A language that is an amalgam of Propositional Dynamic Logic and Description Logic.

Technical Report in fulfillment of the
thesis requirement for the degree of PhD
"Modeling Composition of Semantic Web Services"

No TRHP-04

P.Tsoutsa

Committee: Omiros Ragos[a] Panagiotis Pintelas[b] Chrisafis Hartonas[c]

Greece, 2014

[a]Department of Mathematics, University of Patras, Greece
[b]Department of Mathematics, University of Patras, Greece
[c]Department of Computer Science and Telecommunications, TEI of Larissa, Greece

Abstract

This work presents our on-going work on defining a language that combines the modality of Modal Logic with the expressiveness of the Description Logic. More specifically it is a variant of star free Propositional Dynamic Logic(PDL) where the propositions are sentences (ABOX assertions) described in a fragment of Description Logic($\mathcal{ALC}$). Our goal is to build a dynamic logic that will describe in a structured and formal way the knowledge is derived during the execution of semantic web services, which they are modeled in Semantic Domains using the technique of Role Modeling.

1 The motivation

A formal language for representing information that derive from modeling web services in Semantic Domains using the technique of Role Modeling[1]. The language will form the foundation to model and express the composition problem of semantic web services.

2 The language $\mathcal{L}$

In this section we define the language that we use to encode the information in semantic domains which is a variant of star free PDL enriched with features from Description Logic adequate to express it.

In logic there are two parts to formally define a language to write expressions, the syntax and the semantics. The syntax determines which collections of symbols are legal to form expressions in the language, while the semantics determines the meaning behind these expressions.

In the following section we first present the *syntax* is used to write well-formed expressions. Then we present the *semantics* of the language.

2.1 Syntax

Syntactically, this language is an amalgam of PDL[2] and the $\mathcal{ALC}$[3] Description language. Moreover, the expressiveness of the language is enriched by using constructors that we borrow from the τPDL [4] language to express capabilities and backward possibility.
As far as atomic propositions are concerned, we split the atom and form the propositions by using elements of Description Logic. More precisely, propositions are assertions of three kinds: *concept assertions, relation assertions and variable* assertions. The basic ingredients to construct them are *Concepts, Relations, Individuals and Variables.*

To formally define the *syntax* of the language we first define the following sets:

(i)Let AtC be a countable set of Basic Concept names. We call each item of the set *Concept* and denote them by letters A, B. A *Concept* is a unary predicate.

$$AtC = \{A \mid A \textit{ is a Concept name}\}$$

(ii) Let $\mathcal{R}$ be a countable and disjoint set with AtC of Relation names. We call each item of the set *Relation* and denote them by the letters R, Q. A *Relation* is a binary predicate.

$$\mathcal{R} = \{R \mid R \textit{ is a Relation name}\}$$

(iii) Let $\mathcal{J}$ be a countable and disjoint set with AtC and $\mathcal{R}$ of individual names. We call the items of the set *Individuals* and denote them by literals j, k.

$$\mathcal{J} = \{j \mid j \textit{ is an Indivindual}\}$$

(iv) Let $\mathcal{X}$ be another countable set of variables.

$$\mathcal{X} = \{x \mid x \textit{ is a Variable.}\}$$

A *Concept assertion* is a *Concept* name with an individual enclosed in parentheses and has the form *C(j)* where C is a *Concept* and j is an *Individual*. As in Description Logic concept assertions generally state properties of individuals, e.g. `Parent(John)` expresses that "John is parent".

A *Relation assertion* is a *Relation* name with two individuals enclosed in parentheses and has the form *R(j,k)* where R is a *Relation* and *j,k* are *individuals*. It generally expresses relations between individuals, e.g. `hasChild(John, Mary)` expresses that "John has child Mary".

A *Variable assertion* is a variable name that determines a variable that we need for describing the problem but for inference reasons we are not interested in its value. The intuition in this kind of *atomic sentences* is to check if the variable has a value or not. We may think of such a variable as a registry that can be in one of two states, full or empty (assigned or not). These registries are then Boolean and we may think of each of them as a *propositional variable* that its truth value is true or false.

