www.ascens-ist.eu

ASCENS

Autonomic Service-Component Ensembles

D2.3: Third Report on WP2

Models for Collaborative and Competitive SCEs, and Distributed Implementation of Connectors

Grant agreement number: **257414**
Funding Scheme: **FET Proactive**
Project Type: **Integrated Project**
Latest version of Annex I: **Version 2.2 (30.7.2011)**

Lead contractor for deliverable: **UNIPI**
Author(s): **Saddek Bensalem & Jacques Combaz (UJF-Verimag), Roberto Bruni (ed.) & Andrea Corradini & Fabio Gadducci (ed.) & Ugo Montanari & Matteo Sammartino & Tomoyuki Suzuki & Giacoma V. Monreale (UNIPI), Alain Tcheukam Siwe (IMT), Giacomo Cabri (UNIMORE), Diego Latella & Mieke Massink (ISTI)**

Reporting Period: **3**
Period covered: **October 1, 2012 to September 30, 2013**
Submission date: **November 8, 2013**
Revision: **Final**
Classification: **PU**

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 25–36.

In Task 2.1 we further pursued the investigation on the theoretical foundations of the distributed implementation of connectors, including expressiveness issues for classes of dynamic BIP connectors and further investigating the issue of parallel executions for BIP systems with time constraints. We further continued the study of coalgebraic models for our resource-aware calculus we also discussed last year, and chosen peer-to-peer overlay as a case study. Finally, we freshly tackled nominal automata, investigating the issue of regular expressions for such automata.

In Task 2.2 we focussed mostly on the soft constraints paradigm. A more theoretical investigation concerned the laws for behavioral equivalence in SCCP, the best-known language based on constraints. 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, and looking at the self-organizational aspects of an agent-based system managing the interactions among prosumers introduced early in the project. Furthermore, we pursued the study of coalgebraic semantics for the stochastic calculus introduced early in the project, whose work was previously reported in the sections of Deliverables D2.1 and D2.2 devoted to Task 2.1.

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, on the influences from and towards other work packages, and some comments on the foreseen developments for the forthcoming year.

Contents

1 Introduction

The three tasks of Work Package 2 span the whole duration of the Project; indeed, during months 25–36, 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 third 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 specific case of our main component framework, based on BIP, its dynamic extensions and its distributed implementation (see Section 2.1). On the one side, we propose suitable extensions of BIP where the components and the admissible interactions can evolve dynamically (Section 2.1.1). On the other side we study a BIP implementation where the parallel execution of BIP systems may take into account the timing constraints of the application, thus making in general terms a significant step towards the design of correct and efficient implementations of real-time systems (Section 2.1.2).

The activities in the second strand concerned the *Foundations of resource-aware languages and models* (see Section 2.2). Differently from the second year, the emphasis was less on the development of novel frameworks, and more on the subsequent development of supporting tools and techniques. On the one-side, we discussed a coalgebraic semantics for the *Network Conscious π-calculus* (NCPi), a network-aware extension of π-calculus that we introduced in the first year, and we used the language to model a software architecture for peer-to-peer overlay networks (Section 2.2.1). Also, we considered a specific resource-aware model, namely, nominal automata (where the resources to be manipulated are just names), and we developed a new family of regular expressions for capturing the recognized languages (Section 2.2.2).

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 included an investigation on the modular presentation of preference domains, followed by a brief report on the different applications of the soft constraint paradigm as a widely applicable specification technique (and reported in many Year 2 deliverables). At the end of the second year, soft constraints have become the main facet of this task.

During the third year, a more theoretical research focussed on bisimilarity for soft concurrent constraint programming (SCCP), a shared-memory language reminiscent of CCS, but whose principles can be possibly transferred to the more complex SCEL language from WP1 (Section 3.1.1). A more application-oriented research developed a technique based on the orchestration of declarative and procedural knowledge for solving locally global optimization problems (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 started during the first year related to peer-to-peer energy management has been carried on. During the first two years an agent-based system able to manage the interactions among prosumers was designed, relying on an approach coming from game theory and integrated with web services. In the third year we have focused on the self-organization aspects of the proposed system (Section 4.1.1). During year two another investigation was opened on the efficient modeling of peer-to-peer energy management, as reported in Deliverable D2.2. The distributed power market model proposed by the German project DEZENT was adopted, and an efficient controller for delaying/anticipating the production/consumption of a prosumer proposed (Section 4.1.2).

We then describe more foundational contributions on *Coalgebraic techniques for dynamical systems*, pushing the coalgebraic view of stochastic transition systems (STS) (see Section 4.2). During the first two year the State-to-Function LTSs formalism was introduced and a preliminary investigation of behavioural equivalences carried on, as reported in previous deliverables (albeit on Section 2.2). During year 3 the relationship between FuTS bisimilarity and coalgebraic bisimilarity in the context of a coalgebraic framework for FuTSs was established. Additionally, experiments of application of the FuTS framework for the semantics of discrete time process calculi and Markov automata have been performed and they proved the adequacy of the framework also for these kinds of process calculi. The work on FuTS is reported in Section 4.2.1.

2 On Task 2.1. Resource-aware operational models

Likewise Deliverable D2.2, the advances occurred during months 25-36 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 expressiveness, implementation and verification issues for BIP, as planned in Deliverable D2.2. On the more theoretical side, we report on the study of the expressiveness of suitable extensions of BIP, where different degrees of *dynamism* are introduced. The main objective is to increase expressiveness of the component framework while preserving the decidability of verification techniques. On the more practical side we report on a BIP implementation where the parallel execution of BIP systems may take into account the timing constraints of the application, a fundamental aspect for the correct and efficient implementation of real-time systems.

Concerning the second strand, the emphasis on the third year was less on the development of novel frameworks (this was the object of previous years), and more on the development of suitable tools and techniques for the manipulation of the model and languages previously introduced. In this strand, one main result is concerned with the *Network Conscious π-calculus* (NCPi), a network-aware extension of π-calculus introduced in the first year. In the third year we discussed a coalgebraic semantics for NCPi, and we used NCPi to model a software architecture for peer-to-peer overlay networks. Another main result on which we report here focuses on a specific resource-aware model, namely, nominal automata (where the resources to be manipulated are just names), for which we developed a new family of regular expressions that characterize the recognized languages. The adoption of nominal techniques has been already advocated in Deliverable D5.3 as the right level of abstraction for tackling systems

featuring complex interactions among components. While the focus there was verification, here the interest is more foundational: it lies on the use of such automata as semantic model, obtained by a suitable characterization of the associated languages.

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.

Component-based design relies on the separation of concerns between coordination and computation: the components are loosely coupled sequential computational entities that come equipped with a suitable interface (e.g., made of communication ports) and the connectors can be regarded as links between (the interface ports of) the components. Semantically, each connector can impose suitable constraints on the communications between the components it links together (e.g., handshaking). Then, the evolution of an ensemble can be seen 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, informing the components about the decision. In Deliverables D2.1 and D2.2 we have reported on the overview and comparison of some notable theories of connectors, as well as the mutual embeddings.

Due to the high dynamism of autonomic component ensembles, connectors need to be empowered with mechanisms for resource- and network-awareness, as well as adaptation, reflection and reconfigurability. In Deliverable D2.2 we have started to address the issue of connectors for reconfigurable systems, i.e, systems in which the possible interactions among components are not fully defined at design time and may change during run-time. During the third year of the project, we have investigated the expressive power of reconfigurable and dynamic versions of BIP, aiming to balance the addition of dynamism with the preservation of decidability and the applicability of verification techniques.

Another focus of the second year work has been on the development of a methodology for transforming BIP models with priorities into distributed models that are directly implemented, as well as optimized by exploiting the knowledge of the system. Correct coordination is achieved through additional components implementing conflict resolution and enforcing priorities between interactions. During the third year of the project, we have extended the parallel implementation of BIP by taking into account efficiency in addition to correctness. To this aim, the implementation has been extended to a time-based extension of BIP, where time-constraints of the applications can be taken into account and time-safety properties can be established.

2.1.1 Behaviour, interaction and dynamicity

Different rigorous frameworks are used to specify, design, analyse, compare, prototype and implement suitable connectors. Our previous efforts have been focused at unifying different frameworks, in particular, the BIP component framework [BBS06a], Petri nets with boundaries [Sob10] and the algebras of connectors [BLM06, ABC$^+$09] based on the tile model [GM00]. In [BMM11b] we have shown that BIP without priorities, written BI(P) in the following, is equally expressive to Petri nets with boundaries. Thanks to the correspondence results in [Sob10, BMM11a], we can define an algebra of connectors where a few basic connectors can be composed in series and parallel to generate any BI(P) system. The algebra has also been shown to be a suitable instance of the tile model [BMMS13].

All the above approaches deal well with systems that have static architectures, i.e., systems in which the possible interactions among components are defined at design time and remain unchanged during run-time. Nevertheless, when shifting to connectors for reconfigurable systems or systems that adapt their behavior to changing environments, the situation is less well-understood. In order to contribute to the development of such dynamic connectors, we have studied the extension of BI(P), Petri nets with boundaries and the algebra of connectors with different degrees of dynamism.

Reconfigurable BI(P). As a first step, we focus on a *reconfigurable* version of BI(P). An ordinary BI(P) component is defined by a set of ports P and a finite automaton whose transitions carry subsets of P as labels. An ordinary BI(P) system consists of a finite number of components (fixing the "Behaviour") whose ports are disjoint, together with a set of admissible synchronizations between the transitions of components (fixing the "Interaction"). Neither the set of components, nor the set of interactions can change over time. A reconfigurable BI(P) system allows for the dynamic modification of interactions among components, i.e., the set of available interactions can change as a side-effect of an interaction between components.

Our first result proves that any reconfigurable BI(P) system is equivalent to an ordinary BI(P) system. This result is achieved by introducing a "controller" component for each interaction that can be added or removed at run-time. Roughly, the controller keeps track of whether the managed interaction is currently available or not and forces the components willing to use that interaction to synchronise also with the controller. This mapping shows that the reconfiguration capabilities provided by reconfigurable BI(P) do not increase the expressive power of BI(P). In fact, reconfigurable BI(P) only provides a more compact representation of ordinary systems, while ordinary BI(P) representations may require an exponential blow up in the number of components (it requires one controller for each possible interaction, and interactions are sets of ports). The crux of the proof is the fact that the set of controller components can be defined statically. In fact, the interfaces of components in reconfigurable BI(P) are static, i.e., the set of available ports in every component is fixed. As a consequence, the set of all possible interactions in a system are determined at design time (despite the fact that they can be enabled/disabled at run-time).

