
# Reasoning with Inconsistent Ontologies: Evaluation

**Zhisheng Huang, Frank van Harmelen**
**(Vrije Universiteit Amsterdam)**

**Abstract.**
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D3.4.2(WP3.4)
PION is a framework for reasoning with inconsistent ontologies. Is is based on the concept of "selection functions" to obtain meaningful answers from a selection of the axioms of a given inconsistent ontology. In this document, we investigate several selection functions, test them with several large scale realistic ontologies, and report the evaluation of PION with those experiments.
Keyword list: ontology management, inconsistency, ontology reasoning

| | |
|---|---|
| **Document Id.** | SEKT/2005/D3.4.2/v0.9.0 |
| **Project** | SEKT EU-IST-2003-506826 |
| **Date** | Jan 20, 2006 |
| **Distribution** | unrestricted |

## SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

**British Telecommunications plc.**
Orion 5/12, Adastral Park
Ipswich IP5 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

**Jozef Stefan Institute**
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

**University of Sheffield**
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891, Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

**Intelligent Software Components S.A.**
Pedro de Valdivia, 10
28006 Madrid
Spain
Tel: +34 913 349 797, Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

**Ontoprise GmbH**
Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912, Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

**Vrije Universiteit Amsterdam (VUA)**
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

**Empolis GmbH**
Europaallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540, Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

**University of Karlsruhe**, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592, Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

**University of Innsbruck**
Institute of Computer Science
Techikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475, Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

**Kea-pro GmbH**
Tal
6464 Springen
Switzerland
Tel: +41 41 879 00, Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

**Sirma AI EAD, Ontotext Lab**
135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

**Universitat Autonoma de Barcelona**
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vallès)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

# Executive Summary

PION is a framework for reasoning with inconsistent ontologies, based on the concept selection functions, to obtain meaningful answers, given an inconsistent ontology.

In this document, we propose different selection functions, we test them with several realistic ontologies, and we report the evaluation of PION with those experiments.

The first selection function is simply looking at syntactic connections between axioms in order to decide which axioms are relevant to a query. This relevance requirement is decreased graudally during reasoning in order to obtain a in increasing subset of the axioms until it has selected a subset which is small enough to avoid the inconsistency, but large enough to answer the query.

A second selection function takes into account that we are dealing with ontologies, and uses the concept hierarchy to guide the selection of relevant axioms.

We apply this framework to two medium-sized ontoloies. These ontologies are a few hundred axioms each, are taken from external sources, and are “enriched” in order to make explicit the implicit inconsistencies from which they suffer.

A rather surprising result is that in particular the ontology-oriented selection function gives very good results: it finds very high percentages of correct answers, while managing to avoid returning incorrect answers.

The current document builds on the previous deliverable D3.4.1. The second chapter of the current document recapitulates some of the essential definitions of the earlier deliverable. The third chapter defines a new, more semantically informed selection function. The main new contribution of this deliverable is in the fourth chapter, where the behaviour of the previous and the new selection functions are benchmarked on some non-trivial ontologies.

# Contents

# Chapter 1

# Introduction

The Semantic Web is characterised by scalability, distribution, and multi-authorship. All these characteristics may introduce inconsistencies in the Semantic Web. Limiting language expressivity with respect to negation (like RDF and other languages that are based on negation as failure) can avoid inconsistencies to a certain extent. However, the expressivity of these languages is quite limited. In particular, OWL is already capable of expressing inconsistencies.

There are two main ways to deal with inconsistency. One is to diagnose and repair it when we encounter inconsistencies. In [10], Schlobach and Cornet propose a non-standard reasoning service for debugging inconsistent terminologies. This is a possible approach, if we are dealing with one ontology and we would like to improve this ontology. Another approach is to simply avoid the inconsistency and to apply a non-standard reasoning method to obtain meaningful answers. In this work, we will focus on the latter, which is more suitable for the setting in the web area. For example, in a typical Semantic Web setting, one would be importing ontologies from other sources, making it impossible to repair them, and the scale of the combined ontologies may be too large to make repair effective.

The classical entailment in logics is *explosive*: any formula is a logical consequence of a contradiction. Therefore, conclusions drawn from an inconsistent knowledge base by classical inference may be completely meaningless. In [5], a framework for reasoning with inconsistent ontologies has been proposed. The general task of an inconsistency reasoner is: given an inconsistent ontology, return *meaningful* answers to queries (see chapter 2 for our definition of meaningfulness). The main idea of PION is: given a selection function, we select some consistent subtheory from an inconsistent ontology. Then we apply standard reasoning on the selected subtheory to find meaningful answers. If a satisfying answer cannot be found, the relevance degree of the selection function is made less restrictive thereby extending the consistent subtheory for further reasoning.

In this document, we propose different selection functions, we test them with several large scale realistic ontologies, and we report the evaluation of PION with those experi-

ments.

This document is organised as follows: Chapter 2 overviews the framework of reasoning with inconsistent ontologies, and recalls the essential definitions of a previous deliverable [6]. Chapter 3 defines variants of the selection functions. Chapter 4.1 discusses the implementation issues of the variants of selection functions. Chapter 4 describes the tests on PION. Chapter 5 discusses further work and concludes the document.

# Chapter 2

# Basic definitions

In this chapter we will recap the basic definitions of our previous deliverable [6].