Definition *(Atomic sentences)* : *Concept assertions*, *Relation assertions* and *Variable assertions* form the *atomic sentences* of this language and they are the basic blocks for building propositions in the language $\mathcal{L}$. They are formally defined by the following countable set.

$$AtS := \{C(j), R(j,k), x \mid C \in \mathcal{L}_c, R \in \mathcal{R},\ j,k \in \mathcal{J},\ x \in \mathcal{X}\}$$

Apparently, the set of atomic sentences may contain all *concept assertions*, *relation assertions* and *variable assertions* that can be formed from the set of individuals $\mathcal{J}$.

To define the syntax of the language we also assume a set $\mathbb{R}$ of *Role names* (r $\in \mathbb{R}$), a set P of *Atomic processes* ($\sigma \in P$).
In Table 1 we give the schema for the language $\mathcal{L}$ which consists of the sublanguages $\mathcal{L}_s$, $\mathcal{L}_c$ and $\mathcal{L}_\alpha$. $\mathcal{L}_s$ is the language to express propositions and describe properties of states of a system, $\mathcal{L}_c$ is the language to express concepts that are used to form propositional variables and $\mathcal{L}_\alpha$ is the language to express

properties of actions (actions or other processes transforming the states of the system).

Table 1: The schema of the language

$\mathcal{L}_s$	$\ni$	$\varphi := p\ (p \in AtS) \mid \neg\varphi \mid [\alpha]\varphi \mid \varphi\langle\alpha\rangle \mid C_r\alpha$	
AtS	$\ni$	$p :=\ C(j) \mid R(j,k) \mid x$	$(C \in \mathcal{L}_c,\ R \in \mathcal{R},\ j,k \in \mathcal{J},\ x \in \mathcal{X})$
$\mathcal{L}_c$	$\ni$	$C := A\ (A \in AtC) \mid \sim C \mid C \sqcap C \mid \forall R.C \mid \exists R.C$	
$\mathcal{L}_\alpha$	$\ni$	$\alpha := \sigma\ (\sigma \in P) \mid \varphi \mid \alpha\alpha \mid \alpha + \alpha$	

In the language of sentences $\mathcal{L}_s$, *composite sentences* are built from other sentences using the operators (i) $\neg$ (negation), (ii) $[\alpha]\_$ (box forward necessity), (iii) $\_\langle\alpha\rangle$ (backwards possibility) and (iv) $C_r\_$, for each role name $r \in \mathbb{R}$ (capability operator).
The intuitive meaning of the formula $[\alpha]\varphi$ is that after any execution of an action α the resulting state of the system has the property φ. The backward possibility operator, as in $\varphi\langle\alpha\rangle$, designates the type of a state v that came of a state u of type φ, after the execution of an action α. The formula $C_r\alpha$, for a role name $r \in \mathbb{R}$ and an action α, is a capability statement, designating the proposition that role r has the capability to perform an action α. With this operator we propose a machine readable description for action capabilities of roles.

In the language of concepts $\mathcal{L}_c$, *composite concepts* are built from other concepts using the operators (i) $\sim$ (complement), (ii) $\sqcap$ (intersection), (iii) $\forall R.C$ (value restriction) and (iv) $\exists R.C$ (existential restriction).
Intuitively, concept names stand for set of elements of the domain of interpretation and denote that those elements have a specific property. Relation names stand for binary relations between elements. The complement of a concept C is those elements of the Domain do not have the property C. The intersection of two concepts C and D are elements of the domain that have the property C and the property D. The value restriction operator $\exists R.C$ designates the set of elements that are related through the relation R with elements that have the property that is expressed by the concept C.

In the language of actions $\mathcal{L}_\alpha$, we have (i) atomic actions $\sigma \in AtP$, (ii) sentential tests φ, (iii) sequence of actions $\alpha\alpha$ and (iv) nondeterministic choice of actions $\alpha + \beta$.

