ascens

www.ascens-ist.eu

ASCENS

Autonomic Service-Component Ensembles

D2.4: Fourth Report on WP2

Foundational Frameworks for Autonomic Computational Models

Grant agreement number: **257414**
Funding Scheme: **FET Proactive**
Project Type: **Integrated Project**
Latest version of Annex I: **Version 3.0 (29.4.2014)**

Lead contractor for deliverable: **UNIPI**
Author(s): **Saddek Bensalem & Jacques Combaz (UJF-Verimag), Roberto Bruni (ed.) & Fabio Gadducci (ed.) & Ugo Montanari & Matteo Sammartino & Giacoma V. Monreale & Alain Tcheukam Siwe (UNIPI), Niklas Hoch (VW)**

Reporting Period: **4**
Period covered: **October 1, 2013 to March 31, 2015**
Submission date: **March 12, 2015**
Revision: **Final**
Classification: **CO**

Project coordinator: **Martin Wirsing (LMU)**
Tel: **+49 89 2180 9154**
Fax: **+49 89 2180 9175**
E-mail: **wirsing@lmu.de**

Partners: **LMU, UNIPI, UDF, Fraunhofer, UJF-Verimag, UNIMORE, ULB, EPFL, VW, Zimory, UL, IMT, Mobsya, CUNI**

SEVENTH FRAMEWORK PROGRAMME

Executive Summary

This deliverable reports on the WP2 activities that have been conducted during months 37–48.

In Task 2.1 we further pursued the investigation on the theoretical foundations of the distributed implementation of connectors, including normal forms representation for stateful connectors and distributed implementation of BIP systems based on asynchronous message passing. We further continued the study of NCPi in the context of the Science Cloud case study by giving a full modelling of peers and Distributed Hash Tables (DHTs), and proving routing convergence.

In Task 2.2 we focussed mostly on the soft constraints paradigm. A more theoretical investigation concerned the algebraic structure and properties of soft constraints with bipolar preferences. A more applicative outlook lead to the development of a technique based on the orchestration of declarative and procedural knowledge for solving locally optimization problems that are global.

In Task 2.3 we kept on investigating about issues related to game semantics for agents, focusing on energy trading scenarios with virtual prosumers. In particular we compared the notion of aggregator proposed by the ADDRESS project with a similar notion developed within our approach. Furthermore, we developed coalgebraic semantics for operational models with causal information.

The deliverable is organized around the above structure of themes. Each subsection includes a short description of the novelty and offer pointers towards future work. A concluding section offers some general remarks on the overall satisfaction of the objectives, and on the influences from and towards other work packages.

Contents

1 Introduction

The three tasks of Work Package 2 span the whole duration of the Project; indeed, during months 37–48, we have been working on all of them. We summarize below our contributions to Tasks 2.1, 2.2 and 2.3. For each task the activities are roughly divided along either one or two main strands of research, as outlined here. A more detailed description of each strand and each contribution can be found in subsequent sections.

1.1 Task 2.1 (Resource-aware operational models)

The overall aim of the task is concerned with "the study of resource-aware infrastructures and networking middleware modeled in terms of advanced components, glues and connectors which can support different levels of guarantees, reliability, dynamicity and integration to heterogeneous components" (cf. part B of Project proposal). Along those lines, also for the fourth year we further pursued the two main lines of research that were already identified in the first year.

Concerning the strand on *Foundations of resource-aware connectors*, we further tackled the study of component-based frameworks (see Section 2.1). On the one side, building on mutual correspondence results between BIP, Petri calculi and Petri nets with boundaries reported in [BMS15a], we provide a normal form representation for stateful connectors (Section 2.1.1). On the other side we study a model-to-model transformation to deploy fully distributed BIP implementation, where components communicate only through asynchronous message passing (Section 2.1.2).

The activity in the second strand concerned the *Foundations of resource-aware languages and models* (see Section 2.2). Differently from the previous years, where the emphasis was on the development of novel frameworks, we further explored the preliminary results reported in Deliverable 2.3 on the formal modelling and analysis of the p2p architecture PASTRY, from the Cloud case study, for which we were able to prove routing convergence (Section 2.2.1).

1.2 Task 2.2 (Adaptive SCs: building emergent behavior from local/global knowledge)

This task is concerned with the "develop[ment of] robust mathematical foundations for interaction scenarios [...] address[ing] models that can favor a mixture of static and dynamic analysis tools [...]" (cf. part B of Project proposal). During the second year we introduced a novel strand of research on the topic, under the title *Enhancements of conceptual models for autonomicity* (see Section 3.1), which focussed on the soft constraint paradigm, either with some foundational research (e.g. on modularity) or by showing different uses as a widely applicable specification technique (and reported in many Year 2 and Year 3 deliverables). Since then, soft constraints have become the main facet of this task.

During the fourth year, a more theoretical research focussed on extending the soft framework to bipolar preferences, i.e., such that the combination of two preferences does not necessarily worsen the quality of the solution, in order to enlarge the range of possible applications (Section 3.1.1). A more application-oriented research further developed a technique based on the orchestration of declarative and procedural knowledge for solving locally global optimization problems, as originally reported in Deliverable D2.3 (Section 3.1.2).

1.3 Task 2.3 (Modeling SCEs with collaborative and competitive behavior)

The research on this task has the ambition of "develop[ing] a theory combining as much as possible the flexibility and compositionality of computer science semantic models with the expressiveness of models developed by economic theorists", possibly with an in-depth analysis of the "adapt[ion] and

re-use in this context many [of] coalgebraic tools well-known for ordinary transition systems" (cf. part B of Project proposal). Also this task has two different prongs.

We first report on the application of *Game paradigms for service composition* (see Section 4.1). The work carried out on previous years related to the efficient modeling of peer-to-peer energy management, in general, and to the DEZENT virtual prosumers (producers-consumers) simulator, in particular, has been exploited in the fourth year to focus on the concept of *aggregator*: a new market player whose aim is to reduce the energy cost during peak consumption while maximising profit (Section 4.1.1).

We then describe more foundational contributions on *Coalgebraic techniques for dynamical systems*, pushing the coalgebraic view of causal semantics. Causality information is particularly important for debugging component ensembles from unwanted behaviour, e.g. by tracing back the causes of the malfunction. Yet the state space of operational models with causal information tends to suffer from unbounded growth. We show that by regarding causal relation as *resources* we can apply coalgebraic techniques to derive History-Dependent automata which, in many cases, admit minimal, finite-state representatives. The generality of the construction is witnessed by its application to two very different models: causal trees and Petri nets (see Section 4.2.1).

2 On Task 2.1: Resource-aware operational models

Likewise previous deliverables, the advances occurred during months 37-48 in Task 2.1 are mainly concerned with two lines of research, namely *Foundations of resource-aware connectors* and *Foundations of resource-aware languages and models*, both of them started during the first year.

Concerning the first strand, we further tackled the foundational and practical issues for component-based frameworks such as BIP. On the more theoretical side, we report on the study of finding canonical representations for stateful connectors. The focus is on Petri calculi and Petri nets with boundaries, whose equivalence to BIP was already reported in Deliverable 2.1. On the more practical side we discuss a fully distributed implementation of BIP.

Concerning the second strand, the efforts of the fourth year were spent in applying the theoretical body of work layered in previous years to the Science Cloud case study. In this strand, one main result is concerned with the *Network Conscious π-calculus* (NCPi), a network-aware extension of π-calculus introduced already in the first year and advanced in the later ones. In the fourth year we used NCPi to reason about the p2p architecture PASTRY.

2.1 Strand on "Foundations of resource-aware connectors"

This strand addresses the enhancement of classical models of connectors with new features for tackling the dynamism of component ensembles and more efficient distributed realization.

In Deliverables D2.1, D2.2 and D2.3 we have reported on the overview and comparison of some notable theories of connectors, as well as the mutual embeddings and dynamic extensions.

During the fourth year of the project, we have investigated the possibility of finding canonical representations for stateful connectors whose state space is finite. The result builds on the correspondence between algebras of connectors such as Petri calculi and the operational model of Petri nets with boundaries. The back-and-forth transformation is exploited to find minimal realizations that are unique up to isomorphism.

Another focus of the fourth year work has been on the development of a methodology for transforming high-level BIP models into distributed models that are directly implemented, accounting for correctness, optimization and efficiency. In particular, high-level BIP specifications are first transformed to an intermediate level code, called Send/Receive BIP, which is then used to generate C++

code. The code generation has been tested on a high-level model for a robotics scenario. The distributed code was then validated on simulation, where processes implementing the robots were running on different machines and communicating through a network. Experimental results have shown that the generated code reached the goal fixed by the high level specification.

2.1.1 A normal form for stateful connectors

Component-based design is a modular engineering practice that relies on the separation of concerns between coordination and computation. Component-based systems are built from loosely coupled computational entities, the *components*, whose interfaces comprise the number, kind and peculiarities of communication ports. The term *connector* denotes entities that glue the interaction of components [PW92] by imposing suitable constraints on the allowed communications. The evolution of a network of components and connectors (just *network* for short) is as if played in rounds: At each round, the components try to interact through their ports and the connectors allow/disallow some of the interactions selectively. A connector is called *stateless* when the interaction constraints it imposes are the same at each round; *stateful* otherwise.