## 2.1 Desired properties of inconsistency reasoners

Of course the traditional definition of soundness (formula are only provable if they hold in all models) cannot be used for inconsistency reasoners. Instead, we propose a weaker definition, which captures the intuition that only a small part of the theory is affected by an inconsistency, while the remainder of it is correct. An inconsistency reasoner should be considered sound if the formulas that follow from an inconsistent theory follow from a consistent subtheory using classical reasoning:

**Definition 2.1.1 (Soundness)** *An inconsistency reasoning $\mid\!\approx$ is sound if the following condition holds:*

$$\Sigma \mid\!\approx \phi \Rightarrow (\exists \Sigma' \subseteq \Sigma)(\Sigma' \not\models \bot \textit{ and } \Sigma' \models \phi)$$

Even though an inconsistency reasoner can deal with inconsistent ontologies, it should itself be consistent about its own answers:

**Definition 2.1.2 (Self-consistency)** *An inconsistency reasoner is self-consistent* iff

$$\Sigma \mid\!\approx \phi \Rightarrow \Sigma \not\mid\!\approx \neg\phi$$

These two notions combined define the notions of a meaningful answer and a meaningful reasoner:

**Definition 2.1.3 (Meaningfulness)** *An answer given by an inconsistency reasoner is meaningful iff it is self-consistent and sound.*
*An inconsistency reasoner is said to be meaningful iff all of the answers are meaningful.*

## 2.2 Selection Functions

An inconsistency reasoner uses a selection function to determine which consistent subsets of an inconsistent ontology should be considered in its reasoning process. The general framework is independent of the particular choice of selection function. The selection function can either be based on a syntactic approach, like Chopra, Parikh, and Wassermann's syntactic relevance [3], or based on semantic relevance like for example in computational linguistics as in Wordnet [2].

Given an ontology (i.e., a formula set) $\Sigma$ and a query $\phi$, a selection function $s$ returns a subset of $\Sigma$ at step $k > 0$. Let $\mathbf{L}$ be the ontology language, which is denoted as a formula set. We have the general definition about selection functions as follows:

**Definition 2.2.1 (Selection Functions)** *A selection function $s$ is a mapping $s : \mathcal{P}(\mathbf{L}) \times \mathbf{L} \times N \rightarrow \mathcal{P}(\mathbf{L})$ such that $s(\Sigma, \phi, k) \subseteq \Sigma$.*

**Definition 2.2.2 (Monotonic Selection Functions)**
*A selection function $s$ is called* monotonic *if the subsets it selects monotonically increase or decrease, i.e., $s(\Sigma, \phi, k) \subseteq s(\Sigma, \phi, k + 1)$, or vice verse.*

For monotonically increasing selection functions, the initial set is either an empty set, i.e., $s(\Sigma, \phi, 0) = \emptyset$, or a fixed set $\Sigma_0$. For monotonically decreasing selection functions, usually the initial set $s(\Sigma, \phi, 0) = \Sigma$. The decreasing selection functions will remove some formulas from the inconsistent set step by step until they find a maximally consistent set.

Monotonically increasing selection functions have the advantage that they do not have to return *all* subsets for consideration at the same time. If a query $\Sigma \mid\!\approx \phi$ can be answered after considering some consistent subset of the ontology $\Sigma$ for some value of $k$, then other subsets (for higher values of $k$) don't have to be considered any more, because they will not change the answer of the inconsistency reasoner.

## 2.3 Rating the results

In order to rate the "correctness" of the answers of an inconsistency reasoner, we follow Marquis and Porquets work in [8], and use Belnaps four valued logic [1] to distinguish the following four epistemic status for the answers:

**Definition 2.3.1 (Epistemic status of answers)**
• *Over-determined:* $\Sigma \mid\!\approx \phi$ *and* $\Sigma \mid\!\approx \neg\phi$[1]
• *Accepted:* $\Sigma \mid\!\approx \phi$ *and* $\Sigma \not\mid\!\approx \neg\phi$

[1]Notice that self-consistent reasoners can never provide overdetermined answers

• *Rejected:* $\Sigma \not\mid\!\approx \phi$ *and* $\Sigma \mid\!\approx \neg\phi$
• *Undetermined:* $\Sigma \not\mid\!\approx \phi$ *and* $\Sigma \not\mid\!\approx \neg\phi$

For example, if $\Sigma = \{C \sqsubseteq C_1, C \sqsubseteq C_2\}$ and $\phi = C_1 \sqsubseteq C_2$ then the intuitive answer is undetermined since neither $\phi$ nor $\neg\phi$ are implied by $\Sigma$.

Notice that in contrast to other work on inconsistency reasoning, these states are *not* part of our reasoning framework, in other words: our reasoner $\mid\!\approx$ does not *return* any of these answers, it just returns a boolean two-valued answer. Instead, the above four-valued epistemic states are a language to speak *about* the (boolean) answers of $\mid\!\approx$. Thus, for a query $\Sigma \mid\!\approx \phi$, PION may answer "true", but the epistemic state of this answer is determined by the answer that PION gives on $\Sigma \mid\!\approx \neg\phi$. For evaluation purposes, we will report on epistemic states (e.g. figure 4.1).

## 2.4 Quality categories

The above four epistemic states are all defined in terms of $\mid\!\approx$ itself. We need further notions to compare the behaviour of $\mid\!\approx$ that of $\models$. As *golden standard*, we cannot use the classical semantics of $\mid\!\approx$, after all, if $\Sigma$ is inconsistent, $\Sigma \models \phi$ for any $\phi$, making it not a very useful golden standard. Instead, we resort to an informal notion of "intuitively correct answer". In particular when the inconsistencies in one part of $\Sigma$ are not "connected" (in some informal sense) other parts of the $\Sigma$, it is often quite clear what what the intuitive answer would be. Consider the following trivial example: $\Sigma = \{a, a \rightarrow b, c, \neg c\}$. Intuitively, this theory should imply $b$ (by modes ponens from $\{a, a \rightarrow b\}$), and should not imply $\neg b$, since its inference under $\models$ is only justified by the inconsistency $\{c, \neg c\}$ which is some "unconnected" to the facts about $a$ and $b$. We emphasise that this notion of "intuitive answer" is informal, and must be provided by human inspection. We use the following notions to capture the differences between an answer by an inconsistency reasoner and the intuitive answer:

**Definition 2.4.1 (IA, CIA, CA, RA)**

- **Intended Answer***: $\mid\!\approx$ agrees with the intuitive answer*
- **Counter-intuitive Answer***: $\mid\!\approx$ provides the opposite to the intuitive answer. Namely, the intuitive answer is 'accepted' whereas the $\mid\!\approx$ answer is 'rejected', or vice versa.*
- **Cautious Answer***: The intuitive answer is 'accepted' or 'rejected', but the $\mid\!\approx$ answer is 'undetermined'.*
- **Reckless Answer***: the $\mid\!\approx$ answer is 'accepted' or 'rejected' whereas the intuitive answer is 'undetermined'.*

