



Project IST 026850 SUPER

Semantics Utilized for Process management within and between Enterprises

Deliverable 4.3

Behavioural Process Mediation
Reasoning Component

Leading Partner: OU

Contributing Partner: OU

Security Classification: Public (PU)

April, 2009

Version 1.0

Project	SUPER	SUPER-Project-No	026850
	Behavioural Process Mediation - Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

Project Details

IST Project Number	026850
Acronym	SUPER
Project Title	Semantics Utilised for Process management within and between EnteRprises
Project URL	http://www.ip-super.org
EU Project Officer	Werner Janusch

Authors (Partner)	Barry Norton (OU)		
Deliverable Owner (Partner)	Barry Norton (OU)	E-mail	b.j.norton@open.ac.uk
		Phone	+44 1908 659399

Project	SUPER	SUPER-Project-No	026850
	Behavioural Process Mediation - Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

Versioning and Contribution History

Version	Description	
0.1	Initial version	Barry Norton (OU)
1.0	Revised for reviewers' comments	Barry Norton (OU)

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

Table of Contents

1 Introduction	2
2 Deliverable Alignment	3
2.1 Architecture Alignment	3
2.2 Methodology Alignment	3
2.3 Modelling Stack Alignment	4
2.4 Use Case Alignment	4
3 Background	6
3.1 Behavioural Reasoning Ontology	6
3.2 BPMO2BRO	7
4 Revised Behavioural Reasoning Ontology	8
4.1 Labels, Actions and Clocks	9
4.2 Variables and Processes	11
4.3 Transitions	11
4.4 Terms	12
5 Conclusions and Further Work	21

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

List of Figures

1	SUPER Architecture	3
2	SUPER Methodology	4
3	SUPER Process Ontology Stack	5
4	Behavioural Reasoning Ontology Main Structure (Abbreviated Terms)	9
5	Full Set of Terms	12

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

List of Tables

1 CaSE Syntax

6

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

Executive Summary

The SUPER project has settled on ontology, rule and reasoning support in the form of the WSML-Flight language. The open-source IRIS reasoning engine has been developed to form the SUPER Process Reasoner, whose final release was documented in Deliverable 6.9. Features specifically motivated by SUPER were added in the form of support for: SQL/SparQL-style aggregate functions to improve the use of logical expressions for querying; WSML-Rule-like function symbols support to allow application of the rule-based data mediation approach. This deliverable will show how the approach that was begun in Deliverables 1.7 — on the design of a Behavioural Reasoning Ontology — and 4.8 — on the rule-based translation of the behaviour of processes expressed in the Business Process Modelling Ontology into this form — has been built upon to form an approach for supporting process mediation with behavioural reasoning via this process reasoner.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

1 Introduction

As remarked in SUPER Deliverable 1.7: “reasoning over the behaviours of the semantic business process models produced by the SUPER methodology is an important basis for tasks that are sensitive to process behaviour, such as *process mediation*, and the inclusion of *conformance* and *compliance* considerations in *discovery* and *composition*”. A Behavioural Reasoning Ontology (BRO) was introduced in that deliverable and it was remarked that the plans for the Process Reasoner developed in Work Package 6, namely the consolidation of support for WSML-Flight reasoning with the addition of aggregate functions, forming WSML-Flight-A, and function symbol support from WSML-Rule could hopefully be made amenable to reasoning over behaviours. With this assumption, this deliverable then was intended to provide support to semantics-based process mediation, as described in Deliverables 4.2 and 4.9, using this ontology. Towards this aim, Deliverable 4.8 introduced a translation of process descriptions from the SUPER Business Process Modelling Ontology (BPMO) into the BRO.

In this deliverable we shall explain problems encountered with the intended approach of extending the definitions of the BRO with axioms based on the algebraic theory of the underlying process calculus, the *Calculus for Synchrony and Encapsulation* (CaSE). In place of such an approach based on the existing CaSE calculus we present an encoding of a variant on the calculus and a means for equational reasoning that allows a novel notion of ‘proof-carrying processes’, which we intend to exploit beyond the project. Although the integration with the SUPER toolset was hampered by the need to change the approach — as well as the need for a small extension to the reasoner, as will be explained — there is a clear and convincing path to the inclusion of this technology.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

2 Deliverable Alignment

Since the task of ontological behavioural reasoning was introduced at the month 12 implementation plan update, it is not a core part of the architecture or methodology, nor on the critical path of any other tool. This section will show, however, where the developed support, oriented towards process mediation, could play a part.

2.1 Architecture Alignment

As explained above, the intention of the Behavioural Reasoning Ontology and related ontologies is to increase the power of the Semantic Business Process Reasoner, which is a core part of the architecture as shown in Figure 1; BRO and BPMO2BRO do not therefore change the architecture, but rather enable this component to achieve more in its currently specified functionality, i.e. in reasoning over ontologies.

