General Description of Network Systems

Introduction

Ugo Montanari
Dipartimento di Informatica, University of Pisa
A Variety of Networks

- Communication networks & routing
- Neural networks & connectivism
- Stochastic models of complex systems
- Social networks and concept mining
- Computation, semantics, economics of networks
- Networks of connectors & buffers represented as graphs
Graphs Are Everywhere

- Use of diagrams / graphs is pervasive to Computer Science
Networks

- hypergraphs with labels, structure and observation interface
- labels/buffer content may change with executions
- structure usually changes only via explicit reconfiguration operations
Networks

- Petri nets
- linear time-invariant dynamical systems
- (soft) constraint satisfaction problems
- Bayesian networks
- electric circuits
- computational fields
Petri Nets

Carl Petri, PhD thesis, 1962 – The(4) dining philosophers
Petri Nets

Roberto Gorrieri
Process Algebras for Petri Nets: The Alphabetization of Distributed Systems
Springer, to appear
Linear Time-Invariant Dynamical Systems


FIGURE 1 Modeling by tearing, zooming, and linking. Part (a) shows a black box with terminals. The aim is to obtain a model of the behavior of the variables on these external terminals. Part (b) shows the result of the tearing process: the black box is viewed as a gray box of interacting subsystems. The modeling process proceeds by zooming in on the subsystems one by one, as illustrated in (c). The subsystems are subsequently linked by sharing the variables on their common terminals, as illustrated by (d). The combination of the models of the subsystems and the interconnection constraints leads to a model of the variables on the external terminals. This modeling process has a hierarchical structure, since a subsystem can in turn be modeled by tearing, zooming, and linking.
(Soft) Constraint Satisfaction

Constraint Solving

Constraint programming is a programming paradigm where relations between variables can be stated in the form of constraints. Constraints differ from the common primitives of other programming languages in that they do not specify a step or sequence of steps to execute but rather the properties of a solution to be found.
(Soft) Constraint Satisfaction

**Scientific Applications**

**Constraints and Molecular Biology**

Constraints and Molecular Biology: Constraint Programming techniques can be efficiently used for predicting structure of a protein which is considered one of the most important problem in Computational Biology. The protein structure prediction problem has effectively been transformed to a constraint minimization problem with finite domain and Boolean variables. The Oz language was then used to implement the constraint problem. Certain variables have been defined for the entire constraint problem of predicting the protein structure. Later constraint optimization has been used to minimize the variable surface. A perfect conformation was found on all possible sequences in finding the sequence length and also the optimal surface. Hence constraint techniques can be effectively applied to solving problems in computational biology.

**Commercial Applications**

**XLufthansa**

A project named PARROT was designed and implemented which was aimed at providing efficient means to address the highly complex and costly problem of airline crew scheduling by combining the techniques of Operations Research and Constraint Programming.

**Industrial Applications**

**HIC Project (Constraint Handling in Industry and Commerce)**

This project aims at exploiting CLP in Industrial applications. A system for treasury planning based on CLP technology was developed which required as input the expected liquidity balances for the next 10 working days and the interest rates on the money market. As output it delivers a set of operations that covers all deficits a set of operations that covers all deficits and utilizes all surpluses...
Bayesian Networks

![Bayesian Network Diagram](diagram.png)

\[
\begin{array}{c|c|c|c|c|c|}
T & S & Pr(R) & W & R & Pr(M) \\
\hline
T & T & \frac{9}{10} & T & T & \frac{8}{10} \\
T & F & \frac{6}{10} & T & F & \frac{5}{10} \\
F & T & \frac{4}{10} & F & T & \frac{6}{10} \\
F & F & \frac{1}{10} & F & F & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c|}
P & M & Pr(A) \\
\hline
T & T & 1 \\
T & F & \frac{7}{10} \\
F & T & \frac{8}{10} \\
F & F & \frac{1}{10} \\
\end{array}
\]
An equivalent power system feeding a 300 km transmission line. The line is compensated by a shunt inductor at its receiving end. A circuit breaker allows energizing and de-energizing of the line.
Computational Fields

Lafuente, Loreti, Montanari, A Fixpoint-based Calculus for Graph-shaped Computational Fields, Coordination 2015.

A rescue problem: associate each victim to its closest rescuer.