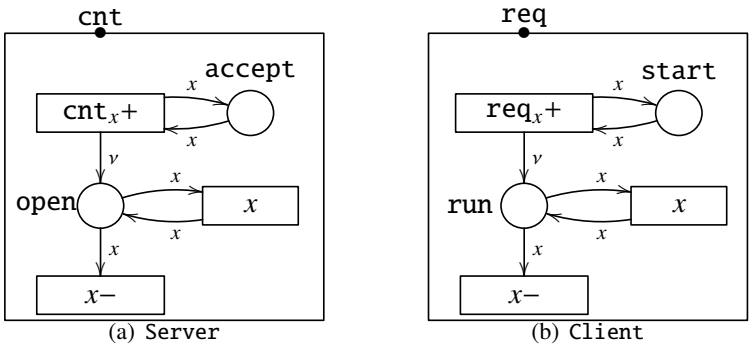
Dynamic BI(P). Our next step is to explore the more complex situations in which the interfaces of the components may change dynamically (i.e., to support the creation/elimination of ports). This requirement also imposes the necessity of handling interactions that can be created/removed dynamically. We take as an inspiring example the notion of correlation sets in web services [Vir07, LPT08], that is used to keep separate sessions between clients and servers. In these cases, when a partner call is made, then an instance of the session is initialized with suitable correlation tokens (e.g., specific message fields) gathered for the partner. Possibly many sessions are opened with the same partners involved. In subsequent interactions, correlation tokens are then exploited to identify the session that interaction is part of. When the session ends, the correlation tokens are discarded. We are confident that, to some extent, SCEL's attribute-based group communication can be handled in a similar way.

To deal with this class of systems we have introduced *dynamic* BI(P), where components may change their interfaces dynamically and new interactions can be added/removed. This allows us to model systems that are possibly infinite state and more expressive than ordinary BI(P) systems.

As an example, let us consider a server component that interacts with a possibly unbounded number of clients in separate sessions. Any session starts with a client request for a new connection. After the initial connection, each client synchronises with the server by using a dedicated, private port until the client disconnects from the server. This behaviour can be modelled as the component depicted in Fig. 1(a).[1] It has one static port `cnt`, two places `accept` and `open` with the following three transitions:

- $\texttt{accept}(x) \xrightarrow{\texttt{cnt}_x +} \texttt{accept}(x) \oplus \texttt{open}(\nu)$: if the server can accept a new connection (i.e., a token x can be consumed from the place `accept`), then it performs the action $\texttt{cnt}_x$ that creates a new dynamic port (as indicated by the symbol +). Then, the server will still accept new connections because the token x is put back to the place `accept`. Now, the component has a new session (i.e., a dedicated port) for interacting with the recently connected client as represented by the symbol ν that is a placeholder for the freshly created port name in the place `open`.

[1] The component modelling the behaviour of a client is depicted in Fig. 1(b), which is analogous to Fig. 1(a).



(a) Server (b) Client

Figure 1: Two dynamic components

- $\texttt{open}(x) \xrightarrow{x} \texttt{open}(x)$: For any open session x, the server can repeatedly perform an action on the corresponding dynamic port. This transition does not alter the set of ports of the component.
- $\texttt{open}(x) \xrightarrow{x-} \emptyset$: An already opened session x is closed after performing the corresponding action $x-$, that synchronises with a request from the client to close the same session.

We illustrate one possible run of the server with two clients in Fig. 2, where the symbol $\bullet$ denotes the initial vacuous tokens. The first transition is obtained by combining the server transition

$$\langle\{\texttt{cnt}\}, \texttt{accept}(\bullet)\rangle \xrightarrow{\texttt{cnt v+}} \langle\{\texttt{cnt}, \texttt{cnt}_\texttt{v}, \texttt{v}\}, \texttt{accept}(\bullet) \oplus \texttt{open}(\texttt{v})\rangle$$

with the following transition of the first client:

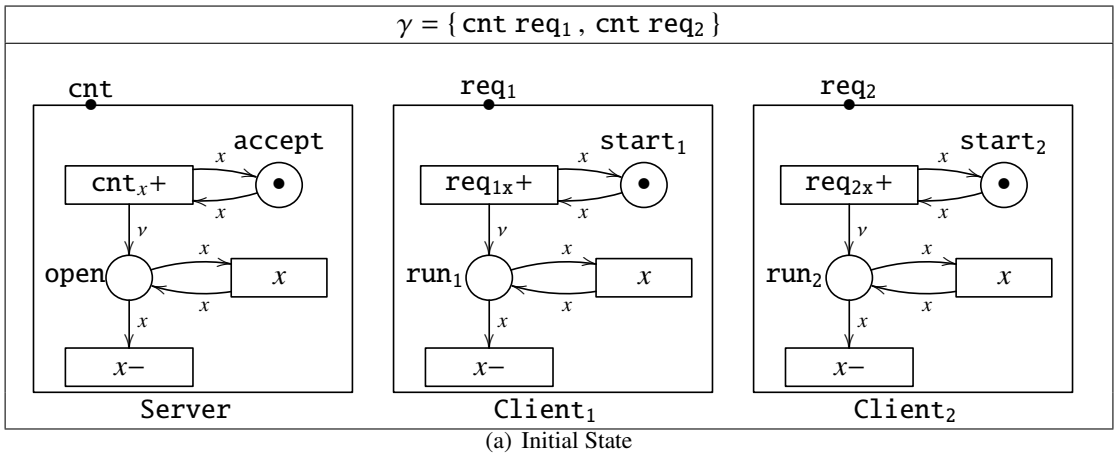
$$\langle\{\texttt{req}_1\}, \texttt{start}_1(\bullet)\rangle \xrightarrow{\texttt{req}_1\,\texttt{m+}} \langle\{\texttt{req}_1, \texttt{req}_{1\texttt{m}}, \texttt{m}\}, \texttt{start}_1(\bullet) \oplus \texttt{run}_1(\texttt{m})\rangle$$

Analogously, for the second transition. Note that suitable replicas $\texttt{cnt}_\texttt{v}$, $\texttt{cnt}_\texttt{w}$, $\texttt{req}_{1\texttt{m}}$, $\texttt{req}_{2\texttt{n}}$ of the static ports $\texttt{cnt}$, $\texttt{req}_1$, $\texttt{req}_2$ have been created locally to each component, and that the set of interactions has been enriched with suitable replicas $\texttt{cnt}_\texttt{v}\,\texttt{req}_{1\texttt{m}}$ and $\texttt{cnt}_\texttt{w}\,\texttt{req}_{2\texttt{n}}$ of the static interactions $\texttt{cnt}\,\texttt{req}_1$ and $\texttt{cnt}\,\texttt{req}_2$ together with freshly created dynamic interactions $\texttt{v}\,\texttt{m}$ and $\texttt{w}\,\texttt{n}$.

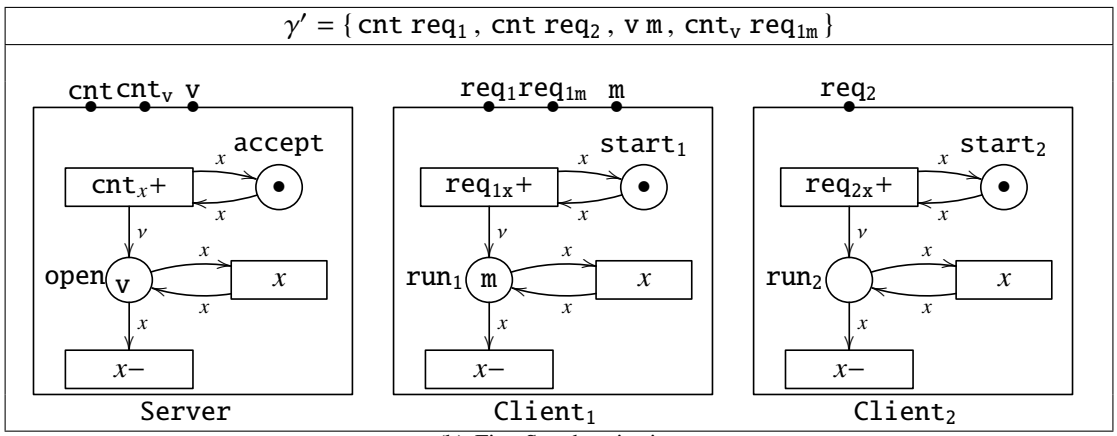
Let s denote the last state reached (Fig. 2(c)). Then, the server can interact with the clients via the interactions $\texttt{v}\,\texttt{m}$ and $\texttt{w}\,\texttt{n}$ as many times as needed, with the system remaining in the same state s:

Finally, the session between the server and the clients can be closed by combining a closing transitions of the server (label $\texttt{v}-$ and $\texttt{w}-$) with a closing transition of one of the client (label $\texttt{m}-$ and $\texttt{n}-$). Then, the set of ports of the server and of the clients are updated consequently, by removing all ports that refer to the session identifiers $\texttt{v}$, $\texttt{w}$, $\texttt{m}$ and $\texttt{n}$.

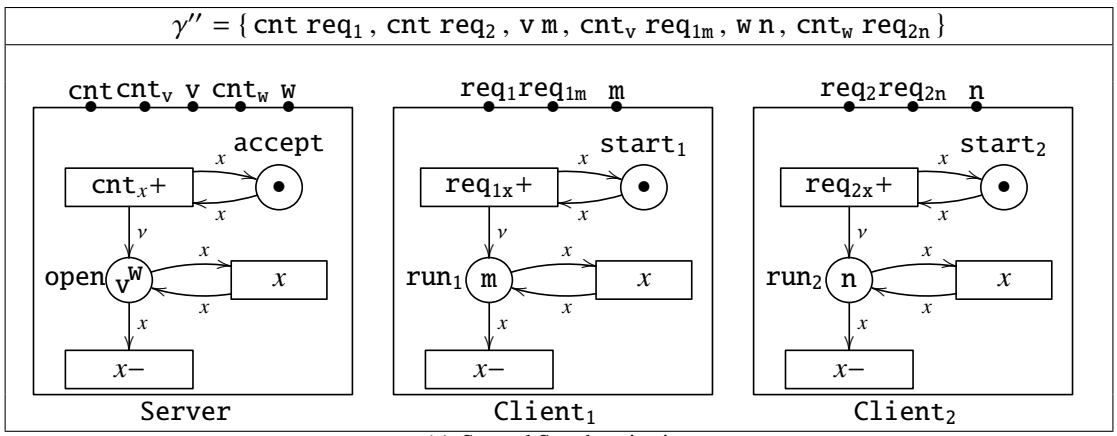
Our second main result shows that reachability is still decidable for dynamic BI(P). This is achieved by tracing a correspondence between dynamic BI(P) systems and Place/Transition (P/T) Petri nets. The result is interesting because: i) it shows that properties like reachability remains decidable and ii) it draws a nice analogy with the correspondence between ordinary BI(P) systems and safe Petri nets shown in [BMM11b]. Roughly, given a dynamic BI(P) system B we define a P/T Petri net $N(B)$ whose places are tuples of states from components in B and whose transitions represent the possible interactions. Note that $N(B)$ is determined statically and may contain more places and transitions than those strictly necessary, i.e., $N(B)$ may contain places that will never be marked as well as transitions that will never be enabled. Still $N(B)$ is finite and it is neither an ordinary automata nor a safe Petri net because: i) it may contain transitions that are attached to two output places; and ii) during a run it may produce more than one token in the same place.