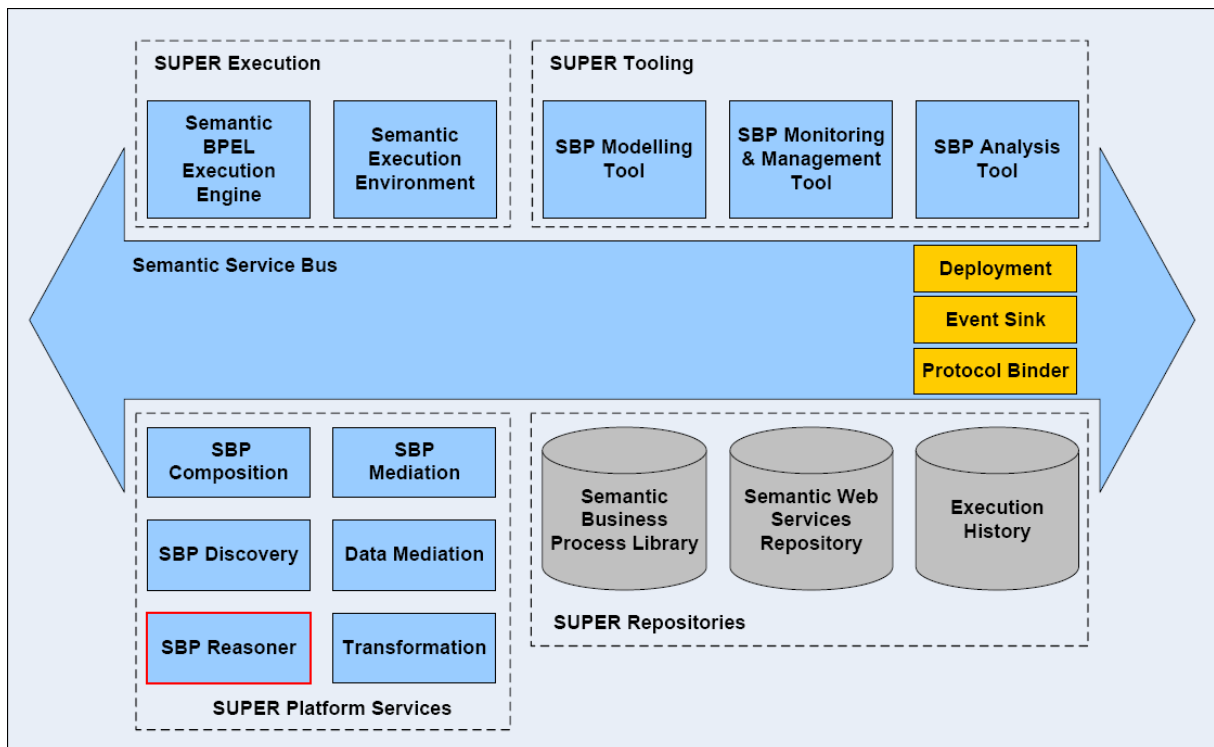


Figure 1: SUPER Architecture

2.2 Methodology Alignment

As shown in the overview of SUPER methodology in Figure 2 four phases depend on the 'ontological foundation', which the Behavioural Reasoning Ontology enriches. The phases to which these ontologies are intended to be of particular benefit are modelling and configuration. Both process mediation and discovery/composition must consider behavioural aspects and each has a design-time and a configuration-time application. In particular reasoning over the Behavioural Reasoning Ontology should be able to

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

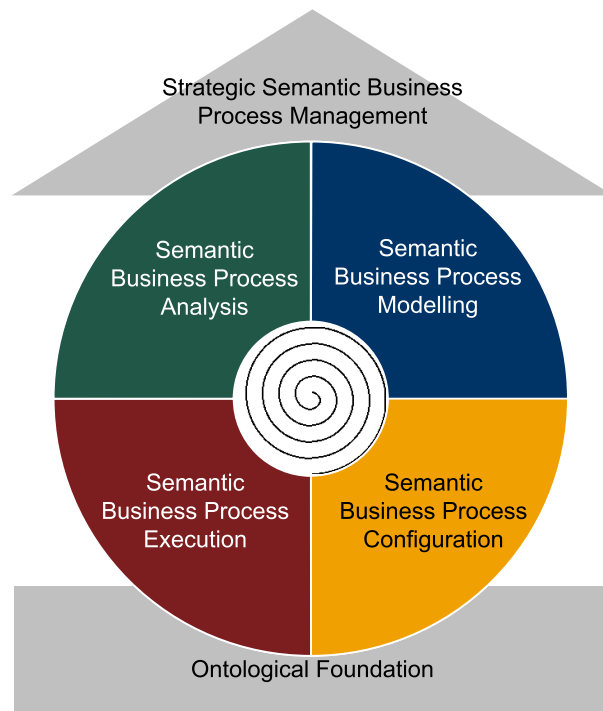


Figure 2: SUPER Methodology

verify the results from such tools concerning: the deadlock freedom of a designed process; the successful behavioural interaction enabled between a set of processes via a mediation process, either designed or discovered; the conformance of a discovered process to its designed counterpart.

2.3 Modelling Stack Alignment

The placement of the Behavioural Reasoning Ontology within the SUPER ontology stack is shown in Figure 3. The final version of the Behavioural Reasoning Ontology and an updated translation from the Business Process Modelling Ontology in BPMO2BRO contribute to the final version of the process stack.

2.4 Use Case Alignment

Use Case alignment follows the use case alignment of SUPER process mediation in general, as described in Deliverables 4.2 and 4.9.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

3 Background

In this section we review the existing work within SUPER on which this deliverable builds.

3.1 Behavioural Reasoning Ontology

CaSE [1] is a process algebra derived from the Calculus of Communicating Systems (CCS) [2] via the Timed Process Language (TPL) [3], which introduces a concept of ‘clock’, measuring the qualitative passage of time, and governed by the principle of maximal progress. Like PMC [4], CaSE is generalised to a set of distinct and independent clocks. As included in BRO each clock in CaSE is governed by the principle of maximal progress, as in CSA [5] but not PMC, and also subject to hiding, which is the novelty of the calculus. The semantic theory follows CCS in three ways: first an *operational semantics* are defined over the syntax using structured operational semantics (SOS) [6] in the form of a labelled transition system, also known in mathematics and logic as a Kripke structure. Secondly a semantic behavioural equivalence is defined over these operational semantics to relate processes; CaSE follows CCS in choosing bisimulation and its *observational* generalisation as equivalences. Finally the semantic behavioural equivalence is projected onto the syntax via axiomatisation to form an algebra.

The original design of the BRO subsisted in the structural representation of the CaSE process algebra, with a conceptual representation of both process terms and equational expressions, with placeholders for three behavioural relations: strong congruence; weak similarity; and temporal observation congruence. The intention was to use the completeness proof for (syntactic equational) axiomatisation of the last of these, together with the (syntactic) axioms themselves, in the form of (ontological) axioms or, more concretely, as WSMML-Flight rules (defining membership in the three relations).

Since the syntax of the calculus is unchanged in our new approach, we re-include that in Table 1. The grammar of terms, \mathcal{E} , is dependent on a number of related sets of the right hand side. In common with CCS, the set Λ names communication channels (in the syntax the names are drawn from a, b, c etc., and subscripts of these), which are the basis for two-party asymmetric synchronous communications. *Actions*, ranged over by α, β etc., are drawn from the set $\Lambda \cup \bar{\Lambda} \cup \{\tau\}$, where τ is called the silent action. In CCS the labels in the labelled transition system-based operational semantics are drawn exclusively from actions. In TPL this set is extended with one further label, σ , representing ticks of a global clock which marks the progress of time qualitatively, i.e. marks out periods where measurable progress of time can happen, but assigns no measurement to the *amount* of time that can progress. As in the predecessors to CaSE, PMC and CSA, this generalises to a family of clocks $\{\rho, \sigma, \dots\}$ moving at different rates.