Victims are red, rescuers black

Fields are expressed via a soft mu calculus, with arcs labelled with monotone functions.
Each rescuer has found the shortest path to its closest victim.
Each victim has chosen the closest rescuer
From algebraic to graph-based syntax

**Algebraic**
- Terms
  - $a | b$
- Operations
  - $\cdot : W \times W \rightarrow W$
- Axioms
  - $x | y \equiv y | x$
- Rewrite rules
  - $a \rightarrow b$

**Graph-based**
- Elements
- Vocabulary
- Equivalence
- Dynamics
- Graphs (diagrams)
  - *flat, hierarchical, etc.*
- Graph compositions
  - *Union, tensor, etc.*
- Homomorphisms
  - *Isomorphism, etc.*
- Transformation rules
From graphs to graph algebras

- Start with a given class of graphs
- Define an equational signature,
  - operators correspond to operations on graphs
  - axioms describe their properties
- Prove once and for all soundness and completeness of the axioms with respect to the interpretation on graphs, as well as surjectivity
- Next, you can safely use the algebra as an alternative, more handy syntax for the graphs
First part: Milner Flowgraphs

Networks are connected via shared global names/channels

- Milner flowgraph algebra for process calculi
- networks of constraints
- network tree decomposition for dynamic programming via scope extension
Product permutation categories: symmetric monoidal categories

Networks are connected via series parallel composition

- bigraphs
  - connector algebras for Petri nets with boundaries
  - signal flow graphs
Third Part: From Structured Graphs to Categories

From structured states to
- structured rewrite rules
- structured transition systems (graphs)
- structured models of computation (categories)

Examples
- Petri nets are monoids
- rewriting logic
- tiles
Roadmap

Milner flowgraphs
- flowgraph algebra for process calculi
- networks of constraints
- tree decomposition for dynamic programming

PROPs and string diagrams
- bigraphs
- connector algebras for Petri nets with boundaries
- signal flow graphs

From structured graphs to categories
- Petri nets
- rewriting logic
- tiles

Conclusions
THE END
General Description of Network Systems

Part 1: Milner Flowgraphs

Ugo Montanari
Dipartimento di Informatica, University of Pisa
Roadmap

Milner flowgraphs

- flowgraph algebra for process calculi
- networks of constraints
- tree decomposition for dynamic programming

- networks are connected via shared global names/channels
Roadmap

Milner flowgraphs

- flowgraph algebra for process calculi
- networks of constraints
- tree decomposition for dynamic programming
Flow Graphs and Flow Algebras

Robin Milner, Flow Graphs and Flow Algebras, JACM 1979
Flow Graphs and Flow Algebras

Robin Milner, Flow Graphs and Flow Algebras, JACM 1979

$Laws of Composition:\n\begin{align*}
(C1)\ & x_1|x_2 = x_2|x_1, \\
(C2)\ & x_1|(x_2|x_3) = (x_1|x_2)|x_3.
\end{align*}$

$Laws of Restriction:\n\begin{align*}
(R1)\ & x_1\!\setminus\!\alpha = x_1 \text{ when } \alpha \not\in \text{names}(L_1), \\
(R2)\ & x_1\!\setminus\!\alpha\!\setminus\!\beta = x_1\!\setminus\!\beta\!\setminus\!\alpha, \\
(R3)\ & (x_1|x_2)\!\setminus\!\alpha = x_1\!\setminus\!\alpha|x_2\!\setminus\!\alpha \text{ when } \alpha, \bar{\alpha} \not\in L_1 \cap \bar{L}_2.
\end{align*}$

$Laws of Relabelling:\n\begin{align*}
(S1)\ & x_1[\ ] = x_1, \\
(S2)\ & x_1[R][S] = x_1[S \circ R], \\
(S3)\ & x_1\!\setminus\!\alpha[R] = x_1[R \cup \lambda/\alpha]\!\setminus\!\beta \text{ when } \alpha \in \text{names}(L_1), \text{name}(\lambda) = \beta \not\in \text{names}(\text{Ran } R), \\
(S4)\ & (x_1|x_2)[R] = x_1[R_1]|x_2[R_2] \text{ where } R_i = R \mid L_i (i = 1, 2).\n\end{align*}$

flow algebra equivalence corresponds to flow graph isomorphism
Denotational Semantics