(a) Initial State



(b) First Synchronisation



(c) Second Synchronisation

Figure 2: A run of the server with two clients

Novelty and future works. In this section we have reported on two suitable extensions of BI(P) with dynamically defined behaviour and interaction. The first extension, called reconfigurable BI(P), has evolved from a previous proposal of the VERIMAG research group within ASCENS. Here we prove that it is equally expressive as ordinary BI(P). The second extension, called dynamic BI(P), can be used to define systems with infinitely many states (contrary to ordinary BI(P) systems), but ensures that state reachability is decidable. Therefore, both extensions still preserve key BI(P) features in terms of analysis and verification.

In the future we plan to study the interplay between probabilities, priorities and dynamicity, possibly in the compositional setting offered by the algebra of Petri nets with boundaries [BMMS13].

2.1.2 Parallel Execution of BIP Programs

Model-based design of real-time systems usually seeks satisfaction of two types of properties: *correctness*, corresponding to the preservation of the essential properties of the application software, established under the assumption that the available resources are sufficient for running the application; and *efficiency*, referring to an optimal use of the available resources (e.g. memory, time, and energy). Existing model-based implementation techniques use specific programming models. Approaches based on logical execution time (LET) [GHKS04, HHK03, AD98] have been introduced to cope with the inherent uncertainty of the underlying platform: a program behaves as if its actions consume exactly their LET. This is achieved by reading for each action its input exactly at its release time and its output exactly at its due time. Time-safety is violated if an action takes more than its LET to execute.

Abstract Models. We improved approaches based on LET by considering more general timing constraints, such as lower bounds, upper bounds, and time non-determinism, expressed by timed automata [AD94]. A timed automaton represents the behavior of a component. It has control states and executes transitions between these states. Transitions may have timing constraints, which are defined on clocks. Clocks can be reset and/or tested along with transition execution. For instance, a transition guarded by the condition $10 \leq x \leq 100$ can execute only if the current value of the clock x belong to $[10, 100]$. More generally, timing constraints c are expressed by the following grammar:

$$c := \texttt{true} \mid \texttt{false} \mid x \leq k \mid x \geq k \mid c \wedge c \mid c \vee c \mid \neg c.$$

We also consider *urgency types* for transitions. They are used to specify the need for a transition to progress when it is enabled (i.e. when its timing constraint evaluates to true) [BGS00]. By default, transitions are *lazy*, that is, non urgent. *Delayable* transitions are urgent just before they become disabled, and *eager* transitions are urgent whenever they are enabled. Examples of timed automata are provided by Figure 3. For instance, component M_1 on the left of Figure 3 has two control states q_1^1 and q_1^2, and two transitions: one from q_1^1 to q_1^2 that resets clock x, and another from q_1^2 to q_1^1 guarded by timing constraint $[10 \leq x \leq 20]$ and having the urgency type delayable.

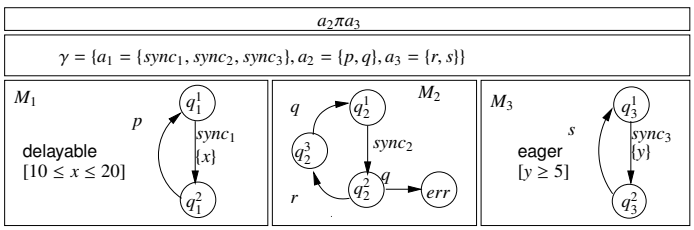


Figure 3: Example of (abstract) composition of components.

We consider that components have private data, and can only communicate when synchronizing through (multiparty) *interactions*. An interaction synchronize transitions of a subset of components via communication *ports*. For instance, transitions labelled by ports $sync_1$, $sync_2$ and $sync_3$ of respectively components M_1, M_2 and M_3 of Figure 3 are synchronized through interaction $a_1 = \{sync_1, sync_2, sync_3\}$. Figure 3 defines also interaction a_2 between ports p and q of M_1 and M_2, and interaction a_3 between ports r and s of M_2 and M_3. An interaction is *enabled* if all participating components can execute the corresponding transitions. We denote by $\gamma(M_1, M_2, M_3)$ the composition of components M_1, M_2 and M_3 for a set of interactions γ (e.g $\gamma = \{a_1, a_2, a_3\}$). Details about the formalization of such composition can be found in [TCBS13].

In addition to interactions, we also consider priorities which are partial order relations between interactions. Priorities are essential for building correct real-time systems. They allow direct expression of real-time scheduling policies used for meeting the timing constraints of the application. Very often these policies also enforce determinism, which is necessary to have reproducible execution. In Figure 3, interaction a_3 has priority over interaction a_2, which is denoted by $a_2 \pi a_3$. That is, if at a given state both interactions a_2 and a_3 are enabled, then execution of a_2 is blocked as long as a_3 is enabled. This situation occurs only if components M_1, M_2, and M_3 are respectively in control states q_1^2, q_2^2 and q_3^2. We denote by $\pi\gamma(M_1, M_2, M_3)$ the application of priority π to the composition of components $\gamma(M_1, M_2, M_3)$ (see [TCBS13] for a detailed formalization).

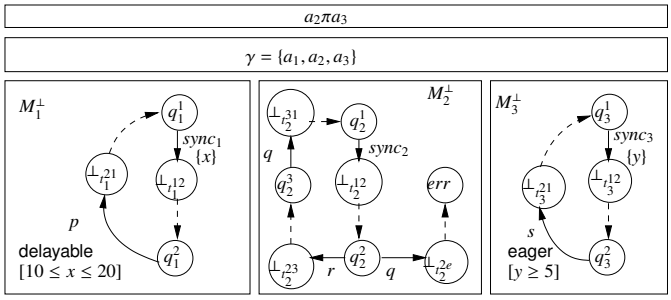


Figure 4: Physical model corresponding to the abstract model of Figure 3.

Physical Models. The operational semantics of such (abstract) composition of timed automata assumes sequential, atomic and instantaneous execution of the interactions. To represent the execution of the application on the target platform, we build physical models from abstract models so that each interaction execution is decomposed into the synchronization itself and the execution of the involved components. Physical models take into account (non zero) execution times of actions. In addition, in a physical model interactions can execute from *partial* states, that is, even if one or more components are still executing. This allows us to generate efficient implementations of real-time applications by exploiting the parallelism offered by hardware platforms (e.g. by multiprocessor or by multicore platforms). The physical model $M^{\|} = \pi\gamma(M_1^\perp, M_2^\perp, M_3^\perp)$ corresponding to the abstract model $M = \pi\gamma(M_1, M_2, M_3)$ of Figure 3 is provided in Figure 4. Notice that $M^{\|}$ and M differ only in their components, i.e. their interactions γ and priorities π remain identical. Components $M_i^\perp$ of the physical model $M^{\|}$ are computed from components M_i of the abstract model M by replacing each transition of M_i by a sequence of two consecutive transitions. The first transition marks the beginning of the execution, and the second transition marks its completion. These transitions are separated by a *partial* state denoted by $\perp$. The corresponding execution time is the time spent at state $\perp$.

Execution from partial states introduced with physical models allows the parallel execution of components and interactions. However, it may lead to incorrect execution, i.e., such that the semantics

of abstract models is not preserved. Consider again the example of Figure 4. After the execution of interaction $\{sync_1, sync_2, sync_3\}$, components M_1 and M_2 may reach respectively states q_1^2 and q_2^2 before component M_3 reaches q_3^2. This happens if execution in M_3 takes more time than those in M_1 and M_2. This situation leads to the execution of interaction $\{p, q\}$ after $\{sync_1, sync_2, sync_3\}$, as the priority $a_2 \pi a_3$ cannot apply when M_3 is not at state q_3^2 (i.e. M_3 is still in its partial state and $\{r, s\}$ is not enabled). This execution is incorrect in the physical model since it is not possible in the corresponding abstract model: in M after the execution of $\{sync_1, sync_2, sync_3\}$ there is no way to execute $\{p, q\}$.

To prevent physical models from incorrect execution, for any interaction a we define a predicate $\texttt{safe}(a)$ characterizing the states of $M^{\|}$ from which it is safe to let a execute (i.e. from which there is no potential violation of the semantics of the abstract model). Clearly, for *global* states (i.e. no component is in a partial state) these predicates satisfy $\texttt{safe}(a) = \texttt{true}$ (i.e. the behavior of physical model is already safe for global states). Ideally, from a partial state $\texttt{safe}(a)$ should evaluate to true if and only if no priority can prevent a from executing in the corresponding reachable global state. This analysis is not possible in practice for real systems due to the inherent complexity of the code associated to transitions executed by the components. Instead of using $\texttt{safe}(a)$, we use over approximated predicates that still lead to correct execution. Their computation is detailed in [TCBS13].

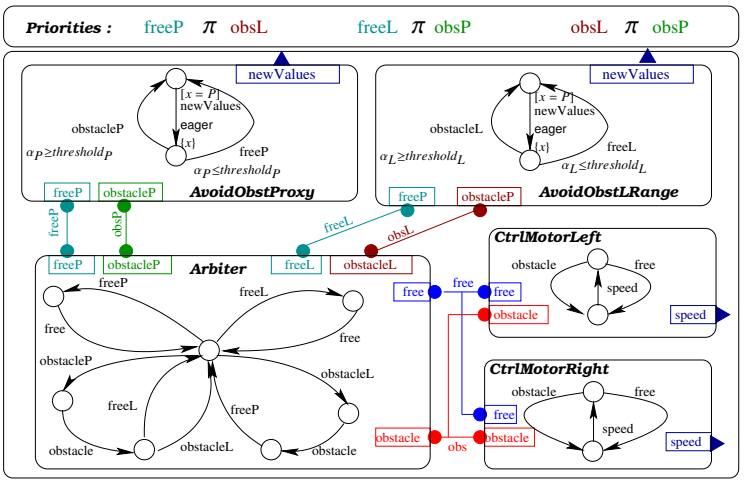


Figure 5: The abstract model of the obstacle avoidance application.

