Extending and Computing the Concept Covering for the Semantic Web*

Tommaso Di Noia[1], Eugenio Di Sciascio[1], Francesco M. Donini[2]
[1]Politecnico di Bari, Via Re David, 200, I-70125, Bari, Italy
{t.dinoia,disciascio}@poliba.it
[2]Università della Tuscia, via San Carlo, 32, I-01100, Viterbo, Italy
donini@unitus.it

1 Introduction

The Semantic Web [15] initiative has begun, slowly yet steadily, to revolutionize the way information is provided on the Internet. The basic idea is to structure information with the aid of markup languages, based on the XML language, such as RDF and RDFS [14], DAML+OIL [7] and OWL [13]. These languages have been conceived to allow for representation of machine understandable, unambiguous, description of web content through the creation of domain ontologies, and aim at increasing openness and interoperability in the web environment.

Description Logics are a family of logic formalisms for knowledge representation that appear extremely suitable in the framework of semantic web reasoning. One of the three versions of OWL, namely OWL-DL, has been defined in view of DL reasoning and allows to express several DL constructs. Unfortunately, classical inference services offered by DLs, *i.e.*, subsumption and satisfiability check, can be insufficient to cope with novel issues that come out when semantic technologies are put to work. One of the most promising yet challenging application scenario of semantic-based approaches is the automated discovery and composition of complex web-services exposing annotated descriptions of their behavior.

In this paper we show that by exploiting a novel inference service in DL, Concept abduction [8], we can reformulate the classical set covering problem in terms of Concept Covering– keeping it tractable at least for the subset of OWL-DL we adopt and with an approximate covering– and propose it for automated discovery and composition of e-services exploiting inherent semantics of their descriptions.

The rest of the paper is structured as follows. In section 2 basic Description Logics syntax and semantics are introduced; then the $\mathcal{ALN}$Description

*Technical Report. n. 21/04/S, Serie Scientifica - Dipartimento di Elettrotecnica ed Elettronica - Politecnico di Bari.

Logic is described with its relations with OWL-DL and the novel non-standard Concept Abduction is briefly described. In section 3 the Concept Covering problem (CCoP) is investigated and redefined in terms of Concept Abduction. A tractable algorithm is then presented to solve a CCoP. Conclusion and future work conclude the paper.

2 Basic Description Logics

Description Logics (DL)[3, 10] are a family of logic formalisms whose basic syntax elements are *concept* names, *e.g.*, `computer`, `CPU`, `software`, and *role* names, such as `hasDevice`, `hasOS`. Intuitively, concepts stand for sets of objects, and roles link objects in different concepts. Formally, concepts are interpreted as subsets of a domain of interpretation Δ, and roles as binary relations (subsets of $\Delta \times \Delta$). Basic elements can be combined using *constructors* to form concept and role *expressions*, and each DL has its distinguished set of constructors. Every DL allows one to form a *conjunction* of concepts, usually denoted as $\sqcap$; some DL include also disjunction $\sqcup$ and complement $\neg$ to close concept expressions under boolean operations. Roles can be combined with concepts using *existential role quantification*, *e.g.*, $\texttt{computer} \sqcap \exists \texttt{hasDevice.monitor}$, which describes the set of computers with a monitor, and *universal role quantification*, *e.g.*, $\texttt{PDA} \sqcap \forall \texttt{hasOS.windows}$, which describes PDAs having exclusively a Windows Operating System on board. Other constructs may involve counting, as number restrictions: $\texttt{computer} \sqcap (\leq\ 1\ \texttt{hasOS})$ expresses computers with only one Operating System, and $\texttt{server} \sqcap (\geq\ 4\ \texttt{hasCPU})$ describes servers endowed with at least four CPUs. Many other constructs can be defined, increasing the expressive power of the DL, up to n-ary relations [4]. Concept expressions can be used in *inclusion assertions*, and *definitions*, which impose restrictions on possible interpretations according to the knowledge elicited for a given domain. For example, we could impose that monitors can be divided into CTR and LCD using the two inclusions: $\texttt{monitor} \sqsubseteq \texttt{LCDMonitor} \sqcup \texttt{CRTMonitor}$ and $\texttt{CRTMonitor} \sqsubseteq \neg\texttt{LCDMonitor}$. Or, that computers for a domestic use have only one operating system as $\texttt{homePC} \sqsubseteq (\leq\ 1\ \texttt{hasOS})$. Definitions are useful to give a meaningful name to particular combinations, as in $\texttt{server} \equiv \texttt{computer} \sqcap (\geq\ 2\ \texttt{hasCPU})$. Historically, sets of such inclusions are called TBox (Terminological Box). The basic reasoning problems for concepts in a DL are satisfiability, which accounts for the internal coherency of the description of a concept (no contradictory properties are present), and subsumption, which accounts for the more general/more specific relation among concepts, that forms the basis of a taxonomy. More formally, a concept C is satisfiable if there exists an interpretation in which C is mapped into a nonempty set unsatisfiable otherwise. If a TBox $\mathcal{T}$ is present, satisfiability is relative to the models of $\mathcal{T}$, that is, the interpretation assigning C to a nonempty set must be a model of the inclusions in $\mathcal{T}$. A concept C subsumes a concept D if every interpretation assigns to C a subset of the set assigned to D. Also subsumption is usually established relative to a TBox, a relation that we denote $\mathcal{T} \models C \sqsubseteq D$. Also a TBox can be said satisfiable if