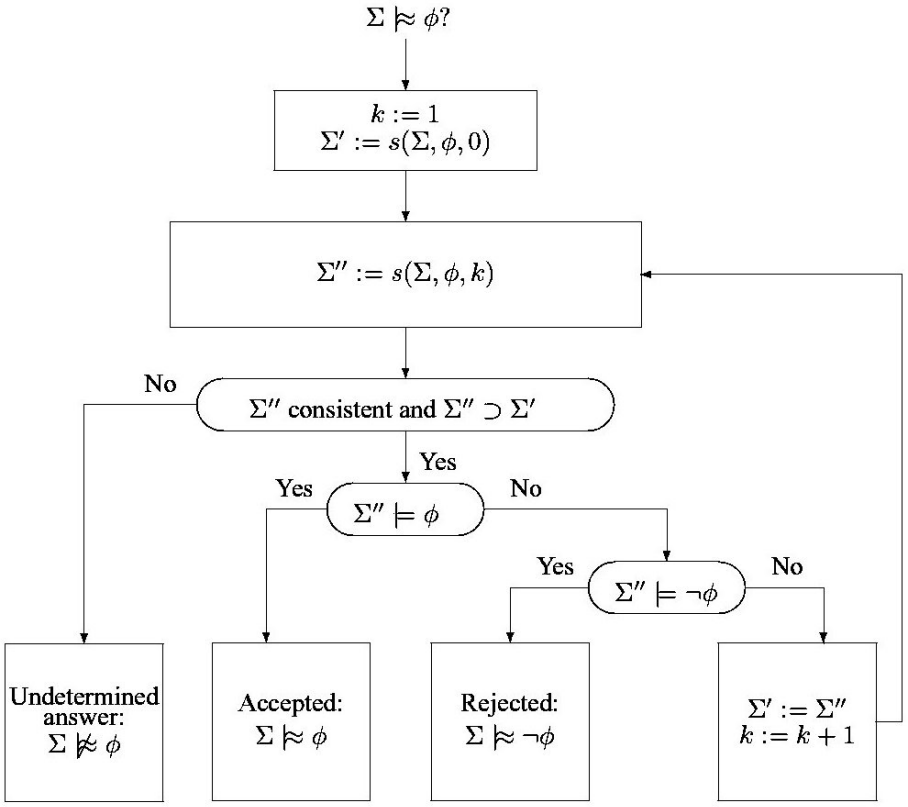


Figure 2.1: Linear Extension Strategy.

## 2.5 Extension Strategies

An inconsistency reasoner that uses a monotonically increasing/decreasing selection function will be called an inconsistency reasoner that uses a *linear extension strategy* and a *linear reduction strategy* respectively.

A linear extension strategy is carried out as shown in Figure 2.1, where $\mid\!\approx$ is used to denote the inconsistency-reasoner, and $\models$ denotes a classical reasoner. Given a query $\Sigma \mid\!\approx \phi$, the initial consistent subset $\Sigma'$ is set. Then the selection function is called to return a consistent subset $\Sigma''$, which extends $\Sigma'$, i.e., $\Sigma' \subset \Sigma'' \subseteq \Sigma$ for the linear extension strategy. If the selection function cannot find a consistent superset of $\Sigma'$, the inconsistency reasoner returns the answer *'undetermined'* (i.e., unknown) to the query. If the set $\Sigma''$ exists, a classical reasoner is used to check if $\Sigma'' \models \phi$ holds. If the answer is 'yes', the inconsistency reasoner returns the *'accepted'* answer $\Sigma \mid\!\approx \phi$. If the answer is 'no', the inconsistency reasoner further checks the negation of the query $\Sigma'' \models \neg\phi$. If the answer is 'yes', the inconsistency reasoner returns the *'rejected'* answer $\Sigma \mid\!\approx \neg\phi$, otherwise the current result is undetermined (def.1), and the whole process is repeated by calling the selection function for the next consistent subset of $\Sigma$ which extends $\Sigma''$.

One of the reasons for concentrating on a linear extension strategy in this work is that an inconsistency reasoner using a linear extension strategy is always meaningful. However, it is clear that the linear extension strategy may result in too many 'undetermined' answers to queries when the selection function picks the wrong sequence of monotonically increasing subsets. It would therefore be useful to measure the successfulness of (linear) extension strategies. Notice, that this depends on the choice of the monotonic selection function.

We call this strategy a *linear* one, because the selection function only follows one possible 'extension chain' for creating consistent subsets. The advantages of the linear strategy is that the reasoner can always focus on the current working set $\Sigma'$. The reasoner doesn't need to keep track of the extension chain. The disadvantage of the linear strategy is that it may lead to an inconsistency reasoner that is undetermined. There exists other strategies which can improve the linear extension approach, for example, by backtracking and heuristics evaluation. We are going to discuss a backtracking strategy in Section 3.1. The second reason why we call the strategy linear is that the computational complexity of the strategy is linear with respect to the complexity of the ontology reasoning. Let $n$ be the cardinality $|\Sigma|$ of an ontology $\Sigma$ and let the complexity of $\models$ be $E$.

**Proposition 2.5.1 (Complexity of Linear Extension)** *The complexity of $\mid\!\approx$ in the linear extension strategy is $n \cdot E$.*

In other words, the linear extension strategy does not significantly increase the complexity of the ontology reasoning, because typically $E$ is already *PSPACE-complete* for standard concept languages [4].

