ? ) Z 0 0 f ! f 0  
\* 0 f 0 0 ) 0 , 0 0 0 ) \* ? ! \* 0 0 0 0 ! 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
! ) Z ? Z 0 0 0 0 ! ? f 0 Z 0 ? f f \* ) f ? 0 0 ) f ) Z \* 0 ? , 0 0 ) \* Z \*  
0 ? ! 0 ? ) ? ? ! 0 ! 0 f , Z \* ? ? , 0 0 0 0 0 f 0 ? Z , ? Z , ! 0 ) ? 0 0  
f \* ! 0 ! \* 0 0 0 0 0 0 0 0 , Z \* 0 0 0 0 0 0 0 0 f Z ! 0 0 Z \* , \* 0 0 ) \* Z Z , \*  
? 0 ! Z \* 0 0 0 0 0 0 ) \* 0 0 0 0 ! , ? \* , f 0 ? 0 ) ? ? ? ! ? Z f ? 0 ) )  
\* , Z ) \* 0 0 ? ? , 0 , , ? 0 0 0 0 ! f 0 0 ? 0 0 0 0 ! ) ? 0 0 0 0 ! f f \* 0  
f ) \* ? Z 0 0 \* f Z 0 ! 0 0 , Z 0 0 0 0 0 0 ! 0 0 f ! ? f ? 0 0 , 0 ) \* )