2.2 Semantics

In the previous paragraph we gave the syntax for the well-formed formulas of this language. In this paragraph we define a formal semantics for all formulas i.e. sentences, concepts and actions.
In order to define a formal semantics we consider structures called *Frames*. A Frame $\mathcal{F}$ is a structure

$$\mathcal{F} \ = \ \langle \ W, \ (\overset{\sigma}{\rightsquigarrow})_{\sigma \in \tilde{P}} \ , \ \tilde{\mathbb{R}} \ \rangle$$

where

- W is a nonempty set of worlds or states of the system
- $\tilde{P}$ is a set of labels and the map $\rightsquigarrow$: $\tilde{P} \to 2^{W \times W}$ assigns a labeled binary relation $\overset{\sigma}{\rightsquigarrow}$ to each $\sigma \in \tilde{P}$
- $\tilde{\mathbb{R}}$ is a nonempty set of Roles, $r \in \tilde{\mathbb{R}}$

The frame is an $\mathcal{L} - Frame$ if $\tilde{P} = P$ and $\tilde{\mathbb{R}} = \mathbb{R}$.

An $\mathcal{L} - Model$ $\mathcal{M}$ where we evaluate the value of formulas is a structure

$$\mathcal{M} \ = \ \langle \ \mathcal{F} \ , \ \Delta \ , \ \varrho \ , \ \mathcal{I}, \ (r^{\mathcal{M}})_{r \in \mathbb{R}} \ \rangle$$

where

- $\mathcal{F}$ is an $\mathcal{L} - Frame$
- Δ is a non empty set (the domain of the interpretation)
- $\varrho : \mathcal{X} \to \mathcal{P}(W)$ is the *interpretation function* for variables, assigning to each variable ($x \in \mathcal{X}$) a subset $\varrho(x)$ of W.
- $\mathcal{I}$ is the *interpretation function* for individuals, concepts and Relations. The function $\mathcal{I}$ is defined as $\mathcal{I} = \mathcal{I}_J \cup \mathcal{I}_C \cup \mathcal{I}_R$.
- for each $r \in \mathbb{R}$, $r^{\mathcal{M}}$ is a map assigning to the role r capabilities that it has at each state $u \in W$, $r^{\mathcal{M}} : W \to \mathcal{P}(\overline{P})$, where $\overline{P} = \{\overset{\sigma}{\rightsquigarrow} | \ \sigma \in AtP\}$ i.e. $r^{\mathcal{M}}(u) \subseteq \{\overset{\sigma}{\rightsquigarrow} | \ \sigma \in AtP\}$

The function $\mathcal{I}_J$ interprets indivinduals, the function $\mathcal{I}_C$ interprets atomic concepts and the function $\mathcal{I}_R$ interprets Relations.

$\mathcal{I}_J \ : \ (J \times W) \longrightarrow \Delta$

$\mathcal{I}_C \ : \ (AtC \times W) \longrightarrow \mathcal{P}(\Delta)$

$\mathcal{I}_R \ : \ (R \times W) \longrightarrow \mathcal{P}(\Delta \times \Delta)$

An individual name $j \in \mathcal{J}$ is used to represent an item of the set Δ which is the domain of the interpretation. We use the letters j, k as individual names and the bold letters $\mathbf{j}, \mathbf{k}$ for the corresponding items of the domain.

$$
\begin{array}{lcl}
(\llbracket j \rrbracket_u^{\mathcal{M}}) & = & \mathcal{I}_J(j,u) \\
(\llbracket A \rrbracket_u^{\mathcal{M}}) & = & \mathcal{I}_C(A,u) \\
(\llbracket R \rrbracket_u^{\mathcal{M}}) & = & \mathcal{I}_R(R,u) \\
(\llbracket \sim C \rrbracket_u^{\mathcal{M}}) & = & \Delta \setminus \llbracket C \rrbracket_u^{\mathcal{M}} \\
(\llbracket C \sqcap D \rrbracket_u^{\mathcal{M}}) & = & \llbracket C \rrbracket_u^{\mathcal{M}} \cap \llbracket D \rrbracket_u^{\mathcal{M}} \\
(\llbracket \forall R.C \rrbracket_u^{\mathcal{M}}) & = & \{\mathbf{j} \in \Delta \mid \forall \mathbf{k} \in \Delta((\mathbf{j},\mathbf{k}) \in \llbracket R \rrbracket_u^{\mathcal{M}} \rightarrow \mathbf{k} \in \llbracket C \rrbracket_u^{\mathcal{M}})\} \\
(\llbracket \exists R.C \rrbracket_u^{\mathcal{M}}) & = & \{\mathbf{j} \in \Delta \mid \exists \mathbf{k} \in \Delta((\mathbf{j},\mathbf{k}) \in \llbracket R \rrbracket_u^{\mathcal{M}} \wedge \mathbf{k} \in \llbracket C \rrbracket_u^{\mathcal{M}})\}
\end{array}
$$

