TeSCO-S: A Framework for defining **TE**mporal **S**emanti**C**s in **O**WL enabled **S**ervices

Monika Solanki
Software Technology Research Laboratory,
De Montfort University
Leicester, UK
monika@dmu.ac.uk

Keywords: Interval Temporal Logic (ITL), Tempura, Ontology, OWL, Validation, Web services

1 Introduction and Motivation

Complex Web services and their compositions are long lived processes, that extend beyond single step execution and may last anywhere from a few mins to a few months. Examples include, web services deployed and composed as e-commerce applications, where an order once placed, can be canceled, changed or put on hold because of unexpected conditions, anytime before its fulfillment. In certain cases a refund may also be requested later, if the service/product does not meet its specifications. In corporate e-business, it may not be a simple database query that generates a document, but an entire business process involving multiple partners. The final generation of the document may span several days. Web services deployed on wireless devices may take more than expected time to provide the requested service due to poor connection facilities. Such applications are inherently reactive [8], [9], [15] systems. They maintain an ongoing interaction with their environment. When describing such systems the focus is not just on what is computed but equally on how it is computed, in terms of interaction capabilities over time.

An important aspect of Web service specification is therefore, the representation of properties and rules that enable and constrain the behaviour of services and their composition over long lived interactions i.e. properties that are **temporal** in nature. The properties need to be specified in a declarative form that is consistent with the language used for describing other aspects of the service. In the context of describing semantic web services, these properties need to be expressed as ontologies, that can be integrated with ontological representation frameworks for web service like OWL-S [18] or WSMO [20]. Further, the augmentation of service specification with temporal properties needs to be supported by an underlying machinery that can execute and validate these properties during service execution i.e. at runtime. In this paper we present "TeSCO-S", a framework for enriching web service interface specifications, described as OWL[5] ontologies with temporal assertions. The TeSCO-S model is based on Interval Temporal Logic (ITL) [12, 13, 11, 2], our underlying formalism for reasoning about service behaviour over periods of time. TeSCO-S provides an OWL ontology for specifying properties in ITL, a pre-processor, "OntoITL" for transforming ontology instances into ITL formulae and an interpreter, "Anatempura" that executes and validates temporal properties in "Tempura", an executable subset of ITL.

Current web service description frameworks suffer from the lack of their ability to provide constructs and concepts that enable reasoning about service behaviour at runtime. Rule languages for the web include RuleML [4] and within the context of semantic web, initiatives such as SWRL[7] and DRS [6]. These approaches are limited to describing only certain kinds of properties. The expressivity of the languages is restricted to specifying static rules and constraints. There are no constructs available for specifying ongoing behavioural semantics or temporal properties of services. Related work in this area is mostly concerned with representation of time as a first-class citizen [14] i.e. reasoning about duration, time and dates. Web Service Specification languages like WSDL[16] and BPEL4WS[19] provide an operational approach to service specification. They do not have the provision for specifying the conditions that restrict the execution of services to a limited set of valid behaviours. In other frameworks like OWL-S[18] and WSMO[20], specification of pre/post-conditions and effects contribute to some extent towards their behavioural description. However they are limited to static descriptions in the sense that they are predicates required to hold only at the initial and final states. TeSCO-S aims to provide a modular approach that enables the specification of temporal properties for services and also their validation at service execution time.

The paper is structured as follows: section 2 introduces the syntax and semantics of Interval Temporal Logic (ITL), our underlying formalism for describing temporal properties. Section 3 describes TeSCO-S and its components. Section 4 describes the relationship between TeSCO-S and existing ontological frameworks for web service specification. Section 5 presents conclusion and future work.

2 Interval Temporal Logic

We base our work on Interval Temporal Logic (ITL) and its executable subset Tempura. Our selection of ITL is based on a number of points. It is a flexible notation for both propositional and first-order reasoning about periods of time. Unlike most temporal logics, ITL can handle both sequential and parallel composition and offers powerful and extensible specification

and proof techniques for reasoning about properties involving safety, liveness and projected time. Timing constraints are expressible and furthermore most imperative programming constructs can be viewed as formulas in a slightly modified version of ITL. Tempura: an executable subset of ITL, provides a framework for developing, analysing and experimenting with suitable ITL specifications.