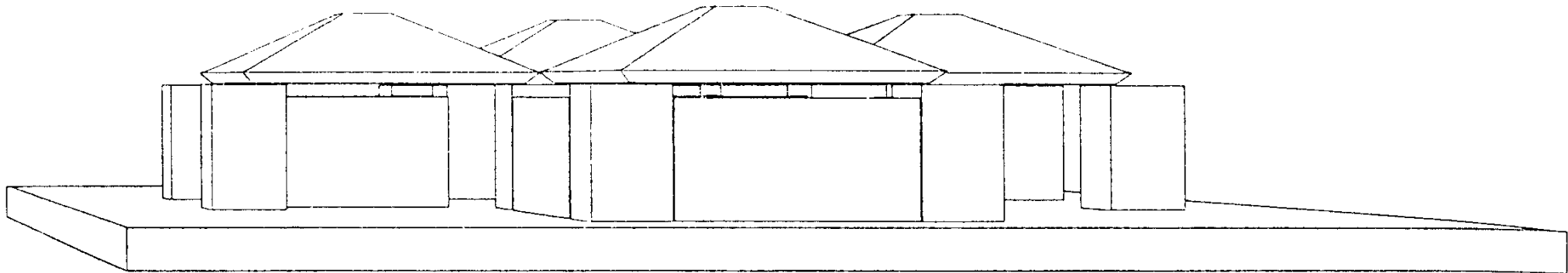


# A Bathhouse by Louis Kahn in Trenton, New Jersey



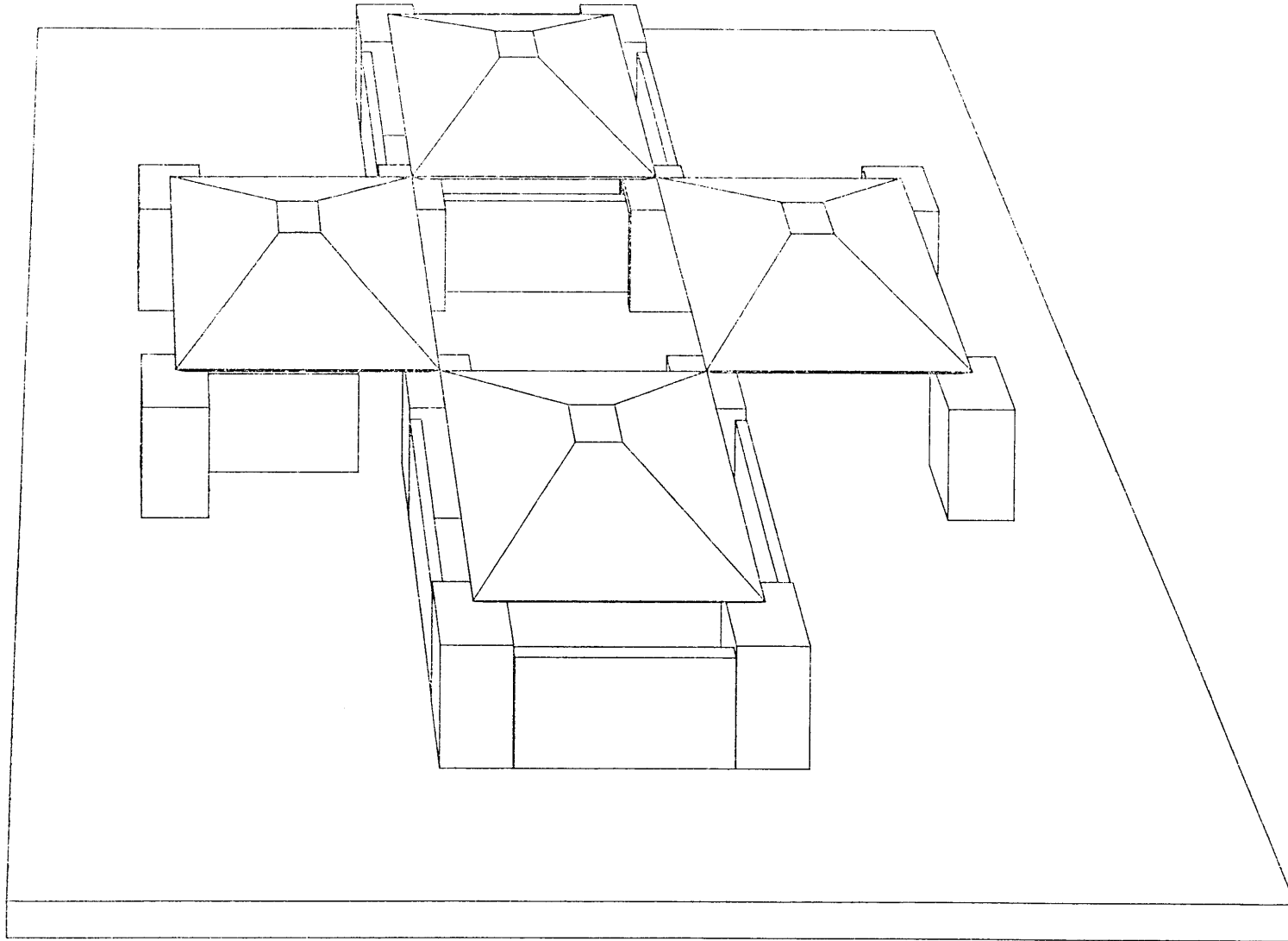
# A View Without Hidden Lines, I

---

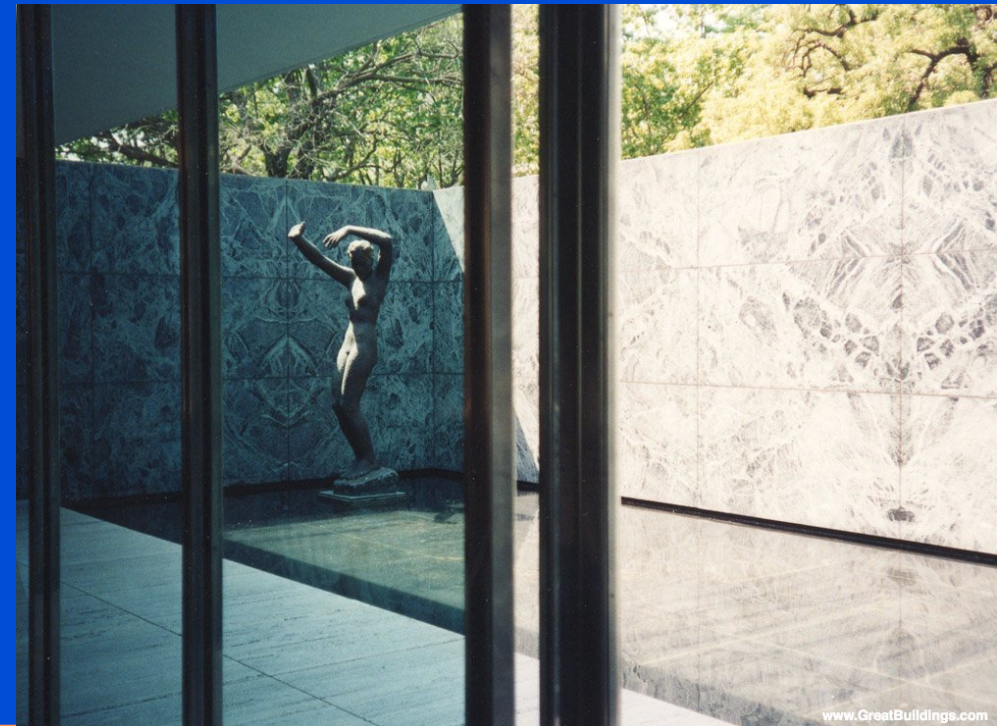




# A View Without Hidden Lines, II

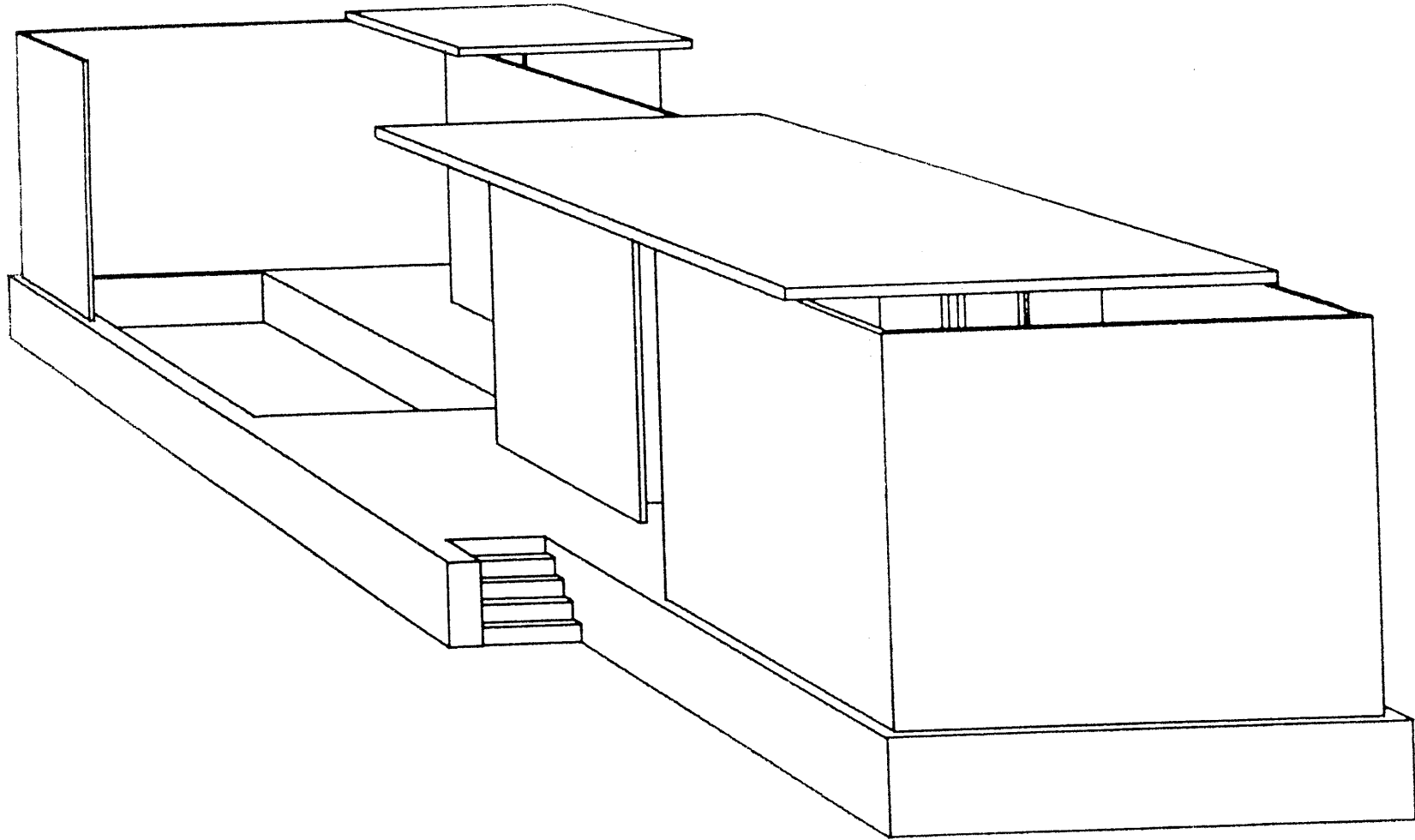


# Mies van der Rohe at Barcelona Exhibition



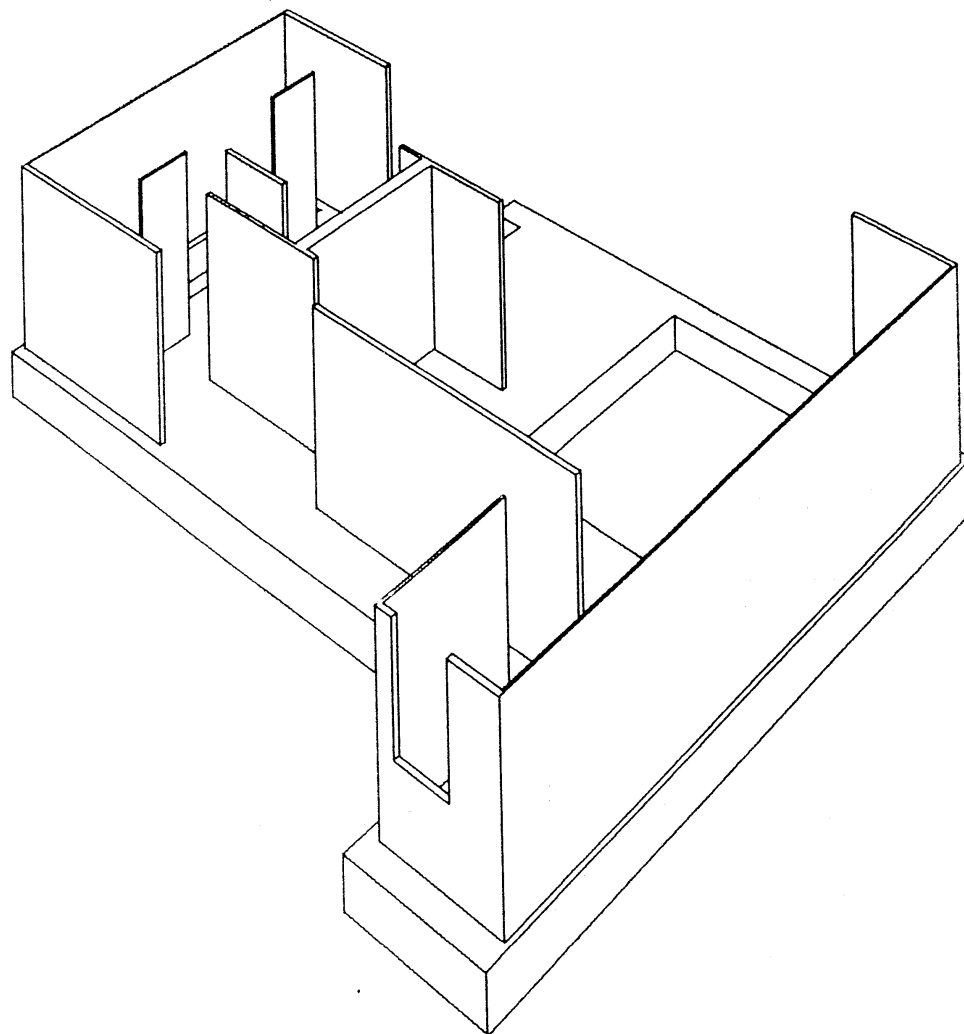


# Mies van der Rohe at Barcelona with no Hidden Lines

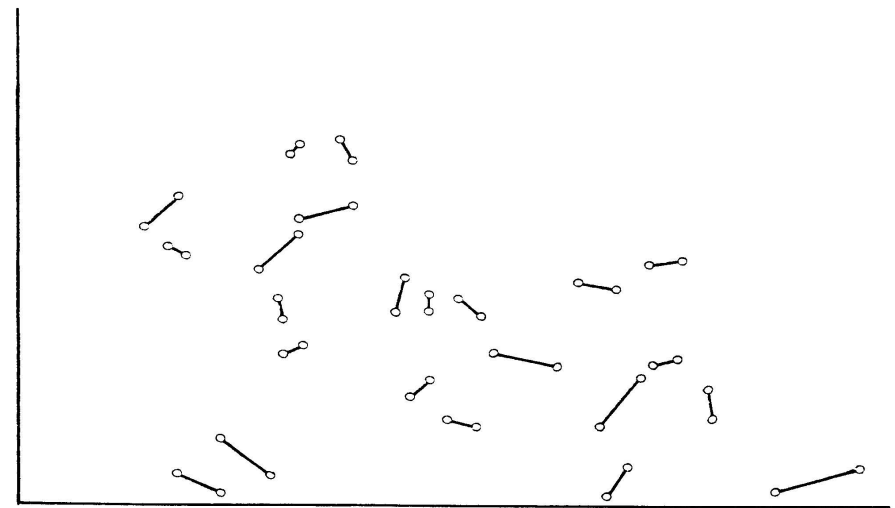
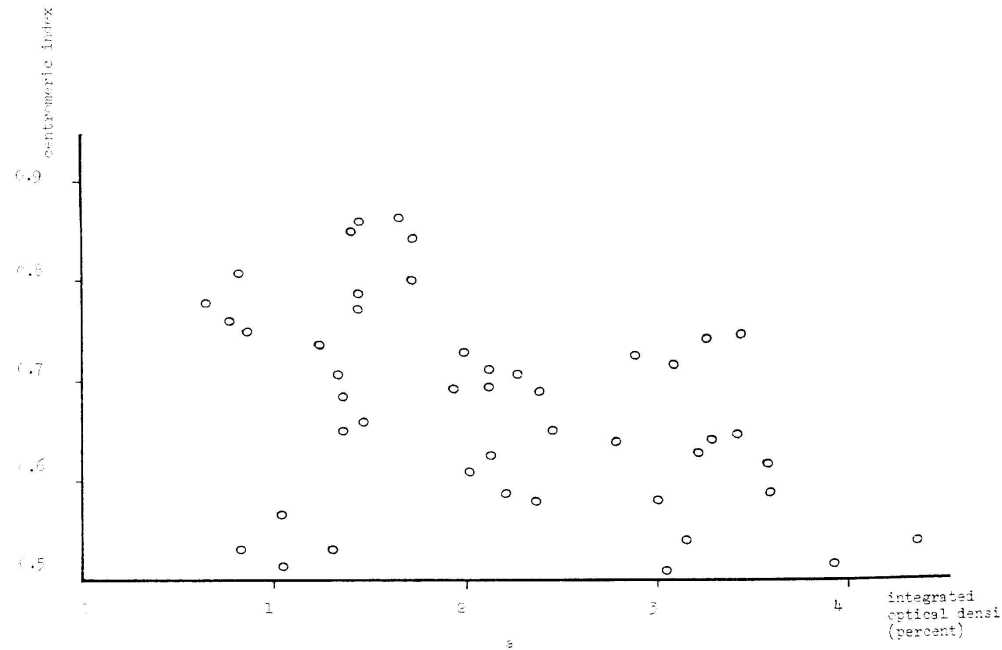


# Mies van der Rohe at Barcelona without the Roof

---



# Optimal Matching of Chromosomes, Via A\* Algorithm



# From Practice To Theory (and Back?)

---

- To theoretical areas
- Concurrent Distributed Models of Computation
- Metamodels: with category theory



# From Practice To Theory (and Back?)

---

Back to more practical matter

- Games for grid scheduling
- Reinforcement learning / optimal profiling for Smart Power Grids
- E-mobility

# Metamodels

---

- A most useful theory
- Explain concepts in a general, uniform way
- Help better design
- Inductive process for defining the metamodel
- Deductive process for checking soundness and generality
  
- Here I will present some comments based on my experience

# Roadmap

---

- Natural phenomena vs. computer science
- Abstract models must be modular and parametric
- Concurrency
- Networks
- Transition systems
- Constraints

# Roadmap

---

- Natural phenomena vs. computer science
- Abstract models must be modular and parametric
- Concurrency
- Networks
- Transition systems
- Constraints



# Natural Sciences and Computer Science

---

- natural sciences discover phenomena, guess and check laws
- mathematics looks for deep results guided by originality and formalization of rational concepts
- computer science both studies natural phenomena and develops design goals