The satisfaction relation $\models^{\mathcal{M}}$ is defined for sentences at a state $u \in W$ in a model $\mathcal{M}$ as follow:

$$
\begin{array}{llll}
u \models^{\mathcal{M}} & x & \text{iff} & u \in \varrho(x) \\
u \models^{\mathcal{M}} & A(j) & \text{iff} & \mathcal{I}(j,u) \in \mathcal{I}(A,u) \\
u \models^{\mathcal{M}} & R(j,k) & \text{iff} & (\mathcal{I}(j,u), \mathcal{I}(k,u)) \in \mathcal{I}(R,u) \\
u \models^{\mathcal{M}} & (B \sqcap C)(j) & \text{iff} & \mathcal{I}(j,u) \in \llbracket B \sqcap C \rrbracket_u^{\mathcal{M}} \\
u \models^{\mathcal{M}} & (\forall R.C)(j) & \text{iff} & \mathcal{I}(j,u) \in \llbracket \forall R.C \rrbracket_u^{\mathcal{M}} \\
u \models^{\mathcal{M}} & (\exists R.C)(j) & \text{iff} & \mathcal{I}(j,u) \in \llbracket \exists R.C \rrbracket_u^{\mathcal{M}} \\
u \models^{\mathcal{M}} & (\sim C)(j) & \text{iff} & u \not\models^{\mathcal{M}} C(j) \\
u \models^{\mathcal{M}} & \neg\varphi & \text{iff} & u \not\models^{\mathcal{M}} \varphi \\
u \models^{\mathcal{M}} & [\alpha]\varphi & \text{iff} & \forall u' \in W \ (u \overset{\alpha}{\rightsquigarrow} u' \Rightarrow u' \models^{\mathcal{M}} \varphi) \\
u \models^{\mathcal{M}} & \varphi\langle\alpha\rangle & \text{iff} & \exists u' \in W (u' \overset{\alpha}{\rightsquigarrow} u \text{ and } u' \models^{\mathcal{M}} \varphi) \\
u \models^{\mathcal{M}} & C_r\alpha & \text{iff} & \overset{\alpha}{\rightsquigarrow} \subseteq r^{\mathcal{M}}(u), \quad r^{\mathcal{M}}(u) = \bigcup\{\overset{\alpha}{\rightsquigarrow} \mid u \in \llbracket C_r\alpha \rrbracket^{\mathcal{M}}\}
\end{array}
$$

Interpretation of processes

$$
\begin{array}{lcl}
\overset{\sigma}{\rightsquigarrow} & \subseteq & W \times W \\
\overset{\varphi}{\rightsquigarrow} & = & \{(u,u) \mid u \models^{\mathcal{M}} \varphi\} \\
\overset{\alpha+\beta}{\rightsquigarrow} & = & \overset{\alpha}{\rightsquigarrow} \cup \overset{\beta}{\rightsquigarrow} \\
\overset{\alpha\beta}{\rightsquigarrow} & = & \overset{\alpha}{\rightsquigarrow}\overset{\beta}{\rightsquigarrow}
\end{array}
$$

2.3 Conclusion and future work

In this report we have presented our on-going work on the syntax and the semantics of a language that is a variant of star free PDL. Our main focus is in modeling *Semantic Web Service Composition* and we approach the composition problem as a planning as satisfiability problem[2]. The difference in our approach is the logic language we will use to encode the problem that is based

on the language we present here. We are currently investigating how will take advantage of the features of the language to model the problem of composition in *semantic domains* using the technique of *Role Modeling*.

References

[1] Kendall, Elizabeth A., 2001, Agent-Oriented Software Engineering with Role Modeling, Lecture Notes in Computer Science, Springer.

[2] David Harel, Jerzy Tiuryn, and Dexter Kozen. 2000. Dynamic Logic. MIT Press, Cambridge, MA, USA.

[3] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (2003), The Description Logic Handbook: Theory, Implementation and Applications , Cambridge University Press.

[4] C.Hartonas, On the Dynamic Logic of Agency and Action, Studia Logica, May 2013.