# Chapter 3

# Variants of Selection Functions

## 3.1 Syntactic Relevance

[3] proposes syntactic relevance to measure the relationship between two formulas in belief sets, so that the relevance can be used to guide the belief revision based on Schaerf and Cadoli's method of approximate reasoning. Given a formula set $\Sigma$, two atoms $p, q$ are directly relevant, denoted by $R(p, q, \Sigma)$ iff there is a formula $\alpha \in \Sigma$ such that $p, q$ appear in $\alpha$. A pair of atoms $p$ and $q$ are $k$-relevant with respect to $\Sigma$ iff there exist $p_1, p_2, \ldots, p_k \in \mathcal{L}$ such that: (a) $p, p_1$ are directly relevant; (b) $p_i, p_{i+1}$ are directly relevant, $i = 1, \ldots, k-1$; and (c) $p_k, q$ are directly relevant (i.e., directly relevant is $k$-relevant for $k = 0$).

The notions of relevance above are based on propositional logics. However, ontology languages are usually written in some fragment of first order logic. We extend the ideas of relevance to those first-order logic-based languages by restricting relevance to the co-occurrence of only the predicate letters or constant symbols. The following definition specialises the general definition of relevance for the case where $\phi$ is a formula in an ontology.

Given a formula $\phi$, we use $I(\phi), C(\phi), R(\phi)$ to denote the sets of individual names, concept names, and relation names that appear in the formula $\phi$ respectively.

**Definition 3.1.1 (Direct symbol-relevance)** *Two formula $\phi, \psi$ are directly relevant iff there is a common name which appears both in formula $\phi$ and formula $\psi$, i.e.,*
$I(\phi) \cap I(\psi) \neq \emptyset \vee C(\phi) \cap C(\psi) \neq \emptyset \vee R(\phi) \cap R(\psi) \neq \emptyset$.

**Definition 3.1.2 (Direct relevance to a set)** *A formula $\phi$ is relevant to a formula set $\Sigma$ iff there exists a formula $\psi \in \Sigma$ such that $\phi$ and $\psi$ are directly relevant.*

We can similarly specialise the notion of k-relevance.

**Definition 3.1.3 (k-relevance)** *Two formulas $\phi, \phi'$ are k-relevant with respect to a formula set $\Sigma$ iff there exist formulas $\psi_0, \ldots \psi_k \in \Sigma$ such that $\phi$ and $\psi_0$, $\psi_0$ and $\psi_1$, . . ., and $\psi_k$ and $\phi'$ are directly relevant.*

**Definition 3.1.4 (k-relevance to a set)** *A formula $\phi$ is k-relevant to a formula set $\Sigma$ iff there exists a formula $\psi \in \Sigma$ such that $\phi$ and $\psi$ are k-relevant with respect to $\Sigma$.*

The above definition of symbol-relevance is based on *any* syntactic overlap between formula (def. 3.1.1). These means that the set of $k$-relevant formula grows very rapidly, with the danger that it becomes inconsistent very rapidly. We can be more conservative in growing the set of $k$-relevant formula by restricting def. 3.1.1 to only the most important overlaps between formula. In this work we are not dealing with arbitrary logical theories $\Sigma$, but we assume that $\Sigma$ represents an ontology. Since the backbone of any ontology is the concept hierarchy, it makes sense to specialise symbol-relevance (i.e. any co-occurring symbol) to concept-relevance:

**Definition 3.1.5 (Direct concept-relevance)**
*An axiom $\phi$ is directly concept-relevant to a formula $\psi$, iff*

| | | |
|---|---|---|
| *(i)* | $C_1 \in C(\psi)$ | *if $\phi$ has the form $C_1 \sqsubseteq C_2$,* |
| *(ii)* | $C_1 \in C(\psi)$ *or* $C_2 \in C(\psi)$ | *if $\phi$ has the form $C_1 = C_2$,* |
| *(iii)* | $C_1 \in C(\psi)$ *or* $\cdots$ *or* $C_n \in C(\psi)$ | *if $\phi$ has the form $disjoint(C_1, \cdots, C_n)$.* |

Notice that clauses (ii) and (iii) of this definition simply amount to restricting the definition of symbol-relevance to co-occurring concept-symbols (instead of arbitrary symbols). The reason the asymmetry in clause (i) (only taking into account co-occurring symbols in the "small" end of a subsumption clause) is to direct the expansion function in the right direction of the subsumption hierarchy. In other words: here we are exploiting the fact that we are dealing with ontologies, and not just with arbitrary logical theories.

We can then define the notion of $k$-concept-relevance analogously as above in the obvious way.

## 3.2 Relevance-based Selection Functions

In inconsistency reasoning we can use syntactic relevance to define a selection function $s$ to extend the query '$\Sigma \mid\!\approx \phi?$' as follows: We start with the query formula $\phi$ as a starting point for the selection based on syntactic relevance. Namely, we define:

$$s(\Sigma, \phi, 0) = \emptyset.$$

Then the selection function selects the formulas $\psi \in \Sigma$ which are directly relevant to $\phi$ as a working set (i.e. $k = 1$) to see whether or not they are sufficient to give an answer to the query. Namely, we define:

$$s(\Sigma, \phi, 1) = \{\psi \in \Sigma \mid \phi \text{ and } \psi \text{ are directly relevant}\}.$$

If the reasoning process can obtain an answer to the query, it stops. Otherwise the selection function increases the relevance degree by 1, thereby adding more formulas that are relevant to the current working set. Namely, we have:

$$s(\Sigma, \phi, k) = \{\psi \in \Sigma \mid \psi \text{ is directly relevant to } s(\Sigma, \phi, k-1)\},$$

for $k > 1$.

This leads to a "fan out" behaviour of the selection function: the first selection is the set of all formulae that are directly relevant to the query; then all formulae are selected that are directly relevant to that set, etc. This intuition is formalised in the following:

**Proposition 3.2.1** *The syntactic relevance-based selection function $s$ is monotonically increasing.*

All of the above holds for either of the two relevance notions defined above (i.e. symbol-relevance or concept-relevance).