$\mathcal{E} ::= \mathbf{0} \mid \alpha.\mathcal{E} \mid \mathcal{E} \setminus a \mid \mathcal{E} + \mathcal{E} \mid \mathcal{E} \mathcal{E} \mid$	$a, \bar{a}, b, \bar{b}, \dots \in \Lambda \cup \bar{\Lambda}$
$\Delta \mid \Delta_\sigma \mid \mathcal{E}/\sigma \mid [\mathcal{E}]\sigma(\mathcal{E}) \mid \lceil \mathcal{E} \rceil \sigma(\mathcal{E}) \mid$	$\rho, \sigma, \dots \in \mathcal{T}$
$\mu X.\mathcal{E} \mid X$	$\alpha, \beta, \dots \in \Lambda \cup \bar{\Lambda} \cup \{\tau\}$
	$\gamma, \delta, \dots \in \Lambda \cup \bar{\Lambda} \cup \{\tau\} \cup \mathcal{T}$

Table 1: CaSE Syntax

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

An action matching a channel name is usually taken to be an input on that channel which can be matched, to form a communication, with an output represented by the action with the complementary decorated form of the name. Such a communication is then represented as a τ , which may also be performed directly as an action representing some internal behaviour. This action has no complement and does not need a communication partner, nor is it visible to any observer, and so is called the *silent* action. In CaSE, as encoded in the M24 version of BRO this action, whether derived through communication or included directly by prefix, has a higher priority than any clock and will preempt transitions in any clock at the state in which it occurs. As we shall see this preemption, called maximal progress, has consequences for the ability to use WSML-Flight to give behavioural semantics to processes.

3.2 BPMO2BRO

The BPMO2BRO ontology consists of WSML-Flight rules, with the addition of WSML-Rule-like function symbols as supported by IRIS as SUPER process reasoner, which infer instances over the BRO ontology from those in the BPMO ontology. As such the current version imports both legacy versions as shown in Listing 3.1.

```

wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace { _"http://ip-super.org/ontologies/BPMO2BRO#",
  bpmo _"http://www.ip-super.org/ontologies/BPMO/20071122#",
  bro _"http://ip-super.org/ontologies/bro#" }

ontology _"http://ip-super.org/ontologies/BPMO2BRO#"
importsOntology { _"http://www.ip-super.org/ontologies/BPMO/20071122#bpmo-1-4",
  _"http://ip-super.org/ontologies/bro#" }

relation behaviouralSemantics(impliesType bpmo#WorkflowElement, impliesType bro#ProcessTerm)

```

Listing 3.1: BPMO2BRO Ontology Header

A simple translation rule, acting on BPMO GoalTask instances, populating the relation between the general concepts in these two ontologies, is shown in Listing 3.2.

```

axiom aGoalTask definedBy
  ?g memberOf bpmo#GoalTask implies
  r(?g) memberOf bro#Clock and
  e(?g) memberOf bro#Channel and
  c(?g) memberOf bro#Clock and
  ie(?g)[bro#hasChannel hasValue e(?g)] memberOf bro#Input and
  x(?g) memberOf bro#ProcessVariable and
  sem(?g)[bro#bindsVariable hasValue x(?g),
  bro#hasBody hasValue sem(?g, 1)] memberOf bro#VariableBinding and
  sem(?g, 1)[bro#hasAction hasValue ie(?g),
  bro#hasBody hasValue sem(?g, 2)] memberOf {bro#Prefix, bro#ProcessTerm} and
  sem(?g, 2)[bro#hasVariable hasValue x(?g)] memberOf {bro#FreeVariable, bro#ProcessTerm} and
  behaviouralSemantics(?g, sem(?g)).

```

Listing 3.2: GoalTask Translation Rule

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

4 Revised Behavioural Reasoning Ontology

The stated intention, in Deliverable 1.7, for extending the BRO was to automatically populate the relations shown in Listing 4.1.

```

relation StrongCongruence (ofType ProcessTerm, ofType ProcessTerm) subRelationOf TemporalObservationCongruence
  nfp
    dc#relation hasValue {aStrongCongruence}
  endnfp

relation TemporalObservationCongruence (ofType ProcessTerm, ofType ProcessTerm) subRelationOf WeakEquivalence
  nfp
    dc#relation hasValue {aTemporalObservationCongruence}
  endnfp

relation WeakEquivalence (ofType ProcessTerm, ofType ProcessTerm)
  nfp
    dc#relation hasValue {aWeakEquivalence}
  endnfp

```

Listing 4.1: Behavioural Relations

In particular the plan was to infer from each ProcessTerm an equation set, according to [7], where each variable is defined by an instance of ExpressionTerm. For this reason both ProcessTerm and ExpressionTerm were subconcepts to a generic Term, where axioms were needed to keep the two separate. The intention was to encode the syntactic axioms of CaSE in WSML-Flight axioms and to infer instances of ExpressionTerm along the lines of the associated completeness proof (the main difference between the ontological axioms and the syntactic ones is that the syntax assumes all terms exist, whereas equational axioms only relate existant instances; rule-type axioms have rather to be used to infer such instances, and such a structure is necessary to avoid implying all possible terms).

Unfortunately while early experiments found some success in relating regular terms, it became clear that no simple encoding could avoid that the *Expansion Law* — which removes parallelism, forming such terms — would be dependent on a *negative precondition*, i.e. would require to infer instances from the absence of others. WSML-Flight is based on a particular version of the ‘closed World assumption’ wherein negation as failure is allowable in some logical expressions, but not in the head of rules (and strong negation not at all), in order to ensure consistency and decidability.

A partial solution to the problem is based on the observance that with minimal change in the existing BPMO translation, due to the consistent use of the ‘stalling’ processes which explicitly control clocks, maximal progress can be dropped from CaSE, forming a calculus here named CaSE^{mpf} . Unfortunately there currently exists no algebraic equational theory for this calculus, but it is a simple matter to show that strong bisimulation forms a congruence and that a simple correction to bisimulation over weak transitions forms an observation congruence. While deriving such a bisimulation continues to be a second-order problem, we can provide reasoning support for doing so by providing a transition and a weak transition relation, thereby furthering the claim that ontological behavioural semantics are given to BPMO, but more importantly we can *validate* such a bisimulation once encoded. In this way we claim to relate to the notion of ‘proof-carrying code’, as we shall discuss in Section 5.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