Experimental Results. This framework has been evaluated for a robotic application on the marXbot platform of ASCENS. We implemented an obstacle avoidance application given by the model of Figure 5. Using BIP [BBS06b], we generated C++ code for the components of the application targeting the marXbot platform. This code has to be executed by a dedicated runtime, the BIP engine, that implements the semantics of the composition, i.e., interactions and priorities. We compared the application running with the parallel engine based on the physical model semantics restricted to safe execution (using over approximations of $\texttt{safe}(a)$), and the same application running with the sequential engine as defined by the abstract model. The performances are measured by varying the period used for reading sensors. For each tested period, we ran the application 5 times under similar conditions. As shown in Figure 6, with the sequential engine the minimal period for a correct operation of the robot is 130 ms. For smaller periods the application is not able to meet the timing constraints. The minimal period with the parallel engine is 60 ms, which drastically improved the reactivity of the robot.

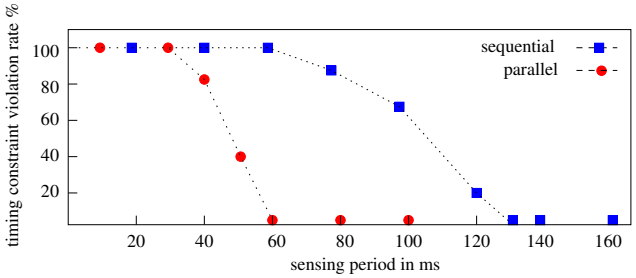


Figure 6: Results for the obstacle avoidance application running on the marXbot platfrom.

Novelty and future work. This contribution extends existing work for parallel execution of BIP systems by taking into account the timing constraints of the application. Similar techniques exist but they are limited to specific types of applications. By dealing with general form of timing constraints, we believe that we made significant steps towards the design of correct and efficient implementations of real-time systems. We are currently improving our method to have a distributed control for interactions execution instead of having a centralized engine.

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 (enriched) 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. With the purpose of devising a general modeling framework for resource-aware calculi, we developed Network Conscious π-calculus (NCPi, see Deliverable D2.1 and D2.2). We report here on the advances conducted during the third year on NCPi that are concerned with the assessment of its categorical semantics and a case study on peer-to-peer overlay networks.

Analogous idea can be transferred to automata-like models. In this case, the concept of nominal automata extends finite automata with the ability to (de)allocate resources and to compare identifiers of resources up to equality. During the third year of the project we started a line of research on nominal automata focusing on the study of the correspondence between languages recognised by an automaton and suitable resource-aware regular expressions. This work complements the nominal techniques adopted in Deliverable D5.3 for the verification of systems featuring complex interactions among components. Here the interest is more foundational, and it lies on the use of such automata as a semantic model. Due to the early stage of the current results, no specific SCEL-related example is provided, even if the resource allocation/deallocation example from Deliverable D5.3 might be adopted also in the present context.

2.2.1 Resource-aware models of computation, with applications to network routing

In Deliverable D2.1 we introduced NCPi and its concurrent variant, κNCPi [MS12]. It is an extension of the π-calculus where processes can create and use network resources (nodes and links) to communicate, and their semantics exhibits routing paths of transmitted data.

We first recap the categorical operational models of NCPi we devised. They were already introduced in Deliverable D2.2, but they have been finalized and published this year [MS13], and presented in extended form in [Sam13]. Then we describe a κNCPi model of `Pastry` [RD01], introduced in [Sam13]. This is relevant for ASCENS because routing in Science Cloud is based on `Pastry`.

NCPi categorical operational semantics. We construct two (presheaf-based) coalgebraic operational models for NCPi, one for observational equivalence and one for observational congruence, following and generalizing the approach of [FT01]. The main novelty is the treatment of complex resources, namely communication networks, where some names (links) are parametrized by other names (sites).

Here is an overview of our approach in categorical terms. We represent communication networks as a category of finite, directed multigraphs, equipped with two allocation operators that create fresh vertices and edges. Then, we construct a category of coalgebras on presheaves indexed by such graphs, where allocation happens according to the allocation operators. In this category, we model the NCPi operational semantics and characterize its observational equivalences. Then, employing categorical constructs called *right Kan extensions*, we transfer the semantics to another category of coalgebras, where bisimilarity is a congruence. These, roughly, are the steps described in [FT01].

We go a little further: we give an explicit characterization of coalgebras in the last category as *saturated* indexed transition systems [BM09], and we show that right Kan extension performs saturation (this was first observed in [BBCG12]). Finally, we show that the categorical operational semantics of our calculus can be implemented as a HD-automaton [MP05].

`Pastry` model. The dynamic nature of peer-to-peer overlay networks makes them an interesting case study for our calculus. Our reference architecture is `Pastry`, which is also used in Science Cloud. 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* (DHT), which are hash tables whose entries are distributed among peers: routing by key in this context amounts to hash table lookup.

Our `Pastry` model is as follows. We begin by formalizing the features of `Pastry` routing that ensure its convergence. These are informally stated in [RD01], but we need a rigorous formulation so that we can prove the correctness of our model. Then we give a κNCPi implementation of a `Pastry` peer. The basic idea is capturing the overlay as a collection of links between peers. Our implementation of a `Pastry` peer has two functions: handling of node joins and provision of routing services to applications. Node joins trigger a complex procedure, ending up with the creation of new links from/to the joining peer. We show that the resulting overlay still guarantees routing convergence. Finally, we model a simple DHT, and we equip it with a labelled semantics whose observations are routing paths for DHT lookups. These are derived via SOS rules that compose atomic forwarding services provided by peers. The interesting point here is that the synchronization mechanism is essentially the same as the ordinary κNCPi one. We prove that we have routing convergence also in this scenario, i.e. lookups always reach the peer responsible for the target key. Further details can be found in Section 5 of Joint Deliberable JD3.1.

Novelty and Future work. The main novelty of this year is the `Pastry` model. Moreover, the NCPi coalgebraic models presented in Deliverable D2.2 have been improved, as detailed in [MS13].

As for future work, taking as reference recent process calculi with nominal data, such as the ψ-calculi [BJPV11], we plan to investigate further applications of the presheaf-based approach. Processes in these calculi are equipped with "assertions" containing names, which can be regarded as the

knowledge of a process, and there are primitives for introducing new assertions. Assertions and their allocation primitives could be modeled as a suitable category of resources for presheaves.

2.2.2 On Algebras of Programs and Resource-Handling Automata

A crucial facet of software development and analysis is the handling of computational *resources*. The need to program the internet and the cloud transports the problem of resource handling from e.g. the realm of operating systems to the most abstract levels of software engineering. Our approach, based on earlier works [KST12b, KST12a], extends finite automata with the ability to allocate and deallocate resources and to compare identifiers of resources up to equality. We define a simple programming language with primitives for allocation (or declaration) and deallocation (or release). We show how to translate programs to automata and how to express program properties by regular expressions.

Let $\mathcal{N}$ be an infinite set of names. We first define *basic nominal regular expressions*—bNRE for short (cf. [KST12a])—given by the following grammar:

$$\mathsf{ne} \quad ::= \quad 1 \;\big|\; 0 \;\big|\; n \;\big|\; \mathsf{ne} + \mathsf{ne} \;\big|\; \mathsf{ne} \circ \mathsf{ne} \;\big|\; \mathsf{ne}^* \;\big|\; \langle_n \mathsf{ne} \rangle_n,$$

where $n \in \mathcal{N}$; bNRE extend classical regular expressions with binder constructs $\langle_n \_ \rangle_n$. Languages are computed as expected (see [KST12b, KST12a]); the binder yields words of the form $\langle n.w \rangle$, meaning that the syntactic scope of n is w.

Automata for such languages have the set of states Q equipped with a function $\|\_\| : Q \to \mathbb{N}$ giving the number of *local registers* of each state. We write $I(q)$ for $\{\star, \oslash\} \cup \{1, \ldots, \|q\|\}$.

Definition 2.1 (Basic layered automaton) *A* basic layered automaton *is a quadruple $\mathcal{H} = \langle Q, q_0, F, tr \rangle$, where Q is as above, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states (we assume $\|q\| = 0$ for each $q \in F \cup \{q_0\}$), and tr is a transition function satisfying the following: for all $q, q' \in Q$ and $\alpha \in I(q) \cup \{\epsilon\}$, $q' \in tr(q, \alpha)$ if and only if*

$$\alpha = \star \wedge \left\|q'\right\| = \|q\| + 1 \;\; or \;\; \alpha = \oslash \wedge \left\|q'\right\| + 1 = \|q\| \;\; or \;\; \alpha \notin \{\star, \oslash\} \wedge \left\|q'\right\| = \|q\| \,.$$

The language of a basic layered automaton is the set of words the automaton accepts; the class of *nominal regular* languages consists of the languages accepted by layered automata. Basic layered automata always have the corresponding regular expression and viceversa. They are closed under union, concatenation, Kleene-star, name-abstraction, intersection and resource-sensitive complementation.

Now we introduce our simple nominal programming language. We use $\mathsf{x}, \mathsf{y}, \ldots$ for program variables (we assume an ordered set *Var* of variables) and $n, m, \ldots$ for natural numbers. Symbols $+$, $-$, $\cdot$ and $\div$ denote the standard arithmetic operations, and $\leq$ is a binary relation on natural. The set of terms T, of formulas F and programs are generated by the following grammar:

$$\begin{aligned} \mathsf{t} \quad &::= \quad \mathsf{x} \;\big|\; \mathsf{n} \;\big|\; \mathsf{t} + \mathsf{t} \;\big|\; \mathsf{t} - \mathsf{t} \;\big|\; \mathsf{t} \cdot \mathsf{t} \;\big|\; \mathsf{t} \div \mathsf{t} \\ \phi, \psi, \ldots \quad &::= \quad \top \;\big|\; \mathsf{t} \leq \mathsf{t} \;\big|\; \phi \wedge \phi \;\big|\; \neg\phi \\ P \quad &::= \quad \mathbf{skip} \;\big|\; \mathsf{x} := \mathsf{t} \;\big|\; P; P \;\big|\; \mathbf{dec}\,(\mathsf{x})\,; P; \mathbf{rel}\,(\mathsf{x}) \;\big|\; \mathbf{if}\,\phi\,\mathbf{then}\,P\,\mathbf{else}\,P \;\big|\; \mathbf{while}\,\phi\,\mathbf{do}\,P \end{aligned}$$

Consider the construct $\mathbf{dec}\,(\mathsf{x})\,; P; \mathbf{rel}\,(\mathsf{x})$: $\mathbf{dec}\,(\mathsf{x})$ declares a *fresh* variable x (initialised to 0), while $\mathbf{rel}\,(\mathsf{x})$ releases it; P is the scope of $\mathbf{rel}\,(\mathsf{x})$. Other constructs are standard. A program is *well-structured* if all variables are used in their scopes.

Transition labels are pairs (ϕ, a) made of a test formula and an action. Actions can be ϵ (do nothing), $[\mathsf{t}/i]$ (substitute t into the i-th variable), $\star$ (declare the next variable) and $\oslash$ (release the last variable). If q is the transition source state, then at most the first $\|q\|$ variables may occur in ϕ and a.