# Natural Phenomena in Computer Science

---

- natural limits to computers and computation processes
  - computability
  - algorithmic complexity
  - security
  - circuit lithography
  - clock frequency
  - feasible parallelism with different grains: fine-grain, coarse-grain
  
- unplanned large global systems
  - internet, www
  - service-oriented computing
  - hybrid cyber-physical systems
  - social networks

# Modular Design Process in Computer Science, I

---

- vertical/horizontal decomposition
  - layers of virtual machines and their codes, overlay networks
  - horizontal modularization for reusability, flexibility
- machine language
  - continuous electronic phenomena vs. discrete steps, as buffer assignment
- operating systems, middleware, hypervisors, available virtual machines
- modular programming languages offer environments for different classes of applications
- specific libraries for reuse

# Roadmap

---

- Natural phenomena vs. computer science
- **Abstract models must be modular and parametric**
- Concurrency
- Networks
- Transition systems
- Constraints



# Modular Design Process in Computer Science, II

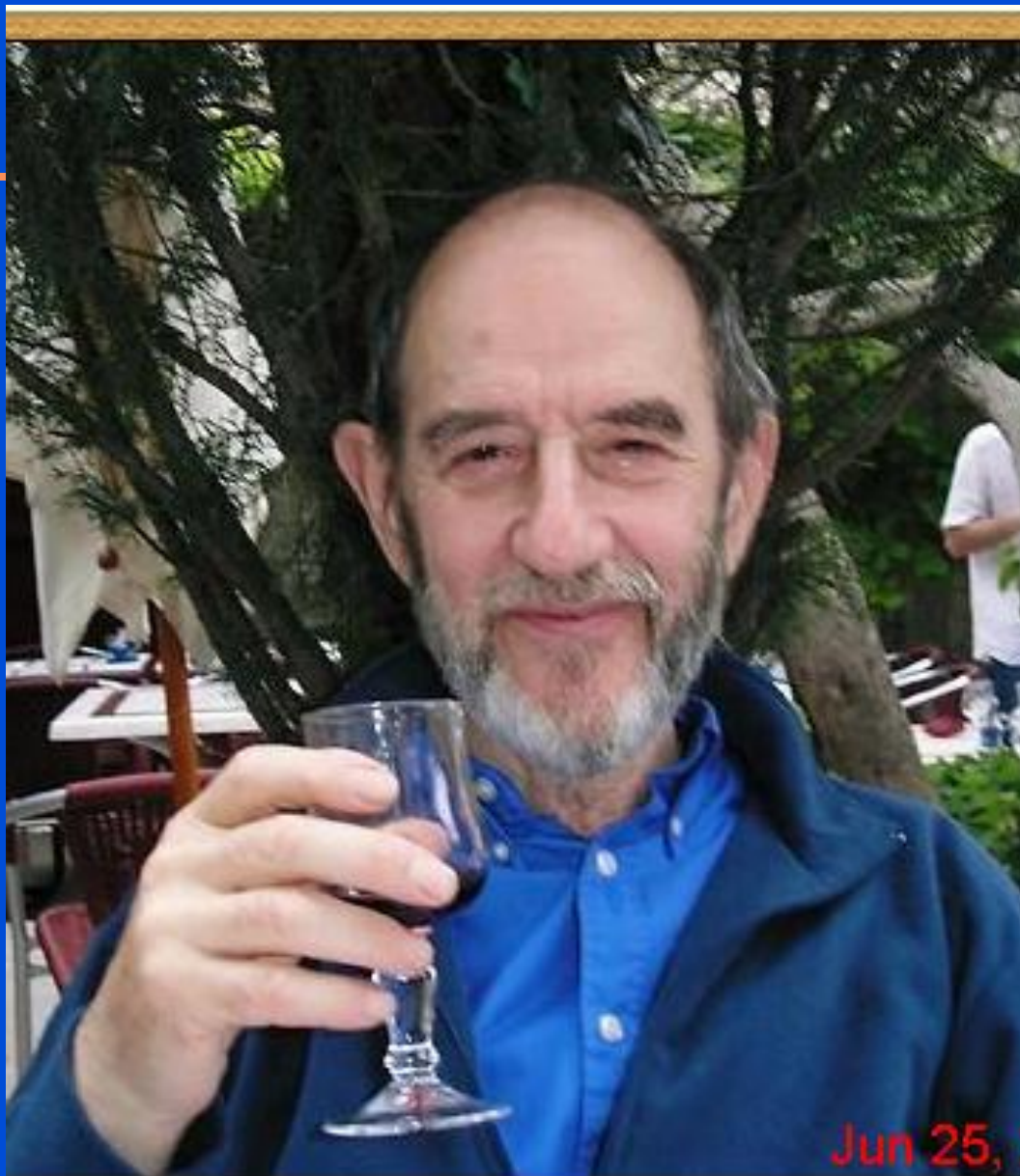
---

- how virtual machines, connectors and interfaces are specified in practice?
- mathematical modeling or engineering design?
- both, but:
  - systems become more complex, global and heterogeneous
  - formal specifications are more and more necessary
- formal definitions themselves must be flexible, modular, hierarchical
- concise and parametric models
- capture all the similar instances of the concept