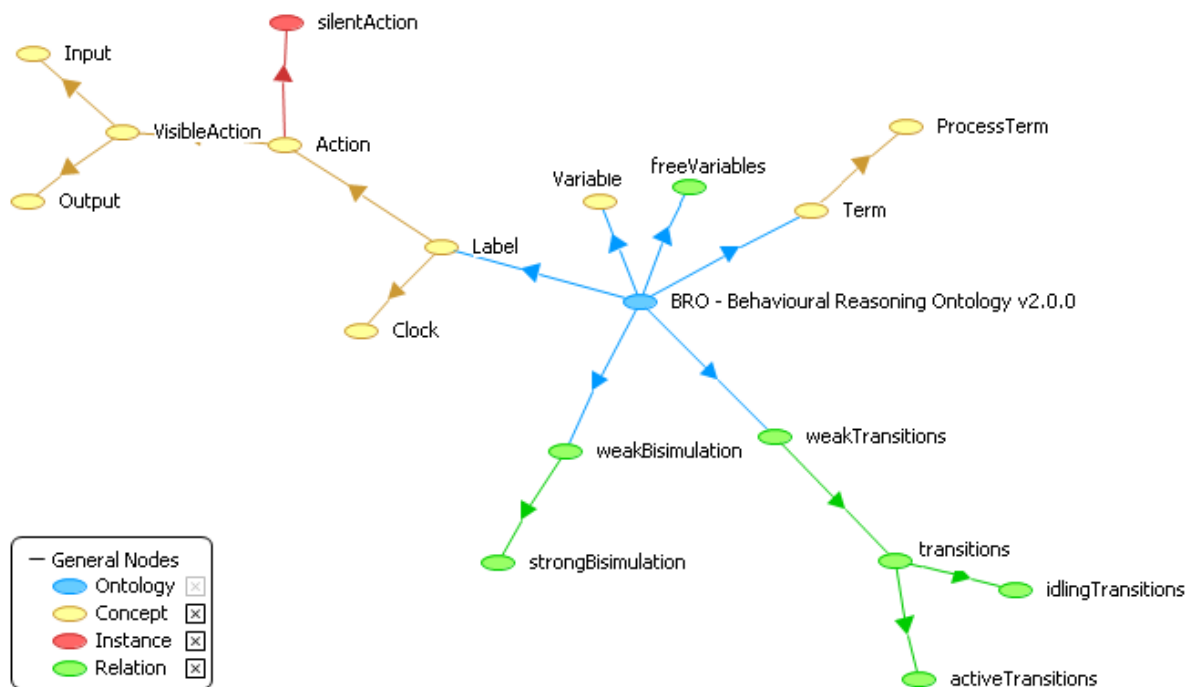


Figure 4: Behavioural Reasoning Ontology Main Structure (Abbreviated Terms)

4.1 Labels, Actions and Clocks

In order to explain the revised and completed Behavioural Reasoning Ontology we proceed through the revised process term definitions, showing in each case what SOS rules derive the labelled transition system (LTS) semantics and how this is encoded as WSMML-Flight rules. The overall structure of the ontology (where the full set of subconcepts to Term are elided) can be seen in Figure 4.

It can be seen on the left-hand side of this figure that, in order to model transitions, an explicit concept of Label has been introduced, of which both Action and Clock are subconcepts. As previously, the silent action is an instance of Action. For reasons that will become clear later both channels (which were once instances of an explicit concept), underlying matching inputs and outputs, and clocks are defined using integer indices. Listing 4.2 shows the relevant definitions¹.

```

concept Label
  nfp
    dc#description hasValue "A label labels a transition ."
    dc#relation hasValue {aLabel}
  endnfp

axiom aLabel
  nfp
    dc#description hasValue "A valid label is either an action or a clock"
  endnfp
  definedBy
    !- ?l memberOf Label and
      (naf ?l memberOf Action) and (naf ?l memberOf Clock).

```

¹Axioms UniqueOutputs and UniqueClocks follow the pattern of UniqueInputs and are not reproduced.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

```

concept Action subConceptOf Label
  nfp
    dc#description hasValue "An action is either silent or a communication on a numbered channel"
    dc#relation hasValue {anAction}
  endnfp

axiom anAction definedBy
  !- ?a memberOf Action and
    ?a != silentAction and
    (naf ?a memberOf Input) and
    (naf ?a memberOf Output).

instance silentAction memberOf Action
  nfp
    dc#description hasValue "A silent action is internal step and not the basis of synchronisation."
  endnfp

concept VisibleAction subConceptOf Action
  nfp
    dc#description hasValue "A visible action is due to a blocking communication on a numbered channel."
    dc#relation hasValue {uniqueInputs}
  endnfp
  hasChannel ofType (1) _integer

concept Input subConceptOf VisibleAction
  nfp
    dc#description hasValue "An input is a blocking wait on a numbered channel until matched by an output on the
      same channel."
    dc#relation hasValue {uniqueOutputs}
  endnfp

axiom uniqueInputs
  nfp
    dc#description hasValue "Each input must have a unique channel index."
  endnfp
  definedBy
  !- ?a[hasChannel hasValue ?n] memberOf Input and
    ?b[hasChannel hasValue ?n] memberOf Input and
    ?a != ?b.

concept Output subConceptOf VisibleAction
  nfp
    dc#description hasValue "An output is a blocking wait on a numbered channel until matched by an input on the
      same channel."
    dc#relation hasValue {uniqueOutputs}
  endnfp

concept Clock subConceptOf Label
  nfp
    dc#description hasValue "Clocks are the basis of multi-party symmetric synchronisations (potentially subject to
      maximal progress)"
    dc#relation hasValue {uniqueClocks}
  endnfp
  hasIndex ofType (1) _integer

```

Listing 4.2: Actions and Clocks in the Behavioural Reasoning Ontology

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

4.2 Variables and Processes

Following Figure 4 around clockwise, we see that the existing concept Variable is preserved; unlike before each instance is also subject to an integer index, allowing us to form a relation FreeVariables, as shown in Listing 4.3, and thereby relate the concept ProcessTerm to Term more accurately than previously by insisting that process terms contain no free variables.

```

concept Variable
  nfp
    dc#description hasValue "Variables range over terms"
    dc#relation hasValue {uniqueVariables}
  endnfp
  hasIndex ofType (1) .integer

concept Term
  nfp
    dc#description hasValue "Terms defines a behaviour up to free variables"
    dc#relation hasValue {aCaseTerm}
  endnfp

relation freeVariables (ofType Term, ofType Variable)
  nfp
    dc#description hasValue "Free variables are variables that are unbound within a term"
    dc#relation hasValue {fvPrefix, fvRestriction,
      fvChoice1, fvChoice2, fvParallel1, fvParallel2, fvTimeout, fvHiding, fvFreeVariable, fvVariableBinding}
  endnfp

concept ProcessTerm subConceptOf Term
  nfp
    dc#description hasValue "A process is a term with no free variables"
    dc#relation hasValue {aProcessTerm}
  endnfp

axiom aProcessTerm definedBy
  !— ?p memberOf ProcessTerm and freeVariables (?p, ?v).