- George Milne, Robin Milner, Concurrent Processes and Their Syntax, JACM 1979
- Tony Hoare, Communicating Sequential Processes, CACM 1978
- Interpreting the algebra in a semantic domain
Roadmap

Milner flowgraphs

- flowgraph algebra for process calculi
- networks of constraints
- tree decomposition for dynamic programming
Networks of (soft) Constraints

- Networks of Constraints, 1971
  - nodes are variables
  - (hyper) arcs are predicates
  - hidden nodes are existentially quantified
  - local propagation makes constraints stronger

- Soft constraints, 1997
  - constraint satisfaction returns a value in a semiring
  - fuzzy, optimization variants

- Semiring definition of constraint composition
  - Constraints themselves are values of a functional semiring
  - $c: (V \rightarrow D) \rightarrow S$
  - $(c_1 \times c_2)\eta = c_1\eta \times c_2\eta$
Roadmap

Milner flowgraphs

- flowgraph algebra for process calculi
- networks of constraints
- tree decomposition for dynamic programming
Back to (Almost) Milner

\[ p, q := p \parallel q \mid (x)p \mid A(\tilde{x}) \mid nil \]

- Problem made of two subproblems \( p \) and \( q \)
- Assignment for \( x \) is determined
- Atomic subproblem of a single constraint \( A \)
- Empty subproblem

+ structural congruence: commutative monoidality of \( \parallel \), \( \alpha \)-conversion, swapping of restrictions + axioms of permutation algebra for nominal structure + scope extension:

\[ (x)(p \parallel q) = (x)p \parallel q \]

\( x \notin fv(q) \)

nodes replaced by the notion of support

Terms up-to structural congruence are (hyper)graphs with hidden nodes: \( A(X) \) is a graph consisting of a single hyperedge and its nodes
Structural Recursion on Terms

- dynamic programming is structural recursion on terms
  - graph operations => interpreted on discrete and continuous domains
  - evaluation should not depend on the particular term, only on the graph
  - many important practical applications
- evaluation of \((x)p(X)\) depends on parameters \(X\)
  - typically exponential in \(|X|\)
  - total complexity is defined as the complexity of the worst restriction subterm
- complexity depends on which term in the equivalence class
- choosing the best term
  - secondary optimization problem of dynamic programming
  - NP complete
Scope extension is key axiom allows us to choose the order of variable elimination
\[(x)(p \parallel q) = (x)p \parallel q \quad x \notin fv(q)\]

Apply from left to right => assign variables as soon as you can

Normal form
Restrictions outside

Canonical forms
Reduced wrt
scope extension

Canonical forms are local optima of SOP
An Example of Evaluation

Connectivity of a social network

- Links with independent failure probability.
- Find the probability that a set of sites are fully connected.
- The evaluation domain are the probability distributions on all the partitions of sites.

\[
R_0(x, y, z) = A(x, y) \parallel B(x, z)
\]

\[
R_{i+1}(x, y, z) = (v)R_i(x, v, z) \parallel R_i(v, y, z)
\]

\[
W_k(v, x) = (z)R_k(x, v, z) \parallel A(v, x) \parallel B(v, z)
\]

The complexity of this families of wheels is logarithmic.
THE END
General Description of Network Systems

Part 2: PROPs and String Diagrams

Ugo Montanari
Dipartimento di Informatica, University of Pisa
Roadmap

PROPs and string diagrams

- bigraphs
- connector algebras for Petri nets with boundaries
- signal flow graphs

Networks are connected via series parallel composition

The basic structure: symmetric monoidal categories

- sequential and parallel composition, permutation of wires
- additional connectors with axioms
- string diagrams, wire-and-box diagrams
- axiomatization = string diagram isomorphism
Roadmap