Stateless connectors. In the case of the algebra of stateless connectors [BLM06], terms are assigned input-output sorts, written $t : (n, m)$ or also $t : n \to m$, where n is the arity of the left-interface (input) and m of the right-interface (output) and they are constructed by composing in series and in parallel five kinds of basic connectors (and their duals, together with identities $\mathsf{I} : 1 \to 1$) that express basic forms of (co)monoidal synchronisation and non-determinism: symmetry, synchronisation, mutual exclusion, hiding and inaction. The symmetry $\mathsf{X} : 2 \to 2$ is self-dual and express the connector that re-wires the ports exchanging their order. It provides the basis for building connectors that permutate in any possible way the order of ports in the interface. The synchronisation $\nabla : 1 \to 2$ (duplicator) and its dual $\Delta\colon 2 \to 1$ (co-duplicator) constrain the ports to be synchronised together, i.e., an action happens on all of them or inaction is enforced. They provide the basis for interactions involving an arbitrary number of ports. The mutual exclusion connector $\wedge : 1 \to 2$ (mutex) and its dual $\vee : 2 \to 1$ (comutex) constrain the only port on one hand to synchronise with exactly one of the ports on the other hand or inaction is enforced. They provide the basis for expressing non-deterministic selection of interactions. The hiding $\bot : 1 \to 0$ and its dual $\top : 0 \to 1$ hide away one port from the interface, without imposing any constraint on the actions that can take place on it. It is neutral when composed with (co)duplicator. The inaction $\downarrow\colon 1 \to 0$ and its dual $\uparrow\colon 0 \to 1$ enforce inaction on one port while removing it from the interface. It is neutral when composed with (co)mutex. The parallel composition $t_1 \otimes t_2$ of two terms $t_1 : (n_1, m_1)$ and $t_2 : (n_2, m_2)$ has sort $(n_1 + n_2, m_1 + m_2)$ and corresponds to place the two connectors side by side, without interaction constraints between them. The sequential composition $t_1; t_2$ is defined only if the right-interface k of $t_1; (n, k)$ matches with the left-interface of $t_2 : (k, m)$, so that $t_1; t_2 : (n, m)$. It corresponds to plug together such interfaces, enforcing port-wise synchronisation. It is intuitive to see that each term $t : (n, m)$ has a corresponding dual $t^c : (m, n)$ (defined recursively by letting $(t_1 \otimes t_2)^c = t_1^c \otimes t_2^c$ and $(t_1; t_2)^c = t_2^c \otimes t_1^c$) and a normal form axiomatisation is provided in [BLM06] whose equivalence classes form a symmetric strict monoidal category (PROduct and Permutation category, PROP [Mac65, HR12]) of tick-tables. Such kind of connectors are stateless, because they impose the same constraints at each round of interaction.

Stateful connectors, Petri nets with boundaries and BIP. The simplest extension to stateful connectors consists of adding one-place buffers as basic terms: $\bigcirc : 1 \to 1$ denotes the empty buffer, willing to receive a "token" when an action is executed on its left port; and $\odot : 1 \to 1$ denotes the full buffer, willing to give the "token" away when an action is executed on its right port. This way,

certain interactions can be dynamically enabled or disabled depending on the presence or absence of "tokens" in the buffers. Such stateful connectors can be put in correspondence, up to bisimilarity, with other models such as BIP [BMM11b] and a compositional variant of Petri nets equipped with interfaces, called Petri nets with boundaries [Sob10, BMM11a, BMMS13]. In particular, we note that the operational semantics of both connectors and Petri nets with boundaries can be expressed in terms of labelled transition systems whose labels are pairs (a, b) of actions a observed on the left-interface and b on the right-interface. A transition with such an observation is written $t \xrightarrow{a}_{b} t'$. In the case of connectors, states are terms of the algebra, while in the case of nets states are markings. In both cases the "sizes" of the interfaces are preserved by transitions, e.g., if $t \xrightarrow{a}_{b} t'$ and $t : (n, m)$, then $|a| = n$, $|b| = m$ and $t' : (n, m)$. Interestingly, the abstract semantics induced by ordinary bisimilarity over such transition systems is a congruence w.r.t. sequential and parallel composition. Regarding the correspondence, first, it is shown that any net $N : m \to n$ with initial marking X can be associated with a connector $t_{N,X} : (m, n)$ that preserves and reflects the semantics of N. Conversely, for any connector $t : (m, n)$ there exists a bisimilar net $N_t : m \to n$ defined by structural recursion on t. Roughly, in both cases, the one-place buffers of the connector correspond to the places of the Petri net, and the presence of tokens depends on the fact that the corresponding buffer is full or not.

The same idea can be followed when Place/Transition Petri nets (P/T nets) with boundaries are considered. In this case, places capacity is unconstrained, i.e., a place can contain any number of tokens. Correspondingly, we start from terms of the P/T Petri calculus, where the basic constructors for the empty and the full buffer are replaced by a denumerable set of constructors $(\!|n|\!)$ for any natural number n, each representing a buffer with n tokens and no capacity constraint.

Normal forms. The problem of finding canonical representatives for bisimilarity classes of stateful connectors is complicated by the fact that the number of buffers is in general not preserved by bisimilarity, as the same "abstract state" can be described by a different combination of places. As a simple example, take a net with two transitions α and β and a place p that is in the post-set of α and in the pre-set of β and has no other connection. Clearly if p is substituted by any number of places connected in the same way to α and β the overall behaviour is not changed.

When one-place buffers are considered, the solution we propose is to translate a term t to the corresponding Petri net with boundaries N_t. Then we build the case graph of N_t taking advantage of the fact that it must be finite (only a finite number of markings is possible). Moreover we observe that marking graphs can be represented up to bisimilarity by a Petri net with boundaries that has one place for each reachable marking of the original net (i.e., one place for each state of the case graph). Finally, the translation of such net to the corresponding connector gives a normal form representation of t, in the sense that any other term t' bisimilar to t will yield the same term (up to suitable permutations).

For the more general case of P/T Petri calculus, given the correspondence in [BMMS13] between P/T Petri calculus terms and P/T nets with boundaries, we can again translate a term t to the corresponding net N_t, but building the marking graph of N_t requires N_t to be bounded.[1] This is equivalent to require that only a finite set of terms are reachable from the term t via transitions. The marking graph can then be minimised and translated to an equivalent P/T Petri calculus term.

Formal details about the construction of normal forms are in [BMM15].

Novelty and future work. We have considered a calculus of connectors that allows for the most general combination of synchronisation, non-determinism and buffering. The touchstone of its generality is its ability of modeling a variety of Petri nets compositionally, up to bisimilarity. Often bisimilarity

[1] A P/T net is *bounded* if there exists a natural number k such that in any reachable marking the number of tokens in any place is less than or equal to k, i.e., the value k is a bound for the capacity of places. Note that the marking graph of a P/T net is finite if and only if the net is bounded.

implies the existence of a minimal representative, but such a construction has not been exhibited yet for Petri nets, at least directly. We interpret the case graph of a net as a transition systems labelled with the synchronizations observable on its boundaries. Then we can minimize such a LTS and reinterpret it univocally as a net and as a term of the calculus. Thus minimality is restricted to a case graph (step) semantics, which we might say observes parallelism but not concurrency. Future work might consider extending the results of this section to other more expressive semantics, observing e.g. causality as discussed in Section 4.2.1. Another direction in which our results could be extended is dealing with systems with a higher degree of dynamism, that adapt their behavior to changing environments: e.g., systems whose structure and interaction capabilities can change at runtime as proposed in [BMM14] and summarised in [BMS15a].

2.1.2 Automated model-based implementation of distributed real-time systems

Last year in Deliverable D2.3 we presented a method allowing parallel execution of Real-Time BIP models based on a (centralized) execution engine responsible for the scheduling of components and their interactions. This year we improved this approach by proposing the automatic generation of fully distributed implementations. Given a set of BIP components glued together through (multi-party) interactions, our method consists in an automated model-to-model transformation which generates a Send/Receive BIP distributed model from the input model. In the Send/Receive BIP model, components communicate only through asynchronous message passing, that is, the n-ary interactions of the input model are transformed into a protocol implemented as message exchanges. The Send/Receive model includes components corresponding to the components of the input model as well as additional components which acts as coordinators whose role is to schedule interactions according to the semantics of BIP. Compared with the method presented in D2.3, the novelty of this work is to allow the presence of several schedulers acting concurrently instead of a single centralized engine. Each scheduler (1) computes the set enabled interactions amongst the subset of interaction it is responsible for, (2) chooses an interaction and a corresponding time for its execution, and (3) notifies the components involved in the selected interaction to plan their execution.

Input model: Real-Time BIP. We assume that the system is a composition $\gamma(B_1, \ldots, B_n)$ of components B_i with respect to a set of multi-party interactions γ [ACS10]. Each component B_i is a timed automaton [AD94] having private data and whose transitions are labelled by ports P_i. A timed automaton defines the timing constraints of the corresponding component, which includes deadlines for the execution of transitions. The method presented here is restricted to timed automata having non-decreasing deadlines, that is, such that an execution of transition cannot decrease the deadlines. An interaction synchronizes transitions of a subset of components via communication *ports*. Formally, an interaction a of γ is a subset of components ports (i.e. $a \subseteq P_1 \cup \ldots \cup P_n$) containing at most one port of each *participating* components $\{B_i\}_{i \in I}$, $I \subseteq \{1, \ldots, n\}$, that is, $|a \cap P_i| = 1$ for $i \in I$, $a \cap P_i = \emptyset$ otherwise. The interaction a is *enabled* if all participating components B_i, $i \in I$, can execute a transition labelled by exactly the (only) port in $a \cap P_i$. Moreover, executing the interaction a corresponds to simultaneously executing transitions labelled by p_i in the participating components B_i.

From High-Level BIP to Send/Receive BIP. The first step for the generation of a distributed implementation from a timed system $TS = \gamma(B_1, \ldots, B_n)$ is to transform TS into a *Send/Receive* BIP model involving only binary (Send/Receive) interactions that can be directly mapped on a distributed platform. Send/Receive BIP models seek for both *(i)* correctness, that is, the Send/Receive model must execute sequences of interactions meeting the semantics of the original model TS, and *(ii)* maximal progress, that is, interactions are scheduled as soon as possible (without discarding correctness).

When trying to execute interactions concurrently we faced the problem of conflicts. Two interactions a and b are *conflicting* if they share (at least) one participating component. We denote by $a\#b$ such a situation. Clearly, conflicting interactions cannot execute in parallel because the semantics of BIP components allows only to execute transitions sequentially. Consequently, when an interaction is planned for execution, all the conflicting interactions are blocked until it executes. That is, planning an interaction a may block all the components participating in all the interactions which are conflicting with a, which may violate the timing constraints of these components. In the proposed Send/Receive model, when scheduling an interaction a we take into account not only the timing constraints of the components participating in a but also the timing constraints of the components participating in any interaction b conflicting with a. That is, for scheduling an interaction a we need to *observe* the state of each of its participating components as well as the state of each of the components involved in an interaction conflicting with a.

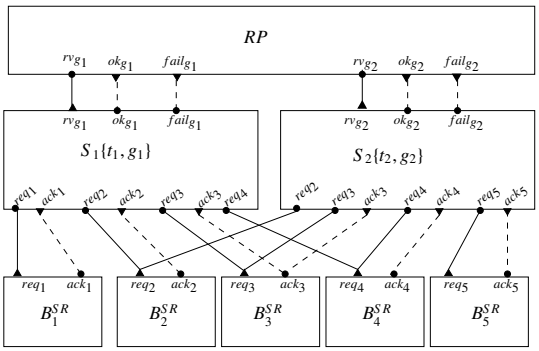


Figure 1: An example of generated Send/Receive architecture.