there exist at least one model (i.e., an interpretation fulfilling all its inclusions in a nontrivial way).
It is easy to see that $\mathcal{T}$ in DLs represents what is called ontology in a knowledge representation system. In the rest of the paper we refer to the $\mathcal{ALN}$(**A**ttributive **L**anguage with unqualified **N**umber restrictions) Description Logic, explained in the next subsection.

2.1 $\mathcal{ALN}$ Description Logic and OWL

Constructs allowed in an $\mathcal{ALN}$ DL are:

- A *atomic concepts.* All the objects belonging to the set represented by A
- $\top$ *universal concept.* All the objects in the domain.
- $\bot$ *bottom concept.* The empty set.
- $\neg A$ *atomic negation.* All the objects not belonging to the set represented by A.
- $C \sqcap D$ *intersection.* The objects belonging both to C and D.
- $\forall R.C$ *universal restriction.* All the objects participating to the R relation whose range are all the objects belonging to C.
- $\exists R$ *unqualified existential restriction.* There exists at least one object participating in the relation R[1].
- $(\geq\ n\ R)|(\leq\ n\ R)|(=\ n\ R)$[2] *unqualified number restrictions.* Respectively the minimum, the maximum and the exact number of objects participating in the relation R.

Here we use a $simple - TBox$ in order to express the relations among objects in the domain. With a $simple - TBox$ in all the axioms (for both inclusion and definition) the left side is represented by a concept name.

Ontologies using the above logic can be easily modeled using languages for the Semantic Web[7, 12, 13]. These languages have been conceived to allow for representation of machine understandable, unambiguous, description of web content through the creation of domain ontologies, and aim at increasing openness and interoperability in the web environment. The strong relations between Description Logics and the above introduced languages for the Semantic Web [2] is also evident in the definition of the OWL language. In fact there are three different sub-languages for OWL:

OWL-Lite. It allows class hierarchy and simple constraints on relation between classes.

[1]Notice that $\exists R$ is equivalent to $(\geq\ 1\ R)$
[2]We write $(=\ n\ R)$ for $(\geq\ n\ R) \sqcap (\leq\ n\ R)$

OWL syntax	**DL syntax**
$< owl : Thing/ >$	$\top$
$< owl : Nothing/ >$	$\bot$
$< owl : Classrdf : ID = "C"/ >$	C
$< owl : ObjectPropertyrdf : ID = "R"/ >$	R
$< rdfs : subClassOf/ >$	$\sqsubseteq$
$< owl : equivalentClass/ >$	$\equiv$
$< owl : disjointWith/ >$	$\neg$
$< owl : intersectionOf/ >$	$\sqcap$
$< owl : allValuesFrom/ >$	$\forall$
$< owl : someValuesFrom/ >$	$\exists$
$< owl : maxCardinality/ >$	$\leq$
$< owl : minCardinality/ >$	$\geq$
$< owl : cardinality/ >$	$=$

Table 1: Correspondence between OWL and DL syntax

OWL-DL. Based on Description Logics theoretical studies, it allows a great expressiveness keeping computational completeness and decidability.

