Semantic Service Composition Framework for Multidomain Ubiquitous Computing Applications

Mohamed Hilia, Abdelghani Chibani, Karim Djouani, and Yacine Amirat

Paris-Est Créteil University (UPEC)
Signals Images & Intelligent Systems Laboratory
mohamed.hilia@evidian.com,
{chibani,djouani,amirat}@upec.fr
http://www.lissi.fr

Abstract. In this paper we propose a semantic framework based on constructive description logic. The main innovative aspect of our work consists in the formalization of a composition in the form of e-contract semantic statements where the semantic and logic correctness/soundness are formally checked. The e-contract model is based on cooperation ontology and includes control rules. This model improves on the one hand the common understanding between heterogeneous domains, and on the other hand, it ensures an efficient control of each service from remote requester and preserves the confidentiality of the know-how and the privacy of the local domains. In the conclusion of this paper we present a health care scenario that demonstrates the feasibility of our framework and the demonstration statements of the e-contract in $\mathcal{BCDL}_0$.

Keywords: Collaborative Provisioning Process, Service Composition, Constructive Description Logics, Theorem Proving.

1 Introduction

Several frameworks and middleware have been proposed for performing service composition in pervasive computing in order to comply with the evolving needs of users and organizations, and to take into account the changes of the execution environment. Most of these approaches facilitate the composition task by offering high level abstractions by using web services and semantic web technologies. However, the composed services can involve the cooperation of services belonging to different domains. Enabling this cooperation poses several heterogeneity issues that concerns the semantics of the operations and their control policies.

Leveraging such issues requires semantic framework that provides composition tools along the formal verification of the semantic soundness and the correctness of the composed service regarding what is requested [16].

During the recent years, several approaches were proposed in the state of the art to provide semantic management tools for using description logics and semantic web ontologies such as WSDL-S, DAML-S, WSMO, SWSL, SAWSDL

C. Liu et al. (Eds.): ICSOC 2012, LNCS 7636, pp. 450–467, 2012.

[19]. The most adopted language for building semantic description of web services is OWL-S [10], the successor of DAML-S. In fact, OWL-S is a high-level ontology that allows the description of semantic web services behavior characteristics using business processes and workflow, and the grounding using web service technical specification language such as WSDL. Services described using OWL-S ontology can be even simple or complex service that corresponds to the composition of a set of simple services. The semantic description of services using OWL-S can be published in a declaratory way, on top of standard directory such as UDDI, to facilitate their discovery by software agents running Description Logics based ontology matchmaker algorithms. Describing complex services capabilities and effects using ontologies and composition languages (e.g. BPEL, OWL-S) to build any cooperative provisioning systems implies the existence of a common understanding on the semantics of provisioning services capabilities of each domain, their associated messaging dialogs and their usage control policies. Description Logics is considered as a powerful tool offering a high expressiveness to formalize semantics, and is associated with decidable inference procedures for reasoning on them.

Beyond representation and expressiveness concerns using DL to describe the composition of multi domain heterogeneous services, an important issue that should be treated is how we can prove the correctness of the resulting composition according to the model theory and the objective of the cooperation. Bozzato et al. proposed a formal framework of service composition calculus that assures that any composite service specification (i.e. the service profile) can be verified according to its semantic and logic correctness/soundness by verifying the applicability conditions of the flow control rules used in the composition using the basic constructive description logic, called $\mathcal{BCDL}_0$ [4]. In this paper, we present a formal framework that deals with main issues of services composition formalization for multi domain environments. Concretely our framework reuses $\mathcal{BCDL}_0$ logics and extends the work initiated in [8] with additional control flows and methodology for service composition. The main innovative aspect in our work consists in the formalization of a composition according to formal cooperation e-contract statements that can be proved. The e-contract is specified using the instantiation and extension of the cooperation ontology. The latter describes the cooperation of both business and provisioning operations of the services of each domain and their corresponding control rules. The combination of the cooperation ontology with e-contract statements improves on one hand the common understanding between heterogeneous domains. On the other hand it insures an efficient control of services execution by remote requester and preserves the confidentiality of the know-how and the privacy of the local domains.

The rest of this paper is organized as follows. Section 2 describes the semantic framework architecture. Section 3 overviews the constructive logic $\mathcal{BCDL}$ and its subsystem $\mathcal{BCDL}_0$. Section 4 details the $\mathcal{BCDL}_0$-based multi-domain cooperation ontology specification. Section 5 presents the proposed integrated model. A use case of the proposed model is studied in section 6 and section 7 presents the related work. Section 8 gives the conclusion and the ongoing work.

2 Semantic Cooperation Framework

2.1 Framework Overview

Our framework is built using the following specification components : e-contract, cooperation ontology and the abstraction views (see Figure.1). The abstraction views provide high-level definition of the interfaces required to invoke local services of each domain involved in the cooperation. Each view is defined according to the following formula : $SV(Si) :: Pcond(I) \Longrightarrow Post(O)$, where SV is the service view label, Si service interface, $Pcond$ are the preconditions over the input parameters I, and $Post$ are the postcondition over the output parameters O. Note that the effects are defined with postconditions. Our approach is not intrusive and does not imply the modification of local services. So local services can be invoked in the same manner and under the same constraints by local or remote applications. Each interface is formalized as set of preconditions and their corresponding effects. For each service we consider two views: view on the provisioning operations and a view on the business operations. The provisioning operations concerns the configuration of any nonfunctional parameters needed for the execution of the business service. For instance, the provisioning of QoS parameters that should be done before, during and after the use of a business service such as a camera based monitoring service that requires the provisioning of the following QoS parameters: codec, data transfer rate and security credentials. Many efforts have been undertaken to define ontologies of quality of service concepts. In our work we define upper concepts that can be mapped into these ontologies.

The e-contract defines the workflow for orchestrating the invocation of services and the rules for controlling their invocation. We consider two types of rules: flow control rules and usage control rules (including QoS conrtol). Each service invocation is formalized as a message exchanged between two domains. This message is represented in $\mathcal{BCDL}$ according to the following semantic quadruple (Speech act, service view, source domain, destination domain). The content of each message is specified according to the concepts of the cooperation ontology. The latter is composed of three blocks. The first block provides an upper ontology that describes the concepts used to describe the workflow messages templates, and imports the external QoS ontologies to deal with QoS consideration [21]. For instance, an action execution message is different from notification message. The second block provides common sense ontology the application domain concepts that are used within speech acts, and describes the concepts used to define control rules. The third block describes the concepts used to define global QoS and usage rules regarding the execution of the multi domain composition such as execution time, cost, energy, etc. The cooperation ontology and e-contract formalization are respectively detailed in sections 4, and 5.