PROPs and string diagrams

- bigraphs
- connector algebras for Petri nets with boundaries
- signal flow graphs
GS-monoidal theory over $\Sigma$

\[
\begin{align*}
&\text{(generators)} \quad f : v \rightarrow w \in \text{GS}(\Sigma) \\
&\text{(tensor)} \quad s : v \rightarrow w, t : v' \rightarrow w' \\
&\text{(composition)} \quad s \otimes t : v \otimes v' \rightarrow w \otimes w' \\
&\text{(permutations)} \quad s : v \rightarrow w, t : w \rightarrow u \\
&\text{(duplicators)} \quad \rho_{v,w} : v \otimes w \rightarrow w \otimes v \\
&\text{(dischargers)} \quad \nabla_v : v \rightarrow v \otimes v \\
&\text{axiomatization of cartesian categories \`a la Lawvere without naturality axioms for duplicator and discharger}
\end{align*}
\]
GS-monoidal theory over $\Sigma$ - Axioms

- Arrows and pairing operator form a monoid:
  - $s \otimes (t \otimes r) = (s \otimes t) \otimes r$
  - $s \otimes id_\varepsilon = id_\varepsilon \otimes s = s$

- Categorical axioms:
  - $s; (t; r) = (s; t); r$
  - $s; id_\varepsilon = id_\varepsilon \otimes s = s$

- Functoriality axiom:
  - $(s \otimes t); (s' \otimes t') = (s; s') \otimes (t; t')$

- Monoidality axioms:
  - $id_{v \otimes w} = id_v \otimes id_w$
  - $\nabla_{\varepsilon} = !\varepsilon = \rho_{\varepsilon, \varepsilon} = id_{\varepsilon}$
  - $\rho_{v;w}, \rho_{w,v} = id_{v \otimes w}$
  - $\rho_{v \otimes w, u} = (id_v \otimes \rho_{w,u}); (\rho_{v,u} \otimes id_w)$
  - $\nabla_{v \otimes w} = (\nabla_v \otimes \nabla_w); (id_v \otimes \rho_{v,w} \otimes id_w)$
  - $\nabla_v; \rho_{v,v} = \nabla_v$
  - $!_{v \otimes w} = !_v \otimes !_w$

- Coherence axioms:
  - $\nabla_v; (id_v \otimes \nabla_v) = \nabla_v; (\nabla_v \otimes id_v)$
  - $\nabla_v; (id_v \otimes !_v) = id_v$

- Naturality axiom:
  - $(s \otimes t); \rho_{v\otimes w} = \rho_{v,w}; (t \otimes s)$
Bigraphs, as the name suggests, are composed by two independent (hyper)graphs on the same set of nodes:

**Place Graph - Locality**

0

\[ v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \]

1

\[ v_0 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \]

**Link Graph - Connectivity**

\[ e_0 \rightarrow v_0 \rightarrow v_4 \]

\[ e_1 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \]

\[ e_2 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \]
Gs-graphs vs. bigraphs

Top view vs. side view
Gs-graphs vs. bigraphs

- They are almost the same:
  - place and links graphs vs. ● and ○ typed nodes and wires
  - controls vs. signature operations of the form $K: \bullet \circ^n \rightarrow \bullet$
  - top view vs. side view
  - restricted nodes vs. $\nu: \varepsilon \rightarrow \circ$ always in the signature

- Main difference: the interfaces
  - gs-graph: strings over the alphabet \{●, ○\}*
    - => assign a different name to each character in the string
  - bigraphs: pairs made by an ordinal $m$ and by a set of names $X$
    - => make a list out of \{0, ..., $m$-1\} $\cup X$

- Theorem: Shuffled support-equivalent bigraphs over a pure signature $K$ are isomorphic to gs-graphs over $K$ with name choices.

- Lean bigraphs have no edges in the link graph attached to no node.
- Correspondingly, gs graphs must be equipped with the axiom

$$\nu; \not\circ = \text{id}_\varepsilon$$
**Roadmap**

PROPs and string diagrams

- bigraphs
- connector algebras for Petri nets with boundaries
- signal flow graphs
A Basic Algebra of Stateless Connectors

- operational semantics
- equipped with a constraint-like denotational semantics
- complete axiomatization
- canonical representation
- most general combination of synchronization and nondeterminism
- Bruni, Lanese, Montanari TCS 2006
# A Basic Algebra of Stateless Connectors