OWL-Full. Using such a language, there is a huge syntactic flexibility and expressiveness. This freedom is paid in terms of no computational guarantee.

The subset of OWL-DL TAGs allowing to express an $\mathcal{ALN}$ description logic is presented in Table 1.

Notice that as in $\mathcal{ALN}$ only *unqualified existential restriction* is allowed, the restriction on the *ObjectProperty* must be $< owl : Thing/ >$.

As an example we present a translation of an inclusion axiom in DL to OWL.

$$\texttt{onSalePC} \sqsubseteq \texttt{homePC} \sqcap (\leq\ 1\ \texttt{hasOS}) \sqcap \forall\texttt{hasOS.winX}$$

```
< owl : Classrdf : ID = "onSalePC"/ >
    < rdfs : subClassOf >
        < owl : intersectionOfrdf : parseType = "Collection" >
            < owl : Classrdf : ID = "homePC"/ >
            < owl : Restriction >
                < owl : onPropertyrdf : resource = "hasOS"/ >
                < owl : maxCardinalityrdf : datatype = "&xsd; nonNegativeInteger" >
                    1
                < /owl : maxCardinality >
                < owl : allValuesFromrdf : resource = "winX"/ >
            < /owl : Restriction >
        < /owl : intersectionOf >
    < /rdfs : subClassOf >
< /owl : Class >
```

In the rest of the paper we will use DL syntax instead of OWL-DL syntax, because the former is more compact.

2.2 Concept Abduction in Description Logics

In [8] the Concept Abduction Problem (CAP) was introduced and defined as a non standard inference problem for Description Logics.

Definition 1 *Let C, D, be two concepts in a Description Logic $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms, where both C and D are satisfiable in $\mathcal{T}$. A* Concept Abduction Problem *(CAP), denoted as $\langle \mathcal{L}, C, D, \mathcal{T} \rangle$, is finding a concept H such that $\mathcal{T} \not\models C \sqcap H \equiv \bot$, and $\mathcal{T} \models C \sqcap H \sqsubseteq D$.*

We use $\mathcal{P}$ as a symbol for a CAP, and we denote with $SOL(\mathcal{P})$ the set of all solutions to a CAP $\mathcal{P}$.
In [8] also minimality criteria for H and a polynomial algorithm to find solutions which are irreducible, for the $\mathcal{ALN}$ DL, have been proposed.

Given a CAP, if H is a conjunction of concepts and no sub-conjunction of concepts in H is a solution to the CAP, then H **is an irreducible solution**. In [9] the *rankPotential* algorithm computing a value representing the length of H is presented.

The solution to a CAP can be read as *what have I to hypothesize in C, and in a second step add to, in order to make C more specific than D?* In other words H is *what is expressed, explicitly or implicitly, in D and is not present in C*, or again *which part of D is not covered by C*. On the basis of the latter remark in the following we will see how to use concept abduction to perform a "concept covering".

A numerical version of the algorithm also exists. In [9] the *rankPotential* algorithm is presented, which computes the length of H solution of the CAP $\langle \mathcal{L}, C, D, \mathcal{T} \rangle$. It will be used in the algorithm presented in the following.

3 Concept Covering via Concept Abduction

In [11] the *best covering problem* in Description Logics was introduced as "...a new instance of the problem of rewriting concepts using terminologies".

That is, given a concept C and a set of concept definitions in a terminology $\mathcal{T}$, find concepts defined in $\mathcal{T}$such that their conjunction can be an approximation of C.