```

Listing 4.3: Variables, Terms and Process Terms

4.3 Transitions

Continuing to follow Figure 4 around clockwise, we see the relations transitions, made up of active transitions and idling transitions, and the larger set of weakTransitions, defined as shown in Listing 4.4.

```

relation activeTransitions (ofType Term, ofType Label, ofType Term) subRelationOf transitions

relation idlingTransitions (ofType Term, ofType Label, ofType Term) subRelationOf transitions

relation transitions (ofType Term, ofType Label, ofType Term) subRelationOf weakTransitions

```

Listing 4.4: Transitions

Due to the subRelationOf declaration, both members of activeTransitions, by which the state is changed, and idlingTransitions, which merely admit a clock without changing the state, are included in transitions.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

It can be seen that transitions are defined to be contained within a wider set of weak transitions, defined as shown in Listing 4.5. Herein further axioms infer that they are prefixed by the transitive closure of the silent transition relation and suffixed by the same.

relation weakTransitions (**ofType** Term, **ofType** Label, **ofType** Term)

axiom weakBackwardsChaining **definedBy**
 weakTransitions (?p, silentAction , ?q) **and**
 weakTransitions (?q, ?l, ?r) **implies**
 weakTransitions (?p, ?l, ?r).

axiom weakForwardsChaining **definedBy**
 weakTransitions (?p, ?l, ?q) **and**
 weakTransitions (?q, silentAction , ?r) **implies**
 weakTransitions (?p, ?l, ?r).

Listing 4.5: Weak Transitions

4.4 Terms

We now consider each process term in turn, based on the rules by which it populates these two new relations, freeVariables and transitions, following them around clockwise as presented in Figure 5. Terms are constrained to be exclusively drawn from this set by the constraining axiom aCaSETerm, as shown in Listing 4.6².

²We remind the reader that CaSE^{mpf} terms are drawn from the same syntax.

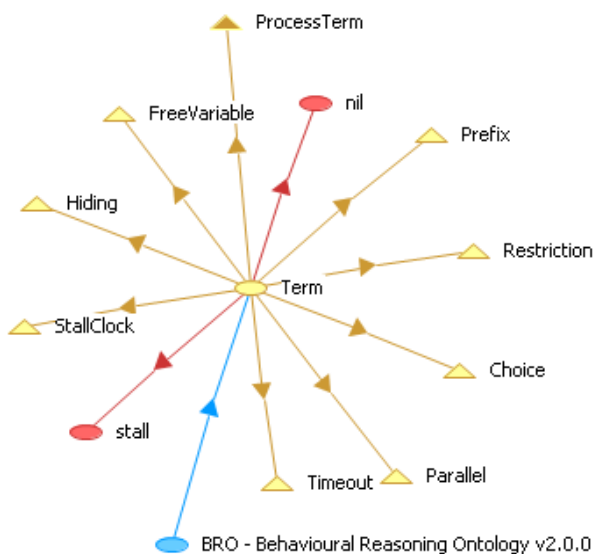


Figure 5: Full Set of Terms

axiom aCaSETerm
definedBy
 !- ?t **memberOf** Term **and**
 ?t != nil **and**
 (naf ?t **memberOf** Prefix) **and**
 (naf ?t **memberOf** Restriction) **and**
 (naf ?t **memberOf** Choice) **and**
 (naf ?t **memberOf** Parallel) **and**
 ?t != stall **and**
 (naf ?t **memberOf** StallClock) **and**
 (naf ?t **memberOf** Timeout) **and**
 (naf ?t **memberOf** Hiding) **and**
 (naf ?t **memberOf** FreeVariable) **and**
 (naf ?t **memberOf** VariableBinding).

Listing 4.6: Variables, Terms and Process Terms

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

We start with nil:

```

instance nil memberOf Term
  nfp
    dc#description hasValue "Nil is the inactive behaviour, i.e. the one which engages in no communications and
    have no effect on the progress of time."
    dc#relation hasValue {transNil}
  endnfp

axiom transIdle
  nfp
    dc#description hasValue "All defined clocks idle on nil."
  endnfp
  definedBy
    ?c memberOf Clock implies
      idlingTransitions ( nil , ?c, nil ).

```

Listing 4.7: The nil process

Note that by definition the term consisting of nil is a process since it contains no free variables, and there is therefore no rule to populate the freeVariables relation. The rule inferring transitions is equivalent to the following rule in SOS form, unchanged from CaSE as presented in the appendix to Deliverable 4.8:

$$\text{Idle} \quad \frac{}{\mathbf{0} \xrightarrow{\sigma} \mathbf{0}}$$

We note that this transition is introduced through the relation of explicitly idling transitions and therefore is not active.

Listing 4.8 considers prefixed terms:

```

concept Prefix subConceptOf Term
  nfp
    dc#description hasValue "A prefix defines a step of behaviour according to an action, a body defines the
    remainder of the behaviour."
    dc#relation hasValue {fvPrefix, transAct, transAct2b}
  endnfp
  hasAction ofType (1) Action
  hasBody ofType (1) Term

axiom fvPrefix
  nfp
    dc#description hasValue "A prefix introduces no new variables, but preserves those of the body."
  endnfp
  definedBy
    ?t[hasBody hasValue ?s] memberOf Prefix and freeVariables(?s, ?v) implies freeVariables(?t, ?v).

axiom transAct
  nfp
    dc#description hasValue "All defined clocks idle on prefixes (note: even on silent actions since there is no
    maximal progress)."
  endnfp
  definedBy
    ?p memberOf Prefix and
    ?c memberOf Clock implies
      activeTransitions (?p, ?a, ?q).

```

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

```

axiom transAct2b
  nfp
    dc#description hasValue "A prefix performs the action then behaves as the body."
  endnfp
  definedBy
    ?p[hasAction hasValue ?a, hasBody hasValue ?q] memberOf Prefix implies
    idlingTransitions (?p, ?c, ?p).

```

Listing 4.8: Prefixed terms

Here the free variables are simply those of the body of the term, none being introduced by an action prefix. The transition rules correspond respectively with the following:

$$\text{Act} \quad \frac{}{\alpha.E \xrightarrow{\alpha} E} \quad \text{Act2b} \quad \frac{}{\alpha.E \xrightarrow{\sigma} \alpha.E}$$

While Act is unchanged from the previous presentation, Act2b replaces Act2 by allowing clock transitions also for silent action prefixes. We note that the former are active transitions, but the latter only idling.