# Parametricity and Modularity of Models

---

- seminal work
  - Rod Burstall, Joe Goguen, Putting Theories Together, IJCAI 1977
  - Lawvere algebraic theories and functorial models
  - combination, parametrization, extension of theories
- institutions
  - Borstall & Goguen, Abstract Model Theory for Specification and Programming, 1991
- efficient implementation, animation, metaprogramming
  - OBJ family: Joe Goguen, Jose Meseguer, Claude Kirchner, Kokichi Futatsugi
  - rewriting logic and MAUDE
  - widely used, e.g. at IMT within project ASCENS



Le bonheur



# Roadmap

---

- Natural phenomena vs. computer science
- Abstract models must be modular and parametric
- Concurrency
- Networks
- Transition systems
- Constraints



# Theory of Concurrency

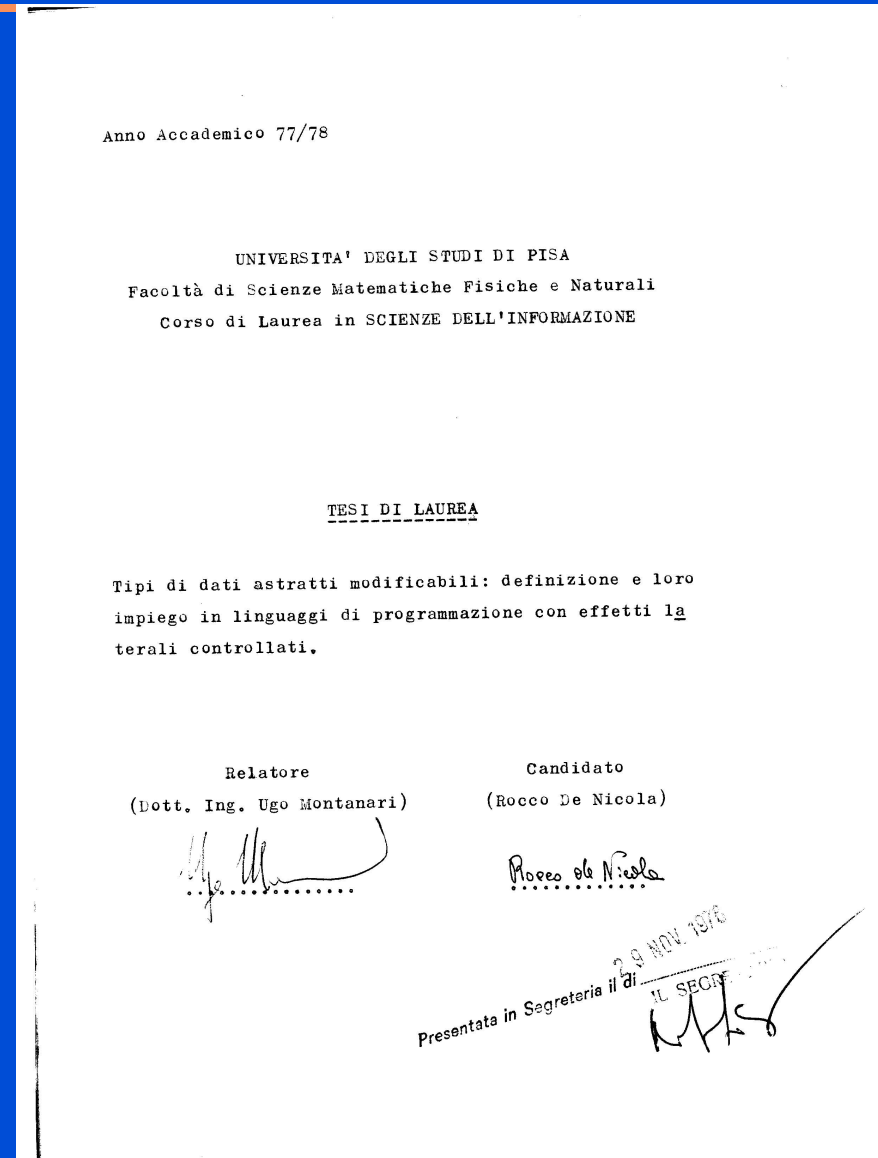
---

- equivalence classes of computations
  - concurrent events may occur in any order
  - implementation chooses one of the possible/the most convenient
  - different speeds are possible
  - essential for program reuse
- how to define equivalence of computations?
  - monoidal categories
  - essentially the same basic construction for all kinds of models of computation
  - Meseguer Montanari 1988, Petri nets are Monoids
  - Bruni and Montanari “Pensare il Mondo” Math series by Einaudi
- related concept of causality: “no concurrency, mutual exclusion”
  - security, noninterference

# Rocco De Nicola



- De Nicola, R., Martelli, A., and Montanari, U., **Communication Trough Message Passing or Shared Memory: a Formal Comparison**, 2nd International Conference on Distributed Computing Systems, Versailles (April 6-10, 1981).



# Ilaria Castellani

---



- Castellani, I., and Montanari, U., **Graph Grammars for Distributed Systems**, H. Ehrig, M. Nagl and G. Rozenberg, Eds.: Graph Grammars and their Application to Computer Science, Springer Lecture Notes in Computer Science Vol. 153 (1982), 20-38.

# Roberto Gorrieri



UNIVERSITÀ DEGLI STUDI DI PISA  
DIPARTIMENTO DI INFORMATICA

DOTTORATO DI RICERCA IN INFORMATICA

Università di Pisa - Genova - Udine

PhD Thesis: TD-2/91

## Refinement, Atomicity and Transactions for Process Description Languages

*Roberto Gorrieri*

**Abstract.** The thesis aims at giving a contribution towards a formal semantics for algebraic process description languages equipped with some mechanisms for changing the level of abstraction, studying such an extension in a variety of different settings. The individual chapters are centered around the common themes of atomicity, refinement and transactions.