In order to define a concept covering two non standard inferences were then used: the least common subsumer (*lcs*)[1] and the *difference* or *subtraction* operation [16]. Unfortunately, as the authors admitted, the difference operator makes sense only for a small set of DLs, and surely not for the $\mathcal{ALN}$ (we do not delve into details, for a complete description see [16]).

In a more formal way the authors of [11] defined a *cover* as follows.

Definition 2 *Let $\mathcal{L}$ be a Description Logic with structural subsumption, $\mathcal{T}$ be a terminology using operator allowed by $\mathcal{L}$, $S_{\mathcal{T}}$ be the set of concept definitions in $\mathcal{T}$, $\mathcal{R} = \{S_i, i \in [1..n]\}$, and D be a concept in $\mathcal{L}$ such that $\mathcal{T} \not\models D \equiv \bot$. A* cover *of a D using $\mathcal{T}$is finding a set $\mathcal{R}_c \subseteq \mathcal{R}$ such that $\sqcap S_i)$, conjunction of all the $S_i \in \mathcal{R}_c$ is such that $D - lcs_{\mathcal{T}}(D, \sqcap S_i) \not\equiv D$.*

That is, a cover is finding a set of concepts defined in $\mathcal{T}$such that they contain the information in D. Notice that a DL with structural subsumption is needed in order to use *concept difference*. In [11] also an hypergraphs based methodology is presented to compute best covers.

We now extend the previous definition, in terms of a Concept Covering Problem, both eliminating limitations on $\mathcal{L}$ to be used and rewriting it in terms of Concept Abduction.

Definition 3 *Let D be a concept, $\mathcal{R} = \{S_1, S_2, ..., S_k\}$ be a set of concepts, and $\mathcal{T}$ be a set of axioms, all in a DL $\mathcal{L}$, where D and $S_1, \ldots, S_k$ are satisfiable in $\mathcal{T}$. The* Concept Covering Problem *(CCoP) for $\mathcal{V}$=$\langle\mathcal{L}, \mathcal{R}, D, \mathcal{T}\rangle$ is finding a pair $\langle\mathcal{R}_c, H\rangle$ such that*

1. *$\mathcal{R}_c \subseteq \mathcal{R}$, and the conjunction of concepts in $\mathcal{R}_c$, $C = \sqcap_{S\in\mathcal{R}_c} S$ is satisfiable in $\mathcal{T}$;*

2. *$H \in SOL(\langle\mathcal{L}, C, D, \mathcal{T}\rangle)$, and $\mathcal{T} \not\models H \sqsubseteq D$.*

We call $\langle\mathcal{R}_c, H\rangle$ a solution *for $\mathcal{V}$, and say that $\mathcal{R}_c$* (partially) covers *D. Finally, we denote $SOLCCoP(\mathcal{V})$ the set of all solutions to a CCoP $\mathcal{V}$.*

Intuitively, $\mathcal{R}_c$ is the set of concepts that partially cover D w.r.t. $\mathcal{T}$, while the abduced concept H covers what is still in D and is not covered by C. There can be several solutions for a single CCoP, depending also on the strategy adopted for choosing concepts in $\mathcal{R}_c$. However, observe that – differently from the standard Set Covering Problem – a complete cover may not exist. Hence, minimizing the cardinality of $\mathcal{R}_c$ is not the aim of a CCoP; the aim is maximizing the covering, hence minimizing H. This argument leads us to the definition of *best cover* and *full cover*.

Definition 4 *A* best cover *for a CCoP $\mathcal{V}$, w.r.t. an order $\prec$ on $\mathcal{L}$, is a solution $\langle\mathcal{R}_c, H_b\rangle$ for $\mathcal{V}$ such that there is no other solution $\langle\mathcal{R}'_c, H'\rangle$ for $\mathcal{V}$ with $H' \prec H_b$.*

There is no solution $\langle\mathcal{R}'_c, H'\rangle$ for $\mathcal{V}$ such that H', the remaining part of D yet to be covered, is *smaller* than H_b. Here, a concept H' is smaller than H with respect to a given order $\prec$. That is, if you can define $\prec$ and $\prec'$ as two orders on a DL concept, it may happen that a best cover w.r.t. $\prec$ is not a best cover w.r.t. $\prec'$ or vice versa. Observe that, since the solution for a concept abduction is not unique in general, there could be two solutions $\langle\mathcal{R}_c, H_1\rangle, \langle\mathcal{R}_c, H_2\rangle$ such that the first is a best cover, while the second one is not. However, when a full cover exists, it is independent of any order $\prec$ on $\mathcal{L}$.