Listing 4.9 considers the restriction of channel names over terms:

```

concept Restriction subConceptOf Term
  nfp
    dc#description hasValue "A restricted term defines a behaviour according to its subterm, but removing behaviours starting with communications on any of the restricted channels."
  endnfp
  hasSubterm ofType (1) Term
  hasChannel ofType (1) _integer

axiom fvRestriction
  nfp
    dc#description hasValue "A restriction introduces no new variables, but preserves those of the subterm."
    dc#relation hasValue {fvRestriction, transRes1a, transRes1b, transRes1c, transRes1d}
  endnfp
  definedBy
    ?t[hasSubterm hasValue ?s] memberOf restriction and freeVariables(?s, ?v) implies freeVariables(?t, ?v).

axiom transRes1a
  nfp
    dc#description hasValue "A restriction has all of the active clock transitions of the subterm, preserving the restriction over the new subterm."
  endnfp
  definedBy
    ?t[hasSubterm hasValue ?s, hasChannel hasValue ?n] memberOf Restriction and
    activeTransitions (?s, ?c, ?s2) and
    ?c memberOf Clock implies
    activeTransitions (?t, ?c, deriv(?t)) and
    deriv(?t)[hasSubterm hasValue ?s2, hasChannel hasValue ?n] memberOf Restriction.

axiom transRes1b
  nfp
    dc#description hasValue "A restriction has all of the idling clock transitions, preserving the whole term."
  endnfp
  definedBy
    ?t[hasSubterm hasValue ?s, hasChannel hasValue ?n] memberOf Restriction and
    idlingTransitions (?s, ?c, ?s2) and
    ?c memberOf Clock implies
    idlingTransitions (?t, ?c, ?t).

```

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

```

axiom transRes1c
  nfp
    dc#description hasValue "A restriction has all of the silent transitions , preserving the restriction ."
  endnfp
  definedBy
    ?t[hasSubterm hasValue ?s, hasChannel hasValue ?n] memberOf Restriction and
    activeTransitions (?s, silentAction , ?s2) implies
      activeTransitions (?t, silentAction , deriv(?t)) and
      deriv(?t)[hasSubterm hasValue ?s2, hasChannel hasValue ?n] memberOf Restriction.

axiom transRes1d
  nfp
    dc#description hasValue "A restriction has all of the silent transitions of the subterm, preserving the
    restriction ."
  endnfp
  definedBy
    ?t[hasSubterm hasValue ?s, hasChannel hasValue ?m] memberOf Restriction and
    activeTransitions (?s, ?a, ?s2) and
    ?a[hasChannel hasValue ?n] memberOf VisibleAction and
    wsml#numericInequal(?m, ?n) implies
      activeTransitions (?t, silentAction , deriv(?t)) and
      deriv(?t)[hasSubterm hasValue ?s2, hasChannel hasValue ?n] memberOf Restriction.

```

Listing 4.9: Restriction of channels over terms

As above it takes only one rules to derive the free variables of the restricted term from the subterm. On the other hand the following SOS rule has to be treated very carefully:

$$\text{Res} \quad \frac{E \xrightarrow{\gamma} E'}{E \setminus a \xrightarrow{\gamma} E' \setminus a} \quad \gamma \notin \{a, \bar{a}\}$$

There are three WSML-Flight rules corresponding to this. transRes1a deals with strictly active clock transitions, introduced as we shall show below by time-outs, where a new term is created as the derivative. On the other hand idling transitions are treated by transRes1b in order that no new term instance is created (otherwise an infinite number of instances, all encoding structurally the same term, would be inferred). transRes1c deals with the active silent transitions of the subterm. Finally transRes1d deals with the visible transactions and thereby the side condition which uses the built-in wsml#numericInequal. We note that general identity comparisons cannot be made in the head of WSML-Flight rules and therefore that this rule, among others, motivates the use of integer indices in the revised ontology.

Listing 4.10 considers non-deterministic choices between the behaviours of a pair of terms, which are chosen between by actions but not broken by clocks, which must proceed on both sides in order to preserve the choice:

```

concept Choice subConceptOf Term
  nfp
    dc#description hasValue "A choice defines a behaviour in which the first step is a non-deterministic choice
    between the behaviour of two subterms, according to whichever performs an action first."
    dc#relation hasValue {fvChoice1, fvChoice2, transSum1, transSum2, transSum3a, transSum3b, transSum3c,
    transSum3d}
  endnfp
  hasLeftSubterm ofType (1) Term
  hasRightSubterm ofType (1) Term

```

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

axiom fvChoice1

nfp

dc#description **hasValue** "The free variables of a choice include those from the left subterm."

endnfp

definedBy

?t[hasLeftSubterm **hasValue** ?s] **memberOf** Choice **and** freeVariables(?s, ?v) **implies** freeVariables(?t, ?v).

axiom fvChoice2

nfp

dc#description **hasValue** "The free variables of a choice include those from the right subterm."

endnfp

definedBy

?t[hasRightSubterm **hasValue** ?s] **memberOf** Choice **and** freeVariables(?s, ?v) **implies** freeVariables(?t, ?v).

axiom transSum1

nfp

dc#description **hasValue** "A non-deterministic choice can be resolved with an action from the left-hand term, in which case the whole process proceeds as per the left-hand derivative."

endnfp

definedBy

?t[hasLeftSubterm **hasValue** ?s] **memberOf** Choice **and**
activeTransitions (?s, ?a, ?s2) **and**
?a **memberOf** Action **implies**
activeTransitions (?t, ?a, ?s2).

axiom transSum2

nfp

dc#description **hasValue** "A non-deterministic choice can be resolved with an action from the right-hand term, in which case the whole process proceeds as per the right-hand derivative."

endnfp

definedBy

?t[hasRightSubterm **hasValue** ?s] **memberOf** Choice **and**
activeTransitions (?s, ?a, ?s2) **and**
?a **memberOf** Action **implies**
activeTransitions (?t, ?a, ?s2).

axiom transSum3a

nfp

dc#description **hasValue** {"A non-deterministic choice is not resolved with a clock, but allows this clock when both subterms do, preserving the choice. In this version both sides make an active transition"}

endnfp

definedBy

?t[hasLeftSubterm **hasValue** ?s1, hasRightSubterm **hasValue** ?s2] **memberOf** Choice **and**
activeTransitions (?s1, ?c, ?s3) **and**
activeTransitions (?s2, ?c, ?s2) **and**
?c **memberOf** Clock **implies**
activeTransitions (?t, ?c, deriv(?t)).

axiom transSum3b

definedBy

?t[hasLeftSubterm **hasValue** ?s1, hasRightSubterm **hasValue** ?s2] **memberOf** Choice **and**
activeTransitions (?s1, ?c, ?s3) **and**
idlingTransitions (?s2, ?c, ?s4) **and**
?c **memberOf** Clock **implies**
activeTransitions (?t, ?c, deriv(?t)) **and**
deriv(?t)[hasLeftSubterm **hasValue** ?s3, hasRightSubterm **hasValue** ?s4] **memberOf** Choice.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

axiom transSum3c
definedBy
 ?t[hasLeftSubterm **hasValue** ?s1, hasRightSubterm **hasValue** ?s2] **memberOf** Choice **and**
 idlingTransitions (?s1, ?c, ?s3) **and**
 activeTransitions (?s2, ?c, ?s4) **and**
 ?c **memberOf** Clock **implies**
 activeTransitions (?t, ?c, deriv(?t)) **and**
 deriv(?t)[hasLeftSubterm **hasValue** ?s3, hasRightSubterm **hasValue** ?s4] **memberOf** Choice.

axiom transSum3d
nfp
 dc#description **hasValue** {"A non-deterministic choice is not resolved with a clock, but allows this clock when both subterms do, preserving the choice."
 "In this version both sides idle so the whole term has an idle self-transition"}
endnfp
definedBy
 ?t[hasLeftSubterm **hasValue** ?s1, hasRightSubterm **hasValue** ?s2] **memberOf** Choice **and**
 idlingTransitions (?s1, ?c, ?s3) **and**
 idlingTransitions (?s2, ?c, ?s4) **and**
 ?c **memberOf** Clock **implies**
 activeTransitions (?t, ?c, ?t).