Until today, two main approaches have been adopted to build cross-organizational (i.e. multi domain) cooperative systems, namely : bottom-up and top-down approaches. In the bottom-up approache [9,20] we start by enumerating the existing services as a starting point to define the cooperation possibilities

in order to get at the final stage a composite service (i.e. collaborative workflow) that satisfies a set of mutual agreement rules. Conversely, the top-down approaches [6,1] start by specifying a set of common objectives that correspond to a global workflow. According to these objectives, each partner implements local processes that represent a part of the global workflow. Unfortunately, these approaches do not deal with cooperation and control of provisioning management processes. On the one hand, the bottom-up approaches listed above offer no control for the interactions model, and the specification lacks of semantic aspects to ensure the workflows interoperability, and on the other hand, top-down approaches respect neither, the existing workflow nor the design flexibility of the control and the cooperation.

In our framework we propose a hybrid approach that allows domain managers to define the composition according to the steps methodology, depicted in (Figure 1).

1. *Abstraction*: Each domain manager defines local abstract views on the services that are used to build the multi domain composite service. Concretely, this leads to create view on local workflow. The abstraction of the inputs and outputs of this workflow view corresponds to the definition of an atomic service profile. The view is described using interfaces and execution point that allow to invoke a specific process in an internal workflow. The advantage of using abstract views is preserving the confidentiality.
2. *Consistency Checking*: In this step, the managers check the consistency of the cooperation ontology after the refinement of the upper ontology concepts with the specific concepts needed for the cooperation. This is performed by using an automatic consistency checker of ontology such as Pellet.
3. *E-Contract*: The domain managers of the participating domains define the cooperation process which defines the cooperation goals. Afterward, they define the atomic processes involved, and they define the flow control between them. The cooperation process is defined by using abstraction views formalized in Section 5.1. Based on the cooperation process definition with semantic concept, the managers set the list of the obligations, authorizations, and prohibitions, and also temporal constraints.ie deadlines. The cooperation ontology model is detailed in Section 4
4. *Formal Verification*: After the definition of the flow control, and usage control rules in $\mathcal{BCDL}_0$ formalism, we use an interactive theorem prover to prove the soundness of the composite service.
5. *Views Creation*: We generate services invocation interfaces. These interfaces represent atomic provisional tasks that should be handled by the participating domains. A provisional task is any process that can change the environment or effect the running way of business services, such as request for an account creation, task approval, permission assignment, role delegation, notification, etc. The behavioral flow control of the cooperation is also managed through interfaces that must be implemented or mapped with internal process interfaces.

6. *Mapping*: This step consists of creating the mapping between tasks of internal processes view and those generated in the previous step. In this step, we must check the compliancy of the internal security policy with the cooperation defined rules. This step make reuse of the existing processes.

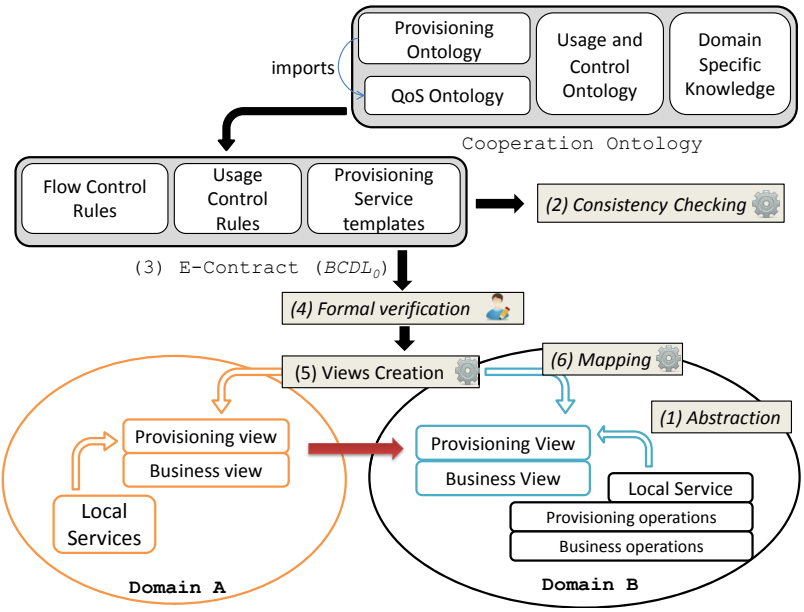

Fig. 1. Global architecture schema of the semantic composition framework

3 Basic Constructive Description Logic ($\mathcal{BCDL}$)

3.1 Towards Constructive Description Logics

Building successfully a cooperative provisioning framework requires the representation of the shared resources (i.e. services, exchanged data) by ensuring the common interpretation and semantic interoperability between the participating domains. Description logics [2] are a family of knowledge representation languages which can be used to represent knowledge of an application domain in a structured and formal way. However, their semantics are restricted to classical reading of description for concepts and individuals [3]. Furthermore, Constructive Description Logics (CDL) has emerged to give new interpretations of DL formulas. CDL aim at modeling knowledge domain and problems that can hardly be treated in the context of classical semantics. This is deeply discussed in [18]. Recently, Ferrari et al. in [5] have proposed $\mathcal{BCDL}$, a constructive description logic based on information terms semantics. This logic allows a constructive interpretation of $\mathcal{ALC}$ formulas. A constructive analysis allows us to exploit the computational properties of its formulas and proofs. $\mathcal{BCDL}_0$ is a subsystem of $\mathcal{BCDL}$ [4].

The constructive interpretation of $\mathcal{BCDL}_0$ is based on the notion of *information term* [5]. Intuitively, an information term α for a closed formula K is a structured object that provides a justification for the validity of K in a classical model.