Definition 5 *A* full cover *for a CCoP $\mathcal{V}$ is a solution $\langle\mathcal{R}_c, H_f\rangle$ for $\mathcal{V}$ such that $\mathcal{T} \models H_e \equiv \top$.*

Having a set $\mathcal{R}$ of concepts $S_i, i = 1..k$, we want to find a subset $\mathcal{R}_c$ of $\mathcal{R}$, if exists, such that the conjunction of all the concepts in $\mathcal{R}_c$ is more specific than, *i.e.*, is subsumed by, D. In other words, we are looking for a set of concepts which completely cover D.

3.1 An Algorithm to Solve a CCoP

Based on the GREEDY-SET-COVER presented in [6], we now present a tractable algorithm, $GREEDYsolveCCoP$ to compute a solution to a CCoP. In the following $GREEDYsolveCCoP$ we use $H = solveCAP(\langle \mathcal{L}, C, D, \mathcal{T} \rangle)$ to state that H is a solution for the CAP $\langle \mathcal{L}, C, D, \mathcal{T} \rangle$.

In [6] is also proved that, for a set covering problem, the solution grows logarithmically in the size of the set to be covered with respect to the minimal one.

```
**Algorithm** $GREEDYsolveCCoP(\mathcal{S}, D, \mathcal{T})$
**input** concepts $D$, $S_i \in \mathcal{R}, i = 1..k$, where $D$ and
$S_i$ are satisfiable in $\mathcal{T}$
**output** $\langle \mathcal{R}_c, H \rangle$
**begin algorithm**
  $\mathcal{R}_c = \emptyset$;
  $D_{uncovered} = D$;
  $H_{min} = D$;
  **do**
    $S_{min} = \top$;
    /* [♣] *Perform a greedy search among* $S_i \in \mathcal{R}$ */
    **for each** $S_i \in \mathcal{R}$
      **if** $\mathcal{R}_c \cup \{S_i\}$ *is a cover* for $D_{uncovered}$ **then**
        $H = solveCAP(\langle \mathcal{L}, S_i, D_{uncovered}, \mathcal{T} \rangle)$;
        /* [♢] *Choose* $S_i$ *based on an order* */
        **if** $H \prec H_{min}$ **then**
          $S_{min} = S_i$;
          $H_{min} = H$;
        **end if**
      **end if**
    **end for each**
    /* [♠] *If a new* $S_i$ *is found then add* $S_i$ to $\mathcal{R}_c$ *and remove it from* $\mathcal{R}$ */
    **if** $S_{min} \not\equiv \top$ **then**
      $\mathcal{R} = \mathcal{R} \backslash \{S_i\}$;
      $\mathcal{R}_c = \mathcal{R}_c \cup \{S_i\}$;
      $D_{uncovered} = H_{min}$;
    **end if**
    /* [♡] *Continue searching until no* $S_i$ *is found* */
  **while**$(S_{min} \not\equiv \top)$;
  **return** $\langle \mathcal{R}_c, D_{uncovered} \rangle$;
**end algorithm**
```

The algorithm tries to cover D *as much as possible*, using the concepts $S_i \in \mathcal{R}$.

♡ If it is not found any new useful $S_i \in \mathcal{R}$, that is any S_i such that it covers D more, then the algorithm terminates.

♣ A greedy approach is used to choose the *candidates* for $\mathcal{R}_c$.

♢ Among the candidates it is chosen the one such that H, solution for the local CAP, is minimal w.r.t. an order $\prec$.