The syntactic relevance-based selection functions defined above usually grows up to an inconsistent set rapidly. That may lead to too many undetermined answers. In order to improve it, we can require that the selection function returns a consistent subset $\Sigma''$ at the step $k$ when $s(\Sigma, \phi, k)$ is inconsistent such that $s(\Sigma, \phi, k-1) \subset \Sigma'' \subset s(\Sigma, \phi, k)$. It is actually a kind of backtracking strategy which are used to reduce the number of undetermined answers to improve the linear extension strategy. We call the procedure an *over-determined processing*(ODP) of the selection function. Note that the over-determined processing does not need to exhaust the powerset of the set $s(\Sigma, \phi, k) - s(\Sigma, \phi, k-1)$, because of the fact that if a consistent set $S$ cannot prove or disprove a query, then nor can any subset of $S$. Therefore, one approach of ODP is to return just a maximally consistent subset. Let $n$ be $|\Sigma|$ and $k$ be $n - |S|$, i.e., the cardinality difference between the ontology $\Sigma$ and its maximal consistent subset $S$ (note that $k$ is usually very small), and let $C$ be the complexity of the consistency checking. The complexity of the over-determined processing is polynomial to the complexity of the consistency checking:

**Proposition 3.2.2 (Complexity of ODP)** *The complexity of over-determined processing is $n^k \cdot C$.*

Note that ODP introduces a degree of non-determinism: selecting different maximal consistent subsets of $s(\Sigma, \phi, k)$ may yield different answers to the query $\Sigma \mid\!\approx \phi$. The simplest example of this is $\Sigma = \{\phi, \neg\phi\}$.

# Chapter 4

# Test and Evaluation

## 4.1 Implementation and Prototype

In [6] we have reported on our prototype of an inconsistency reasoner (PION) implemented in SWI-Prolog.[1] PION implements an inconsistency reasoner based on an linear extension strategy and the syntactic relevance-based selection function as discussed in Sections 2.5 and 3.1. The selection function returns the first maximal consistent subset for its over-determined processing. PION is powered by XDIG, an extended DIG Description Logic interface for Prolog [7]. PION supports the TELL requests in DIG data format and in OWL, and the ASK requests in DIG data format. A prototype of PION is available for download at the website[2].

## 4.2 General Approach and Selection of Data

### 4.2.1 General Approach

The goal of our evaluation is to measure the behaviour of PION on ontologies that are realistic, both in size, expressivity, structural properties, etc. In our evaluation, we will

1. select realistic inconsistent ontologies
2. run a realistic set of queries on these ontologies using PION
3. tabulate how often PION gives answers of the different types defined in section 2.4: intended, counter-intuitive, cautious or reckless.

[1] <http://www.swi-prolog.org>
[2] <http://wasp.cs.vu.nl/sekt/pion>

4. tabulate performance characteristics of PION, both in terms of run-time and in terms of numbers of steps taken

### 4.2.2 Selection of Data

Concerning the requirement for *realistic* ontologies: Some aspects of ontologies can be measured reasonably well (number. of concepts, depth of hierarchy, expressivity of the language, etc). However, it is not at all a priori clear which properties of ontologies will affect the behaviour of PION. Therefore, we want to use *realistic* ontologies for our tests. In order to avoid experimental bias as far as possible, we want to use ontologies that have been constructed by third parties, and that are also used by third parties (preferably used by others than their authors).

Of course, we require ontologies that contain *inconsistencies*, since on consistent ontologies, $\not\approx$ coincides with $\models$, and PION simply reduces to a very inefficient implementation of a standard DL reasoner[3].

We can distinguish different processes by which ontologies can become inconsistent (see also SEKT deliverable D3.6.2 [12]):

- **migration:** inconsistencies may arise because of migrating an ontology to another formalism. The DICE ontology used in deliverable D3.6.2 [12] is an example of this.
- **clarification:** many current ontologies are expressed in RDF/RDF-Schema, which implies that at first sight they do not include inconsistencies. However, as we discuss in the next subsection, important implicit assumptions underly such ontologies. When such assumptions are made explicit through a mechanism of "clarification" (again: discussed in the next subsection), significant inconsistencies do show up.
- **merging:** even though two individual ontologies may each be consistent, their combination may well end up being inconsistent.

Of these three sources of inconsistency, the first one (migration) would not yield an appropriate benchmark. Ontology-reasoning as done by PION is meant to be deployed at *run-time* of an application system (e.g. for answering user-queries from an inconsistent ontologies), while migration is a task that is done *off-line* during the ontology-engineering stage (this dual structure is also reflected in the global SEKT architecture, with the "engineering" components and the "runtime" components: PION is intended as a runtime component, while migration is a process at engineering time.

[3]It would iteratively call a standard DL reasoner over increasingly large subsets of the ontology until a sufficiently large subset was obtained to answer the query. This kind of strategy is well known to be inefficient in the general case.

The third source (merging) would have indeed provided useful datasets for benchmarking PION. We briefly considered using the same benchmark as in SEKT deliverable D3.6.2 [12] (merging SUMO and CYC), but in the end refrained from these experiments because of worries over efficiency-limitations (many of the SUMO/CYC experiments in [12] suffered from time-outs because of the size of the ontologies).

Consequently, we have chosen to perform the PION benchmarking on inconsistencies caused by the clarification process. This will be discussed in the next section

### 4.2.3 Semantic Clarification

Already in earlier work, we have used with some succes a method called *semantic clarification*: RDF(S) ontologies are by definition free from inconsistencies, but closer inspection of such ontologies has shown that this is only due to the fact of the impoverished language: the underlying conceptualisation does actually contain inconsistencies, but these simply do not show up in the ontology because some aspects of the conceptualisation cannot be made explicit due to the impoverished language. In particular, disjointness assumptions between classes are often clear from the names given to these classes, but cannot be expressed explicitly in RDF(S).

In [9] we have shown that making such implicit assumptions explicit (at the cost of using a more expressive ontology language, particularly some fragment of OWL DL) does reveal important inconsistencies in many ontologies. This involves automatically adding disjointness statements to an ontology by assuming that all the direct siblings in a well-defined is-a hierarchy are disjoint. Most of these disjointness statements are indeed correct, and reflect the implicit modelling assumptions underlying the ontology. However, some of them turn out to be overspecified (between classes that are not actually disjoint), and they will cause the ontology to become inconsistent. [9] shows how these overspecified disjointness statements can be pinpointed using debugging techniques (see also SEKT deliverable D3.6.1 [11]). Here we will investigate how well we can reason in the presence of such overspecified disjointness statements.