In the transformed Send/Receive BIP model, interactions are implemented by a protocol between the original components and a set of new components acting as schedulers, each of them being responsible for a subset of interactions. Intuitively, a Send/Receive BIP model is a set of independent components communicating through asynchronous message passing. Given a timed system $\gamma(B_1, \ldots, B_n)$ and a partition $\{\gamma_j\}_{j=1}^m$ of the interactions γ, the corresponding Send/Receive BIP model has the three layers of components (see Figure 1 for an example of Send/Receive three-layered architecture).

- The *Component Layer* consists of a transformation of each component B_i into Send/Receive component B_i^{SR}. Components B_i^{SR} send asynchronously request messages to notify the scheduler layer about their current state, and wait for acknowledge messages to know which are the next transitions to execute and when.

- The *Scheduler Layer* consists of m scheduler components, each one being responsible for a subset γ_j of interactions. Based on requests sent by components, a scheduler may decide on the execution of an interaction at a given time. If so, it sends back acknowledge messages to participating components specifying the transitions to be executed and when to execute them. Schedulers resolve *internal* conflicts locally. Internal conflicts are conflicts between interactions that belong to the same subset γ_j of interactions.

- The *Reservation Protocol Layer* resolves the conflict between the schedulers. A conflict occurs when two different schedulers try to schedule conflicting interactions. The Reservation Protocol layer implements algorithms solving the committee coordination problem, based on the idea of message-count technique [Bag89]. Conflicts are resolved by ensuring that each participating number is used only once. Various implementations of the Reservation Protocol has been

considered in [BBJ+10], including the one based on dining philosophers [CM88], which was considered in the work presented here.

Notice that a scheduler can plan the execution of interaction a at a given time only if this does not violate the timing constraints of the components participating in a, as well as the timing constraints of the components participating in interactions conflicting with a. This is a necessary condition for building correct implementations. Another correctness condition is that the delays encountered at execution (transmission time of messages, execution time of components, etc.) should allow the protocol implemented by the schedulers to execute the interactions before the components reach their deadlines.

Code generation and experiments. In a last step we automatically generate C++ code from the Send/Receive model. Send/Receive interactions are implemented using standard message passing primitives, e.g. TCP sockets. Send/Receive components of the three layers presented above are straightforwardly translated into independent C++ processes [BBCT14]. Notice that during the execution it may happen that a deadline is violated if the response from schedulers to a request from a component is received after its next deadline. This corresponds to a situation in which the execution platform is not fast enough for meeting the timing constraints of the model [ACS10], which is detected by our implementation and triggers an exception handled by application specific code.

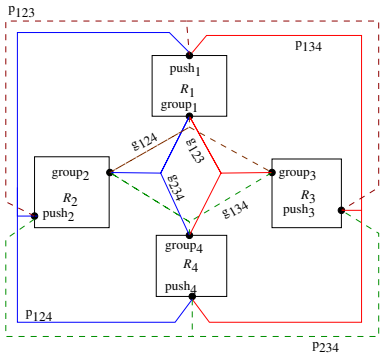


Figure 2: The BIP model for the robotics application.

To validate our approach we generated the code for a robotics application reminiscent of the scenario presented in Deliverable D7.1. It consists in a set of communicating robots that collaborate to move an object from a given location to a target destination, assuming that a single robot is not able to perform this tasks alone. Initially, the robots are randomly distributed over an arena. When three of them are sufficiently close they group themselves in order to push the object to the target destination. The tricky part when implementing such a scenario is to ensure that the robots will agree on a single group in a distributed setting. The system may fail if a robot is part of two different groups (which may go in two different directions, thus breaking the groups), or if the robots make inconsistent decisions. In our approach, the system is specified independently from the execution platform using high level primitives such as multi-party interactions. Figure 2 shows the model corresponding to the robotics application, in which the grouping of three robots is simply implemented as an interaction between the corresponding three components. From this model we automatically generated the distributed code which was validated on simulation. Processes implementing the robot were running different machines communicating through a network. Experimental results show that the generated implementation reached the goal fixed by the high level specification (see Figure 3 for an example of

a trace of simulation, where after some random exploration, three robots have met and reached the object and moved it to the target destination, while a fourth robot does not become part of the group).

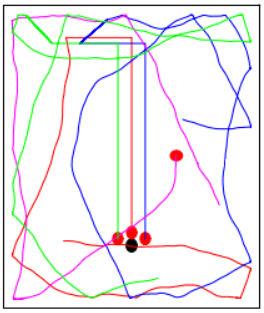


Figure 3: Simulation Results

2.2 Strand on "Foundations of resource-aware languages and models"

The strand focused on *Foundations of resource-aware languages and models* addresses the development of novel specification frameworks based on (nominal) process calculi and automata.

Resource-aware calculi are languages with an explicit notion of computational resource and with primitives for resource allocation. An explicit, run time allocation of new resources is essential for adaptivity and autonomicity, as studied in ASCENS, since the additional resources, which can possibly include new knowledge items, can dynamically improve the behavior of agents when needed. A paradigmatic example is the creation of new communication channels in the π-calculus.

Concerning this strand, the efforts of the fourth year were spent in applying the theoretical body of work layered in previous years to the Science Cloud case study. In this strand, one main result is concerned with the *Network Conscious π-calculus* (NCPi), a network-aware extension of π-calculus introduced already in the first year and advanced in the second and third year (as reported in the corresponding deliverables D2.x), in particular assessing its categorical semantics and using it for a preliminary model of p2p overlay networks in D2.3. In the fourth year the latter was extended to a full case study, by adopting NCPi to reason about the p2p architecture PASTRY.

2.2.1 PASTRY

In [MS14a] we present a NCPi [MS14b] model of the p2p architecture PASTRY [RD01]. In PASTRY, peers have unique identifiers, logically ordered in a ring. The main operation is routing by key: given a message and a target key, the message is delivered to the peer whose identifier is numerically closest to the key. PASTRY is typically used for implementing Distributed Hash Tables (DHTs), that are hash tables whose entries are distributed among peers: routing by key in this context amounts to hash table lookup: given a message and a target key, the message is delivered to the peer whose identifier is numerically closest to the key. An important property of PASTRY is *routing convergence*: messages always get to the peer responsible for the message's key. This property is informally stated in [RD01]. In [MS14a] we express it in formal terms, which allows us to prove the correctness of our model.

Peer model. The key idea is modeling identifiers as sites, and the routing table and the leaf-set of a peer as two collections of links $\mathcal{L}_{RT}$ and $\mathcal{L}_{LS}$, which form the overlay network of a peer. Notice that these links are *logical*, not physical connections: the overlay network is an abstraction of the

underlying physical one. When proving convergence of routing, we assume that failures happening at the physical level are handled locally.

A peer with identifier a is modeled as the process

$$\texttt{Peer}(a, \mathcal{L}_{RT}, \mathcal{L}_{LS}) = (O_{RT})(O_{LS})\,\texttt{Control}(a, O_{RT}, O_{LS}) \mid \texttt{RT}(\mathcal{L}_{RT}, O_{RT}) \mid \texttt{LS}(\mathcal{L}_{LS}, O_{LS})$$
$$\texttt{Control}(a, O_{RT}, O_{LS}) = \texttt{JoinH}(a) + \texttt{Route}(O_{RT}, O_{LS})$$

Processes RT and LS allow querying and modifying routing table and leaf-set. These operations are called internally via the names in O_{RT} and O_{LS}. The process Control implements the control logic of a peer, JoinH executes the distributed protocol for node join, ending up with the creation of new links from/to the joining peer, and the process Route makes the overlay network of a available to applications. We prove that the routing convergence is preserved after each execution of the node join protocol.

DHT model. Now we want to model routing behavior for a simple Distributed Hash Table (DHT), where observations are routing paths of DHT lookups. In order to do this, we introduce a new type of link: $a \trianglerighteq k$ means that the peer with identifier a is responsible for the key k.

We can model a Distributed Hash Table over a PASTRY system made of peers $a_1, \ldots, a_n$ as follows. Suppose the DHT has m key-value pairs $\langle k_i, v_i \rangle$, and let a_{k_i} be the identifier of the peer responsible for k_i, i.e., the closest to k_i among $a_1, \ldots, a_n$.

$$\texttt{DHT} = \texttt{Peer}(a_1) \mid \cdots \mid \texttt{Peer}(a_n) \mid \texttt{H} \qquad \texttt{H} = \texttt{Entry}(k_1, v_1, a_{k_1}) \mid \cdots \mid \texttt{Entry}(k_m, v_m, a_{k_m})$$
$$\texttt{Entry}(k, v, a) = a \trianglerighteq k \mid k(b).\overline{a}bv.\texttt{Entry}(k, v, a)$$

Here H represents the DHT content as the parallel composition of processes that handle the table's entries. The idea is implementing a DHT lookup request for a key k as a message with destination k, carrying the identifier b of the sender. Upon receiving this message, the handler $\texttt{Entry}(k, v, a)$ for $\langle k, v \rangle$ replies to b with a message containing v. Notice that node joins in PASTRY may introduce new key-value pairs in the DHT. However, for simplicity, we assume that the DHT is fixed. The addition of a key k with value v could be modeled by using the routing mechanism to find the peer with id a_k closest to k, and then spawning a new process $\texttt{Entry}(k, v, a_k)$. In [MS14a] we show that routing convergence holds also: in this context, such property says that DHT lookups always get to their destinations.

Novelty and future work. This work applies the Network Conscious π-calculus, presented in Deliverable 2.2, and has been introduced in Deliverable 2.3. A description of the PASTRY model can also be found in [BMS15a]. The Cloud Computing case study is related to our DHT model, because DHTs are used to handle its storage service. The novelty of this year is the streamlining of the model and its publication in [MS14a].

3 On Task 2.2: Adaptive SCs: building emergent behavior from local/global knowledge

During the fourth year, the work on this task focussed even more on soft constraints techniques, for which the novel strand on *Enhancements of conceptual models for autonomicity* was introduced since the second year. Foundational issues were tackled in terms of adding expressive power to the formalism by moving the preference domains from absorptive semirings to residuated monoids. This subsumes the work on bipolar preferences already proposed in the literature, while still recovering

the most used heuristics (local consistency and bucket elimination) for reducing search complexity. Moreover, during the third year, in a more application-oriented research, a technique based on the orchestration of declarative and procedural knowledge has been developed for solving locally global optimization problems. The work reported in Deliverable 2.3 has been expanded by introducing both a textual and a graphical notation for optimization problems, and dynamic programming algorithms for variable elimination that is based on an exploration of such representations.