<table>
<thead>
<tr>
<th>Ordinary structure</th>
<th>Dual structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>symbolic</td>
</tr>
<tr>
<td>symmetry</td>
<td>$\gamma: 2 \rightarrow 2$</td>
</tr>
<tr>
<td>duplicator</td>
<td>$\nabla: 1 \rightarrow 2$</td>
</tr>
<tr>
<td>bang</td>
<td>$!: 1 \rightarrow 0$</td>
</tr>
<tr>
<td>mex</td>
<td>$\nabla: 1 \rightarrow 2$</td>
</tr>
<tr>
<td>zero</td>
<td>$\mathbf{0}: 1 \rightarrow 0$</td>
</tr>
</tbody>
</table>

Configuration $\xrightarrow{\text{input}}$ Configuration’

$\gamma \frac{x \otimes y}{y \otimes x} \gamma$ with $x, y \in \{\text{tick, untick}\} \quad \nabla \xrightarrow{\text{tick}} \nabla \xrightarrow{\text{untick}} \nabla \xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla \\
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla \xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla \\
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla \xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
\xrightarrow{\text{untick}} \nabla \xrightarrow{\text{tick}} \nabla
A Basic Algebra of Stateless Connectors: Some Axioms

\[
\begin{align*}
\langle \rangle & = \cdot \cdot \\
\times & = \cdot \cdot \\
\cdot \cdot & = \cdot \cdot \\
(1) & \quad (2) \\
\cdot \cdot & \cdot \cdot \\
\cdot & \cdot \\
(3) & \quad (4) \\
\cdot \cdot & \cdot \cdot \\
\cdot & \cdot \\
(5) & \quad (6) \\
\cdot & \cdot \\
\cdot & \cdot \\
(7) & \\
\end{align*}
\]
1-Safe Nets with Boundaries: Notation

ordinary notation

directed, left-to-right notation
 transitions disappear

essentially basic connectors with one-place buffers
Nets with Boundaries: Composition

Boundaries = attach points for transition fragments

Composition = can combine multiple transitions
Petri Calculus: Strong Semantics

\[ P ::= \circ | \bullet | 1 | X | \Delta | \nabla | \perp | T | \land | V | \downarrow | \uparrow | P \otimes P | P ; P \]

Basic stateless algebra plus 1-place buffers

\[ \circ \xrightarrow[0]{1} \bullet \quad (TkI) \quad \bullet \xrightarrow[1]{0} \circ \quad (TkO) \]

one token arrives \hspace{1cm} one token leaves

Configuration \xrightarrow{\text{input}} \text{Configuration’} \xrightarrow{\text{output}}

Ordinary bisimilarity, when taking the string of inputs/outputs as label
**Petri Calculus: Other Buffer Semantics**

\[
P ::= \bigcirc | \bigcirc | l | X | \Delta | \nabla | \perp | T | \land | V | \downarrow | \uparrow | P \otimes P | P ; P
\]

Configuration \(\xrightarrow{\text{input}}\) Configuration’

\[
\begin{align*}
\bigcirc \xrightarrow{1} & \bigcirc \\
\bigcirc \xrightarrow{1} & \bigcirc \\
\bigcirc \xrightarrow{1} & \bigcirc
\end{align*}
\]

(TkO2)

Both TkI2, TkO2

\[
\begin{align*}
\bigcirc & \xrightarrow{1} \bigcirc \\
\bigcirc & \xrightarrow{1} \bigcirc
\end{align*}
\]

(Ugo Montanari - IFIP WG2.2, Singapore, September 12-160916)
The Case of P/T Nets

\[ P ::= (n) \mid I \mid X \mid \Delta \mid \nabla \mid \perp \mid \top \mid \wedge \mid \vee \mid \downarrow \mid \uparrow \mid P \otimes P \mid P ; P \]

Configuration \( \frac{\text{input}}{\text{output}} \) Configuration’

\[
\begin{align*}
    n, h, k &\in \mathbb{N} \quad k \leq n \\
    \frac{h}{k} \quad &\rightarrow \\
    (n) \quad &\rightarrow (n+h-k) \quad (\text{TKIO}_{n,h,k})
\end{align*}
\]
Connector Algebra for Petri Nets

- most general combination of synchronization and nondeterminism, with buffers
- correspondence with BIP by Sifakis, REO by Arbab and Span(Graph) by Katis, Sabadini and Walters
- coalgebraic theory for $\text{F}(X) = \text{P}(A \times X)$
- bialgebraic theory: operations preserve bisimilarity
- standard representatives of equivalence classes for finite Petri nets
Roadmap