Clearly, not all RDF(S) hierarchies will result in inconsistencies when disjointness statements are added. [9] shows that the well-known UNSPC product catalogue produces no inconsistencies (since its structure is simply too weak), while the Teknowledge Transport Ontology yields 150 inconsistencies after adding 89 disjointness axioms to 450 classes.

Furthermore, our choice of ontologies was limited by the computational power required to run many queries over them (see below). This precluded the choice of such interesting ontologies such as the well known SUMO top-level ontology, and the MILO mid-level ontology (intended to act as a bridge between the high-level abstractions of SUMO and domain ontologies ).

### 4.2.4 Description of selected data

Based on these considerations, we have decided on the use of the following ontologies:

- *Transportation Ontology*[4]: some 450 concepts in the transportation domain, describing different types of transportation connections (e.g. Railways, Highways, Waterways, Pipelines), different kinds of transportation vehicles (e.g. Land Vehicles, Water Vehicles, Air Vehicles), different Transportation Authorities and Regulations, Transportation Organisations and Transportation Personnel. The Ontology was constructed under US Government funding from sources in the military, the government and commerce: Universal Joint Task List[5]; Glossary of Landform and Geologic Terms[6]; Householders Goods Forwarders Association of America[7]; Information about government organisations[8]; sea and shipping terms[9]; and general transportation terms from Congressional Research Service (CRS) reports[10]

- Communication Ontology: some 200 concepts on communication technology, describing different types of radio, television and telephone systems as well as Internet technology.

- Enriched MadCow Ontology: The MadCow ontology is a small but well-known tutorial ontology explicitly designed to illustrate OWL DL expressivity. It displays a number of non-trivial inconsistencies after disjointness statements have been added. This ontology was not added because of realistic content or size, but because of its intricate use of OWL constructs.

### 4.2.5 Queries

For each of the selected ontologies, we ran a significant number of subsumption queries (encoded in a particular way, see below).

In principle, an ontology on $n$ concepts generates some $n^2$ potential subsumption queries. For the Transportation ontology, this would amount to some 20.000 queries. The main bottleneck in such a number of queries is not even the amount of CPU time required, but mainly the evaluation of the results. Remember that the notion of an "intended answer" is only defined by human inspection.

[4] <http://ontology.teknowledge.com/>
[5] <http://www.dtic.mil/doctrine/jel/cjcsd/cjcsm/m3500_4b.pdf>
[6] <http://www.statlab.iastate.edu/soils/nssh/629.htm>
[7] <http://www.hhgfaa.org/public/industryterms1.asp#B>
[8] <www.dot.gov>
[9] <http://www.trans-inst.org/seawords.htm>
[10] <http://www.ncseonline.org/NLE/CRS/>

| ontology | relevance | queries | IA | CA | RA | CIA | IA% |
|---|---|---|---|---|---|---|---|
| MadCow+ | symbol | 2594 | 2538 | 0 | 54 | 2 | 98% |
| | concept | 2594 | 2402 | 192 | 0 | 0 | 93% |
| Communication | symbol | 6576 | 6396 | 8 | 164 | 8 | 97% |
| | concept | 6576 | 6330 | 246 | 0 | 0 | 96% |
| Transport | symbol | 6258 | 5504 | 0 | 752 | 2 | 88% |
| | concept | 6258 | 6228 | 30 | 0 | 0 | 99% |

Table 4.1: Running PION on realistic ontologies

Given these prohibitive costs, we have decided to run a randomly generated subset of subsumption queries, on the assumption that this does not introduce any bias into the query-set.

For each concept $C$ in those ontologies, we create an instance $i_C$. We make both a positive instance query $i_C \in C'$ and a negative instance query $i_C \in \neg C'$ for some concepts $C'$ in the ontologies. Querying $i_C \in C'$ is equivalent to asking the subsumption of $C \sqsubseteq C'$ since $i_C$ is an arbitrary instance of $C$ without any further known properties or restrictions.

## 4.3 Logical Results

Table 4.1 shows the results of running PION the selected ontologies using the following abbreviations:

| | | |
|---|---|---|
| relevance | = | using selection function based on symbol-relevance or concept-relevance |
| queries | = | number of queries |
| IA | = | number of Intended Answers |
| CA | = | number of Cautious Answers |
| RA | = | number of Reckless Answers |
| CIA | = | number of Counter-Intuitive Answers |
| IA% | = | rate of Intended Answers: IA/queries |

All data underlying these results are available at <`http://wasp.cs.vu.nl/sekt/pion/test/`>.

A first observation is to notice what happens when switching from symbol relevance to concept relevance. The Intended Answers (IA) drop, which is a disadvantage, but the Counter-Intuitive answers (CIA) also drop, in fact to 0. Also, the Cautious Answers rise and the Reckless Answers drop (again to 0).

Thus, when precision is preferred over recall[11], concept-relevance is much better than

[11] Recall measuring the percentage of intended answers returned, and precision measuring the percentage

| Example | Queries | IA | CA | RA | CIA | IA Rate(%) | ICR Rate(%) |
|---|---|---|---|---|---|---|---|
| Bird | 50 | 50 | 0 | 0 | 0 | 100 | 100 |
| Brain | 42 | 36 | 4 | 2 | 0 | 85.7 | 100 |
| MarriedWoman | 50 | 48 | 0 | 2 | 0 | 96 | 100 |
| MadCow | 254 | 236 | 16 | 0 | 2 | 92.9 | 99 |

IA = Intended Answers, CA = Cautious Answers, RA = Reckless Answers, CIA = Counter-Intuitive Answers, IA Rate = Intended Answers(%), ICR Rate = IA+CA+RA(%).