2.1 ITL: Syntax and Semantics

An interval is considered to be a (in)finite sequence of states, where a state is a mapping from variables to their values. An Interval σ in general has a length $|\sigma| \geq 0$ and a nonempty sequence of $|\sigma| + 1$ states $\sigma_0 \ldots \sigma_{|\sigma|}$. Thus the smallest intervals have length 0 and one state.

The syntax of ITL is defined below where μ is an integer value, a is a static variable (doesn't change within an interval), A is a state variable (can change within an interval), v a static or state variable, g is a function symbol and p is a predicate symbol.

- Expressions

  $exp \ ::= \ a \mid A \mid g(exp_1, \ldots, exp_n) \mid \iota \, a : f$

- Formulae

  $f \ ::= \ p(exp_1, \ldots, exp_n) \mid \neg f \mid f_1 \wedge f_2 \mid \forall v \cdot f \mid skip \mid f_1; f_2 \mid f^*$

ITL contains conventional propositional operators such as $\wedge$, $\neg$ and first order ones such as $\forall$ and $=$. There are temporal operators like "; *(chop)*", "* *(chopstar)*" and "skip". Additionally in ITL, there are temporal operators like $\bigcirc$(next) and $\Box$(always). Expressions and Formulae are evaluated relative to the beginning of an interval.

Some of the frequently used abbreviations are further listed in Table 1.

Table 1: Frequently used non-temporal and temporal abbreviations

$\exists v \bullet f$	$\widehat{=} \neg\forall v \bullet \neg f$	exists
$\bigcirc f$	$\widehat{=}$ skip ; f	next
more	$\widehat{=} \bigcirc true$	non-empty interval
empty	$\widehat{=} \neg more$	empty interval
inf	$\widehat{=} true \,;\, false$	infinite interval
finite	$\widehat{=} \neg inf$	finite interval
$\Diamond f$	$\widehat{=} finite \,;\, f$	sometimes
$\Box f$	$\widehat{=} \neg\Diamond\neg f$	always
$\Diamond_i f$	$\widehat{=} f \,;\, true$	some initial subinterval
$\Box_i f$	$\widehat{=} \neg(\Diamond_i \neg f)$	all initial subintervals
$\Diamond_a f$	$\widehat{=} finite \,;\, f \,;\, true$	some subinterval
$\Box_a f$	$\widehat{=} \neg(\Diamond_a \neg f)$	all subintervals
halt f	$\widehat{=} \Box(\text{empty} \equiv f)$	terminate interval when
fin exp	$\widehat{=} \iota a{:}\ fin(exp = a)$	end value
keep f	$\widehat{=} \Box_a(\text{skip} \supset f)$	all unit subintervals
$\bigcirc exp$	$\widehat{=} \iota a{:}\ \bigcirc(exp = a)$	next value
$exp_1 \leftarrow exp_2$	$\widehat{=} finite \wedge (fin\ exp_1) = exp_2$	temporal assignment
exp_1 *gets* exp_2	$\widehat{=} keep\,(exp_1 \leftarrow exp_2)$	gets
stable *exp*	$\widehat{=} exp\ gets\ exp$	stability
$len(exp)$	$\widehat{=} \exists I \bullet (I = 0) \wedge (I\ gets\ I + 1) \wedge I \leftarrow exp)$	interval length

We do not present the informal and formal semantics of ITL, due to lack of space. We refer the interested reader to [12, 13, 11, 2] for further details about the logic.