3.1 Strand on "Enhancements of conceptual models for autonomicity"

While classical constraint satisfaction problems (CSPs) search for the boolean assignment of a set of variables that may satisfy a family of requirements, its soft variant extends the (possibly ordered) domain of assignments, thus modeling preferences: the aim is to provide an environment where suitable properties (e.g. on constraint propagation) could be proven and inherited by all the instances.

Soft technology focussed mostly on structures modelling negative preferences, i.e., such that the combination of two preferences always worsens the quality of the solution. This property is underlying the earlier proposals (see e.g. [BMR97, SFV95] and [RvBW06, Ch. 9.3]), and it is the base for various extensions, such as those concerning partial inverses [CS04, BG06, Bov09]. However, the literature on CSPs has recognised the need of *positive* preferences in what is called a *bipolar* setting, see e.g [DP08] (as well as other articles of that special issue). These situations might not be dealt with by manipulating the problem to end up with negative preferences only (see e.g. [BPRV10, Section 8]), and it should be taken into account the fact that adding some values may actually decrease the overall cost. Recovering the heuristics developed for the negative case has thus been the main purpose of recent work on soft technology, starting from bipolar semiring introduced in [FW07] and adding a partial inverse operator along the lines of what has been done for negative preferences in [BG06].

ASCENS is concerned with the behaviour of systems that through interactions (also with the environment) accomplish both individual (local) and collective (global) goals by optimizing the use of resources. These systems can be modelled as a multi-criteria optimization problem whose solution is a set of values, one for each variable modelling the problem, which maximize or minimize a set of objective functions, while satisfying all the constraints of the problem. However, often a globally optimal solution may be impossible to find, and specific strategies are needed to solve them.

Building on the application of the Constraint Logic Programming (CLP) to the automotive case study via the CIAO system (as reported in Deliverables D2.2, D7.2, and JD2.2), last year in Deliverable D2.3 a technique based on the coordination of declarative and procedural knowledge has been proposed for tackling the kind of problems discussed above. It consists in decomposing the original optimization problem of the ensemble in many local problems concerning its components. Each local problem will consist of all the objective functions and constraints concerning only resources that are local to one component. Separately, each local optimization problem will be modelled and solved by using the soft CLP framework. A coordination strategy is then used to verify that the local solutions satisfy all global constraints. If it is not so, the orchestrator implementing the coordination strategy iteratively determines some parameters of the declarative implementation, which it will modify and which can lead to new optimal local solutions representing a satisfactory global solution.

This year, the work reported in Deliverable D2.3 has been expanded by introducing both a textual and a graphical notation for optimization problems, and dynamic programming algorithms for variable elimination that is based on an exploration of such representations. In particular, we introduced an algebraic specification for soft Constraint Satisfaction Problems (CSPs), similar to a process calculus, based on nominal structures (namely permutation algebras). Soft CSPs are represented as terms over the variables of the problem, consisting of the parallel composition of subproblems; atomic subproblems represent the optimization with respect to a single constraint. A key feature is the representation

of variable elimination via the restriction operator. A graphical notation for optimization problems, nominal hypergraphs, is then introduced. They are hypergraphs with an interface where names are assigned to some vertices. These vertices are variables for the overall problem. Other vertices are variables of subproblems that must be eliminated in the optimization process. We show that nominal hypergraphs can be given an algebraic structure according to the process calculus previously introduced. Moreover, we show that (isomorphic) nominal hypergraphs can be described by (congruent) terms of the specification. Finally, a bottom-up visit of a hierarchical nominal graph yields a dynamic programming algorithm with a given variable elimination strategy.

3.1.1 Residuation for bipolar preferences in soft constraints

Soft constraints refer to a family of formalisms adopted for modelling and optimising problems of a combinatorial nature. They usually take the form of a set of variable assignments, each one of them equipped with a value denoting the degree of preference for that assignment as part of the solution to the problem. In the worst case, such solution can be obtained through a combinatorial exploration of all the possible alternatives. However, many techniques for reducing search complexity have been advocated and adopted in practice.

A large body of work has been devoted to the study of algebraic structures for sets of preferences guaranteeing that those techniques can be generalised. The key idea is to consider suitable operators such that the application of e.g. a local consistency algorithm sums up to the manipulation of the values associated to a given set of assignments.

Soft technology focussed mostly on structures modelling negative preferences, i.e., such that the combination of two preferences always worsens the quality of the solution. However, the literature on CSPs has recognised the need of *positive* preferences in what is called a *bipolar* setting, see e.g [DP08] (as well as other articles of that special issue). Consider, for instance, a scenario where a manufacturer needs to buy some spare parts, and it has to check the offering of many firms. Each product unit has a price (let it be just a natural number), and clearly the main object of the manufacturer is to minimise the overall cost. A bulk sale by a producer may occur, so that a discount can be applied.

Recovering the heuristics developed for the negative case is the aim of our work. Our starting point is the bipolar semiring introduced in [FW07]: we add a partial inverse operator along the lines of what has been done for negative preferences in [BG06], the resulting algebraic structure being a residuated monoid enriched over a join semi-lattice. Our soft constraints are based on such structure. Residuated monoids have a long tradition in logic [Ono03]: they are shown here to offer the right level of abstraction for modelling bipolar preferences, since they allow to recast two of the standard techniques (local confluence and mini-bucketing) proposed in the soft constraint literature.

Monoids in bipolar preferences The first step is to define a suitable algebraic structure for modelling bipolar preferences. Our choice falls on resisted monoids, that is, six-tuples $\langle A, \leq, \otimes, \ominus, \mathbf{1}, \bot\rangle$ such that $\langle A, \leq, \bot\rangle$ is a join semi-lattice, $\langle A, \otimes, \mathbf{1}\rangle$ is a monoid, $\ominus : A \times A \to A$ is a function and 1. $\forall a \in A. a \otimes \bot = \bot$, and 2. $\forall a, b, c \in A. b \otimes c \leq a \iff c \leq a \oplus b$.

Residuation conveys the meaning of division, and it allows for generalizing the classic heuristics for local consistency. Standard soft CSP instances (i.e. *Classical*, *Fuzzy*, *Probabilistic* and *Weighted CSPs*) verify the properties of residuated monoids. They are all invertible, meaning that $\forall a, b \in A. a \leq b \implies a = b \otimes (a \ominus b)$, and for these problems the $\ominus$ operator can be used to (partially) relax constraints. Indeed, it is easy to show that our residuated monoids are tropical semirings, i.e., semirings with a sum operator $a \oplus b = \bigvee\{a, b\}$ that is idempotent. If $\mathbf{1}$ is also the top, we would end up in what are called absorptive semirings [Gol03] in the algebra literature, which in turn are known as c-semirings in the soft constraint jargon [BMR97] (see e.g. [BG06] for a brief survey on residuation

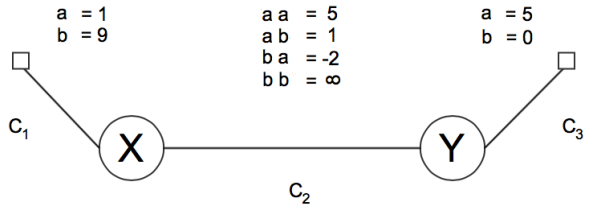


Figure 4: A soft CSP based on the residuated monoid W_m.

for such semirings). Indeed, it is precisely the lack of the latter requirement on $\mathbf{1}$ that is making our structures suitable for modelling bipolar preferences: combined with monotonicity, imposing $\mathbf{1}$ to be the top means that preferences are negative, that is, $\forall a, b \in A. a \otimes b \leq a$.

Constraint systems In the following we fix a residuated monoid $\mathbb{R} = \langle A, \leq, \otimes, \oplus, \mathbf{1}, \bot \rangle$.

Let V be a set of variables and $D = \bigcup_{x \in V} D_x$ a finite domain of interpretation for V. Then, a *constraint* $(V \to D) \to A$ is a function associating a value in A to each assignment $\eta : V \to D$.

Even if a constraint involves all the variables in V, it may depend on the assignment of a finite subset of them, its *support*. For instance, a binary constraint c with $supp(c) = \{x, y\}$ is a function $c : (V \to D) \to A$ that depends only on the assignment of variables $\{x, y\} \subseteq V$, meaning that two assignments $\eta_1, \eta_2 : V \to D$ differing only for the image of variables $z \notin \{x, y\}$ do coincide (i.e., $c\eta_1 = c\eta_2$). The support corresponds to the classical notion of scope of a constraint. An assignment over a support X of size k is concisely represented by a tuple t in D^k: we often write $c_X(t)$ instead of $c_X\eta$, and $supp(t)$ defines the set of variables assigned in t.

A *soft constraint satisfaction problem* (soft CSP or SCSP) is a triple $\langle V, D, C \rangle$, where $C \subseteq C_{V,D}$ is a finite set of constraints. The *solution* of a soft CSP $P = \langle V, D, C \rangle$ is the constraint $Sol(P) = \bigotimes_{c \in C} c$.

A simple running example A standard example of a domain of values for a soft problem is the tropical semiring $W_0 = \langle [0, \infty), \geq, +, 0 \rangle$, for $\geq$ the inverse of the standard order (thus ∞ is the bottom and 0 the top, respectively), and addition as the monoidal operator. Our proposal add capped subtraction ($a -_0 b$ is $a - b$ if positive, 0 otherwise), thus making W_0 a residuated monoid. We generalise the structure above by considering the interval $[m, \infty)$, for m any non-positive integer: we denote it as W_m.

Fig. 4 shows a weighted CSP as a graph with values from W_m. Variables and constraints are represented respectively by nodes and by undirected arcs (unary for c_1 and c_3, and binary for c_2), and values are written to the right of each tuple. The domain of the variables contains only the elements a and b. The solution of the weighted CSP of Fig. 4 associates an element to every domain value of the variables x and y. Such an element is obtained by combining all the constraints together. For instance, for the tuple $\langle a, a \rangle$ (that is, $x = y = a$), we compute the sum of 1 (the value assigned to $x = a$ in constraint c_1), 5 ($\langle x = a, y = a \rangle$ in c_2) and 5 ($y = a$ in c_3). The resulting value for this tuple is 11.

We repeat the same procedure for tuples $\langle a, b \rangle \to 2$, $\langle b, a \rangle \to 12$, and $\langle b, b \rangle \to \infty$, where the latter value means that the assignment is not acceptable. The best assignment is given by the tuple $\langle a, b \rangle$.