Table 4.2: Running PION on artificial test-ontologies

general symbol-relevance. This is explainable because concept-relevance chooses much smaller sets than symbol-relevance, and suffers less from over-expanding the set of selected axioms to obtain an inconsistent subset.

Even though the tested strategy is very simple (a basic linear extension strategy, using an elementary syntactic relevance function), the resulting systems makes for a very high quality approximation, with $> 90\%$ recall and 100% precision on the above dataset.

However, we would like to point out that the high rate of the intended answers includes many 'undetermined' answers.

For completeness, we also repeat in table 4.2 the results of running PION with only the symbol-relevance function on some small test examples, as reported in [6]. Although not run on realistic examples, these figures are in line with what we observed in table 4.1

## 4.4 Efficiency Results

### 4.4.1 Runtime

Whereas the previous section reported on the logical correctness or otherwise of PION's answers, in this section we report on the performance characteristics.

Figure 4.1 shows the average runtime in seconds per query when using symbol-relevance and concept-relevance on the three ontologies. All the tests were done on a low-end PC, with 550Mhz CPU, 256 MB memory, running Windows 2000.

This shows that answering queries with concept relevance took much less time than with symbol relevance (roughly a factor $^1/_2 - {}^1/_3$ in all cases). This is all the more interesting because not only does the runtime go down when using concept relevance, we saw in the previous section that the quality of the answers goes up. This shows that the heuristic underlying concept relevance (using the class-hierarchy as the main guiding principle for the selection function) is indeed an improvement over looking for arbitrary

of non-counter-intuitive answers.

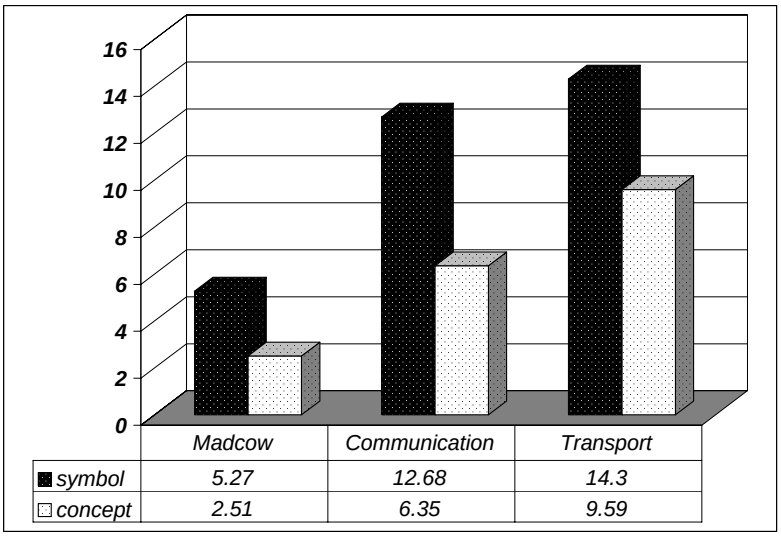


Figure 4.1: average **runtime in seconds** required per query when using symbol-relevance and concept-relevance on the three ontologies.

co-occurring symbols, as done in the original work Chopra et al. [3].

### 4.4.2 Number of steps

It is also interesting to see to what this reduction in runtime can be attributed. Although the reduction is about a factor of $^1/_2 - {}^1/_3$ in all cases, the underlying mechanisms seem to be very different.

Figure 4.2 shows the average number of expansion steps required per query before PION was able to give an answer. This is the value of the parameter $k$ in definition 3.1.5. Put differently, it is the number of iterations that must be made in the diagram of figure 2.1 before PION provides an answer.

Figure 4.3 shows the average number of backtracking steps per query for processing over-determined axiom sets. Remember that when the current set of considered axioms becomes itself inconsistent (in other words, it becomes overdetermined), PION backtracks to a maximally consistent subset. This is called over-determined processing, or ODP for short.

All these data are represented in a more compact way in table 4.3

The above shows that that the run-time gain observed in figure 4.1 is actually causes

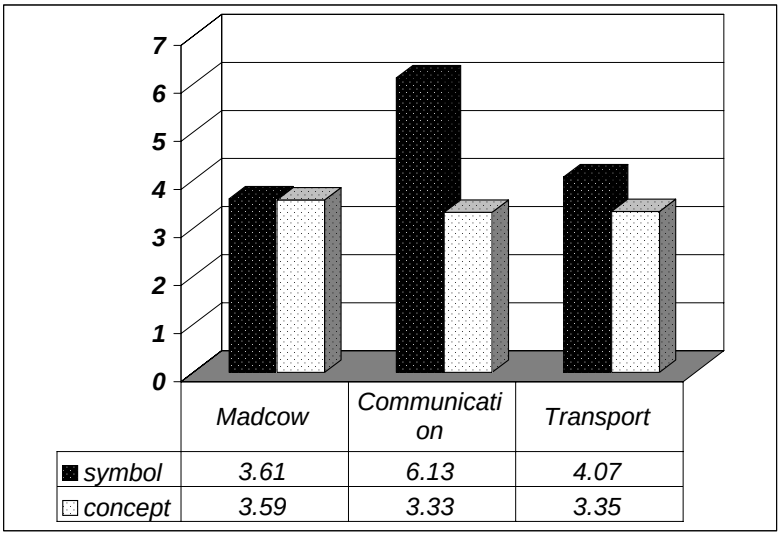


Figure 4.2: average number of **expansionsteps** required per query when using symbol -relevance and concept-relevance on the three ontologies.