Listing 4.10: Non-deterministic choices between terms

The rules represent the following SOS rules:

$$\text{Sum1} \frac{E \xrightarrow{\alpha} E'}{E + F \xrightarrow{\alpha} E'} \quad \text{Sum2} \frac{F \xrightarrow{\alpha} F'}{E + F \xrightarrow{\alpha} F'} \quad \text{Sum3} \frac{E \xrightarrow{\sigma} E' \quad F \xrightarrow{\sigma} F'}{E + F \xrightarrow{\sigma} E' + F'}$$

In the case of Sum3 four separate WSML-Flight rules are needed due to the separation of active and idling transitions, all the combinations of which must be considered in order for (in the last rule transSum3d) idling on both sides to give rise to an idle transition only.

Listing 4.11 considers parallel compositions of the behaviours of a pair of terms³:

concept Parallel **subConceptOf** Term
nfp
 dc#description **hasValue** "A parallel term defines a behaviour according to the concurrent interaction of the behaviours defined by its subterms."
endnfp
 hasLeftSubterm **ofType** (1) Term
 hasRightSubterm **ofType** (1) Term

axiom transCom1
nfp
 dc#description **hasValue** "An action from the left-hand term induces the same in its parallel composition, but preserving the right-hand component."
endnfp
definedBy
 ?t[hasLeftSubterm **hasValue** ?s1, hasRightSubterm **hasValue** ?s2] **memberOf** Parallel **and**
 activeTransitions (?s1, ?a, ?s3) **and**
 ?a **memberOf** Action **implies**
 activeTransitions (?t, ?a, deriv(?t)) **and**
 deriv(?t)[hasLeftSubterm **hasValue** ?s3, hasRightSubterm **hasValue** ?s2] **memberOf** Parallel.

³The rules fvParallel1 and fvParallel2 match though for non-deterministic choice and so are elided.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

axiom transCom2

nfp

dc#description **hasValue** "An action from the right – hand term induces the same in its parallel composition, but preserving the left – hand component."

endnfp

definedBy

?t[hasLeftSubterm **hasValue** ?s1, hasRightSubterm **hasValue** ?s2] **memberOf** Parallel **and**
activeTransitions (?s2, ?a, ?s3) **and**
?a **memberOf** Action **implies**
activeTransitions (?t, ?a, deriv(?t)) **and**
deriv(?t)[hasLeftSubterm **hasValue** ?s1, hasRightSubterm **hasValue** ?s3] **memberOf** Parallel.

axiom transCom3a

nfp

dc#description **hasValue** "An input from the left – hand term can be matched by an output on the same channel from the right – hand term to induce a silent transition to the composition of the derivatives in a parallel composition."

endnfp

definedBy

?t[hasLeftSubterm **hasValue** ?s1, hasRightSubterm **hasValue** ?s2] **memberOf** Parallel **and**
activeTransitions (?s1, ?a, ?s3) **and**
?a[hasChannel **hasValue** ?n] **memberOf** Input **and**
activeTransitions (?s2, ?b, ?s4) **and**
?a[hasChannel **hasValue** ?n] **memberOf** Output **implies**
activeTransitions (?t, silentAction, deriv(?t)) **and**
deriv(?t)[hasLeftSubterm **hasValue** ?s3, hasRightSubterm **hasValue** ?s4] **memberOf** Parallel.

axiom transCom4a

nfp

dc#description **hasValue** {"A parallel composition a clock is permitted, preserving the composition through pairwise derivatives, if both sides allow it .",
"In this version both sides make an active transition"}

endnfp

definedBy

?t[hasLeftSubterm **hasValue** ?s1, hasRightSubterm **hasValue** ?s2] **memberOf** Parallel **and**
activeTransitions (?s1, ?c, ?s3) **and**
activeTransitions (?s2, ?c, ?s4) **and**
?c **memberOf** Clock **implies**
activeTransitions (?t, ?c, deriv(?t)) **and**
deriv(?t)[hasLeftSubterm **hasValue** ?s3, hasRightSubterm **hasValue** ?s4] **memberOf** Parallel.

Listing 4.11: Parallel compositions of terms

The rules⁴ encode the following SOS rules which mean that parallel compositions are broken neither by actions nor clocks, but where actions can proceed independently, clocks must synchronise and where complementary visible actions can synchronise to produce a silent action, as in the following SOS rules:

$$\text{Com1} \frac{E \xrightarrow{\alpha} E'}{E \mid F \xrightarrow{\alpha} E' \mid F} \quad \text{Com2} \frac{F \xrightarrow{\alpha} F'}{E \mid F \xrightarrow{\alpha} E \mid F'}$$

$$\text{Com3} \frac{E \xrightarrow{\alpha} E', F \xrightarrow{\bar{\alpha}} F'}{E \mid F \xrightarrow{\tau} E' \mid F'} \quad \text{Com4b} \frac{E \xrightarrow{\sigma} E', F \xrightarrow{\sigma} F'}{E \mid F \xrightarrow{\sigma} E' \mid F'}$$

⁴The rule transSum3b is elided since it matches transSum3a exactly except in the reversal of input and output. Similarly the elided rules Com4b to Com4d are elided since they match the pattern of rules Sum3b to Sum3d.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

Note that Com4b drops the side condition to Com4, as previously presented, which enforces maximal progress.

Continuing clockwise around Figure 5 the remaining terms concern timed behaviour — time-out introducing a clock transition, as prefix introduces an action transition; stall and relative stall being inactive like nil, but more restricted in their timed behaviour; and hiding removing clock transitions, as restriction removes action transitions, but replacing them with the same silent behaviour — and recursion. For the sake of brevity we elide these definitions and refer the interested reader to the full ontology definitions at the SUPER website.