Soft Local Consistency Since finding the optimal solution of a soft CSP is NP-hard [RvBW06], many heuristics have been designed to tackle this inherent complexity. Inference consists in adding constraints that make a problem more explicit, changing neither the set of solutions nor their costs.

The idea behind Local Consistency algorithms is to *safely move* costs (i.e., without changing the solution) towards smaller arity constraints. Local consistency is thus a family of increasingly harder

properties about a (soft) CSP [RvBW06]. The control parameter is the size of the sub-network (i.e., the number of tuple variables) involved. The larger the tuples, the harder the property is. The simplest form of local consistency is *node consistency*, which accounts for unary constraints. The next one is *arc consistency*, which accounts for binary constraints.

Each local consistency property comes with its enforcing algorithm, which works in polynomial time. In soft CSPs, the effect of such transformation is a *move* of values. For instance, the enforcement of arc consistency moves values from binary to unary and zero-arity constraints. Enforcing algorithms are based on the concept of *local consistency rule*, as defined e.g. in [BG06].

An example of application is shown for the soft CSP in Figure 5, with $X = \{x, y\}$ and $D_x = D_y = \{a, b\}$: we move e.g. values from c_{xy} to c_x (from subfigure 1 to 3) and to c_y (from subfigure 3 to 4). Note that after some steps all the positive weights are removed.

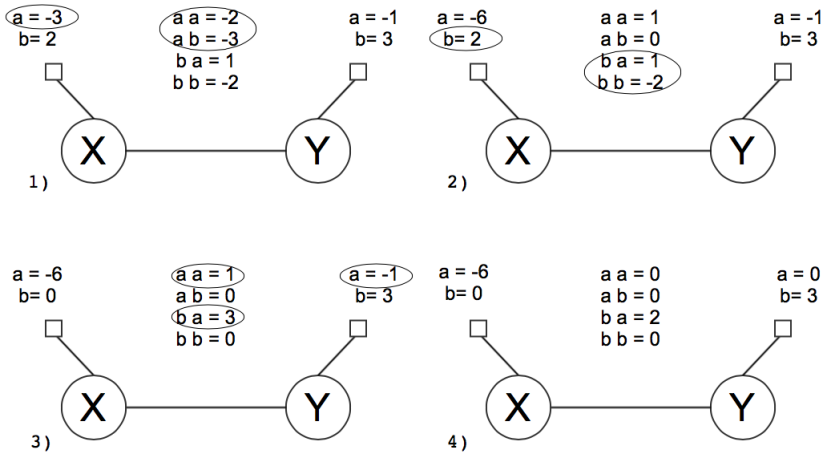


Figure 5: The original soft CSP over W_m is in 1: applying the consistency rule moves all positive preferences and as much negative preferences as possible to unary constraints.

Novelty and Future Works We propose a formalism for modelling bipolar preferences in soft constraints that includes a (partial) inverse operator of $\otimes$, along the line of [BG06, Bov09].

We have shown how choosing a residuated monoid as the domain of preferences allows for recasting two well-known heuristics adopted for negative preferences: local consistency (as shown in the running example), and bucket elimination. In general, the $\ominus$ operator can be exploited as a means to compute a difference between exact solutions and their approximation in case optimality is difficult (with the bucket algorithm) or impossible (when the preference structure is not invertible) to reach.

We plan to further investigate the properties of $\ominus$, either by relaxing its properties or by exploiting its current formulation in order to compute even better bounds for soft CPSs and to solve such problems more efficiently.

3.1.2 A coordinating approach to global optimization problems

Large optimization problems tend to be overly complex to solve and often a globally optimal solution may be impossible to find. For this reason specific strategies are needed to solve them. We propose two approaches for their solution. The first one consists in the coordination of declarative knowledge, that is the exact specification of the complete optimization problem, and procedural knowledge, that is the specific knowledge about subproblems and their, possibly approximated, resolution strategies.

We consider Soft Constraint Satisfaction Problems (SCSPs) and we introduce an algebraic specification for their representation, similar to a process calculus. Then we provide an evaluation of terms as cost functions, and we show that cost functions indeed form an algebra for the specification, where operators are optimization steps. If a scope extension axiom is removed, terms also indicate a variable elimination strategy, i.e., a solution for the *secondary optimization problem*. Evaluation of terms without such axiom can be implemented via dynamic programming with the given order of variable elimination. The procedural part concerns heuristic choices about which dynamic programming strategy should be employed and how different ad-hoc approximation heuristics could be applied. The second approach considers graph representations of optimization problems, where vertices correspond to variables and edges to tasks, whose cost depends on the values of the adjacent variables. We show that the aforementioned specification provides an axiomatization of these graphs, which allows their evaluation as cost functions. Terms without the scope extension axiom are equivalent to graphs with an explicit hierarchical structure, which also describes a dynamic programming algorithm. We apply our approaches to the e-mobility case study.

Declarative vs procedural knowledge. We present an approach to the coordination of declarative and procedural knowledge, which is particularly relevant for ASCENS. We introduce an algebraic specification for SCSPs, similar to a process calculus, based on nominal structures (namely *permutation algebras*, see, e.g., [Pit13, GMM06]). SCSPs are represented as terms over the variables of the problem, consisting of the parallel composition of subproblems; atomic subproblems represent the optimization with respect to a single constraint. A key feature is the representation of variable elimination via the restriction operator. In fact, since a restricted variable can only occur in its scope, its value can be determined when solving the subproblem it encloses.

For instance

$$(x_1)((x_2)A(x_1, x_2) \parallel (x_3)B(x_1, x_3))$$

represents a problem with three variables x_1, x_2, x_3 and two constraints A, B, sharing x_1. One can solve the subproblem $(x_2)A(x_1, x_2)$ with respect to x_2 right away, thus eliminating x_2 and obtaining a solution $A'(x_1)$ parametric with respect to x_1; similarly for $(x_3)B(x_1, x_3)$. Then x_1 can be eliminated in $(x_1)(A'(x_1) \parallel B'(x_1))$, which yields the global solution.

The specification provides axioms for modifying the structure of the problem. They are common in process algebras, and have a natural interpretation in terms of problems. For instance: subproblems can be solved in any order, the formal names of their variables are irrelevant and so on. In particular, *scope extension* (actually, scope reduction) becomes a crucial operation: it allows variables to be eliminated *earlier*, when solving a smaller subproblem. This produces a more efficient solution of the secondary optimization problem of dynamic programming.

The computation of an optimal solution is formalized as an evaluation of terms in a suitable domain of *cost functions*, giving a cost to each assignment of free variables. This domain is indeed an algebra of the SCSP specification: constructors are interpreted as operations over cost functions, performing optimization steps, and axioms become useful properties. In particular, they tell that cost functions are preserved under rearranging the structure of the problem to get a more efficient computation. Moreover, the underlying nominal structure provides a notion of *support*, i.e., the set of variables that are relevant for the computation. This is the key for a finite, efficient representation of cost functions.

We apply our approach to the e-mobility case study, where parking systems are regarded as terms of our specification, and we interpret them as cost functions telling the cost of parking cars inside or outside certain parking areas. The corresponding interpretation of the parallel composition operator becomes remarkably more complex than the classical point-wise sum of costs, but our framework is flexible enough to accommodate it. See [HMM+15] for details and an example.

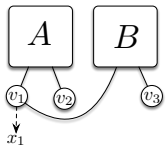
The procedural, heuristic part of our approach has a role in several steps of the solution procedure. Given the data of a parking problem, we may want to proceed exactly or with a restricted version of the available choices. Then the structure of the problem decomposition (the secondary optimization problem of dynamic programming) should be determined, taking advantage of heuristic information about the problem domain, either specified by design or acquired by employing reinforcement learning techniques. Finally, the size of the tables to be stored, which is usually the limiting factor in dynamic programming implementations, could be reduced by suitable approximation techniques.

Optimization problems as graphs. In [HMS15] we introduce a graphical notation for optimization problems, called *nominal hypergraphs*. They are hypergraphs with an *interface* where names are assigned to some vertices. These vertices are variables for the overall problem. Other vertices are variables of subproblems that must be eliminated in the optimization process. We show that nominal hypergraphs can be given an algebraic structure according to the specification presented in the previous section. Moreover, we show that (isomorphic) nominal hypergraphs can be described by (congruent) terms of the specification.

For instance, consider the optimization term

$$(x_2)A(x_1, x_2) \parallel (x_3)B(x_1, x_3)$$

Here x_1 is free, meaning that solutions are parametric with respect to x_1. The problem can be represented as a nominal hypergraph $\eta \rhd G$, that is a hypergraph G together with a partial function η, mapping some vertices of G to variable names. The graph has two hyperedges, labeled A and B, and three vertices v_1, v_2, v_3, corresponding to x_1, x_2 and x_3. Actually, only x_1 becomes an interface name, because it is the only variable the problem "exposes". Other variables are not part of the interface, meaning that they are taken up to α-conversion. The hypergraph is depicted below: the dashed line describes the domain of definition of η, namely $\eta(v_1) = x_1$.



The existence of an algebra of nominal hypergraphs allows us to recursively compute the cost function of a nominal hypergraph by performing the computation on any of the corresponding terms. However, nominal graphs still lack a description of the variable elimination strategy. We describe this information as *hierarchical* nominal hypergraphs, that are trees describing the decomposition of a nominal hypergraph in terms of nested components, each corresponding to a subproblem. Such trees correspond to terms without the scope extension axioms, i.e., where the scope of restrictions is fixed. An example of a term, and the corresponding hierarchical nominal hypergraph, is shown in Figure 6. Here the scope of each restriction determines a component in the tree, where a vertex for the restricted name is added. Notice that the root is the empty graph because all names are restricted.

A bottom-up visit of a hierarchical nominal graph yields a dynamic programming algorithm with the given variable elimination strategy. In [HMS15] we describe an algorithm for the parking optimization problem. The algorithm starts from the cost functions for the leaves, i.e., for single parking zones. Then, it performs a bottom-up visit of the tree and eliminates variables accordingly. More precisely, whenever an edge from G to G' is traversed, with G and G' discrete hypergraphs, variables $G \setminus G'$ are eliminated. In terms of cost functions, this means that the cost function of G' is computed from that of G by fixing the value of eliminated variables to "inside".