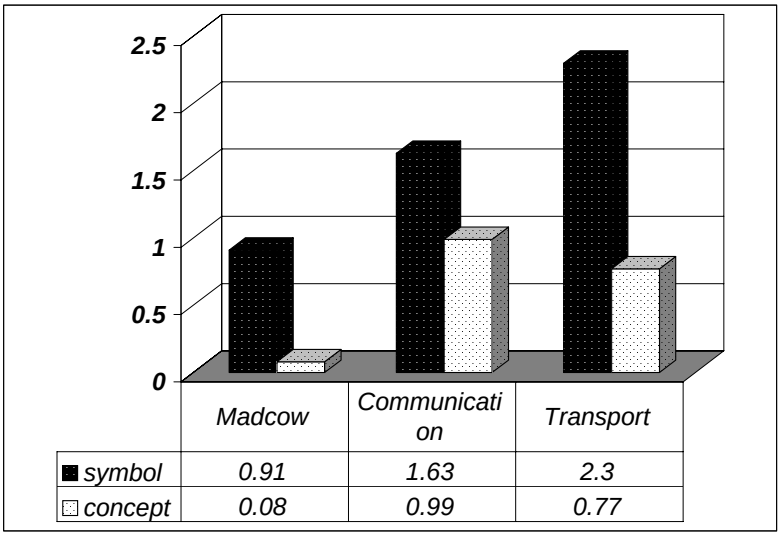


Figure 4.3: average number of **ODP steps** required per query when using symbol -relevance and concept-relevance on the three ontologies.

| ontology | relevance | Time (Sec.) | Expansion(steps) | ODP(steps) |
|---|---|---|---|---|
| MadCow+ | symbol | 5.27 | 3.61 | 0.91 |
| | concept | 2.51 | 3.59 | 0.08 |
| Communication | symbol | 12.68 | 6.13 | 1.63 |
| | concept | 6.35 | 3.33 | 0.99 |
| Transport | symbol | 14.30 | 4.07 | 2.30 |
| | concept | 9.59 | 3.35 | 0.77 |

Table 4.3: Run-time results and number of steps required for the benchmarks

by a variety of mechanisms. In the Communication and Transport ontologies, the gain is caused by a reduction in both the number of expansion steps (almost half), and the number of ODP steps. In the MadCow+ ontology on the other hand, the gain in runtime is entirely attributable to a reduction in the number of ODP steps (by a factor of 10).

Although not shown in these figures, the number of ODP steps has an interesting distribution over the different queries. Although the average number of required ODP steps is never very far from 1, the distribution is actually very uneven: For example when using concept-relevance on the Communication ontology, some 10-15% of all queries does not need any ODP processing at all (i.e. ODP=0); most cases (on the order of 80-90%) does need ODP backtracking, but PION finds a suitable consistent subset in a single step (ODP=1), and only a very small number of queries (on the order of 1%) needs multiple backtracking steps, some as many as 100 steps (although these cases are very rare, much below 1%). Similar results are obtained with concept-relevance on the Transport ontology: 95% of queries can be solved with 0 or 1 ODP step, and only 5% needs any significant backtracking.

Studying which properties of ontologies determine such behaviours is a point for future work.

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusion

In earlier deliverables, we have presented a framework for reasoning with inconsistent ontologies, based on the notion of selection functions.

We have also implemented a prototype of this framework called PION. In the current report, we have provided the evaluation report of the prototype by applying it to the several inconsistent ontology examples. The tests show that our approach can obtain intuitive results in most cases for reasoning with inconsistent ontologies. Considering the fact that standard reasoners always results in either meaningless answers or incoherence errors for queries on inconsistent ontologies, we can claim that PION can do much better, because it can provide a lot of intuitive, thus meaningful answers. This is a surprising result given the simplicity of our selection function.

## 5.2 Future Work

In future work, we are going to test PION with more large-scale ontology examples, i.e. essentially repeating the experiments above, and see if we can determine which properties of ontologies determine the success or failure of our approach on these ontologies. Obvious candidates are the terminologies that were used to benchmark the diagnosis of inconsistencies, in particular the SUMO/CYC combination.

We are also going to investigate different approaches for selection functions. In particular, we have an interest in non-uniform, semantically inspired selection functions using domain-specific background knowledge. Another option is to use weak background knowledge like co-occurrence of concept-names on the Web as the basis for a lightweight semantic selection function.

An essential limitation of this benchmarking effort is that it is a “laboratory” bench-

mark: it tells us how percentages of answers given of various types, it gives us insight in the run-time performance, it shows the impact of having different selection functions. However, it does *not* show us whether an actual user in an actual application scenario would have benefitted from the answers given by PION. For this we are currently studying the ontology constructed in the "BT case study". This ontology is used for question answering, but does in fact turn out to be inconsistent. This would appear to give opportunities for deploying PION in an application setting.

# Bibliography


[1] N. Belnap. *Modern Uses of Multiple-Valued Logic*, chapter A useful four-valued logic, pages 8–37. Reidel, Dordrecht, 1977.

[2] A. Budanitsky and G. Hirst. Semantic distance inwordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA, 2001.

[3] S. Chopra, R. Parikh, , and R. Wassermann. Approximate belief revision preliminary report. *Journal of IGPL*, 2000.

[4] F. Donini. *Description Logic Handbook*, chapter Complexity of reasoning, pages 96–136. OUP, 2003.

[5] Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'05*, 2005.

[6] Z. Huang, F. van Harmelen, A. ten Teije, P. Groot, and C. Visser. Reasoning with inconsistent ontologies: a general framework. Project Report D3.4.1, SEKT, 2004.

[7] Zhisheng Huang and Cees Visser. Extended dig description logic interface support for prolog. Deliverable D3.4.1.2, SEKT, 2004.

[8] P. Marquis and N. Porquet. Resource-bounded paraconsistent inference. *Annals of Mathematics and Artificial Intelligence*, 39:349–384, 2003.

[9] S. Schlobach. Semantic clarification by pinpointing. In *Proceedings of the second European Semantic Web conference*, LNCS. Springer Verlag, 2004.

[10] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the eighteenth International Joint Conference on Artificial Intelligence, IJCAI'03*. Morgan Kaufmann, 2003.

[11] S. Schlobach and Zh. Huang. Inconsistent ontology diagnosis: Framework and prototype. Project Report D3.6.1, SEKT, 2005.

[12] S. Schlobach, Zh. Huang, and R. Cornet. Inconsistent ontology diagnosis: Evaluation. Project Report D3.6.2, SEKT, 2006.