3 TeSCO-S: Temporal Semantics for OWL enabled Services

TeSCO-S is a framework for semantically annotating and validating web service specifications with temporal properties. TeSCO-S uses OWL as the ontology representation language. The semantics of the formulae and expressions modeled are the semantics as defined in ITL. TeSCO-S currently does not include any reasoning support for temporal specification. It however provides an interpreter that executes and validates the temporal formulae encoded in the ontology. The current version of TeSCO-S includes the following components:

- An OWL Ontology for first order formulae, expressions and temporal constructs as defined in ITL and its executable subset "Tempura".

- A pre-processor that transforms ontological representations of ITL and tempura constructs defined in the ontology above to concrete formulae and expressions.
- An interpreter,"AnaTempura" that provides execution support for "Tempura", an executable subset of ITL.

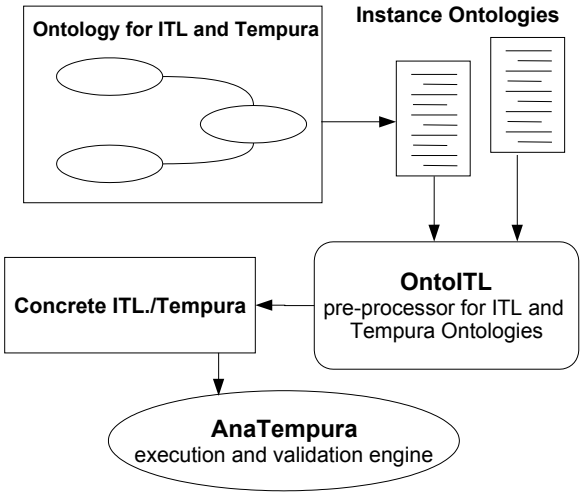

Figure 1: *The TeSCO-S framework*

The following sections present a brief discussion of each of these components.

3.1 An Ontology for Expressing Temporal Formulae and Expressions

Dynamic constraints, rules and temporal properties for e.g. security policies and Assumption - Commitment properties for web services [17, 10] are specified as instances of the core ITL Ontology [1]. The objective is to express as concepts and properties, the syntactical framework of ITL and Tempura. As discussed in section 2, the syntax of ITL is defined primarily by Expressions and Formulae. Expressions can be of various types for e.g. static and state variables, functions, and constants. Similarly formulae can be subclassed as being atomic: e.g. "skip", composite: e.g. "$f_1 ; f_2$" and predicates: e.g. "$isValidISBN(booTitle, ISBNNumber)$" amongst others. We use the Protege OWL Editor [3] for modeling the ontology. A graph showing the class hierarchies in the ontology, generated by OWLViz from the protege-OWL plugin is shown in the following figure 2. Expressions and formulae in the ontology are built incrementally. The root class of all Formulae is

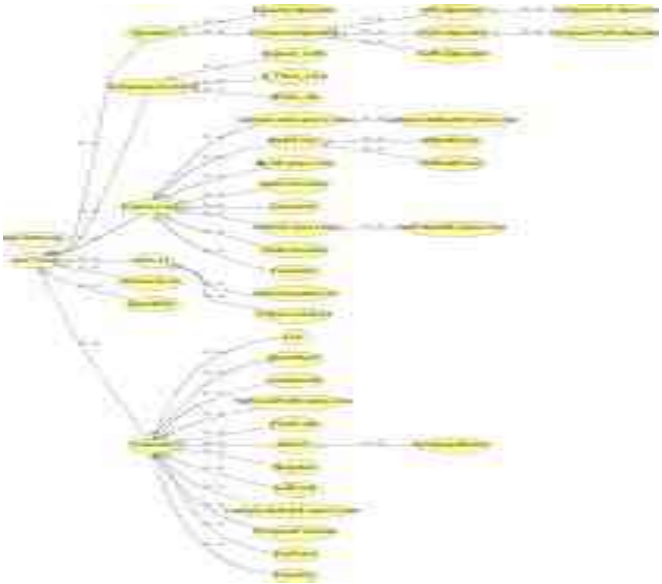

Figure 2: *The ITL/Tempura Ontology*

"`Formula`", while that of Expressions is "`Expression`". Formula has several subclasses such as "`Atomic`", "`Composite`" and "`Prefixed`" amongst others."`TempuraFormula`", defines all those formulae that can be executed by Anatempura. Classes