PROPs and string diagrams

• bigraphs
• connector algebras for Petri nets with boundaries
• signal flow graphs
Signal Flow Graphs

- Foundations of control theory

- Coalgebraic theory for $F(X) = A \times X$

- PROP treatment
    - a sound and complete graphical theory of vector subspaces over the field of polynomial fractions, with relational composition
    - buffers are derivatives in the operational calculus (e.g. via Laplace transforms)
    - deterministic functional vs deadlock-prone relational
Signal Flow Graphs

- Signal Flow Graphs (SFGs) are stream processing circuits widely adopted in Control Theory since at least the 1950s.
- Constructed combining four kinds of gate

\[
\begin{align*}
\text{copier} & \quad k \\
\text{adder} & \quad x \\
\text{amplifier} & \quad k \\
\text{register} & \quad x
\end{align*}
\]

\[ k \in k \]

\[
\begin{align*}
k_1 & \quad 0 \\
k_2 & \quad k_1 \\
k_3 & \quad k_1 \\
\vdots & \quad \vdots
\end{align*}
\]
Signal Flow Graphs

Two examples:

Both circuits implement the generating function

\[ \frac{1}{(1-x)^2} = 1x + 2x^2 + 3x^3 + \ldots \]

Can we check this statically?
The Calculus of SF Diagrams

Circuit diagrams of $\text{Circ}$ are generated by the grammar

$$c, d ::= \begin{array}{c}
\bullet \\
\circ \\
\boxed{k} \\
\boxed{x} \\
\boxed{C} \\
\boxed{d}
\end{array}$$
Full Abstraction

A counterexample

\[
\left[\begin{array}{c}
\begin{array}{c}
\text{x} \\
\text{x}
\end{array}
\end{array}\right] = \left[\begin{array}{c}
\begin{array}{c}
\text{y}
\end{array}
\end{array}\right] = \left[\begin{array}{c}
\begin{array}{c}
\text{x} \\
\text{x}
\end{array}
\end{array}\right]
\]

\left\{ \begin{array}{c}
\text{x} \\
\text{x}
\end{array} \right\} \not\in \left\{ \begin{array}{c}
\begin{array}{c}
\text{y}
\end{array}
\end{array} \right\} \not\in \left\{ \begin{array}{c}
\begin{array}{c}
\text{x} \\
\text{x}
\end{array}
\end{array} \right\}

\]

We say that \( \begin{array}{c}
\begin{array}{c}
\text{x} \\
\text{x}
\end{array}
\end{array} \) has **deadlocks** and \( \begin{array}{c}
\begin{array}{c}
\text{x} \\
\text{x}
\end{array}
\end{array} \) needs **initialisation**.
Full Abstraction

Theorem
For any $c$ and $d$ in Circ deadlock and initialisation free

$$[c] = [d] \iff \langle c \rangle = \langle d \rangle$$
Linear Time-Invariant (LTI) Discrete Dynamical Systems.

- Fong, Sobocinski, Rapisarda, A Categorical Approach to Open and Interconnected Dynamical Systems, LICS 2016
  - biinfinite streams, i.e. infinite past, no requirement of initial value 0
  - Linear Time-Invariant (LTI) discrete dynamical systems as categories of corelations of matrices
  - they are (interpreted) SMT
  - an operational semantics which fully agrees with the denotational semantics
  - no axiomatization at the moment
THE END
General Description of Network Systems

Part 3: From Structured Graphs To Categories

Ugo Montanari
Dipartimento di Informatica, University of Pisa
Roadmap

From structured graphs to categories
• Petri nets
• rewriting logic
• Tiles

From structured states to
• structured rewrite rules
• structured transition systems (graphs)
• structured models of computation (categories)

with left adjoints which preserve colimits
Roadmap

From structured graphs to categories

- Petri nets
- rewriting logic
- tiles
Petri Nets as an Algebra

• Petri nets are monoids
  – Algebra of (concurrent) computations via the lifting of the monoidal structure of markings to steps and computations
    • sequential composition “;” (of computations)
    • plus identities (idle steps)
    • plus parallel composition ⊕ (of markings, steps and computations)
    • plus functoriality of ⊕ (concurrency!)
    • leads to a (strictly) symmetric strict monoidal category of computations