The novelty of having the transitions induced by rules is the ability to introduce bisimulation relations as shown in Listing 4.12.

```
relation strongBisimulation (ofType ProcessTerm, ofType ProcessTerm) subRelationOf weakBisimulation
```

```
axiom strongBisimulation1 definedBy
  !- strongBisimulation(?p, ?q) and
  transitions (?p, ?l, ?p2) and
  naf ( transitions (?q, ?l, ?q2) and
       strongBisimulation (?p2, ?q2)).
```

```
axiom strongBisimulation2 definedBy
  !- strongBisimulation(?p, ?q) and
  transitions (?q, ?l, ?q2) and
  naf ( transitions (?p, ?l, ?p2) and
       strongBisimulation (?p2, ?q2)).
```

```
relation weakBisimulation (ofType ProcessTerm, ofType ProcessTerm)
```

```
axiom weakBisimulation1 definedBy
  !- weakBisimulation(?p, ?q) and
  weakTransitions (?p, ?l, ?p2) and
  naf ((weakTransitions (?q, ?l, ?q2) and
       weakBisimulation (?p2, ?q2)) or
       weakBisimulation (?p2, ?q)).
```

```
axiom weakBisimulation2 definedBy
  !- weakBisimulation(?p, ?q) and
  weakTransitions (?q, ?l, ?q2) and
  naf ((weakTransitions (?p, ?l, ?p2) and
       weakBisimulation (?p2, ?q2)) or
       weakBisimulation (?p, ?q2)).
```

Listing 4.12: Bisimulation relations

Note that although rules to derive these relations are necessarily second-order, by stating these as constraining axioms we allow a set of relation instances proving the relatedness under bisimulation of two processes to be checked by the SUPER process reasoner. By combining the revised BRO with BP MO2BRO, in the motivating context of process mediation, we could suggest that the process mediator concept from BP MO, reproduced with explicit namespacing in Listing 4.13, could be the subject (in an optional extension to BP MO) of the axiom shown in Listing 4.14.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

concept bpmo#ProcessMediator **subConceptOf** bpmo#Mediator
 bpmo#hasSourceProcess **ofType** (1) bpmo#Process
 bpmo#hasTargetProcess **ofType** (1 *) bpmo#Process
 bpmo#hasMediationProcess **ofType** (0 1) bpmo#MediationProcess
 bpmo#hasSWSMediator **ofType** (0 1) bpmo#SemanticCapability

Listing 4.13: BPMO Process Mediator

axiom provenProcessMediation **definedBy**
 !- ?med[bpmo#hasSourceProcess **hasValue** ?s,
 bpmo#hasMediationProcess **hasValue** ?m,
 bpmo#hasTargetProcess **hasValue** ?t] **memberOf** bpmo#ProcessMediator **and**
 bpmo2bro#behaviouralSemantics(?s, ?sp) **and**
 bpmo2bro#behaviouralSemantics(?m, ?mp) **and**
 bpmo2bro#behaviouralSemantics(?t, ?tp) **and**
 (naf bro#weakBisimilation(?sp, ?comp) **and**
 ?comp[bro#hasLeftSubterm **hasValue** ?mp,
 bro#hasRightSubterm **hasValue** ?tp] **memberOf** bro#Parallel).

Listing 4.14: Proof-carrying Process Mediation Axiom

Unfortunately such an axiom cannot be included in the stack at the time of writing since the need for integer indices for BRO concepts cannot be met by the WSML-Flight axioms that make up BPMO2BRO without a small reasoner extension discussed in the following section. But for this need, unmet only due to the delay in this deliverable due to the change, this profitable approach would have been pursued within the project rather than after, where it will be considered in the standardisation of BPMO in the STI Working Group also discussed in the following section.

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

5 Conclusions and Further Work

This deliverable has achieved two major objectives towards the aims of the SUPER project. First the use of ontology-based technologies to give behavioural semantics to the Business Process Modelling Ontology has been furthered. The approach of giving a state-transition system via ontology rules is to be considered for application in the OASIS Semantic SOA Reference Ontology¹, currently under public review towards standardisation. If accepted it could quite easily make SUPER BPMO the first process model that is directly compliant with this reference ontology. The inability to directly produce integer indices in translating from BPMO into the revised BRO will be addressed via a specific form of function symbol for which a reasoner extension can be produced to produce a hash index from.

Secondly the encoding of the constraints necessary for process relations to form bisimulation-based behavioural equivalences enables a form of checking to be effected by ontology-based reasoning. This has been illustrated in the context of process mediation, but which would apply equally well to behaviour-aware discovery. In this way any consumer of ontology-based process descriptions can be sure of the validity of results produced by such services and tools, even though actually finding the result is beyond the power of their reasoner. For this reason we compare the approach pioneered here with the field of *proof-carrying code* [8] wherein the proof of certain verification results, which require higher-order reasoning (in many cases with human intervention in an interactive theorem prover) to be attached to code, and automatically checked by any system where such code is to be executed.

Both of these advantages of the SUPER BPMO are to be pursued in the preparation of a post-project Business Process Modelling Ontology which has already been initiated as a deliverable within the STI Conceptual Models for Services Working Group².

¹<http://www.oasis-open.org/committees/semantic-ex/>

²<http://cms-wg.sti2.org/>

Project	SUPER	SUPER-Project-No	026850
	Business Process Mediation Reasoning Component	Work Package 4	
Document	Deliverable 4.3	Date	20.04.09

References

- [1] Barry Norton, Gerald Lüttgen, and Michael Mendler. A compositional semantic theory for synchronous component-based design. In *14th International Conference on Concurrency Theory (CONCUR '03)*, number 2761 in LNCS. Springer-Verlag, 2003.
- [2] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.
- [3] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117(2):221–239, 1995.
- [4] Henrik Reif Anderson and Michael Mendler. An asynchronous process algebra with multiple clocks. In *Proceedings of 5th European Symposium on Programming Languages and Systems (ESOP'94)*, volume 788 of LNCS, pages 58–73. Springer, 1994.
- [5] Rance Cleaveland, Gerald Lüttgen, and Michael Mendler. An algebraic theory of multiple clocks. In *Proceedings of 8th International Conference on Concurrency Theory (CONCUR'97)*, volume 1102 of LNCS, pages 166–180. Springer, 1997.
- [6] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN-19, Computer Science Department, Århus University, Denmark, 1981.
- [7] A. J. R. G. Milner. A complete axiomatisation for observational congruence of finite-state behaviours. *Information and Computation*, 81(81):227–247, 1989.
- [8] George C. Necula and Peter Lee. Proof-carrying code. Technical Report CMU-CS-96-165, Carnegie Mellon University, 1996.