3.2 $\mathcal{BCDL}_0$: Syntax and Semantics

Let define the language $\mathcal{L}$ for $\mathcal{ALC}$ based on the following sets : NC a set of *Concepts names*, NR set of *Roles names*, NI a set of *Individual names*, and Var a set of *Individual Variables names*. $\mathcal{BCDL}_0$ grammar is defined as follows : $C, D := A \mid \neg\ C \mid C\ \sqcup\ D \mid C\ \sqcap\ D \mid \exists\ R.\ C \mid \forall\ R.\ C$. Where $C,\ D \in$ NC, $R \in$ NR, and A an atomic concept. The $\mathcal{BCDL}_0$ grammar is the same as $\mathcal{ALC}$. The generated concepts by the latter, enable the construction of the following $\mathcal{BCDL}_0$ formulas K, such that : $K := \bot \mid t\ :\ C \mid A\ \sqsubseteq\ C \mid\ (s,\ t) : R$, where s,t $\in$ NI $\cup$ Var.

Let $\mathcal{N} \subseteq NI, \mathcal{L}_N$ be the list of formulas generated by the finit subset $\mathcal{N}$. An *Interpretation* (*Model*) $\mathcal{M}$ for $\mathcal{L}_N$ is the pair $\left(\mathcal{D}^{\mathcal{M}}\ ,\ .^{\mathcal{M}}\right)$. $\mathcal{D}^{\mathcal{M}}$ defines the domain, corresponding to an not empty set, and $.^{\mathcal{M}}$ is a valuation function such that : for every $c \in \mathcal{N},\ c^{\mathcal{M}} \in \mathcal{D}^{\mathcal{M}}$, for every $A \in$ NC, $A^{\mathcal{M}} \subseteq \mathcal{D}^{\mathcal{M}}$, and for every $R \in$ NR, $R^{\mathcal{M}} \subseteq \mathcal{D}^{\mathcal{M}} \times \mathcal{D}^{\mathcal{M}}$.

3.3 $\mathcal{BCDL}_0$ Computational Interpretation

The constructive interpretation of $\mathcal{BCDL}_0$ is based on *information terms*. Formally, given $\mathcal{N} \subseteq NI$ and a closed formula K of $\mathcal{L}_{\mathcal{N}}$, the set of information terms $IT_{\mathcal{N}}(K)$ can be defined by induction on K :

$$
\begin{array}{ll}
IT_{\mathcal{N}}(K) & = \{tt\}\,, \text{iff K is a } \textit{closed formula} \\
IT_{\mathcal{N}}(c : C_1 \sqcap C_2) & = \{(\alpha, \beta) | \alpha \in IT_{\mathcal{N}}(c : C_1) \text{ and } \beta \in IT_{\mathcal{N}}(c : C_2)\} \\
IT_{\mathcal{N}}(c : C_1 \sqcup C_2) & = \{(k, \alpha) | k \in 1, 2 \text{ and } \alpha \in IT_{\mathcal{N}}(c : C_k)\} \\
IT_{\mathcal{N}}(c : \exists\ R.C) & = \{(d, \alpha) | d \in \mathcal{N} \text{ and } \alpha \in IT_{\mathcal{N}}(d : C)\} \\
IT_{\mathcal{N}}(c : \forall\ R.C) & = \left\{\phi : \mathcal{N} \to \bigcup_{d \in \mathcal{N}} IT_{\mathcal{N}}(d : C) |\ \phi(d) \in IT_{\mathcal{N}}(d : C)\right\} \\
IT_{\mathcal{N}}(A \sqsubseteq C) & = \left\{\phi : \mathcal{N} \to \bigcup_{d \in \mathcal{N}} IT_{\mathcal{N}}(d : C) |\ \phi(d) \in IT_{\mathcal{N}}(d : C)\right\}
\end{array}
$$

$\mathcal{BCDL}$ reasoning technique is compatible with the realizability relation of K formula by a given information term. The realizability relation is defined as follows:

Realizability: Let $\mathcal{M}$ be a Model for $\mathcal{L}_N$, K a closed formula and $\eta \in IT_{\mathcal{N}}(K)$. The *realizability* relation is defined as $M \rhd \langle\eta\rangle\ K$ by induction on the structure of K.

$$
\begin{array}{ll}
\mathcal{M} \rhd \langle tt \rangle\ K & \text{iff } \mathcal{M} \vDash K \\
\mathcal{M} \rhd \langle (\alpha, \beta) \rangle\ c : C_1 \sqcap C_2 & \text{iff } \mathcal{M} \vDash \langle \alpha \rangle\ c : C_1 \text{ and } \mathcal{M} \vDash \langle \beta \rangle\ c : C_2 \\
\mathcal{M} \rhd \langle (k, \alpha) \rangle\ c : C_1 \sqcup C_2 & \text{iff } \mathcal{M} \vDash \langle \alpha \rangle\ c : C_k \\
\mathcal{M} \rhd \langle (d, \alpha) \rangle\ c : \exists\ R.C & \text{iff } \mathcal{M} \vDash (c : d) : R \text{ and } \mathcal{M} \vDash \langle \alpha \rangle\ d : C \\
\mathcal{M} \rhd \langle \phi \rangle\ c : \forall\ R.C & \text{iff } \mathcal{M} \vDash c : \forall\ R.C, \text{ and, for every } d \in \mathcal{N}, \mathcal{M} \vDash (c, d) : R \text{ implies } \mathcal{M} \vDash \langle \phi(d) \rangle\ d : C \\
\mathcal{M} \rhd \langle \phi \rangle\ A \sqsubseteq\ C & \text{iff } \mathcal{M} \vDash A \sqsubseteq\ C, \text{ and, for every } d \in \mathcal{N} \text{ if } \mathcal{M} \vDash \langle tt \rangle\ d : A \text{ then } \mathcal{M} \vDash \langle \phi(d) \rangle\ d : C
\end{array}
$$

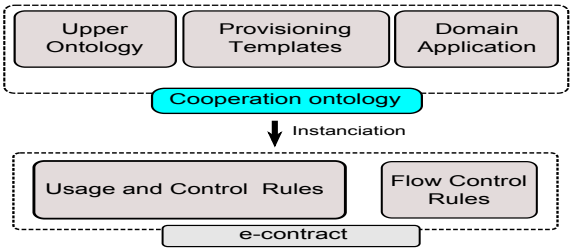

Fig. 2. Semantic model for multi-domain service composition

Definition (Theory) : A theory T consists of $TBox$ and $ABox$. A $TBox$ is a finit set of formulas of the form $A \sqsubseteq C$. An $ABox$ is a set of role assertions and concept assertions :

- *Role assertion* is a formula of the kind (c, d):R , with $c, d \in$ NI and $R \in$ NR
- *Concept assertion* is a formula of the kind $t : C$, with $t \in$ NI and $C \in$ NC

The theories defined by $\mathcal{BCDL}_0$ are sound with the respect of information term semantics. In [4], the authors have shown the natural deduction calculus $\mathcal{ND}$ as the proof calculus for $\mathcal{BCDL}_0$, and gives the soundness theorem according to realizability relation and the natural deduction proofs of $\mathcal{ALC}$ formulas.