Transitions out of q must satisfy two conditions: there should always be a unique outgoing transition whose test is true (*sequential condition*); for each transition (ϕ, a) to q', $\|q'\| = \|q\|+1$ whenever $a = \star$, $\|q'\| = \|q\| - 1$ whenever $a = \oslash$ and $\|q'\| = \|q\|$ otherwise (*layered condition*).

Definition 2.2 (Sequential program automaton) *A* sequential program automaton *is a quadruple $\mathcal{H} = \langle Q, q_0, F, tr\rangle$, where Q is a set of states endowed with local registers, q_0 is the initial state (with no local registers), F is a set of final states (with no local registers), tr is a transition function, and each state satisfies the sequential condition and the layered condition.*

Configurations of sequential program automata are defined as pairs (q, f) of a state q and an assignment function $f\colon \{\mathsf{x}_1, \ldots, \mathsf{x}_{\|q\|}\} \to \mathbb{N}$. If $q = q_0$ then the configuration is *initial*, if $q \in F$ then it is *final*. Since we consider only well-structured programs, which are closed terms, assignments in initial and final configurations must be the empty function $\emptyset \to \mathbb{N}$. Given two configurations (q, f) and (q', f'), a sequential program automaton $\mathcal{H}$ moves by a transition labelled with (ϕ, a) if, roughly speaking, ϕ holds when its variables are assigned by f; then the automaton moves to q' by updating the assignment f to f' according to the action a.

As nominal regular expressions of sequential program automata, we introduce the following:

$$\mathsf{nP} ::= 1 \mid 0 \mid (\phi, a) \mid \mathsf{nP} + \mathsf{nP} \mid \mathsf{nP} \circ \mathsf{nP} \mid \mathsf{nP}^* \mid (\phi, \mathbf{dec}\,(\mathsf{x}))\mathsf{nP}(\psi, \mathbf{rel}\,(\mathsf{x})),$$

where ϕ and ψ range over F and $a \in \{\epsilon, \star, \oslash\}$. As before, here we consider closed expressions only.

As a main result, we show that every well-structured program has the corresponding sequential program automaton and every sequential program automaton has the corresponding nominal regular expression. Therefore, every well-structured program has the corresponding nominal regular expression and its corresponding well-structured program is closed under α-conversion.

Novelty and future work. The research thread on nominal automata arises from the stay of Dr. Tomoyuki Suzuki in Pisa. It elaborates on previous work reported in [KST12b, KST12a], and it relates with the work on similar automata presented in Deliverable D.5.3. There, new automata for context-free nominal languages have been proposed, especially designed for modelling Dick language patterns of systems. The main focus of that research is on the study of the properties that have practical relevance in the verification of such systems. Here, instead, the focus is on the correspondence between languages recognised by an automaton and regular expressions.

Future works consider the extension of the characterization theorem to concurrent languages.

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

During the third year, the work on this task focussed on soft constraints techniques, one of its main facets, for which the novel strand on *Enhancements of conceptual models for autonomicity* was introduced in the second year. Some foundational issues were tackled in terms of adding modularity to the formalism via an operator for giving an order to preference domains. This was followed by a brief report on the different applications of the soft constraint paradigm as a specification technique (either in the extension of programming languages or in the representation of knowledge), fully reported on many Year 2 deliverables. During the third year, a more theoretical research focussed on bisimilarity for soft concurrent constraint programming (SCCP), a shared-memory language reminiscent of CCS. In a more application-oriented work, a technique based on the orchestration of declarative and procedural knowledge has been developed for solving locally global optimization problems.

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 could be proven and inherited by all the instances.

Moving from its roots on the specification of optimisation problems, (soft) constraints have been adopted for specification purposes into (either declarative or procedural) programming languages. Similarly, their use has been advocated as a tool for knowledge representation and manipulation.

The best known language based on constraints is possibly the (Soft) Concurrent Constraint Programming (SCCP): a shared-memory language which includes a procedural part, based on guarded primitives like ask and tell [SRP91, BMR06]. The language is reminiscent of classical process calculi such as Milner's CCS, and current work concerns the embedding of the relevant parts of the SCCP paradigm into SCEL. Even if SCCP operational semantics was defined early on, so far an exploration of the behavioural equivalence was missing. We envision two payoffs for such an exploration. First of all, the issue of behavioral equivalence is still an open one for the calculi developed inside WP1. Since the shared memory paradigm adopted by SCCP is clearly reminiscent of the one adopted by SCEL, developing a theory for the former should provide insights into the latter calculus. Second, note that the minimal primitives of SCCP are also included into the KnowLang reasoner (to be used as an external reasoner for SCEL). The actual work on deriving an equational presentation of SCCP bisimilarity may help to develop suitable proof techniques for KnowLang. This section reports some preliminary results, still with no example, which are deferred to a later moment with a more developed theory (possibly including an equational presentation) of the bisimilarity equivalence.

ASCENS is concerned with the behaviour of systems that through interactions among them and 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, these problems tend to be complex to solve and 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 to the automotive case study via the CIAO system reported last year in Deliverables D2.2, D7.2 and JD2.2, 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 local to one component. Separately, each local optimization problem will be modelled and solved by using the soft constraint logic programming 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. The solution is indeed guaranteed to be just an acceptable global solution: it may or may not be globally optimal.

3.1.1 Saturated bisimilarity for soft constraints

The work in [SRP91] established a denotational semantics for CCP and an equational theory for infinite agents. More recently, [ABP+11] was able to show that the equational axioms were underlying a specific weak bisimilarity among agents, thus providing a clear operational understanding. The key ingredient was a complete lattice as the domain of the store, with least upper bound for constraint

$$A ::= \ \mathbf{stop} \ \Big| \ \mathbf{tell}(c) \ \Big| \ \mathbf{ask}(c) \to A \ \Big| \ A \parallel A \ \Big| \ \exists_x A \ \Big| \ Z \ \Big| \ \mu_Z A$$

Table 1: The syntax of deterministic soft CCP

combination. The soft version introduced in [BMR06] dropped the upper bound for combination in exchange of a more general monoidal operator. Thus, the domain was potentially just a (not necessarily complete) partial order, possibly with finite meets and a residuation operator (kind of inverse of the monoidal one) in order to account for algorithms concerning constraint propagation. Indeed, the main use of soft constraints has been as a generalization of classical constraint satisfaction problems, hence the emphasis on e.g. local consistency techniques [RvBW06, Section 3], and the lack of investigation about e.g denotational semantics.

The aim of the work is to connect the soft and the crisp paradigm by investigating both a denotational and an operational semantics for soft CCP. The domain will be a mix of those investigated in the two communities, namely, an infinitary monoid whose underlying set of elements form a compete lattice. After recalling some basic notion, we present a first correspondence result between the operational semantics and the semantics based on observable stores and fair computations.

Basic notions. We now move our attention to the domain of values we are going to consider. We will consider as known by the reader complete lattices (CLs) and commutative monoid with identity (IMs), possibly with infinite composition $\bigotimes_{i\in I} a_i$ for I a (possibly countable) set of indexes.

Definition 3.1 (CL-enriched IM) *A CL-enriched IM (CLIM) is a four-tuple $\mathbb{S} = \langle A, \leq, \otimes, \bot\rangle$ such that $\langle A, \leq\rangle$ is a CL, $\langle A, \otimes, \bot\rangle$ is an IM, and furthermore it holds*

- $\forall a \in A. \forall X \subseteq A.\, a \otimes \bigwedge X = \bigwedge\{a \otimes x \ \big| \ x \in X\}$

Recall that an element $a \in A$ is $\otimes$-compact (or $\otimes$-finite) if whenever $a \leq \bigotimes_{i\in I} a_i$ then there exists a finite subset $J \subseteq I$ such that $a \leq \bigotimes_{j\in J} a_j$.

Definition 3.2 (constraints) *Let V be a (possibly ordered) set of variables; and let D be 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 $\eta : V \to D$ of the variables.*

The set of constraints (with finite support) clearly forms a CLIM, where the structure is lifted from $\mathbb{S}$. Furthermore, it enjoys some *cylindric* properties. Indeed, the cylindric CLIM of constraints $\mathbb{C}$ is defined as $\langle C, \leq, \otimes\rangle$ such that

- $c_1 \leq c_2$ if $c_1\eta \leq c_2\eta$ for all $\eta : V \to D$
- $(c_1 \otimes c_2)\eta = c_1\eta \otimes c_2\eta$ for all $c_1, c_2 \ \in C$
- $(\exists_x c)\eta = \bigwedge_{d\in D} c\eta[x := d]$ for all $c \in C, x \in V$

Deterministic Soft CCP We now introduce our deterministic language. In the following we fix a cylindric CLIM of constraints $\mathbb{C} = \langle C, \leq, \otimes\rangle$, for $C^{\otimes}$ the subset of $\otimes$-compact constraints.

In Table 1 we recall the deterministic fragment of soft CCP: with respect to the standard proposal, we replaced the procedure call with the simpler to handle recursion operator μ_Z. $\mathcal{A}$ is the set of agents, $\mathcal{Z} = \{Z, Z_1, Z_2 \ldots\}$ is the set of recursion variables, and $c \in C^{\otimes}$ is a compact constraint.

In Table 2 we provide a reduction semantics for our SCCP: a pair $\langle \Gamma, \to\rangle$, for $\Gamma = \mathcal{A} \times C$ the set of configurations and $\longrightarrow \subseteq \ \Gamma \times \Gamma$ a relation between them.

R1	$\langle \mathbf{tell}(c), \sigma\rangle \longrightarrow \langle \mathbf{stop}, \sigma \otimes c\rangle$	**Tell**	**R4**	$\dfrac{\langle A, \sigma\rangle \longrightarrow \langle \mathbf{stop}, \sigma'\rangle}{\langle A \parallel B, \sigma\rangle \longrightarrow \langle B, \sigma'\rangle}$	**Par2**
R2	$\dfrac{d \leq \sigma}{\langle \mathbf{ask}(d) \longrightarrow A, \sigma\rangle \longrightarrow \langle A, \sigma\rangle}$	**Ask**			
			R5	$\dfrac{\langle A, c \otimes \exists_x \sigma\rangle \longrightarrow \langle B, \sigma'\rangle}{\langle \exists^c_x A, \sigma\rangle \longrightarrow \langle \exists^{\sigma'}_x B, \sigma \otimes \exists_x(\sigma' \otimes \exists_x \sigma)\rangle}$	**Hide**
R3	$\dfrac{\langle A, \sigma\rangle \longrightarrow \langle A', \sigma'\rangle}{\langle A \parallel B, \sigma\rangle \longrightarrow \langle A' \parallel B, \sigma'\rangle}$	**Par1**			
			R6	$\dfrac{\langle A[^A/_Z], \sigma\rangle \longrightarrow \langle B, \sigma'\rangle}{\langle \mu_Z A, \sigma\rangle \longrightarrow \langle B, \sigma'\rangle}$	**Rec**

Table 2: Reduction semantics for SCCP (the symmetric rules of **R3** and **R4** are not shown).