$(x_2)((x_1)A(x_1, x_2) \parallel (x_3)B(x_2, x_3))$

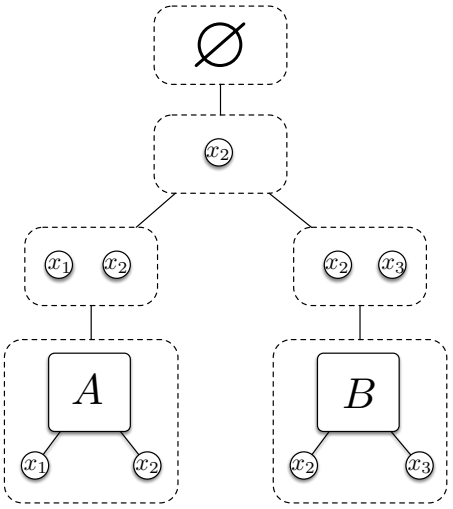


Figure 6: A hierarchical term and the corresponding hierarchical nominal hypergraph.

Novelty and future work. This work is a development of the approach presented in Deliverable D2.3. A brief overview of the declarative vs procedural approach, and of its application to the e-mobility case study, can be found in [HMM+15]. The approach based on graphs is novel.

As future work, we could integrate the implementation of the described coordination techniques into the CIAO Prolog solver of [MMH12], extended with Java routines. A simple demo where coordination of local choices for the parking problem is achieved by increasing the price of the most crowded zones has been programmed in CIAO and Java and described in [BNG+13]. Moreover, an interesting piece of future work would be to extend our approach to graphs which are incrementally modified, e.g., extended, at run time. The resulting scenario could consist of a (soft) (concurrent) constraint component together with a mobile pi-calculus-like process algebra component. A good example of this combination is cc-pi [BM07].

4 On Task 2.3: Modeling SCEs with collaborative and competitive behavior

One side of this task is concerned with the application of *Game paradigms for service composition*: after the investigation carried in the last two years on minority games for service composition, as needed by, e.g., a peer-to-peer energy management scenario, this year the model has been further put to test by checking its behaviour with respect to a distributed power market model proposed by the German project DEZENT. On a more foundational issue, *Coalgebraic techniques for dynamical systems* pushes the coalgebraic view of systems by proposing a semantics-preserving, compact representation for the causality relations, if any, holding among the different components of an ensemble.

4.1 Strand on "Game paradigms for service composition"

A fruitful application of adaptation in service component is peer-to-peer energy management, displaying different trading strategies on the energy market. The domestic energy market is changing with the increasing availability of energy generating home-devices, which provide energy for the house where they are installed as well as enabling the selling of surplus energy. After its proposal in the second year, Deliverable 2.3 reported the advances in modelling a scenario where households have the chance

to trade energy for purchasing to and for selling from a number of different actors, further including the ability to automatically retrieve data to be used in adaptive and collaborative aspects, such as those affecting the consumption of electricity. In the fourth year further experimentation took place, according to a distributed power market model proposed by the German project DEZENT and by providing a Java implementation of a controller for delaying/anticipating the production/consumption of a prosumer.

4.1.1 Aggregators in the DEZENT context of regenerative power production

Building energy infrastructures with high efficiency and renewable energy sources is an important yet challenging task for a sustainable future. In the PhD thesis by Alain Tcheukam Siwe, recently approved at IMT Lucca, the main issue is to make elastic the demand for, and the supply of, electricity of prosumers (producers-consumers), in order to optimize their energy costs based on power market conditions and on suitable constraints on their power production/consumption. The chosen distributed power market model has been proposed by the German project DEZENT. We have defined an efficient controller for delaying/anticipating the production/consumption of a prosumer. The controller aims at minimizing the electricity cost achieved at the end of a day, taking into account realistic constraints. We performed extended experimental studies based on the available DEZENT simulator and on the Java implementation of the optimal controller.

ADDRESS [BRV+08, PBB+09, BSD+10] is a large-scale integrated R&D project co-founded by the European Commission under the 7th Framework Programme, in the energy area for the "Development of Interactive Distribution Energy Networks". ADDRESS stands for Active Distribution network with full integration of Demand and distributed energy RESourceS and has been carried out by a Consortium of 25 partners from 11 European countries. The main goal of the ADDRESS project was to enable the active participation of domestic and small commercial consumers to electricity system markets and the provision of services to the different electricity system participants (see Figure 7 and Figure 8, taken from [PBB+09]). The active participation of domestic and small commercial consumers is suggested to be managed by a new market player called "aggregator" whose objective is to exploit the flexibility of power consumers for building active demand services to the power market. Figure 7 represents the ADDRESS architecture. In ADDRESS, consumers are the providers of flexibility; the aggregator, see Figure 8, is the key mediator between consumers, markets and other power system participants.

In the activity of the fourth year, we compared the notion of aggregator proposed by the ADDRESS project with the similar notion developed within our approach developed in ASCENS. The operation of balancing group managers (BGM) previously experimented within the DEZENT tool was extended in a decentralized way and our concept of aggregator was introduced as characterized by a set of virtual prosumers. Each virtual prosumer exploits the control model defined in our approach, but (s)he is neutral in the sense that (s)he essentially neither consumes nor produces energy, as (s)he can only sell in the power market the energy previously bought and stored. Actually, a virtual prosumer consumes a little amount of energy, due to the overhead of the energy storing processes. Thus the behavior of the virtual prosumer is similar to that of a rechargeable battery.

The aim of the aggregator in this model is twofold: (i) to reduce the energy cost of the consumer's population during peak energy consumption; and (ii) to maximize its profit. In order to evaluate the impact of our aggregator, we compare two types of power market situations: (i) a neutral situation in which there is no aggregator in the power market; and (ii) the active situation in which an aggregator is present. Each simulation (with respect to the type of the power market situations) was run separately and in the same conditions. The space of the experiments depended essentially on three parameters: (i) the free market power cost, which can exhibit high or low variance: for this we chose real data from

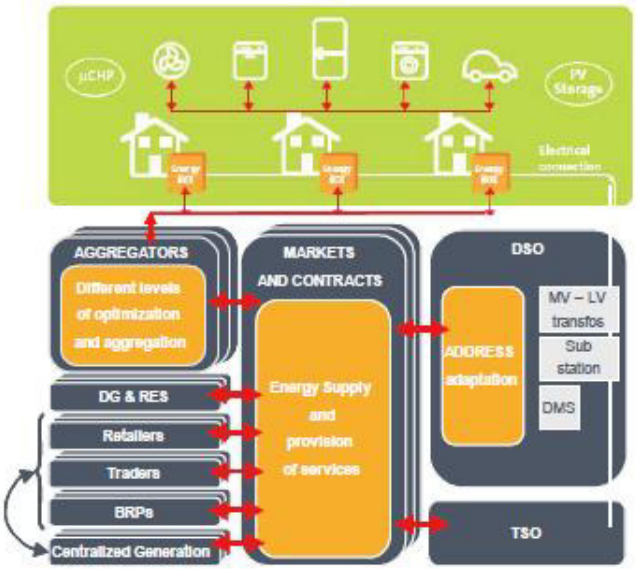


Figure 7: ADDRESS conceptual architecture.

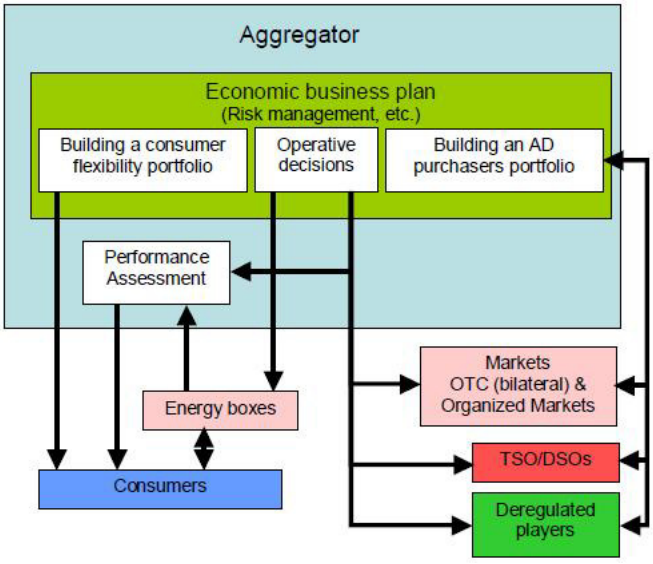


Figure 8: ADDRESS project architecture

the day ahead market prices of Switzerland (date: March 9, 2013) [eu] and Italy (date: June 18, 2013) [ita] respectively; (ii) the prosumers environment, namely heavy production or heavy consumption, in which the total amount of the electricity produced in the subnet is respectively greater than or less than the total amount needed in the subnet. In the heavy consumption situations, the additional, needed power is made available at the large power plant level, at a price which depends on the time of the day; and (iii) the available energy reserve capacity of the virtual prosumers characterizing the aggregator: it is either finite or infinite.

The experiments were conducted on the cluster at the IMT Institute of Advanced Studies Lucca. The best results of the simulations corresponded to larger differences in energy global cost, to the presence of undersupply conditions and to an infinite reserve capacity of virtual prosumers. Our results turned out to be compatible and similar to those described for the ADDRESS project [MS14c].

4.2 Strand on "Coalgebraic techniques for dynamical systems"

Autonomic, as well as adaptable and reconfigurable systems, are complex objects where aspects like feedback and stability play a key role. Describing and analyzing those aspects calls for a systemic view of systems, where classical models of non-deterministic computation might be integrated with tools from control theory. In this context, SCEs could be described and analyzed as dynamical systems. One of the involved issues is to establish the causality relations, if any, among different components of a SCE. This additional information can be useful for debugging purposes. For instance, in the robotic case-study, one could discover which event prevented a robot from accomplishing a given task. However, it is pivotal to establish tractable representations, possibly finite state, for such information. Besides its use as an off-the-shelf technique for modeling purposes, the final outcome of our research on causality would be a theoretical framework for addressing distributivity issues within SCEL.

4.2.1 Causality models of computation

Causal semantics explicitly represents causal dependencies among actions. This semantics is often given as an operational model where states are equipped with causal information that may grow along the computation. However, the unbounded growth of causal information may lead to an infinite state space. In order to construct compact, in some cases finite-state, models, we characterize causal semantics in a category of coalgebras over presheaves, in the style of Turi-Plotkin approach to the semantics of nominal calculi, where events and causal relations are regarded as "resources" and are equipped with suitable allocation operators. From such coalgebras, using a well-known categorical equivalence, we can automatically derive a History-Dependent automaton which, in many cases, has a minimal, finite-state representative. We apply this framework to *causal trees* and *Petri nets*.