Theorem 1 (Soundness) : Let $\mathcal{N}$ be a finit subset of NI and let $\pi :: \Gamma \vdash K$ be a proof of $\mathcal{ND}$ over $\mathcal{L}_\mathcal{N}$ such that the formulas K in Γ are closed. Then :

- $\Gamma \vDash K$
- For every model $\mathcal{M}$ and $\gamma \in IT(\Gamma)$, $\mathcal{M} \rhd \langle\phi\rangle\, \Gamma$ implies $\mathcal{M} \rhd \langle\phi^{\pi}_{N}(\gamma)\rangle\, K$

4 Cooperation Ontology Specification

In this section, we specify the main components of the multi-domain cooperation ontology (see Figure 2). This ontology is divided into three ontologies, which are respectively, provisioning ontology, usage and control ontology, and the domain specific ontology . Each ontology is specified by using constructive description logics formulas.

4.1 Provisioning Ontology Specification

This ontology described by its $TBox$ (see Table 1) expresses the provisioning part of the multi-domain cooperation ontology. The main concept in this theory, is the *ProvisioningTask*. It represents the atomic unit of work in a provisioning process. This concept is generalized by the concept $Task$ which is related to three concepts, namely, the target runtime domain, $Domain$, a $Hook$ to be plugged-in into a specific abstract view interface, and a list of states represented by the concept, $TaskState$.

Table 1. Provisioning Ontology

$Add \sqsubseteq ProvisioningMessage$	$AutomaticTask \sqsubseteq ProvisioningTask$
$AutomaticTask \sqsubseteq \neg ManualTask$	$Delete \sqsubseteq ProvisioningMessage$
$Lookup \sqsubseteq ProvisioningMessage$	$ManualTask \sqsubseteq ProvisioningTask$
$ManualTask \sqsubseteq \neg AutomaticTask$	$Modify \sqsubseteq ProvisioningMessage$
$ProvisioningAction \sqsubseteq Action$	$ProvisioningTask \sqsubseteq Task$
$ProvisioningTask \sqsubseteq \exists executedBy\ Domain$	$\sqcap \exists hasHook\ Hook \sqcap \exists hasState\ TaskState$
$Search \sqsubseteq ProvisioningMessage$	$\exists assignedTo\ Thing \sqsubseteq Action$
$\exists executedBy\ Thing \sqsubseteq Task$	$\top \sqsubseteq \forall executedBy\ Domain$
$\exists hasProvisioningMessage\ Thing \sqsubseteq ProvisioningAction$	$\exists hasPerformer\ Thing \sqsubseteq ManualTask$
$\top \sqsubseteq \forall hasPerformer\ Role$	$\exists hasState\ Thing \sqsubseteq Task$
$\top \sqsubseteq \forall hasProvisioningMessage\ ProvisioningMessage$	$\exists hasHook\ Thing \sqsubseteq Task$
$\top \sqsubseteq \forall hasState\ TaskState$	$\exists hasTargetObject\ Thing \sqsubseteq ProvisioningAction$

We note that, each task is performed by a physical atomic action represented by the concept *Action*. An *Action* is the physical operation assigned to a *Task*. The *ProvisioningTask* is managed by executing the provisioning message described by the concept *ProvisioningMessage* on a specific object (e.g Service, Resource). This provisioning action can be classified as *AcceptedAction* if the domain validates the request, otherwise as *RefusedAction*. For example : adding an access account *Object* for the subject *Cardiology Doctor* on the monitoring service Policy. A provisioning task can be either automatically triggered by provisioning management system, or manually performed by a user with the requested role *Role*.

4.2 Control Access and Usage Rules Ontology Specification

Table 2 gives the control rules to access the shared resources. These rules, inspired from XACML standard, are based on deontic logic formalization of the *Prohibition*, *Obligation* and *Permission*.

The *Policy* concept has an access effect which is an authorization (*Permit* concept $\top$) or a prohibition (*Deny* concept $\bot$) . It is applicable on a Target. The policy is a set of rules *Rule* and propositional formula specified as a *Condition*. Each rule is defined by a domain on a target *Target*. A target is a set of simplified conditions for the provisioning action.

4.3 Domain Specific Ontology Specification

In this ontology, we express the domain specific concepts and their relations to model the knowledge of this domain. This ontology is used to add additional concepts to the cooperation ontology that are used for the data conversion from a domain to another during the cooperation execution. For instance, the concept *Account* as a specific *Object* in a domain. We note the following formula, $Account \sqsubseteq Object$. This refinement permit to specify the local concepts such as security concepts without disclosing any information neither about their internal structures nor their content. In such way, we preserve the privacy and the confidentiality of the shared information which is an important issues in multi-domain applications.

Table 2. Usage and Control Rules Ontology [13]

$ABACPolicy \sqsubseteq AccessControlPolicy$	$Access \sqsubseteq Action$
$AccessControlPolicy \sqsubseteq SecurityPolicy$	$Action \sqsubseteq Event$
$Agent \sqsubseteq Resource$	$Agent \sqsubseteq \neg Object$
$Attribute \sqsubseteq Descriptor$	$Attribute \sqsubseteq \neg Time$
$Attribute \sqsubseteq \neg Location$	$Location \sqsubseteq Descriptor$
$Location \sqsubseteq \neg Time$	$Location \sqsubseteq \neg Attribute$
$Object \sqsubseteq Resource$	$Object \sqsubseteq \neg Agent$
$Parameter \sqsubseteq Attribute$	$Person \sqsubseteq Agent$
$PersonalData \sqsubseteq Object$	$Time \sqsubseteq Descriptor$
$Time \sqsubseteq \neg Location$	$Time \sqsubseteq \neg Attribute$
$\exists\ assignedTo\ Thing \sqsubseteq Permission$	$\top \sqsubseteq \forall\ assignedTo\ SecurityAttribute$
$\exists\ controledBy\ Thing \sqsubseteq Action$	$\top \sqsubseteq \forall\ controledBy\ SecurityPolicy$
$\exists\ hasAttribute\ Thing \sqsubseteq ABACPolicy$	$\exists\ grants\ Thing \sqsubseteq Permission$
$\exists\ hasDescriptor\ Thing \sqsubseteq Condition$	$\exists\ describedBy\ Thing \sqsubseteq Resource$
$\exists\ hasParameter\ Thing \sqsubseteq Action$	$\top \sqsubseteq \forall\ describedBy\ Descriptor$
$\exists\ hasValidity\ Thing \sqsubseteq SecurityAttribute$	$\top \sqsubseteq \forall\ controls\ Action$
$\exists\ identifies\ Thing \sqsubseteq PersonalData$	$\exists\ controls\ Thing \sqsubseteq SecurityPolicy$
$\exists\ isAssignedTo\ Thing \sqsubseteq SecurityAttribute$	$\top \sqsubseteq \forall\ grants\ Action$
$\exists\ managesBy\ Thing \sqsubseteq SecurityPolicy$	$\exists\ has\ Thing \sqsubseteq SecurityPolicy$
$\exists\ on\ Thing \sqsubseteq Access$	$\exists\ specifies\ Thing \sqsubseteq SecurityPolicy$
$\exists\ performedBy\ Thing \sqsubseteq Event$	