Fair computations To define fair computations (Definition 3.3), we introduce the notions of enabled and active agents. Observe that any transition is generated either by an agent **tell**(c) or **ask**$(c) \longrightarrow A$. We say that an agent A of such a shape is *active* in a transition $t = \gamma \to \gamma'$ if it generates such transition; i.e. if there exists a derivation of t where **R1** or **R2** (Table 2) are used to produce a transition of the form $\langle A, d\rangle \to \gamma''$. Moreover, we say that an agent A is *enabled* in a configuration γ if there exists γ' such that A is *active* in $\gamma \to \gamma'$.

Definition 3.3 (Fair Computations) *A computation $\gamma_0 \to \gamma_1 \to \gamma_2 \to \ldots$ is fair if for each agent A enabled in some γ_i there exists $j \geq i$ such that P is active in γ_j.*

The definition is well given: since the calculus is deterministic and its syntax freely generated, the occurrence of an agent can be traced along a computation.

Definition 3.4 (Observables) *The* result *Result(ξ) of a (possibly infinite) computation ξ of the shape $\langle Q_0, \sigma_0\rangle \to \langle Q_1, \sigma_1\rangle \to \langle Q_2, \sigma_2\rangle \to \ldots$ is the constraint $\bigvee_i \sigma_i$.*

If a finite computation is fair, then it ends on a state $\langle \mathbf{stop}, \sigma\rangle$, and its result coincides with the store of the last configuration. As for crisp programming [SRP91], all the fair computations of a configuration have the same result, i.e., given a configuration γ and two computations ξ_1, ξ_2 of γ, if ξ_1 and ξ_2 are fair, then $Result(\xi_1) = Result(\xi_2)$.

Since the language is deterministic, the fair computations originating from a configuration are either all finite or infinite: the proposition further states that each of these have the same result. So, we now denote as $Result(\langle A, \sigma\rangle)$ the unique result of the fair computations originating from $\langle A, \sigma\rangle$.

Saturated Barbed Bisimulation As proposed in [ABP+11] with crisp languages, we define a *barbed* equivalence between two agents [MS92]. Since *barbs* are basic observations (predicates) on the states of a system, in this case barbs correspond to the compact constraints in $C^{\otimes}$, and we say that $\langle A, \sigma\rangle$ verifies c, or that $\langle A, \sigma\rangle \downarrow_c$ holds, if $c \leq \sigma$.

Definition 3.5 (Semi-saturated barbed bisimilarity) *A semi-saturated barbed bisimulation is a symmetric relation R on configurations such that whenever $(\langle A, \sigma\rangle, \langle B, \rho\rangle) \in R$*

1. *if $\langle A, \sigma\rangle \downarrow_c$ then $\langle B, \rho\rangle \Downarrow_c$*
2. *if $\langle A, \sigma\rangle \longrightarrow \gamma'_1$ then there exists γ'_2 such that $\langle B, \rho\rangle \longrightarrow^* \gamma'_2$ and $(\gamma'_1, \gamma'_2) \in R$*
3. *$(\langle A, \sigma \otimes d\rangle, \langle B, \rho \otimes d\rangle) \in R$ for every $d \in C^{\otimes}$*

We say that γ_1 and γ_2 are semi-saturated barbed bisimilar ($\gamma_1 \sim_{sb} \gamma_2$) if there exists a semi-saturated barbed bisimulation R such that $(\gamma_1, \gamma_2) \in R$. We write $A \sim_{sb} B$ if $\langle A, \perp\rangle \sim_{sb} \langle B, \perp\rangle$.

We let $\longrightarrow^*$ denote the reflexive and transitive closure of $\longrightarrow$, and say that $\gamma \Downarrow_c$ holds if there exists γ' such that $\gamma \longrightarrow^* \gamma'$ and $\gamma' \downarrow_c$. Then, the main result establishes that $A \sim_o B$ if and only if $A \approx_{sb} B$.

Novelty and future work. The current research follows a recent stream of work on the derivation of labelled operational semantics, and their characterization via semi-saturated bisimilarity, for systems whose semantics is either operational but without labels, or actually denotational in principle. The starting point has been the theory of reactive systems [LM00], and some updates in the direction of this work can be found in [BGM11] and in the work on the crisp version of CCP [ABP+11]. Immediate future work will focus on a labelled operational semantics, in order to prove the correspondence of a standard notion of weak bisimilarity with the semi-saturated one just introduced.

3.1.2 A coordinating approach to global optimization problems

Within ASCENS, systems are modelled as self-aware, self-adaptive and autonomic components running in ensembles (dynamic aggregations of components), which through interactions among them and with the environment accomplish both individual (local) and collective (global) goals by optimizing the use of resources. So, for example, in the e-mobility case study, the traffic system is modelled as ensembles of electrical vehicles with the goal to optimize the usage of resources (electricity, parking places, etc) while ensuring the fulfillment of individual goals (such as reaching in time the destination) and collective goals (avoiding traffic jam or guaranteeing that all vehicles find a spot where to park).

The behaviour of this kind of systems can obviously 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. Therefore, the general form of this kind of problems is the following:

$$\begin{array}{lll} \text{minimize or maximize} & f_i(x_1, \ldots, x_m) & \forall i \in= \{1, \ldots, n\} \\ \text{such that} & c_1, \ldots, c_k & \text{are satisfied} \end{array}$$

where $c_1, \ldots, c_k$ denote constrains on the variables $x_1, \ldots, x_m$.

Typically, some of the objective functions aim at optimizing the usage of local resources of components, others instead optimize global resources. Analogously, some constraints will be on local resources and others on global ones.

As an example, let us consider the parking lot optimization problem. It consists in finding the best parking lot $P_i \in P = \{P_1, \ldots, P_m\}$ for each vehicle V_i of an ensemble $V = \{V_1, \ldots, V_n\}$ in terms of three factors: (1) the distance from the current location l_i of V_i to the parking lot location l_j, (2) the distance from l_j to the appointment location l_{A_i}, (3) the cost c_j of the parking lot P_j. It is obviously needed to also ensure that the initial energy level of the vehicle is enough to reach the parking lot.

In order to solve the problem, we have to find an assignment of parking lots to vehicles that minimizes the cost (given by the three factors above) of the choice, while satisfying all the constrains.

In order to formalize the problem we introduce the following decision variables

$$x_{ij} = \begin{cases} 0 & \text{if } V_i \text{ is not assigned to the parking lot } P_j \\ 1 & \text{if } V_i \text{ is assigned to the parking lot } P_j \end{cases}$$

We denote by:

- y_j the number of available parking spots of P_j
- e_i the energy level of V_i
- $E(l_i, l_j)$ the energy needed to go from l_i to l_j
- c_{ij} the cost of V_i to use P_j, where $c_{ij} = D(l_i, l_j) + c_j + D(l_j, l_{A_i})$, with $D(l_i, l_j)$ denoting the minimum distance from l_i to l_j and $D(l_j, l_{A_i})$ the minimum distance from l_j to l_{A_i}.

So the formalisation of the problem is the following

$$
\begin{array}{lll}
\text{minimize } f_i = & \sum_{j=1}^{j=m} c_{ij}x_{ij} & \forall i \in \{1,\ldots,n\} \\
\text{such that} & \sum_{j=1}^{j=m} x_{ij} = 1 & \forall i \in \{1,\ldots,n\} \\
& \sum_{j=1}^{j=m} E(l_i, l_j)x_{ij} \leq e_i & \forall i \in \{1,\ldots,n\} \\
& x_{ij} \in \{0,1\} & \forall i \in \{1,\ldots,n\}\ \forall j \in \{1,\ldots,m\} \\
& \sum_{i=1}^{i=n} x_{ij} \leq y_j & \forall j \in \{1,\ldots,m\}
\end{array}
$$

It is easy to note that each objective function concerns only one vehicle, it indeed aims at optimizing the use of resources which are local to the vehicle. Also the constraints in the first three rows are on local resources, while those in the last row are global.

However, these problems tend to be complex to solve and often a globally optimal solution may be impossible to find. For this reason specific strategies are needed to solve them.

To tackle them, we propose a technique based on the coordination of declarative and procedural knowledge. 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 local to one component. Separately, each local optimization problem will be simply and naturally modelled and solved by using the soft constraint logic programming 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. The solution is indeed guaranteed to be just an acceptable global solution: it may or may not be globally optimal.

In Section 3.6.2 of Joint Deliverable JD3.2, the application of this technique to the parking lot problem is described. It leads to several local optimization problems, one for each vehicle of the ensemble, and to a coordination strategy that checks if local optimal choices can be satisfied by the parking lots, i.e., if the parking spots of each parking lots are enough to satisfy the requests of the vehicles. If it is so, the global problem is (optimally) solved, otherwise it queries the declarative knowledge again to compute new local solutions, but now by increasing the costs of the parking lots which received too many requests. The procedure is repeated, with suitable variations, until a global solution is found.

To show how this problem can be solved, we proposed an implementation with CIAO [BCC+97] for the declarative part and Java for the orchestrator (see Section 4.3.1 of Deliverable D7.3).

Our technique for solving the optimization problems of large ensembles, besides being more efficient than the algorithms finding the global optimum, in general results more flexible: parameters of the declarative knowledge can be easily changed during the execution from both the orchestrator and the components of the ensembles. In this setting, SCLP is very convenient since, besides offering a natural way to model and solve optimization problems [MMH12], the parameters of its fact/clause-based declarative implementation are more easily accessible than an ordinary imperative module structure.

Novelty and future work. With respect to the second year, during which just local optimization problems were studied (see for example [MMH12]), now we focus our attention on global optimization problems. As explained above, we propose a technique based on a coordination strategy of the declarative and procedural knowledge.

However, we plan to continue the study by considering approximate optimization techniques based on an approximate implementation of a dynamic programming procedure, still using a coordination among the declarative and procedural part.

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*, deepening the investigation carried last year on minority games for service composition (see Deliverable D2.2), as needed by, e.g., a peer-to-peer energy management scenario. On a more foundational issue, *Coalgebraic techniques for dynamical systems* pushes the coalgebraic view of state-to-function automata, a uniform framework for the formal definition of the operational semantics of stochastic process calculi, where each transition intuitively carries on also the cost of its execution.

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 year 2, dutifully reported on the previous deliverable, Sections 4.1.1 and 4.1.2 report 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.

4.1.1 Smart meter aware domestic energy trading agents

In the third year we have carried on the work started during the first year related to peer-to-peer energy management. During the first two years we designed and developed an agent-based system able to manage the interactions among prosumers (producers and consumers of energy, mainly by solar panels and wind turbines), consumers, and genco (large-scale energy producers). The system relies on an approach coming from game theory and it is integrated with web services in order to enable agents contracting energy to automatically retrieve data to be used in adaptive and collaborative aspects.