♠ If the greedy search returns a new S_i, it is removed from $\mathcal{R}$ and added to $\mathcal{R}_c$.

The above algorithm itself is polynomial in time and the complexity source is in the solution of the CAPs and the comparison in [♢]. When the adopted DL in such that satisfiability and subsumption can be decided in polynomial time, a CCoP can be solved in nondeterministic polynomial time by guessing at the same time a subset $\mathcal{R}_c$ of $\mathcal{R}$, and a concept H, and then checking in polynomial time that $C \sqcap H$ is satisfiable in $\mathcal{T}$(hence C is satisfiable too), that $\mathcal{T} \models C \sqcap H \sqsubseteq D$, and that $\mathcal{T} \not\models H \sqsubseteq D$. On the other hand, the NP-complete problem EXACT-SET-COVER could be easily reduced to our Full Cover Problem, while Best Cover Problem is already NP-hard when $H_1 \prec H_2$ is defined as $|H_1| < |H_2|$ [5]. Therefore, deciding whether there exists a best/full solution for a CCoP is an NP-complete problem when $\mathcal{L}$=$\mathcal{ALN}$. For the $\mathcal{ALN}$ DL, in [8] a polynomial algorithm (*findIrred*) is proposed to find irreducible solutions for a CAP, and in [9] the tractable *rankPotential* is presented to rank concepts. Using such algorithms, *GREEDYsolveCCoP* can be solved in polynomial time.

4 Conclusion and Future Work

Motivated by the need of new reasoning services to bring the Semantic Web initiative to its full potential, in this paper we have set up an extension, based on Concept Abduction, to the definition of Concept Covering for Description Logics and proposed an algorithm to perform such service for an $\mathcal{ALN}$ DL, a subset of OWL-DL. Based on the above mentioned extension best cover and exact cover are also defined.

Based on *findIrred*, an existing algorithm for Concept Abduction presented in [8] and *rankPotential* presented in [9], a prototype system implementing *GREEDYsolveCCoP* has been developed within the MAMAS matchmaker service[3]. Further optimizations for *GREEDYsolveCCoP* are under investigation.

References

[1] Franz Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI2003)*, pages 319–324, 2003.

[2] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In Dieter Hutter and Werner Stephan,

[3]MAMAS is available in beta-version at `http://193.204.59.227:8080/MAMAS-devel/`

editors, *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003.

[3] A. Borgida. Description Logics in Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the Decidability of Query Containment under Constraints. In *Proceedings of the Seventeenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[5] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Uniform Tableaux-Based Approach to Concept Abductiona and Contraction in ALN DL. In *Proceedings of the 17th International Workshop on Description Logics (DL'04)*, 2004. To appear.

[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The Massachusetts Institute of Technology, 1990.

[7] DAML+OIL Specifications. www.daml.org/2001/03/daml+oil-index.html, 2001.

[8] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Abductive matchmaking using description logics. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 337–342, Acapulco, Messico, August 9–15 2003. Morgan Kaufmann, Los Altos.

[9] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. A system for principled Matchmaking in an electronic marketplace. In *Proc. International World Wide Web Conference (WWW '03)*, pages 321–330, Budapest, Hungary, May 20–24 2003. ACM, New York.

[10] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in Description Logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[11] Mohand-Sad Hacid, Alain Leger, Christophe Rey, and Farouk Toumani. Computing Concept Covers: a Preliminary Report. In *Proceedings of the 15th International Workshop on Description Logics (DL'02)*, volume 53 of *CEUR Workshop Proceedings*, 2002.

[12] D.L. McGuinness, R. Fikes, J. Hendler, and L.A. Stein. DAML+OIL: An Ontology Language for the Semantic Web . *IEEE Intelligent Systems*, 17(5):72–80, 2002.

[13] OWL. www.w3.org/TR/owl-features/.

[14] Resource Description Framework. http://www.w3.org/RDF/.

[15] Semantic Web. http://www.w3.org/2001/sw/.

[16] G. Teege. Making the difference: A subtraction operation for description logics. In *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 540–550. MK, 1994.