5 E-Contract Modeling

5.1 Service Specification

A service specification is an expression of the form $p\,(x) :: P \Longrightarrow Q$ where: p is a label that identifies the service; x is the input parameter of the service (to be instantiated with an individual name from $\mathcal{N}$); P and Q are concepts over $\mathcal{L}_N$. P is called the service precondition, denoted by $Pre(p)$, and Q the service postcondition, denoted by $Post(p)$. The service implementation is modeled as a function : $\Phi_p : \bigcup_{t\in\mathcal{N}} IT_{\mathcal{N}}(t : P) \rightarrow \bigcup_{t\in\mathcal{N}} IT_{\mathcal{N}}(t : Q)$. We denote by the pair $(p(x) :: P \Longrightarrow Q, \Phi_p)$ (or with (p, Φ_p)) a service definition over $\mathcal{L}_{\mathcal{N}}$. The service specification provides the formal description of the behavior of the service in terms of pre- and post- conditions. The function p represents a formal description of service implementation (i.e. of the input/output function). Note that the service definition is based on multi-domain cooperation ontology.

5.2 Provisioning Service Modeling

In this section, we provide an example of atomic service specification that constitutes the atomic provisioning action.

Request : The request provisioning service is a query for action execution. It is an action with a destination domain *Domain*, and it requests for an action execution represented by *RequestAction* concept. The postconditions or effects

of this action are answers from the target domain. The latter, can accept the requested action, *AcceptedAction*, or refuse it,$RefusedAction$, with the associated explanation message, $Message$. This provisioning service is formalized in Table 3.

Table 3. Request action specification

$Request(action)$ ::
$RequestAction \sqcap \exists\, hasTargetDomain.Domain \sqcap$ $\Longrightarrow AcceptedAction \sqcup (RefusedAction \sqcup \exists\, hasMessage.Message)$

The notion of correctness of implementation with respect to the process specification is modeled as follows :

Uniform Resolvability Let define $\mathcal{L}_{\mathcal{N}}$ as language over $\mathcal{N}$, a service definition $(p(x) :: P \Longrightarrow Q, \Phi_p)$ over $\mathcal{L}_{\mathcal{N}}$ and a model $\mathcal{M}$ for $\mathcal{L}_n$. Φ_p *uniformly solves* $p(x) :: P \Longrightarrow Q$ iff , for every individual name $t \in \mathcal{N}$, and every $\alpha \in IT_{\mathcal{N}}(t : P)$ such that $\mathcal{M} \rhd \langle\alpha\rangle\, t : P$, $\mathcal{M} \rhd \langle\Phi_p(\alpha)\rangle\, t : Q$

In the rest of this section, we propose a list of provisioning services for the provisioning management in multi-domain environment. As we have mentioned, each functionality is specified by using a single speech act, which represent an atomic service. In our framework we establish a list of predefined actions. As example of these actions : request for action execution, approval of requested action, delegate an action, assign an action to a domain.

Uniform Resolvability Definition. Let *action* be an individual name which represent the input of *Request* speech act. In our setting, the service implementation correspond to function mapping information terms for the precondition into information terms for the postcondition. This function formalizes the behavior of the effective implementation of the web services. In particular let us consider the implementation $\Phi_{Request}$ of the request service. Let *action* be the individual name representing an *RequestAction*. The input of request is any information term for $\alpha \in IT_{\mathcal{N}}(action : Pre(Request))$. *action* can be seen as a reference to a database record providing the information required by the service pre-condition and can be seen as a structured representation of such information. Let assume that α has the following form : α = (tt, (domainA, tt)); this information term means that *action* is a request action with the target domain *domainA*. Now, let $\beta = \Phi_{Request}(\alpha) \in IT(action : Post(Request))$. If β= (1, tt), this classify *action* as accepted. Otherwise β could be (2, (tt, (refusal_message, tt))) which classifies *action* as refused and specifies that there is a message *message* to comment this refusal. To conclude, we remark that the intended model $\mathcal{M}$ we use to evaluate the correctness of the system is implicitly defined by the knowledge base of the system. Indeed, *action* : Pre(Request) is valid in $\mathcal{M}$ if and only if in our system action effectively codify a request and *domainA* is classified as a target Domain.

In this case, since $\Phi_{Request}$ uniformly solves the service specification, we know that *action* : Post(Request) is valid in $\mathcal{M}$, this trivially corresponds to the fact that, looking at its knowledge base, the domain can generate its acceptance.

5.3 Service Composition Calculus $\mathcal{PC}$

A *service composition* is the process that combines the existing services to build a new process called the composite process. A composite service in an environment $E = \{\mathcal{L}_{\mathcal{N}}, T, \eta, (p_1, \Phi_1), ..., (p_n, \Phi_n)\}$ is defined as follows :

$$\frac{p\,(x) :: P \Longrightarrow Q}{\begin{array}{l} \Pi_1 : p_1\,(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2\,(x) :: P_2 \Longrightarrow Q_2 \\ \ldots \\ \Pi_n : p_n\,(x) :: P_n \Longrightarrow Q_n \end{array}}\ rule$$

Where

- $p\,(x) : P \Longrightarrow Q$ is a service specification over E
- *rule* is one of the rules of the service composition calculus $\mathcal{PC}$
- For every $i \in \{1, \ldots, n\}$, $\Pi_i : p_i(x) :: P_i \Longrightarrow Q_i$ is a service composition over E that meets the applicability conditions of rule

5.4 Control Flow Rules and Composition Process

The atomic provisioning management processes presented above can be combined to express the cooperative provisioning process by using the flow control rules expressed in constructive description logic. These rules are inspired from the basic control flow pattern [15]. They define the basic modeling patterns of business processes behavior. Furthermore, They have been widely used to evaluate the features of existing workflows systems [14].