In the last period we have focused on the self-organization aspects of the proposed system [CC13b]. To be more specific, we have proposed a model in which small-scale seller agents dynamically and autonomously decide either to address the market as lone operators or by aggregating into Virtual Power Plants, from time to time in order to adapt to different situations. This iterated decisional process depends on highly variable market related factors, thus our goal was to design a net of agents able to autonomously react to this dynamic environment.

In collaboration with WP4, we studied the chance of changing the collaboration pattern at runtime, depending on the situations [CC13a]. In this case, each seller can have different advantages from different collaboration patterns in different times, so a dynamic adaptation of the entire organization can be useful to grant the best global collaboration. Even if in this case there are only two patterns (lone operator and Virtual Power Plant), this is a good example for our study.

Novelty and future work. The novelty of the approach presented in the third year is the chance of changing the collaboration pattern at runtime, which, as far as we know, is not proposed in existing approaches and was not conceived in the previous version of our own. With regard to future work, we will investigate how to refine our model (i) by considering agents with different behaviors and in particular different ways to calculate the economic risk and (ii) by investigating alternative market models for evaluation and comparison purposes.

4.1.2 Prosumer planning 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, just presented for evaluation 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. We have chosen to follow the distributed power market model 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.

Novelty and future work. The work by Alain Tcheukam Siwe has been granted an award by ENEL Foundation and CRUI Foundation, which will allow the continuation of the research after the PhD period. The plan is to interact with ENEL researchers for evaluating the applicability of the ideas put forward in the thesis. An open issue is the global effect of the introduced controller in a prosumer population, since each prosumer could make use of it. We believe that the issue can lead to a congestion problem similar to that of the minority game problem proposed in the economics literature.

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. Besides classical weighted automata, a better suited tool, developed inside ASCENS, is represented by state-to-function automata. The more operational side of the formalism has been reported in the first two years as part of Task 2.2, while the recent developments have a more abstract nature and supersede the work on weighted automata previously reported on Task 2.3. Besides its use as an off-the-shelf technique for modeling purposes, the final outcome of the research on state-to-function automata would be a theoretical framework for casting the stochastic version of SCEL.

4.2.1 Coalgebraic bisimilarity vs. behavioural equivalence for State-to-Function LTSs

As reported in Deliverable D2.1, state-to-Function Labelled Transitions Systems (FuTSs) have been proposed as a uniform framework for the formal definition of the operational semantics of stochastic Process Calculi (SPC) in the SOS style [DLLM14]. FuTSs are equipped with their native notion of bisimilarity; in Deliverable D2.2, we reported on a coalgebraic view of FuTSs and, in particular, on the *existence* of a *final coalgebra* for the relevant functor and on the *coincidence* between FuTS-*bisimilarity* and coalgebraic *behavioural equivalence* [LMd12].

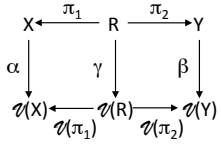


Figure 7: Diagram of coalgebraic bisimulation

During the third year, we complemented the above result with the study of the relationship between FuTS-*bisimilarity* and *coalgebraic bisimilarity*; furthermore, we gave further examples of FuTS semantics e.g. for deterministic (discrete) Timed Process Calculi and Markov Automata.

In the sequel, after briefly recalling some notions, we state the main result; the reader is referred to [LMd13] for details and to D2.2 for a broader introduction and discussion on the approach.

Let $\mathbb{C}$ be a commutative semiring and $\mathbf{FTF}(S, \mathbb{C})$ denote the class of total functions from set S to $\mathbb{C}$ with finite support. In the simplest case, a *total, deterministic* FuTS over $\mathcal{L}$ and $\mathbb{C}$ is a tuple $(S, \mathcal{L}, \mathbb{C}, tr)$ where S is a countable, non-empty, set of states, $\mathcal{L}$ is a countable, non-empty, set of transition labels, $\mathbb{C}$ is a commutative semiring, and $tr \subseteq S \times \mathcal{L} \times \mathbf{FTF}(S, \mathbb{C})$ is the transition relation, which is required to be total on S and $\mathcal{L}$ and deterministic[2]; for such a FuTS, in the sequel, we use the notation (S, θ), where θ is the transition function and $\mathcal{L}$ and $\mathbb{C}$ are assumed known from the context[3]. A notion of bisimilarity for FuTSs has been defined and it has been shown that it is a natural generalization of standard (stochastic) process bisimilarities, such as those presented in [Hil96, Her02][4]. We let $\sim_{\mathcal{F}}$ denote $\mathcal{F}$-bisimilarity on S, for FuTS $\mathcal{F} = (S, \theta)$. Finally, a coalgebraic view of FuTSs on **Set** has been presented, where each FuTS $\mathcal{F} = (S, \theta)$ is interpreted as a coalgebra on an approriate general functor $\mathcal{V}_{\mathbb{C}}^{\mathcal{L}}$. It has been shown that $\mathcal{V}_{\mathbb{C}}^{\mathcal{L}}$ has a final coalgebra and that, for all FuTS $\mathcal{F} = (S, \theta)$, the *behavioural equivalence*[5] $\approx_{\mathcal{F}}^{\mathcal{V}_{\mathbb{C}}^{\mathcal{L}}}$ it induces on S coincides with $\sim_{\mathcal{F}}$.

In the Project third year we continued our coalgebraic study of FuTSs and we showed that, under certain conditions on the semiring $\mathbb{C}$, behavioural equivalence coincides with *coalgebraic bisimilarity*, a standard notion of bisimilarity in the context of Category Theory. We first recall the definition of coalgebraic bisimilarity for generic functor $\mathcal{V}$. An $\mathcal{V}$-coalgebra (R, γ) with $R \subseteq X \times Y$ is called a *coalgebraic bisimulation* of two $\mathcal{V}$-coalgebras (X, α) and (Y, β) if $\alpha \circ \pi_1 = \mathcal{V}(\pi_1) \circ \gamma$ and $\beta \circ \pi_2 = \mathcal{V}(\pi_2) \circ \gamma$. Here $\pi_1 : R \to X$ and $\pi_2 : R \to Y$ are the projections from R to X and Y, respectively. See Figure 7. For a $\mathcal{V}$-coalgebra $\mathcal{G} = (X, \alpha)$, two elements $x_1, x_2 \in X$ are called *coalgebraically bisimilar* if there exists a coalgebraic bisimulation R of (X, α) and (X, α) such that $R(x_1, x_2)$, notation $x_1 \trianglelefteq_{\mathcal{G}}^{\mathcal{V}} x_2$.

We now present our main new result. Let $\mathbb{C}$ be a semiring which (i) admits (right) multiplicative inverse, i.e. for $c \in \mathbb{C} \setminus \{0_{\mathbb{C}}\}$ it holds that $c \cdot_{\mathbb{C}} c' = 1_{\mathbb{C}}$, for some $c' \in \mathbb{C}$, and (ii) satisfies the *zero-sum-property*, i.e. a sum $c_1 +_{\mathbb{C}} \ldots +_{\mathbb{C}} c_n = 0_{\mathbb{C}}$ iff $c_i = 0_{\mathbb{C}}$ for $i = 1, \ldots, n$.

Theorem 4.1 *For all label sets $\mathcal{L}$, semiring $\mathbb{C}$ with (right) multiplicative inverse and satisfying the zero-sum-property and $\mathcal{V}_{\mathbb{C}}^{\mathcal{L}}$-coalgebra $\mathcal{F}$, behavioural equivalence $\approx_{\mathcal{F}}^{\mathcal{V}_{\mathbb{C}}^{\mathcal{L}}}$ coincides with coalgebraic bisimulation $\trianglelefteq_{\mathcal{F}}^{\mathcal{V}_{\mathbb{C}}^{\mathcal{L}}}$.*

[2]That is, for all $s \in S, \ell \in \mathcal{L}$ there exists a unique $\mathscr{P} \in \mathbf{FTF}(S, \mathbb{C})$ s.t. $str\ell\mathscr{P}$.

[3]For the purpose of this deliverable we consider total deterministic finite support FuTSs, since they are general enough as a model for SCPs, including those capturing non-deterministic behaviour.

[4]See Deliverable D2.2. The relevant definition is recalled in the following. Let $\mathcal{F} = (S, \theta)$ be a FuTS over $\mathcal{L}$ and $\mathbb{C}$; we say that an equivalence relation $R \subseteq S \times S$ is an $\mathcal{F}$-bisimulation if $(s_1, s_2) \in R$ implies $\sum_{s' \in [s]_R} \theta(s_1)(\ell)(s') = \sum_{s' \in [s]_R} \theta(s_2)(\ell)(s')$, for all $s \in S$ and $\ell \in \mathcal{L}$, where $\sum$ is the n-ary extension of the sum of $\mathbb{C}$ and $[s]_R$ is the equivalence class of state s (w.r.t. R); $s_1, s_2 \in S$ are $\mathcal{F}$-bisimilar (notation $s_1 \sim_{\mathcal{F}} s_2$) if $(s_1, s_2) \in R$ for some $\mathcal{F}$-bisimulation R for S.

[5]We recall that, by definition of final coalgebra (Ω, ω), there exists a *unique* morphism $M_{\mathcal{F}}$ from any $\mathcal{V}_{\mathbb{C}}^{\mathcal{L}}$-coalgebra $\mathcal{F}$ to (Ω, ω) and that $s_1, s_2 \in S$ are *behaviourally equivalent* iff $M_{\mathcal{F}}(s_1) = M_{\mathcal{F}}(s_2)$.

This result complements the coincidence between FuTSs bisimilarity $\sim_{\mathcal{F}}$ and behavioural equivalence $\approx_{\mathcal{F}}^{\mathcal{V}_{\mathbb{C}}^{\mathcal{L}}}$, thus providing an additional argument in support of the concrete notion of FuTSs bisimilarity and of concrete standard notions of stochastic bisimilarity, which are instances thereof. The restrictions on the semiring, in fact, are satisfied by all SPCs, since the relevant semiring is the set of non-negative real numbers.

In addition to the above result, during the third project year, we have carried out further experimentation with using the FuTSs framework for the definition of quantitative Process Calculi, including a (discrete) time process calculus, and Markov automata (see e.g. [EHZ10]). The interested reader is referred to [LMd13] for a discussion on the topic.