Causal trees. In [DD90] a technique for deriving a causal semantics from a labelled one is introduced. The basic idea is to explicitly decorate each atomic subprocess with a set of *causes*; when an action is performed, causes of the involved subprocesses are observed and a new event is generated and incorporated in the continuation. The abstract semantics of processes is a *causal tree*, a variant of Milner's synchronization trees with enriched action labels, specifying the set of causes for each edge. The key issue is that the causal transition system is usually not suitable for verification, due to its size.

In [BMS15b] we present an approach to obtain equivalent, but more compact models for the causal semantics of concurrency. Our main inspiration is [MP97], where Montanari and Pistore tackle the issue of providing a minimization procedure for Petri Nets with a causal semantics by introducing *causal automata*. However, they rely on properties of a specific class of Petri Nets to eventually compute an ordinary labelled transition system (LTS) and minimization is performed with respect

to ordinary LTS bisimulation. Instead, our approach deals with general causal LTSs, and employs a standard categorical framework to automatically get (often finite-state) operational models for them.

Following Montanari and Pistore, we first give a set-theoretical construction that performs the following two reduction steps on causal LTSs:

(i) **Reduction of labels.** Each causal process is equipped with a partial order over its events, representing causal relations determined by past transitions. Then events that are not maximal according to the ordering, i.e., all but the most recent ones, are removed from labels.

(ii) **Reduction of states.** Only *immediate causes* of atomic subprocesses are kept, i.e., events that are maximal in the ordering with respect to at least one of the subprocesses. Intuitively, we keep causes for the most recent transitions. Then states are identified up to a suitable order-preserving notion of isomorphism, and transitions are enriched with maps that keep track of the original identity of events.

At each step we prove that behavioral equivalences are preserved.

Then we recast this construction in a categorical setting, where it becomes much more natural and simple. The key idea is to represent events as names, and the occurrence of a new event as name generation. This allows us to give a presheaf-based coalgebraic representation of causal semantics, along the lines of the Fiore-Turi approach to the semantics of nominal calculi [FT01]. In particular, we represent causal relations as a category $\mathbf{P}_m$ of partial orders over event names and suitable maps between them, and we equip it with an *event allocation operator* $\delta\colon \mathbf{P}_m \to \mathbf{P}_m$ that allocates new events and relates them to their causes in the partial order. Presheaves $\mathbf{Set}^{\mathbf{P}_m}$ express the association between causal processes and causal relations among the processes' events. We define a category of coalgebras over presheaves $\mathbf{Set}^{\mathbf{P}_m}$ where δ is employed to model event generation along transitions. This category is well-behaved, in the sense that it has a final coalgebra, whose equivalence coincides with coalgebraic bisimilarity. Here we represent the LTS of step (i) as a coalgebra on a suitable *presheaf of causal processes*, which we call *causal coalgebra*. We show that coalgebraic bisimilarity on this coalgebra is essentially Darondeau-Degano's strong causal bisimilarity, so the image of the causal coalgebra in the final one is indeed formed by causal trees.

The state space explosion issue still exists in the causal coalgebra, because the poset of a causal process keeps growing along transitions. However, for "well-behaved" presheaves, according to [CKM10], it is always possible to recover the *support* of a causal process, i.e., the minimal poset on its events. This allows identifying all (equivalent) causal processes with the same support, which in many cases yields a finite state-space, and is crucial for the equivalence between coalgebras over well-behaved presheaves and *History Dependent Automata* [MP05, CM10], that are finite automata suitable for verification. Surprisingly, for the presheaf of causal processes, computing the support amounts to removing all non-immediate causes. Indeed, we show that translating the causal coalgebra along the equivalence achieves (ii). Behavioral equivalence is preserved along this translation by construction.

Petri Nets. In [BMS14] we develop the approach presented in the previous section to construct compact, sometimes finite, operational models for causality in Petri Nets. Previous models, such as *behavior structures* [TR88], are unsatisfactory from the point of view of verification, because the causal information associated to states may grow indefinitely, leading to an infinite state space.

We model the behavior of a labelled Petri net as a *causal case graph*. Recall that a case graph is a labelled transition graph where states are markings and transitions are steps, representing many firings happening simultaneously. Actually, we consider case graphs where transitions correspond to single firings, labelled with the fired transition's label, because our causal data encode information about concurrency. A causal case graph (CG in short) has states and transitions decorated with causal

information. More precisely: a) in each state, tokens keep the set of events that led to their creation, and there is a poset describing the causal dependencies among events; b) transition labels include the set of most recent events associated to tokens that enabled the firing, and a *fresh* event, different from all those occurring in the source state, annotated with the fired transition's action label.

We define a notion of bisimilarity for CGs where causal information plays a key role: only states with the same causal dependencies among past events, namely the same poset, are compared. This fact is crucial for the equivalence with history preserving bisimilarity. Another important aspect is that transitions draw fresh events from an infinite set of event names. For each firing, we have *infinitely many* transitions in the CG, one for each possible fresh event. In this way we implement *event generation* in the same way name generation is represented, e.g., in nominal calculi.

We, then, derive three refinements of the causal CG, partly along the lines of the refinement of causal LTSs presented in the previous section:

- an *abstract CG*, where posets in states are *canonical representatives* of isomorphism classes, which allows for a *finitely branching* CG;
- an *immediate causes CG*, where we only take immediate causes;
- an *immediate causes CG with symmetries*, where we decorate each state with a *symmetry*, that is a group of poset isomorphisms that act as the identity on the state. Transitions are reduced accordingly, by taking only one representative transition among "symmetric" ones. Symmetries in states allow for the computation of minimal models, because CGs that are not isomorphic, but bisimilar under a given permutation, have a unique minimal realization, where that permutation becomes part of a state.

Then we provide categorical models of these CGs. We take as index category for presheaves a suitable category of *labelled posets* up to isomorphism, representing causal relations between events decorated with actions. Notice that the index category for causal LTSs is made of unlabelled posets: the categorical representation of labelled posets and their operations is more complex than the unlabelled case. We construct a *causal coalgebra* by translating the abstract CG. The important result is that coalgebraic and ordinary bisimilarity are equivalent. Then we introduce a notion of nominal sets over causal markings, and we show that an HD-automaton built over such a nominal set, corresponding to the immediate causes CG, can be automatically obtained from the causal coalgebra via the categorical equivalence mentioned in the previous section. Actually, the translation equips states with symmetries, so the last refinement step is also achieved.

Novelty and future work. This work follows and develops the line of research on coalgebraic operational models for resource-aware formalisms, which also includes coalgebraic models for the Network-Conscious π-calculus [MS14b] presented in Deliverable D2.2 and in [BMS15a]. Future research will be devoted to investigate coalgebraic models of formalisms with other kinds of resources and resource allocation mechanisms.

5 Concluding remarks and deliverable cross influence

This deliverable reports the advances of the last year on Work Package 2, which contributed to the foundation of different "strands" within the project. The advances of each strand have been summarised as the final paragraph of the sections detailing the topics, including some details on future activities that specifically concern each contribution. We try here instead to relate the work in each strand with the advances in other work packages. We also mention two strands from previous years on which we have not reported in this deliverable, yet have been exploited in the fourth year.

Foundations of resource-aware operational models. As planned since Deliverable D2.2, the work on connectors has focussed on the expressiveness and implementability of BIP extensions. The work aims at offering a flexible and comprehensive framework for rigorous SCEs design and it is related to the work conducted in Work Package 5, as reported in Deliverable D5.4, where BIP is used for the compositional verification of timed systems. This is made evident by the work of stateful connectors and Petri nets with boundaries, since nets are the automata-like model underlying BIP. Self-evidently, the work on the BIP extensions is going to be fed in the improvement of BIP as a suitable verification tool and as the target of the SCEL implementation, also reported in Deliverable D5.4. The relevant publications for the fourth year are [BMM14, BMS15a, BMM15, BBCT14].

The work on NCPi as reported in Deliverables D2.3 and D2.4 has been extended with a coalgebraic semantics and exploited in the context of the Science Cloud scenario: the routing reachability is part of the verification phase of EDLC. The main contributions are [MS14b, MS14a, BMS15a]

Enhancements of conceptual models for autonomicity. This topic deals with the extension of the soft constraint approach, in order to make it more flexible and expressive. Since the third year the soft constraint paradigm had been integrated with other frameworks and used in different work packages. In particular, Deliverable D2.3 proposed an observational semantics for a Linda-like formalism with ask/tell operators, while Deliverable D2.4 extends the soft constraint technology by introducing a larger class of domains for the preference values.

In Deliverable D3.4 the KnowLang reasoner adopts ask/tell operators for communicating with the knowledge basis. The observational semantics for the Linda-like formalism reported in Deliverable D2.3 can be considered also for KnowLang a reasoning technique that may help designers to impose requirements on the interaction, e.g., requiring the verification of suitable liveness properties. However, also the more general form of domain proposed here should be adopted in order to enrich the possible domain for the Knowledge basis.

The integration of soft constraints in SCEL has been under development in WP1 since the third year. The current proposal is reported in Deliverable D1.4 and [DLL+15] should benefit from both the advancements in soft constraint technology reported in Deliverable D2.3 and Deliverable D2.4

The other relevant publications for the fourth year are [HMS15, HMM+15], about dynamic programming techniques and declarative versus procedural specification, and [MS14b, BMS14, BMS15b], about coalgebraic techniques for dynamical systems. A more recent contribution is [LLM15], where a semiring structure, analogous to the one employed in soft-constraints formalisms, is used to define a simple calculus called SMuC that can be used to program and coordinate the activities of distributed agents via computational fields.

Conceptual models for autonomicity. The white-box approach based on the neat separation between application logic and adaptation logic proposed in previous years has reached a mature stage and has evolved to the CoDa (for Control Data) methodology. This proposal informed some of the pattern constructions reported in Deliverable D4.4, beside serving as a reference model for developing a specification, rapid prototyping and analysis tool written in Maude and called MESSI (for Maude Ensemble Strategies Simulator and Inquirer). The work on MESSI has also been fundamental for the development of the subsequent MISSCEL simulator exploited in the ASCENS tool-chain. This work is reported in [DLL+15].

Coalgebraic techniques for dynamical systems. The work on the coalgebraic models for stochastic systems reported in Deliverable D2.3 has provided a suitable blueprint for the theoretical foundation of the stochastic version of SCEL, reported in [DLL+15].

References

[ACS10] Tesnim Abdellatif, Jacques Combaz, and Joseph Sifakis. Model-based implementation of real-time applications. In Luca P. Carloni and Stavros Tripakis, editors, *EMSOFT*, pages 229–238. ACM, 2010.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[Bag89] R. Bagrodia, editor. *Process synchronization: design and performance evaluation of distributed algorithms*. IEEE, 1989.