The control flow rules are detailed below as well as the applicability conditions (AC). The proof of these applicability conditions implies the correctness of the composition.

Sequence: It expresses the fact that a task is performed after the completion of another one in the same process.

Parallel Split: It expresses the splitting of a task into multiple parallel tasks

Synchronization: It represents the convergence of two or more tasks into a single synchronization point. The outgoing task is enabled after the execution of all the incoming tasks.

Exclusive Choice: Based on the truth of fulfilled condition, the flow selects a single ongoing task.

Simple Merge: It consists of the convergence of two or more tasks into a single task. The outgoing tasks are performed when an incoming task is triggered.

Table 4. Flow control rules

$\dfrac{p(x) :: P \Rightarrow Q}{\begin{array}{l}\Pi_1 : p_1(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Sequence*	$AC = \begin{cases} T, x : P \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : P_1 \\ T, x : Q_{k-1} \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : P_k, \text{for } k \in \{2, \ldots, n\} \\ T, x : Q_n \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : Q \end{cases}$
$\dfrac{p(x) :: P \Rightarrow Q}{\begin{array}{l}\Pi_1 : p_1(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Parallel Split*	$AC = \begin{cases} T, x : P \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : P_k, \text{for } \ k \in \{1, \ldots, n\} \\ T, x : Q_1 \sqcap \ \ldots \ \sqcap\ Q_n \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : Q \end{cases}$
$\dfrac{p(x) :: P \Longrightarrow Q}{\begin{array}{l}\Pi_1 : p_1(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Synchronization*	$AC = \begin{cases} T, x : P_1 \sqcap \ \ldots \ \sqcap\ P_n \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : P \\ T, x : Q_k \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : Q, \text{ for } \ k \in \{1, \ldots, n\} \end{cases}$
$\dfrac{p(x) :: P \Longrightarrow Q}{\begin{array}{l}\Pi_1 : p_1(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Exclusive Choice*	$AC = \begin{cases} T, x : P \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : P_1 \sqcup \ \ldots \ \sqcup\ P_n \\ T, x : Q_k \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : Q, \text{for } \ k \in \{1, \ldots, n\} \end{cases}$
$\dfrac{p(x) :: P \Longrightarrow Q}{\begin{array}{l}\Pi_1 : p_1(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Simple Merge*	$AC = \begin{cases} T, x : P \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : P_k, \text{ for } k \in \{1, \ldots, n\} \\ T, x : Q_1 \sqcup \ \ldots \ \sqcup\ Q_n \mid\!\!\!\overline{\mathcal{BCDL}_0}\ x : Q \end{cases}$

6 Scenario : Healthcare Monitoring

In this section, we present a scenario under implementation of platform for the tele-rehabilitation of patients at home. This complex platform is defined according to the requirements and discussions with clinical physicians at UPEC university hospital. The platform is specified as a service composition using several atomic services involving three main domains: the hospital (domainA), the patient home (Domain B) and the emergency agency (Domain C), see Figure 3. The tele-rehabilitation coaching platform allows physicians to order and monitor any rehabilitation program. The latter is composed as a plan of physical activities along with medications that should be undertaken by the patient under the control of the system and the physician. The home is equipped with set of smart devices such as mobile robot equipped with smart display, ip camera, and wearable sensors such as accelerometers and vital sensors that allows on one hand, to monitor the progress of the rehabilitation, and on the other hand, triggering alarms to prevent any damages when an incident happens. Putting such critical service partly under the control of a system requires that the rehabilitation program and its corresponding provisioning tasks requires the verification of the

consistency and the correctness of the composite service. The figure 3 depicts the main services that are used to define the composite service. We denote two provisioning tasks. The first one concerns the creation of an account for a physician to monitor the rehabilitation devices and sensors at the patient's home and the second task concerns the provisioning of the rehabilitation program inside the memory of the robot, which will schedule the program according to the approval of the patient. If the patient refuses the program the coach is notified and can in this case postpone the program or communicate with patient through the robot smart display. If an incident happens during the execution of the rehabilitation program a notification is sent to the emergency agency (domain C) that will check the notification and order the intervention of an ambulance first aid officer at the patient home

Note that to allow the emergency agency checking the situation, temporal credentials are given in the notification message that allow an officer to control remotely the robot and its camera.

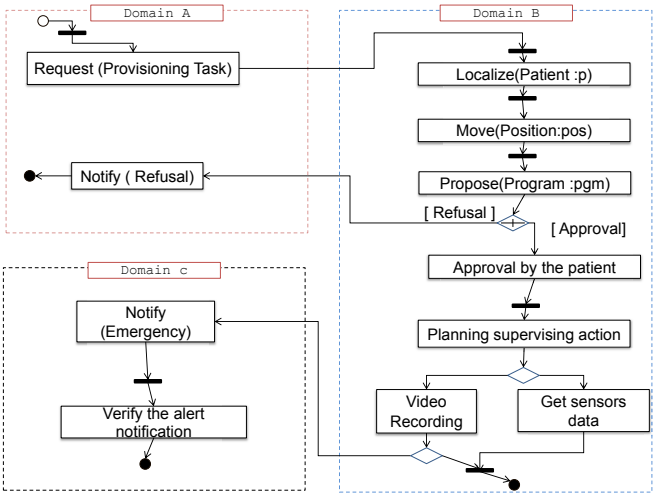

Fig. 3. Service access provisioning process

Let define the required services to successfully build this cooperation, Let's follow the methodology of our framework described in section 2.1. The fist step consists of adding the concepts to the upper ontology to meet the required view definition concepts. We have implemented the ontology by using the ontology editor *protégé*, and we have used its embedded pellet consistency checker. The workflow of the multidomain service composition is defined as follows :

1. The Request corresponds to the provisioning operation of uploading the rehabilitation plan in the robot memory. This request describes the environment criteria and the ambient conditions in which the request will be performed

2. The Robot receives the request and updates the list of the patient daily activities by adding the requested program. When the program time arrives, the robot starts the execution of these task :
   (a) Use localization service to localize the patient in the house by using the RFID identification
   (b) Move to the patient
   (c) Propose the Provisioned Task for performing the activity with the different information
   (d) If the patient refuse, we notify the Coach. Otherwise, the robot must plan for the exercise supervising
   (e) When the patient finishes his activities, the recorded data about These activities are serialized, afterwards, these data are sent to the coach to analyze them and propose other activities

For the lack of space, the steps concerning the e-contract formalization, the creation of services views and the formal validation are detailed in the appendix.