have properties and restrictions associated that define the kind of parameters that are required to build the expression or formula. Consider a temporal[1] "`Composite Formula`",

$$M = 0 \wedge \bigcirc M = 1 \ldots (1),$$

where M is a state variable. The Class and property specification of a"`Composite formula`" in the ITL Ontology are defined as [2]

```
class:      Composite
subclass:   Formula
properties  :hasPrefixSubFormula
            :hasSuffixSubFormula
            :hasInfixOpertator
```

The ITL formula (1) above can be built in an incremental way, from the classes defined in the ontology, as shown in the table below:

Class	Formula
Equality	$M = 0$
Equality	$M = 1$
Prefixed	$\bigcirc(M = 1)$
Composite	
hasPrefixSubFormula	$M = 0$
hasInfixOperator	$\wedge$
hasSuffixSubFormula	$\bigcirc M = 1$
	$M = 0 \wedge \bigcirc(M = 1)$

Similarly, the following Tempura formula,

exists M:{M=0 and M gets M+1 and len(3)}.

can be incrementally built as above, explicitly specifying it as an instance of class "`TempuraFormula`". The ontology thus provides a modular way to build complex first order temporal formulae and expressions.

3.2 OntoITL: A pre-processor for Temporal Ontologies

OntoITL is a pre-processor that generates concrete ITL and executable Tempura formulae from instance ontologies. The instances are defined using the Core Ontology as described in section 3.1 or from ontologies that import these instances. It provides as output, complete information about instances of State and Static variables, Expressions, Formulae and Temporal Formulae modeled in the ontology. An example output of the pre-processor is shown in the figure 3: OntoITL offers several options to store the generated ITL and Tempura formulae. It also provides the facility to directly pass the tempura formula to the AnaTempura interpreter, that executes the formulae and validates temporal properties. Alternatively, OntoITL stores the generated outputs in files that can be executed via the TclTk interface of AnaTempura.

3.3 AnaTempura: Runtime Validation of Tempura specification

AnaTempura (available from [2]), which is built upon C-Tempura, is an integrated workbench for the runtime verification of systems using ITL and its executable subset Tempura. An overview of the run-time analysis process in AnaTempura is depicted in Figure 4.

AnaTempura generates a state-by-state analysis of the system behaviour as the computation progresses. At various states of execution, values for variables of interest are passed from the system to AnaTempura. The Tempura properties are validated against the values received. If the properties are not satisfied AnaTempura indicates the errors by displaying what is expected and what the current system actually provides. Therefore the approach is not just a "keep tracking" approach i.e. giving the running results of certain properties of the system. By not only capturing the execution results but also comparing them with formal properties, AnaTempura performs the validation. The general architecture that employs Anatempura for validation of service properties is shown in figure 5.

4 Relationship with Exisiting standards

TeSCO-S provides an OWL ontology for modelling temporal properties. It can thus be integrated very easily with exisitng standards for rule languages in OWL e.g. SWRL and with web service standards that describe service interfaces in OWL. As an example, the definition of "`StateVaraible`" can be extended to define it as a *SWRL variable*. Similarly *head* and *body* atoms can be defined in terms of "`Predicate`" in the ITL ontology.Temporal properties that define execution monitoring of OWL-S services can be readily defined as Tempura formulae, which AnaTempura validates at runtime.

[1] We consider a trivial example here, due to lack of space.