[BBCT14] Saddek Bensalem, Marius Bozga, Jacques Combaz, and Ahlem Triki. Rigorous system design flow for autonomous systems. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part I*, volume 8802 of *Lecture Notes in Computer Science*, pages 184–198. Springer, 2014.

[BBJ+10] B. Bonakdarpour, M. Bozga, M. Jaber, J. Quilbeuf, and J. Sifakis. From high-level component-based models to distributed implementations. In *EMSOFT*, pages 209–218, 2010.

[BG06] S. Bistarelli and F. Gadducci. Enhancing constraints manipulation in semiring-based formalisms. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *ECAI 2006*, volume 141 of *FAIA*, pages 63–67. IOS Press, 2006.

[BLM06] Roberto Bruni, Ivan Lanese, and Ugo Montanari. A basic algebra of stateless connectors. *Theor. Comput. Sci.*, 366(1-2):98–120, 2006.

[BM07] Maria Grazia Buscemi and Ugo Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In *ESOP*, pages 18–32, 2007.

[BMM11a] Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. A connector algebra for P/T nets interactions. In *CONCUR'11*, volume 6901 of *LNCS*, pages 312–326. Springer, 2011.

[BMM11b] Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. Connector algebras, petri nets, and BIP. In Edmund M. Clarke, Irina Virbitskaite, and Andrei Voronkov, editors, *Perspectives of Systems Informatics - 8th International Andrei Ershov Memorial Conference, PSI 2011, Novosibirsk, Russia, June 27-July 1, 2011, Revised Selected Papers*, volume 7162 of *LNCS*, pages 19–38. Springer, 2011.

[BMM14] Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. Behaviour, interaction and dynamics. In *Specification, Algebra, and Software - Essays Dedicated to Kokichi Futatsugi*, volume 8373 of *LNCS*, pages 382–401. Springer, 2014.

[BMM15] Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. A normal form for stateful connectors, 2015. Submitted.

[BMMS13] Roberto Bruni, Hernán C. Melgratti, Ugo Montanari, and Pawel Sobocinski. Connector algebras for C/E and P/T nets' interactions. *Logical Methods in Computer Science*, 9(3), 2013.

[BMR97] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of ACM*, 44(2):201–236, 1997.

[BMS14] Roberto Bruni, Ugo Montanari, and Matteo Sammartino. A coalgebraic semantics for causality in Petri nets. *J. Log. Algebr. Program.*, 2014. Submitted.

[BMS15a] Roberto Bruni, Ugo Montanari, and Matteo Sammartino. Reconfigurable and Software-Defined Networks of Connectors and Components. In Martin Wirsing, Matthias Hölzl, Nora Koch, and Philip Mayer, editors, *Software Engineering for Collective Autonomic Systems: Results of the ASCENS Project*, volume 8998 of *Lecture Notes in Computer Science*. Springer Verlag, Heidelberg, 2015.

[BMS15b] Roberto Bruni, Ugo Montanari, and Matteo Sammartino. Revisiting causality, coalgebraically. *Acta Inf.*, 52(1):5–33, 2015.

[BNG+13] Tomáš Bureš, Rocco De Nicola, Ilias Gerostathopoulos, Nicklas Hoch, Michal Kit, Nora Koch, Giacoma Valentina Monreale, Ugo Montanari, Rosario Pugliese, Nikola Serbedzija, Martin Wirsing, and Franco Zambonelli. A Life Cycle for the Development of Autonomic Systems: The e-Mobility Showcase. In *Proceedings of the 3rd Workshop on Challenges for Achieving Self-Awareness in Autonomic Systems*, Philadelphia, USA, September 2013.

[Bov09] Simone Bova. Local computation schemes with partially ordered preferences. In Claudio Sossai and Gaetano Chemello, editors, *ECSQARU 2009*, volume 5590 of *LNCS*, pages 887–898. Springer, 2009.

[BPRV10] S. Bistarelli, M. S. Pini, F. Rossi, and K. B. Venable. From soft constraints to bipolar preferences: modelling framework and solving issues. *Experimental and Theoretical Artificial Intelligence*, 22(2):135–158, 2010.

[BRV+08] Regine Belhomme, Ramon C. Real de Asua, Giovanni Valtorta, Andrew Paice, François Bouffard, Rudy Rooth, and Arturi Losi. ADDRESS - active demand for the smart grids of the future. In *CIRED Seminar: Smart Grids for Distribution*, pages 1–4, 2008.

[BSD+10] Regine Belhomme, Maria Sebastian, Alioune Diop, Marianne Entem, François Bouffard, Giovanni Valtorta, Angelo De Simone, Ramon Cerero, Cherry Yuen, Seppo Karkkainen, and Wolfgang Fritz. ADDRESS technical and commercial architecture. Deliverable ADDRESS D1.1, `http://www.addressfp7.org/`, 2010.

[CKM10] Vincenzo Ciancia, Alexander Kurz, and Ugo Montanari. Families of symmetries as efficient models of resource binding. *ENTCS*, 264(2):63–81, 2010.

[CM88] K. M. Chandy and J. Misra. *Parallel program design: a foundation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[CM10] Vincenzo Ciancia and Ugo Montanari. Symmetries, local names and dynamic (de)-allocation of names. *Inf. Comput.*, 208(12):1349–1367, 2010.

[CS04] M.C. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154(1–2):199–227, 2004.

[DD90] Philippe Darondeau and Pierpaolo Degano. Causal trees: Interleaving + causality. In *Semantics of Systems of Concurrent Processes*, pages 239–255, 1990.

[DLL+15] Rocco De Nicola, Diego Latella, Alberto Lluch Lafuente, Michele Loreti, Andrea Margheri, Mieke Massink, Andrea Morichetta, Rosario Pugliese, Francesco Tiezzi, and Andrea Vandin. The SCEL Language: Design, Implementation, Verification. In Martin Wirsing, Matthias Hölzl, Nora Koch, and Philip Mayer, editors, *Software Engineering for Collective Autonomic Systems: Results of the ASCENS Project*, volume 8998 of *Lecture Notes in Computer Science*. Springer Verlag, Heidelberg, 2015.

[DP08] Didier Dubois and Henri Prade. An introduction to bipolar representations of information and preference. *Intelligent Systems*, 23(8):866–877, 2008.

[eu] European power exchange. `http://www.epexspot.com`.

[FT01] Marcelo P. Fiore and Daniele Turi. Semantics of name and value passing. In *LICS*, pages 93–104, 2001.

[FW07] H. Fargier and N. Wilson. Algebraic structures for bipolar constraint-based reasoning. In Khaled Mellouli, editor, *ECSQARU 2007*, volume 4724 of *LNCS*, pages 623–634. Springer, 2007.

[GMM06] Fabio Gadducci, Marino Miculan, and Ugo Montanari. About permutation algebras, (pre)sheaves and named sets. *Higher-Order and Symbolic Computation*, 19(2-3):283–304, 2006.

[Gol03] J. Golan. *Semirings and Affine Equations over Them: Theory and Applications*. Kluwer, 2003.

[HMM+15] Nicklas Hoch, Giacoma Valentina Monreale, Ugo Montanari, Matteo Sammartino, and Alain Tcheukam Siwe. From Local to Global Knowledge and Back. In Martin Wirsing, Matthias Hölzl, Nora Koch, and Philip Mayer, editors, *Software Engineering for Collective Autonomic Systems: Results of the ASCENS Project*, volume 8998 of *Lecture Notes in Computer Science*. Springer Verlag, Heidelberg, 2015.

[HMS15] Nicklas Hoch, Ugo Montanari, and Matteo Sammartino. Dynamic programming on nominal graphs. *Proceedings of GaM 2015*, 2015. Submitted.

[HR12] P. Hackney and M. Robertson. On the category of props, 2012. arXiv:1207.2773.

[ita] Gestore mercati elettrici. `http://www.mercatoelettrico.org`.

[LLM15] Alberto Lluch-Lafuente, Michele Loreti, and Ugo Montanari. A fixpoint-based calculus for graph-shaped computational fields, 2015. Submitted.

[Mac65] S. MacLane. Categorical algebra. *Bulletin of the American Mathematical Society*, 71(1):40–106, 1965.

[MMH12] Giacoma Valentina Monreale, Ugo Montanari, and Nicklas Hoch. Soft constraint logic programming for electric vehicle travel optimization. *CoRR*, abs/1212.2056, 2012.

[MP97] Ugo Montanari and Marco Pistore. Minimal transition systems for history-preserving bisimulation. In *STACS*, pages 413–425, 1997.

[MP05] Ugo Montanari and Marco Pistore. Structured coalgebras and minimal hd-automata for the π-calculus. *Theor. Comput. Sci.*, 340(3):539–576, 2005.

[MS14a] Ugo Montanari and Matteo Sammartino. Network-conscious pi-calculus - a model of pastry. *Proceedings of LSFA 2014*, 2014. To appear.

[MS14b] Ugo Montanari and Matteo Sammartino. A network-conscious π-calculus and its coalgebraic semantics. *Theor. Comput. Sci.*, 546(0):188–224, 2014.

[MS14c] Ugo Montanari and Alain Tcheukam Siwe. Prosumers as aggregators in the dezent context of regenerative power production. In *IEEE SASO Workshops*, 2014.

[Ono03] H. Ono. Substructural logics and residuated lattices – an introduction. *Trends in Logic*, 20:177–212, 2003.

[PBB+09] Eefje Peters, Regine Belhomme, Carlos Battle, François Bouffard, Seppo Karkkainen, Daan Six, and Maarten Hommelberg. Address: Scenarios and architecture for the active demand development in the smart grids of the future. In *CIRED 20th International Conference on Electricity Distribution*, pages 1–4, 2009.

[Pit13] A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*, volume 57 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2013.

[PW92] Dewayne E. Perry and Er L. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17:40–52, 1992.

[RD01] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, pages 329–350, 2001.

[RvBW06] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming*. Elsevier, 2006.

[SFV95] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *IJCAI 1995*, pages 631–639. Morgan Kaufmann, 1995.

[Sob10] Pawel Sobocinski. Representations of Petri net interactions. In *CONCUR'10*, volume 6269 of *LNCS*, pages 554–568. Springer, 2010.

[TR88] Boris A. Trakhtenbrot and Alexander Moshe Rabinovich. Behavior structures and nets of processes. *Fundamenta Informaticae*, 11(4):357–403, 1988.