The next step corresponding to the service invocation. According to the approach we defined above the provisioning messages that are grounded using Service Provisioning Markup Language (SPML). SPML is dedicated to service provisioning and combined with web services. It allows to define provisioning operations such as Add, Delete, modify in both synchronous and asynchronous manner. The objects of the provisioning are generated using the mapping with cooperation ontology to XML data structures defined according DSML. For example, the SPML operation with the DSML profile for adding an account is as follows :

Example of SPML Add User Request

```
<spml:addRequest requestID="0123456789">
    <spml:data>
       <dsml:attrname="objectclass">
                 <dsml:value>User</dsml:value>
       </dsml:attr>
       <dsml:attrname="ID">
                 <dsml:value>Hubert Staub</dsml:value>
       </dsml:attr>
       <dsml:attrname="Organization">
                  <dsml:value>DomainB</dsml:value>
       </dsml:attr>
    </spml:data>
</spml:addRequest>
```

The last step of our work consists of service specification that is grounded as web service specification. For instance, interfaces defined in service views are grounded according to the SPML web service standard in the case of provisioning operations, while business operation are grounded into SOAP web services.

7 Related Work

In the recent years, services composition is a hot research topic especially in ubiquitous computing and ambient intelligence. Most of the proposed approaches have tackled services composition from two view point: (i) composition or assembly of new application functionalities using planning or workflow approaches and (ii) modeling and matchmaking users and services specifications using description logics and semantic web ontologies such as SA-WSDL, DAML-S/OWL-S, WSMO, SWSL, etc. These ontologies provides composition applications with the capability of discovering and invoking services automatically. These works are discussed in several surveys [19,17]. Unfortunately, most of these approaches is based on semantic representation that give no way to verify correctness of the composition. Theorem proving based techniques are used to prove soundness and correctness. [12] introduces a method for automatic composition of semantic Web services using Linear Logic theorem proving for DAML-S services description. In this approach, services are expressed by extra-logical axioms. Linear Logic, as a resource conscious logic, enables people to define attributes of Web services formally (including qualitative and quantitative values of non-functional attributes). In addition, Linear Logic has close relationship with π-calculus, which is the formal foundation of many Web service composition languages. This idea of this work has motivated the work recently presented in[11]. This approach is based on the proofs-as-processes paradigm originally introduced by Abramsky, Bellin and Scott. In comparison to the work of [12], they preserved the original theory of Bellin and Scott by using CLL in conjunction with the standard polyadic π-calculus syntax. With respect to the application of $\mathcal{BCDL}$ in service composition, [4] proposed $\mathcal{BCDL}_0$ formalization to proof the correctness of the composition with regards to the service request. The Correctness can be checked directly by verifying the applicability conditions of the composition rules. However, using only the rules proposed in this work is insufficient in the case of multi domain service composition. For instance, the lack of flow control operators such as synchronization or simple merge makes impossible to formalize most of the provisioning services. In our work we have extended their model theory by adding the missing rules.

8 Conclusion and Future Work

In this paper, we proposed semantic framework for the multi-domain cooperative provisioning services. This framework offers an integrated formalization model that allows to specify services composition semantics and their corresponding cooperation policy for multi domain environment. The model is built using the basic constructive description logic $\mathcal{BCDL}$. This logic is characterized by its correctness and soundness properties. In this paper, we exploit their computational properties to proof the correctness of the cooperative provisioning services with the regards to the established e-contact. The ongoing work, we are implementing the proposed model by using ISABELLE theorem prover to automate the verification of cooperation e-contract.

References

1. Ayed, S., Boulahia, N.C., Cuppens, F.: Deploying access control in distributed workflow. In: Proceedings of the Sixth Australasian Conference on Information Security, AISC 2008, vol. 81, pp. 9–17 (2008)
2. Baader, F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Bozzato, L.: Kripke Semantics and Tableau Procedures for Constructive Description Logics. PhD thesis, Università Degli Studi Dell'inubria (2009)
4. Bozzato, L., Ferrari, M.: A note on semantic web services specification and composition in constructive description logics. Journal of Syntax and Semantics (2010)
5. Ferrari, M., Fiorentini, C., Fiorino, G.: BCDL: Basic constructive description logic. Journal of Automated Reasoning 44(4), 371–399 (2010)
6. Hafner, M., Breur, M., Breu, R., Nowak, A.: Modelling inter-organizational workflow security in a peer-to-peer environment. In: IEEE International Conference on Web Services, pp. 533–540 (2005)
7. Hilia, M.: Methodology steps (2012), `https://dl.dropbox.com/u/12278812/icsoc/appendix/scenario_proof.pdf` (accessed July 30, 2012)
8. Hilia, M., Chibani, A., Amirat, Y., Djouani, K.: Cross-organizational cooperation framework for security management in ubiquitous computing environment. In: Proceedings of the 23rd International Conference on Tools with Artificial Intelligence, pp. 464–471 (2011)
9. Lin, D., Ishida, T.: Interorganizational workflow collaboration based on local process views. In: Asia-Pacific Services Computing Conference, pp. 789–794. IEEE Computer Society, Los Alamitos (2008)
10. Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D., Sirin, E., Srinivasan, N.: Bringing semantics to web services with owl-s. Journal of World Wide Web Internet and Web Information Systems 10(3), 243–277 (2007)
11. Papapanagiotou, P., Fleuriot, J.: A theorem proving framework for the formal verification of web services composition. In: Proceedings of the 7th International Workshop on Automated Specification and Verification of Web Systems, pp. 1–16 (2011)
12. Rao, J., Küngas, P.: Logic-based web services composition: From service description to process model. In: Proceedings of the International Conference on Web Services (ICWS), pp. 446–453 (2004)
13. Reul, Q., Zhao, G., Meersman, R.: Ontology-based access control policy interoperability. In: Proc. 1st Conference on Mobility, Individualisation, Socialisation and Connectivity, MISC (2010)
14. Roman, D., Toma, I.: A CTR-based approach to service composition patterns. In: Third International Conference on Next Generation Web Services Practices, NWeSP, pp. 13–18 (October 2007)
15. Russell, N., Ter Hofstede, A., Mulyar, N.: Workflow controlflow patterns: A revised view (2006)
16. Sheng, Q., Benatallah, B., Maamar, Z., Ngu, A.: Configurable composition and adaptive provisioning of web services. Journal of IEEE Transactions on Services Computing 2, 34–49 (2009)
17. Stavropoulos, T., Vrakas, D., Vlahavas, I.: A survey of service composition in ambient intelligence environments. Journal of Artificial Intelligence Review, 1–24 (2011)