The thesis is divided into two parts. In the first part, the main goal is the development of a calculus with an operation of action refinement, to be properly understood as the analogous in a concurrent framework of the procedure call mechanism in sequential computation theory. We start by introducing a mechanism for handling atomic actions in CCS (Chapter 3), extending it to a form of refinement where the process which refines an action should be executed atomically (Chapter 4) and then, in Chapter 5 and 6, we present our idea of refinement as an operation which implements at the linguistic level the operation of context-free substitution at the semantic level.

In the second part, the transaction concept — the analogous in a “truly concurrent”, distributed setting of the atomicity concept in the “interleaving”, centralized one — is exploited for giving a faithful translation of process description languages into Petri nets (Chapter 8 and 9). Finally, in Chapter 10, we study the relationships between fragments of Linear Logic and net theory. In particular, the connective of linear implication gives the mean for looking at net computations at a lower, more decentralized level of abstraction.

March 1991

ADDR: Corso Italia 40, 56100 Pisa, Italy. TEL:+39-50-510111 - TLX:590291 DIPISAI - E\_MAIL:di.unipi.it



# Roadmap

---

- Natural phenomena vs. computer science
- Abstract models must be modular and parametric
- Concurrency
- Networks
- Transition systems
- Constraints

# All Kinds of Networks

---

- distributed systems
  - interfacing hardware and software components
  - communication networks
- graph models of computation
  - connector networks
  - neural networks
  - expressing choice (nondeterminism) with concurrency and compositionality
- connector algebras
  - Bruni, Montanari and Sobocinski, universal properties
  - extension (Frobenius algebras)
  - pure quantum states and operations on them



# Roadmap

---

- Natural phenomena vs. computer science
- Abstract models must be modular and parametric
- Concurrency
- Networks
- Transition systems
- Constraints

# Labeled Transition Systems (LTS), I

---

- most natural operational model: system moves from state to state
  - labels are observable behavior
  - actions, time constraints, probabilities, quantitative parameters
- behavioral properties
  - temporal logic, model checking for finite LTS, billions of states
  - black box equivalence of states and systems
  - minimal automaton
- adequate logic
  - two systems are equivalent iff they satisfy the same set of formulas
  - they are different iff there exists a formula which differentiates between them.

# Labeled Transition Systems, II

---

- thousands of notions of temporal logic/observational equivalence
- categorical universal notions
  - coalgebras, bialgebras
  - unique concept of bisimilarity and modal/temporal logic: difference due to
    - » base category
    - » behavior functor
- nominal calculi: names as
  - communication channels, links, keywords, session identifiers
- History Dependent (HD) automata
  - yield finite state LTS for model checking
  - several PhD students supervised by Ugo Montanari:
    - Marco Pistore, 1999, Emilio Tuosto, 2003, Filippo Bonchi, 2008, Vincenzo Ciancia, 2008), Matteo Sammartino, 2013

# Filippo Bonchi

---



PH.D. THESIS

## Abstract Semantics by Observable Contexts

Filippo Bonchi

SUPERVISOR  
Ugo Montanari

May 6, 2008

# Vincenzo Ciancia



## Nominal Sets, Accessible Functors and Final Coalgebras for Named Sets

Vincenzo Ciancia

Supervisor: Prof. Ugo Montanari

Co-Supervisors: Prof. Gian Luigi Ferrari, Prof. Giorgio Ghelli

# Matteo Sammartino

---





# Roadmap

---

- Natural phenomena vs. computer science
- Abstract models must be modular and parametric
- Concurrency
- Networks
- Transition systems
- Constraints



# Constraint Systems

---

- Widely used concept, with different formal meanings
- Simpler version, networks of constraints: hypergraphs
  - nodes are variables
  - hyperarcs are constraints which must be satisfied by the adjacent variables
  - usually finite set of possible values of variables
- difficult problem: NP-complete
- efficient algorithms in special cases
  - hierarchical decomposition into small components
  - linear dynamic programming algorithms
  - essentially a tree of logical clauses
- approximate solutions via propagation, relaxation

# Soft Constraints

---

- soft constraints
  - degree of validity or a cost or a set of cases
  - with addition, multiplication
- constraint semirings: idempotent addition, commutative multiplication
  - logical, fuzzy, tropical semiring
  - cartesian product, functional domains
  - Hoare powerdomain for multicriteria (Pareto) optimization
- functional domains: internalizing variables
  - constraints assign logical or soft values to variable assignments
  - sets of clauses: obvious interpretation of sum, product and restriction
  - recursive clauses, several clauses for the same predicate -> DATALOG
  - system defined constraints  $\Rightarrow$  constraint programming
  - Herbrand constraint system: (soft) constraint logic programming

# Francesca Rossi



UNIVERSITÀ DEGLI STUDI DI PISA  
DIPARTIMENTO DI INFORMATICA

DOTTORATO DI RICERCA IN INFORMATICA  
Università di Pisa - Genova - Udine

PhD Thesis: TD-14/93

## Constraints and Concurrency

*Francesca Rossi*

**Abstract.** The research of this thesis is centered around the notion of *constraint programming*. In particular, we study constraint problems and constraint logic languages, with the aim of understanding how a logic language, either concurrent or not, may gain in efficiency, expressive power, modularity, and flexibility, from the fact that it deals with constraints.