[2] Due to lack of space, the ontological representation has been specified here in plain text format with limited expressivity. For more details, please refer the actual ontology

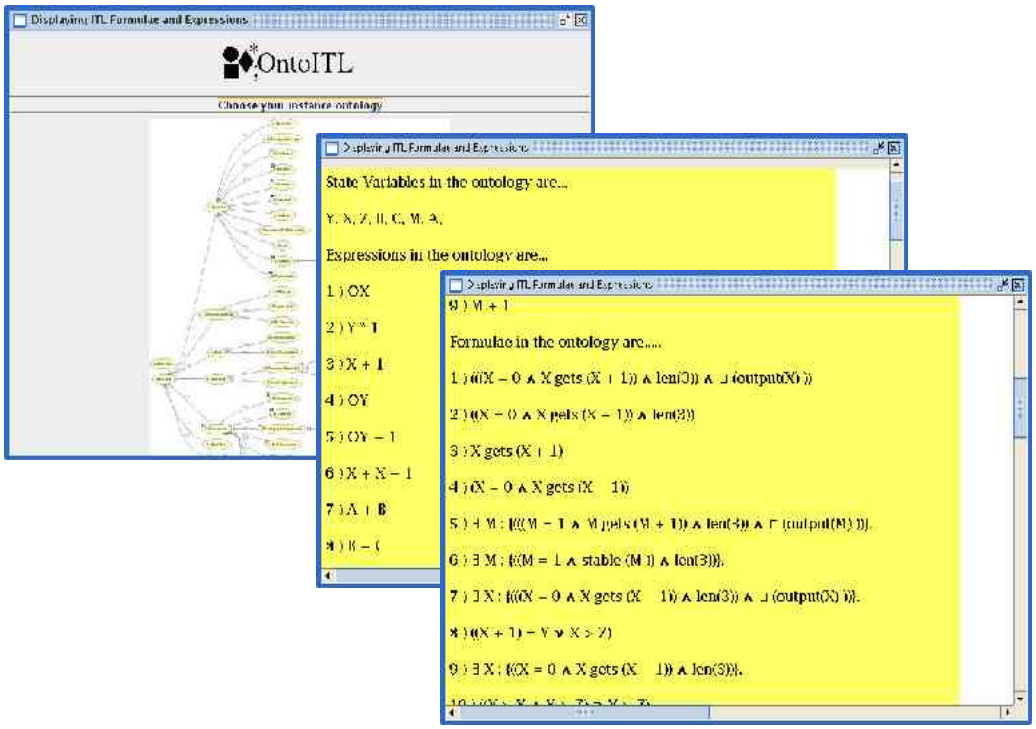

Figure 3: *Some Screen Shots of OntoITL*

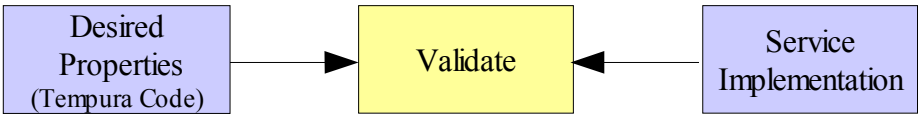

Figure 4: The Analysis Process

5 Conclusion and Future Work

In this paper, we provide a modular approach, TeSCO-S, to building and executing temporal properties of services, with interfaces described as OWL ontologies. TeSCO-S is based on Interval Temporal Logic (ITL) and Tempura, its executable subset. Our pre-processr "OntoITL" enables transformation of the bulky XML representation of temporal properties into concrete ITL and Tempura formulae, that can be handled readily by AnaTempura. The ontology within the TeSCO-S framework can be used by service providers to describe temporal capabilities of services. Service requestors and composing agents can use "OntoITL" and Anatempura for on-the-fly transformation and validation of these temporal properties. The ontology provides constructs not only for specifying temporal expressions and formulae, but general first order predicates and formulae as well. It can therefore, also be used to specify pre-conditions/post-conditions and effects in frameworks like OWL-S and WSMO [3]. Ongoing work in TeSCO-S is providing reasoning support over temporal ontologies and tools for reverse engineering ITL formulae to build temporal ontologies. It is also planned to have a protege plugin for defining temporal ontologies, that could be used along with the OWL-S editor for modelling OWL-S services.

References

[1] An Ontology for ITL and Tempura.
`http://www.cse.dmu.ac.uk/~monika/Pages/Ontologies/OntoITL.owl`.