18. Troelstra, A.: Aspects of constructive mathematics. Journal of Studies in Logic and the Foundations of Mathematics 90, 973–1052 (1977)
19. Urbieta, A., Barrutieta, G., Parra, J., Uribarren, A.: A survey of dynamic service composition approaches for ambient systems. In: Proceedings of the First International Conference on Ambient Media and Systems, pp. 1–8 (2008)
20. Weigand, H., van den Heuvel, W.J.: Cross-organizational workflow integration using contracts. Decision Support Systems 33(3), 247–265 (2002)
21. Zhou, C., Tien Chia, L., Sung Lee, B.: DAML-QOS ontology for web services. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2004, pp. 472–479 (2004)

Appendix: Methodology Steps

Given the cooperation environment $E_{Cooperation}$ defined by the services, and the flow control rules of $\mathcal{PC}$. We define the new process $CoachingAndMonitoring$ as the composition Π of the states specifications. The behavior of this process is defined as follows: The coaching platform domain uses the *Request* service (Service Composition Π_1), to query the provisioning target (The Robot) to localize the patient and then to move until his position. It first invokes the *Request* service, then he proceeds for localizing the patient by using *Localise* service, Afterwards, he moves into the patient by calling $Move$ service. The service $ProposeActivity$ proposes to the patient to requested Request. The answer of the sequence composition of these services is then combined by $ProposeActivity$ service (Service Composition Π_4) which propose the activity to the patient. Then $ProcessActivity$ (Π_5) by means of an Exclusive Choice rule of two services. The proposed action is then refused, or it is accepted. In the last case, an accepted proposal action is generated and the rehabilitation program is added to the patient daily activities. The program is then processed by the $ProcessPlanningSupervision$ (Π_7) service. This service triggers the parallel execution of the $VideoRecording$ service and the $SensorsDataRetreival$ service. The latter produces a recording data of the specified target. These services must be synchronized before doing the rest of the composite service. Thus, the $DoActivitySynthesis$ (Π_8) is specified by using synchronization flow control rule. This service serializes and protects these data and notifies the coach with the successful end of the request process. Then this data is notified to the coach by using the service $ProcessRecordingData$ (Π_9).

Now let discuss how the composite process computes information terms by explaining a sample execution. Let *action* be a coach request represented by the concept $CoachRequest$ with the associated provisioning action $prvg\_action$.

Π :: $CoachingAndMonitoring$ *Service* Then, a call to the $CoachingAndMonitoring$ service over *action* has as input information term :

$$\alpha_1 = (tt, (prvg\_action, tt)) \ \in \ IT_{\mathcal{N}}(x : Pre(CoachingAndMonitoring))$$

Following the composition Π, the execution of $CoachingAndMonitoring$ service starts with the sequence rule, and the first invoked service is :

$\Pi_1 : Request\,(ProvisioningAction : action) ::$
$CoachRequest \sqcap \exists\, hasProvisioningAction.ProvisioningAction$
$\Longrightarrow (AcceptedProvisioningAction \sqcap \exists\, hasProgram.Program) \sqcup RefusedAction$

Π_1 : Request Service This service process the information term α_1. Let suppose that the *domainB* accepts the provisioning request and produces the program *program* for the patient $patient_1$. The provisioning action is codified in the information term :

$\beta_1 = \Phi_{Request}(tt, (prvg\_action, tt)) \in IT_{\mathcal{N}}(x : Post(Request))$. Let us assume that β_1 has the following form : $\beta_1 = (2, (tt, (program_1, (tt, (program\_person_1, tt)))))$. Where $program\_person_1$ represents the concerned person or (Target) by the program. The execution of the *Request* service is forwarded by the *Localize* service execution.

$\Pi_2 : Localize(Object) ::$
$LocalizationAction \sqcap \exists isLocalizable.Object \Longrightarrow Target \sqcap \forall\, hasLocation.Position$

This service performs a localization action, it needs a target to be localized. As result, it generates the objects target with its associated position.

Π_2 : Localize Service. According to the applicability conditions of the Sequence rule, we have the proof : $\pi_1 :: T_{Cooperation}, x : Post(Request) \vdash_{\overline{\mathcal{BCDL}_0}} x : Pre(Localize)$. The corresponding operator $\phi_{\mathcal{N}}^{\pi_1}$ allows us to extract from β_1 the information term α_2 such that :

$\alpha_2 = (tt, (program\_person_1, tt)) \in IT_{\mathcal{N}}(x : Pre(Localize))$. The Localize service consists of generating the position of the specified target, In this case, the target is the patient. The position of this target is codified in the information term : $\beta_2 = \Phi_{Localize}((tt, (program\_person_1, tt)) \in IT_{\mathcal{N}}(x : Post(Localize))$. Let assume that β_2 has the following form : $\beta_2 = (tt, (position\_program\_person_1, tt))$.

Now, let consider the applicability condition of the sequence rule, in particular :

$\pi_2 :: T_{Cooperation}, x : Post(Localize) \vdash_{\overline{\mathcal{BCDL}_0}} x : Pre(Move)$. The corresponding operator $\phi_{\mathcal{N}}^{\pi_2}$ allow us to extract from β_2 the information term α_3 such that:

$\alpha_3 = (tt, (position\_program\_person_1, tt)) \in IT_{\mathcal{N}}(x : Pre(Move))$. After that, the execution of the *Move* service is occurred. For the lack of space, the rest the formal validation is detailed in [7].