Novelty and future work. The main novelty of the work performed during the third year concerns the study of the relationship between behavioural equivalence and coalgebraic bisimilarity in the context of FuTSs. In particular, it has been shown that, under certain assumptions on the relevant semiring, the two equivalences coincide. This adds up to the result of the second year, which was showing the coincidence between (concrete) FuTS bisimilarity and behavioural equivalence, settling the relationship between FuTS bisimilarity and coalgebraic bisimilarity in the context of a coalgebraic framework for FuTSs. Additionally, during the third year, experiments of application of the FuTS framework for the semantics of discrete time process calculi and Markov automata have been performed and they proved the adequacy of the framework also for these kinds of process calculi. Future research needs to reveal under what algebraic conditions of the semirings, or similar structures, or the coalgebraic conditions on the format of the functors involved, standard bisimulation, FuTS-bisimulation, coalgebraic bisimulation and behavioural equivalence will amount to similar identifications also for the above mentioned models. In particular, the study of nested functors (i.e. compositions of functors) seems to be promising.

5 Concluding remarks and deliverable cross influence

This deliverable reports the advances of the third 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.

Foundations of resource-aware connectors As planned in Deliverable D2.2, the work on connectors has focused on the expressiveness and implementability of BIP extensions. This work aimed at offering a flexible and comprehensive framework for rigorous SCEs design and is related to the work conducted in Work Package 5, as reported in Joint Deliverable JD3.1, where distributed implementation of enhanced versions of BIP are discussed and in Deliverable D5.3, where BIP is used for verification of information flow security issues and for the compositional verification of timed systems.

Foundations of resource-aware languages and models The work on NCpi has progressed to assess its categorical semantics and applied to the `Pastry` model used in Science Cloud. The recently delivered Ph.D. Thesis of Matteo Sammartino [Sam13] collects all main results. A more detailed account of the model, and the verification of routing convergence, is detailed in Joint Deliverable JD3.1, together with the sketching of future directions concerning the `Pastry` model. Nominal automata techniques are further exploited in Deliverable D5.3 to deal with the analysis of massive numbers of components dynamically adapting to new requirements and environmental conditions. The work on

stochastic calculi from years one and two has converged with the strand on "Coalgebraic techniques for dynamical systems". This work is pivotal in the quantitative analysis of case studies, as considered e.g. in Joint Deliverable JD3.1 and WP5. Likewise, the work on exact fluid lumpability from previous years has been moved to verification analysis in WP5.

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. During the third year the soft constraint paradigm has been integrated with other frameworks and used in different work packages. In the Deliverable D7.3 and in the Joint Deliverable JD3.2, the technique for solving locally global optimization problems and its application to the parking lot problem are described, together with an implementation with CIAO for the declarative part and Java for the orchestrator. In the Deliverable D3.3, the soft constraints for KnowLang are considered as a suitable reasoning technique that will help designers impose constraining requirements for special liveness properties. The integration of soft constraints in SCEL is still under development in WP1.

Game paradigms for service composition The idea of introducing collaboration patterns at run-time has emerged in close collaboration with WP4 and is highly valuable for its potential applications to be further investigated in year four. Moreover, the smart grid scenario shares interesting analogies with Science Clouds, for what is concerned with competition aspects, like auctions mechanisms for offering/obtaining computational resources from peer cloud nodes and a more direct application for reducing power consumption in cloud farms.

Finally we mention two strands from year two on which we have not reported in this deliverable.

Negotiate, commit, execute The idea is to transfer transferring some of the concepts related to compensation-based semantics, as reported in the recently delivered Ph.D. Thesis of Anne Kersten [KK13], in the design of SCEL. This work is still under development.

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 that also served as 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 Joint Deliverable JD3.2 and Deliverable D1.3.

References

[ABC+09] Farhad Arbab, Roberto Bruni, Dave Clarke, Ivan Lanese, and Ugo Montanari. Tiles for Reo. In *WADT'08*, volume 5486 of *LNCS*, pages 37–55. Springer, 2009.

[ABP+11] Andrés Aristizábal, Filippo Bonchi, Catuscia Palamidessi, Luis Fernando Pino, and Frank D. Valencia. Deriving labels and bisimilarity for concurrent constraint programming. In *FOSSACS*, volume 6604 of *LNCS*, pages 138–152. Springer, 2011.

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

[AD98] Christophe Aussaguès and Vincent David. A method and a technique to model and ensure timeliness in safety critical real-time systems. In *ICECCS*, pages 2–12. IEEE Computer Society, 1998.

[BBCG12] Filippo Bonchi, Maria Grazia Buscemi, Vincenzo Ciancia, and Fabio Gadducci. A presheaf environment for the explicit fusion calculus. *Journal of Automated Reasoning*, 49(2):161–183, 2012.

[BBS06a] Ananda Basu, Marius Bozga, and Joseph Sifakis. Modeling heterogeneous real-time components in BIP. In *Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM 2006)*, pages 3–12. IEEE Computer Society, 2006.

[BBS06b] Ananda Basu, Marius Bozga, and Joseph Sifakis. Modeling heterogeneous real-time components in BIP. In *SEFM*, pages 3–12. IEEE Computer Society, 2006.

[BCC+97] F. Bueno, D. Cabeza, M. Carro, M. Hermenegildo, P. López-García, and G. Puebla. The ciao prolog system. reference manual. Technical Report CLIP3/97.1, School of Computer Science, Technical University of Madrid (UPM), 1997.

[BGM11] Filippo Bonchi, Fabio Gadducci, and Giacoma Valentina Monreale. Towards a general theory of barbs, contexts and labels. In *APLAS*, volume 7078 of *LNCS*, pages 289–304. Springer, 2011.

[BGS00] Sébastien Bornot, Gregor Gößler, and Joseph Sifakis. On the construction of live timed systems. In *TACAS*, volume 1785 of *LNCS*, pages 109–126. Springer, 2000.

[BJPV11] Jesper Bengtson, Magnus Johansson, Joachim Parrow, and Björn Victor. Psi-calculi: a framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science*, 7(1), 2011.

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

[BM09] Filippo Bonchi and Ugo Montanari. Reactive systems, (semi-)saturated semantics and coalgebras on presheaves. *Theoretical Computer Science*, 410(41):4044–4066, 2009.

[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 *Ershov Memorial Conference 2011*, volume 7162 of *LNCS*, pages 19–38. Springer, 2011.

[BMMS13] Roberto Bruni, Hernán Melgratti, Ugo Montanari, and Pawel Sobocinski. Connector algebras for C/E and P/T nets' interactions. *Logical Methods in Comp. Sci.*, 9(3:16):1–65, 2013.

[BMR06] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Soft concurrent constraint programming. *ACM Transactions in Computational Logic*, 7(3):563–589, 2006.

[CC13a] Giacomo Cabri and Nicola Capodieci. Runtime change of collaboration patterns in autonomic systems: Motivations and perspectives. In *AINA Workshops*, pages 1038–1043. IEEE Computer Society, 2013.

[CC13b] Nicola Capodieci and Giacomo Cabri. Managing deregulated energy markets: an adaptive and autonomous multi-agent system application. In *SMC Special Sessions*. IEEE Computer Society, 2013.

[DLLM14] Rocco De Nicola, Diego Latella, Michele Loreti, and Mieke Massink. A uniform definition of stochastic process calculi. *ACM Computing Surveys*, 46(1), 2014. to appear.

[EHZ10] Christian Eisentraut, Holger Hermanns, and Lijun Zhang. On probabilistic automata in continuous time. In *LICS*, pages 342–351. IEEE Computer Society, 2010.

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

[GHKS04] Arkadeb Ghosal, Thomas A. Henzinger, Christoph M. Kirsch, and Marco A. A. Sanvido. Event-driven programming with logical execution times. In *HSCC*, volume 2993 of *LNCS*, pages 357–371. Springer, 2004.

[GM00] Fabio Gadducci and Ugo Montanari. The tile model. In *Proof, Language, and Interaction*, pages 133–166. The MIT Press, 2000.

[Her02] Holger Hermanns. *Interactive Markov Chains*, volume 2428 of *LNCS*. Springer, 2002.

[HHK03] Thomas A. Henzinger, Benjamin Horowitz, and Christoph M. Kirsch. Giotto: a time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84–99, 2003.

[Hil96] Jane Hillston. *A Compositional Approach to Performance Modelling*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1996.

[KK13] Anne Kauer-Kersten. *Formal Specification, Verification and Analysis of Long-running Transactions*. PhD thesis, IMT Alti Studi Lucca, 2013.

[KST12a] Alexander Kurz, Tomoyuki Suzuki, and Emilio Tuosto. A characterisation of languages on infinite alphabets with nominal regular expressions. In *IFIP TCS*, volume 7604 of *LNCS*, pages 193–208. Springer, 2012.

[KST12b] Alexander Kurz, Tomoyuki Suzuki, and Emilio Tuosto. On nominal regular languages with binders. In *FoSSaCS*, volume 7213 of *LNCS*, pages 255–269. Springer, 2012.

[LM00] James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In *CONCUR*, volume 1877 of *LNCS*, pages 243–258. Springer, 2000.

[LMd12] Diego Latella, Mieke Massink, and Erik de Vink. Bisimulation of labeled state-to-function transition systems of stochastic process languages. In *ACCAT*, volume 93 of *ENTCS*, pages 23–43. Open Publishing Association, 2012.

[LMd13] Diego Latella, Mieke Massink, and Erik de Vink. Coalgebraic bisimulation of futs. Technical Report 09, ASCENS, 2013.

[LPT08] A. Lapadula, R. Pugliese, and F. Tiezzi. A formal account of ws-bpel. In *COORDINATION'08*, volume 5052 of *LNCS*, pages 199–215. Springer, 2008.

[MMH12] Giacoma V. Monreale, U. Montanari, and Nicklas Hoch. Soft constraint logic programming for electric vehicle travel optimization. In *WLP*, volume abs/1212.2056 of *CoRR*, 2012.

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

[MS92] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *ICALP*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.

[MS12] Ugo Montanari and Matteo Sammartino. Network conscious π-calculus: A concurrent semantics. In *MFPS*, volume 286 of *ENTCS*, pages 291–306. Elsevier, 2012.

[MS13] Ugo Montanari and Matteo Sammartino. A network-conscious π-calculus and its coalgebraic semantics. *Theoretical Computer Science*, 2013. to appear.

[RD01] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, volume 2218 of *LNCS*, pages 329–350. Springer, 2001.

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

[Sam13] Matteo Sammartino. *A Network-Aware Process Calculus for Global Computing and its Categorical Framework*. PhD thesis, University of Pisa, 2013.

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

[SRP91] Vijay A. Saraswat, Martin C. Rinard, and Prakash Panangaden. Semantic foundations of concurrent constraint programming. In *POPL 1991*, pages 333–352. ACM Press, 1991.

[TCBS13] Ahlem Triki, Jacques Combaz, Saddek Bensalem, and Joseph Sifakis. Model-based implementation of parallel real-time systems. In *FASE*, volume 7793 of *LNCS*, pages 235–249. Springer, 2013.

[Vir07] Mirko Viroli. A core calculus for correlation in orchestration languages. *J. Log. Algebr. Program.*, 70(1):74–95, 2007.