In particular, we first develop a general scheme for constraint consistency algorithms, and we propose a new algorithm, called perfect relaxation, which is very efficient for some classes of constraint problems. Then we analyze the Constraint Logic Programming (CLP) framework, and we show that it can conveniently use the perfect relaxation algorithm as an efficient constraint handler for finite domain constraints.

Although being very declarative and natural for constraint programming, the CLP framework lacks some flexibility. In fact, constraint solving algorithms cannot be specified at the CLP program level, and thus the constraint system cannot be extended by the user. Moreover, it is restricted to sequential computations.

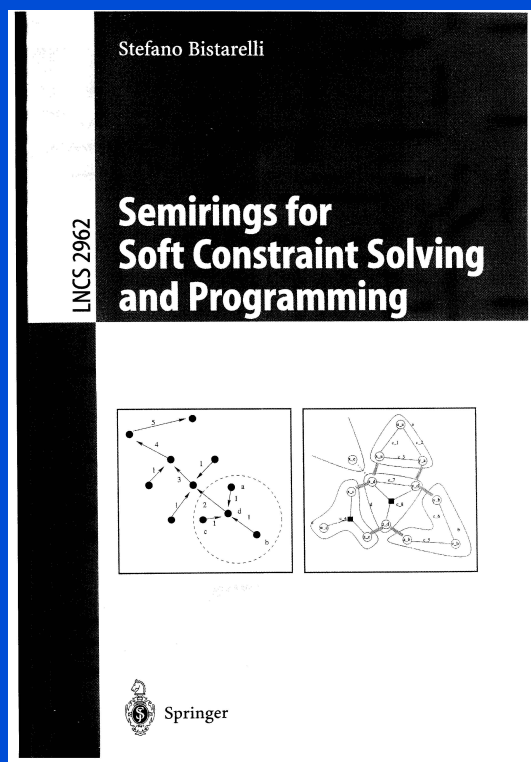
Therefore we turn our attention to the concurrent constraint (cc) programming framework, and we show that it has the desired flexibility, since constraint algorithms and program agents are homogeneous, which means that any algorithm can be expressed as a cc program. We also show that such homogeneity is of crucial importance, since it allows a non-monolithic representation of the underlying constraint system, which in turn yields the possibility of expressing the correct level of concurrency contained in a cc program. In fact, by adopting a distributed view of the constraint system underlying any cc program, and by using a context-dependent computation model, we develop a true-concurrency semantics for cc programs which is based on partial orders and it is able to show all the concurrency contained in a given program, and then we extend it in order to capture all the nondeterminism as well.

Finally, we apply the results obtained for cc programs to other fields. First, by extending Petri nets via context conditions (besides pre- and post-conditions). Then, by proposing an abstract machine for concurrent systems which is based on the obtained results (mainly, a context-dependent model of computation and a flat distributed representation of the computation state).

March 1993

ADDR: Corso Italia 40-56125 Pisa - Italy - TEL:+39-50-510111 - TLX:590291 DIPISAI - E-mail: rossi@di.unipi.it

# Stefano Bistarelli



UNIVERSITÀ DEGLI STUDI DI PISA  
DIPARTIMENTO DI INFORMATICA

DOTTORATO DI RICERCA IN INFORMATICA  
UNIVERSITÀ DI PISA

PH.D. THESIS: TD-2/01

## Soft Constraint Solving and Programming: a general framework.

*Stefano Bistarelli*

**Abstract.** The *Soft Constraints* idea is able to capture many real life situations that cannot be represented and solved with the classical crisp constraint framework. Several extensions were proposed in the last years, but we show that we can use

*One Ring to rule them all,  
One Ring to find them,  
One Ring to bring them all and in the darkness bind them.*  
(from "The Lord of the Rings" of Tolkien, J.R.R.)

March 2001

ADDR: Corso Italia 40, 56125 Pisa, Italy. TEL: +39-50-887268. FAX: +39-50-887226.  
E-MAIL: [bista@di.unipi.it](mailto:bista@di.unipi.it). HOME-PAGE: <http://www.di.unipi.it/~bista/>.

# Procedural vs. Declarative Knowledge

---

- constraints as knowledge combined with
  - process oriented
  - service oriented
  - contract, session paradigms
  - concurrent constraint (cc) programming, cc-pi
- declarative and procedural aspects
  - parametric, partially specified
- SCEL: Service Component Ensemble Language
  - European FP7 FET IP Project ASCENS
  - abstractions that allow one to directly represent behaviors
  - knowledge and aggregations according to specific policies

# Marzia Buscemi

---



UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"  
DIPARTIMENTO DI MATEMATICA E APPLICAZIONI R.CACCIOPOLI

—————  
Dottorato in Matematica Applicata e Informatica - XIV Ciclo

## **Models and Security Verification of Mobile Systems**

MARIA GRAZIA BUSCEMI

# Conclusion

---

- metamodels are very useful
  - in theory: exploiting the essential concepts
  - in practice:
    - » modeling tools
    - » verification
    - » simulation, animation
- underlying theoretical computer science, mathematics
  - category theory
  - logic
  - probability
- towards implementation
  - software engineering
  - theory of algorithms