[2] ITL and (Ana)Tempura Home page on the web.
`http://www.cse.dmu.ac.uk/~cau/itlhomepage/ itlhomepage.html`.

[3] The protege ontology editor and knowledge acquisition system.

[3] To be addressed in a future paper

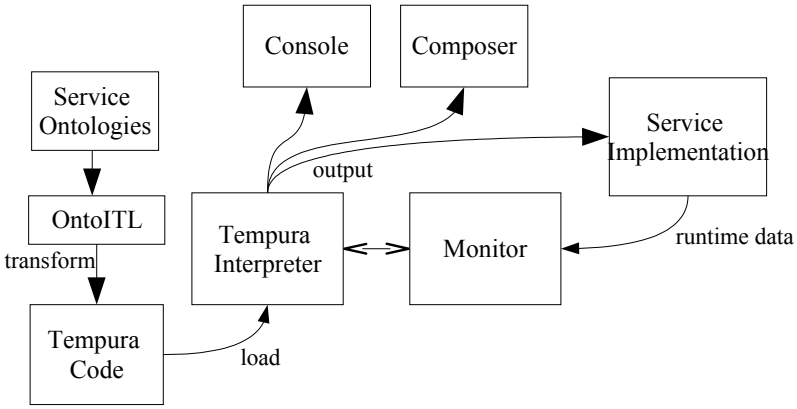

Figure 5: General Architecture

[4] The Rule Markup Initiative. http://www.dfki.uni-kl.de/ruleml/.

[5] OWL Web Ontology Language Reference, 10 February 2004. `http://www.w3.org/TR/owl-ref/`.

[6] Drew McDermott and Dejing Dou . Representing Disjunction and Quantifiers in RDF Embedding Logic in DAML/RDF. International Semantic Web Conference, 2002.

[7] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML . Technical report, W3C Member Submission 21 May 2004.

[8] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, New York, 1991.

[9] Z. Manna and A. Pnueli. *The Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.

[10] Monika Solanki, Antonio Cau, Hussein Zedan. Introducing Compositionality in Web Service Descriptions. Suzhou, China, May 26-28 2004. 10th International Workshop on Future Trends in Distributed Computing Systems - FTDCS 2004, IEEE Computer Society Press.

[11] B. Moszkowski. *Executing temporal Logic Programs.* Cambridge University Press, Cambridge, England, 1986.

[12] B. Moszkowski. *Programming Concepts, Methods and Calculi, IFIP Transactions, A-56.*, chapter Some Very Compositional Temporal Properties, pages 307–326. Elsevier Science B. V., North-Holland, 1994.

[13] B. Moszkowski. *Compositionality: The Significant Difference*, volume 1536 of LNCS, chapter Compositional reasoning using Interval Temporal Logic and Tempura, pages 439–464. Springer Verlag, Berlin, 1996.

[14] Feng Pan and Jerry R. Hobbs. Time in OWL-S. In *Proceedings of AAAI Spring Symposium Series on Semantic Web Services*, 2004.

[15] A Pnueli. Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends. pages 510–584, 1986.

[16] Roberto Chinnic, Martin Gudgin,Jean-Jacques Moreau, Sanjiva Weerawarana. Web Services Description Language (WSDL) Version 1.2, 2003. http://www.w3.org/TR/2003/WD-wsdl12-20030124/#intro.

[17] Monika Solanki, Antonio Cau, and Hussein Zedan. Augmenting semantic web service descriptions with compositional specification. In *Proceedings of the 13th international conference on World Wide Web*, pages 544–552. ACM Press, 2004.

[18] The OWL-S Coalition. OWL-S 1.0 (Beta) Draft Release., 2004. `http://www.daml.org/services/owl-s/1.0/`.

[19] Tony Andrews et al. Business Process Execution Language for Web Services, Version 1.1, 2003. http://www-106.ibm.com/developerworks/library/ws-bpel/.

[20] Web Service Modelling Ontology, 2004. http://www.wsmo.org.