• Collective Token Philosophy (CTPh)
  – \( \pi(\_ \) (commutative processes)

• Individual Token Philosophy (ITPh)
  – \( \mathcal{P}(\_ \) (concatenable processes)
Collective vs. Token View

Best-Devillers vs. Goltz-Reisig processes
The ITPh Story, I

\[ \mathcal{P}(N) \xrightarrow{\mathcal{U}(\_)} \text{SMonCat} \xrightarrow{\langle u \downarrow \_ \rangle} \text{PreOrd} \]

\[ \text{PTNets} \xrightarrow{D(\_)} \text{DecOcc} \]

\[ \text{Safe} \xrightarrow{\mathcal{U}(\_)} \text{Occ} \]

\[ \text{Winskel’ s chain of coreflections} \]

\[ \text{Sassone’ s chain of adjunctions} \]
Pre-Nets

- Under the CTPh, the construction $\pi(\_)$ is completely satisfactory
  - $\pi(\_)$ is left adjoint to the forgetful functor from $\text{CMonCat}^\oplus$ to Petri
  - $\pi(\_)$ can be conveniently expressed at the level of (suitable) theories (e.g. in PMEqtl)

- But the CTPh does not model concurrency

- We argue that, under the ITPh, all the difficulties are due to the multiset (marking) view of states

- Pre-nets are the natural implementation of P/T nets under the ITPh
  - pre-sets and post-sets are strings, not multisets!
Pre-Nets

- Under the ITPh, the construction $\mathcal{Z}(\_)$ is completely satisfactory
  - $\mathcal{Z}(\_)$ is left adjoint to the forgetful functor from $\text{SMonCat}^\otimes$ to PreNets
  - $\mathcal{Z}(\_)$ can be conveniently expressed at the level of (suitable) theories (e.g. in PMEqtl)
  - All the pre-nets implementations $R$ of the same P/T net $N$ have isomorphic $\mathcal{Z}(R)$
  - $\mathcal{P}(N)$ can be recovered from (any) $\mathcal{Z}(R)$
Other Left Adjoint Constructions

Quite similar developments:

- Term rewriting (2-categories), Jose Meseguer
- Logic programming (double categories), Andrea Corradini
- Graphs (DPO / SPO), Paolo Baldan, Andrea Corradini, Leila Ribeiro
- Process Calculi (Tiles, monoidal double categories), Fabio Gadducci, Roberto Bruni
Roadmap

From structured graphs to categories

- Petri nets
- rewriting logic
- tiles
Main Ingredients of Rewriting Logic:
- Signature $\Sigma$ of system configurations
- Structural axioms $E$
- Rewrite rules over $[t]_E$

Categorically:
- States form a cartesian category $\mathcal{L}_{\Sigma,E}$ – the Lawvere Theory associated with $(\Sigma,E)$
  - natural number as objects
  - substitutions as arrows
  - composition = substitution application
- Proof terms form a cartesian 2-category

2-Computad
- Rewrite rules over the arrows of $C$
Theory and Applications of RL

11th International Workshop on Rewriting Logic and its Applications
Eindhoven, 2016

20th anniversary since its first edition in Asilomar, California, in 1996.
Theory and Applications of RL

- **Foundations**
  - termination, confluence, narrowing, partial evaluation, rewriting strategies
  - graph rewriting
  - rewriting-based calculi and explicit substitution

- **Rewriting as a Logical and Semantic Framework**
  - programming language semantics, concurrency models, distributed systems real-time, hybrid, and probabilistic systems

- **Rewriting Languages**
  - rewriting-based declarative languages
  - implementation techniques
  - tools supporting rewriting languages

- **Verification Techniques**
  - temporal, modal and reachability logics for dynamic properties of rewrite theories
  - rewriting-based theorem proving, including (co)inductive theorem proving
    - constraint solving and satisfiability
    - verification and analysis of programs

- **Applications**
  - applications in logic, mathematics and physics
  - rewriting models of biology, chemistry, and membrane systems
  - security specification and verification
  - specification and verification of critical systems
  - applications to model-based software engineering
  - applications to engineering and planning
Roadmap

From structured graphs to categories

- Petri nets
- rewriting logic
- tiles
Tiles, Logically

initial input interface

\[ u \downarrow \alpha \quad \beta \]

\[ \circ \quad \circ \quad \circ \]

\[ \circ \quad \circ \quad \circ \]

\[ \circ \quad \circ \quad \circ \]

final input interface

\[ t \]

\[ s \]

initial output interface

\[ v \]

final output interface
Tiles, Categorically

- Double Monoidal Category
  - Objects, horizontal arrows, vertical arrows and cells
  - Horizontal 1-category: objects and horizontal arrows
  - Vertical 1-category: objects and vertical arrows
  - Horizontal 2-category: vertical arrows and cells
  - Vertical 2-category: horizontal arrows and cells

- Monoidal operation on objects, horizontal arrows, vertical arrows and cells

- Any two operations of vertical, horizontal and monoidal structure commute, e.g.
  - vs(hs(A)) = hs(vs(A))
  - (A ;h B) ;v (C ;h D) = (A ;v C) ;h (B ;v D) exchange law
  - (A ;h B) x (C ;h D) = (A x C) ;h (B x D)
Axiomatization of Double Categories in PMEqtl

\[ \text{Alg}_{T'}(\text{Alg}_T(\text{Set})) \sim \text{Alg}_T(\text{Alg}_{T'}(\text{Set})) \sim \text{Alg}_{T \otimes T'}(\text{Set}). \]

fth DCAT is CAT \( \otimes \) CAT renamed by:

- sorts (Object, Object) to Object . (Arrow, Arrow) to Square .
- sorts (Arrow, Object) to Harrow . (Object, Arrow) to Varrow .
- ops d left to w . c left to e . d right to n . c right to s .
- ops _;_ left to _*_ . _;_ right to _* _

endfth
TL & D-Computads

• **Main Ingredients of Tile Logic:**
  – \((\Sigma_H, E_H)\): system configurations
  – \((\Sigma_V, E_V)\): Observations

  – Tiles: \(\alpha: \langle p \rangle_{E_H} \rightarrow \langle q \rangle_{E_H}\)

• **D-Computad**
  – Monoidal category of configurations \(H\)
  – Monoidal category of observations \(V\)
  – Tiles over the arrows of \(H\) and \(V\)
• In (Unconditional) RL:
  – Rewrites $\alpha$: $f(x) \Rightarrow g(x)$ can be applied
    • in any context $C[.]$: $C[\alpha(x)]: C[f(x)] \Rightarrow C[g(x)]$
    • with any argument $p(y)$: $\alpha(p(y)): f(p(y)) \Rightarrow g(p(y))$
  – Horizontal composition is total

• In TL:
  – Rewrites are synchronized via observations
    – applicable in context if $C$ accept the effect of $\alpha$
    – applicable with argument $p$ if it provides the trigger
  – Horizontal composition is partial
Tiles

- **Foundations**
  - Double monoidal categories
  - Generalization of string diagrams
  - Abstract bisimulation semantics
  - Bisimulation as a congruence
  - Simply typed double $\lambda$-calculus & double cartesian closed categories

- **Applications**
  - Process algebras
  - Open process algebras
  - Synchronized hyperedge replacement
  - Concurrent systems
  - Logic programming
  - Open systems
Tile Generalization of String Diagrams

- an example about located processes
- Fresh locations are labeled by fresh location identifiers \( \lambda \)
- if two processes share the same location and one of them creates a fresh location
- then its subprocesses will be placed on the new location,
- whereas the other process will remain linked to the old location.
Roadmap

Networks

- Milner flowgraph algebras
  - Denotational process algebras
  - (Soft) constraint networks
- Networks as components & connectors
  - Petri nets
  - Signal flow graphs
  - Electric circuits
  - PROPS: product permutation categories
- From graphs to categories
  - Petri nets
  - Rewriting logic
  - Process calculi
- Conclusions
Conclusion

- Towards a general theory of networks:
  - John Baez, Caltech
- Additional kinds of networks
  - Electric circuits
  - Neural networks
  - Bayesian networks: recent work by Bart Jacobs and Fabio Zanasi
  - Proof nets
  - Feynman diagrams
- Additional views
  - nondeterministic, concurrent, probabilistic, stochastic, quantum, combined
- Additional interpreted domains
  - Cyberphysical systems
  - Hardware and software architectures
  - Heterogeneous systems
THE END