# Generating Random Benchmarks for Description Logics


**Quentin Elhaik(*), Marie-Christine Rousset(*), Bernard Ycart(**)**

(*) L.R.I., URA C.N.R.S , University of Paris-Sud
Building 490, 91405, Orsay Cedex, France
{quentin, mcr}@lri.lri.fr
(**) LMC/IMAG, University of Grenoble
BP 53, 38041 Grenoble Cedex 9, France
Bernard.Ycart@imag.fr


## 1 Introduction

The computational complexity of Description Logics (DL) reasoning problems has been extensively studied (see e.g. [6] for a recent survey). Many results on the worst-case time/space complexity for the corresponding inference algorithms are now available. Those results show that, under worst-case conditions, testing subsumption, classifying a Tbox, checking satisfiability of a knowledge base (KB) composed of a Tbox and an Abox, or answering queries over such a KB require runtime resources that grow exponentially or even worse with the size of the KB, for any reasonably expressive DL. However, despite their theoretical intractability, DL systems have been used for real applications. This suggests that worst-case examples are not realistic. Very few attempts have been done to test runtime performance of implementations of DL algorithms on real or artificial KB's. The comparative analysis of six DL systems described in [8] outlines the difficulty of obtaining real KB's.

In this paper, we address the problem of generating *random benchmarks* for DL systems. The advantage of generating random benchmarks is twofold. First, it is a way to provide a great number of artificial knowledge bases whose structure and distribution can be tuned in order to possibly account for some known real-world distributions. Second, it makes it possible to perform average-case complexity analysis for DL algorithms, also called *probabilistic analysis* [7]. In particular, it is an open question in the DL community how the inference algorithms behave *on average*. The recent insights that have been gained into the understanding of the SAT problem were due to the performance analysis of random instances of the problem [9].

Average-case analysis requires precise models of the distribution of input instances. Ideally, those models should reflect real-world distributions. However, since we usually do not have a good understanding of real-world distributions, average-case analysis has to rely on *random samples* of inputs, which are generated according to some *probability distribution*. When the set of possible inputs is finite (e.g. Aboxes with $k$ objects relative to a fixed Tbox), it is natural to consider the *uniform distribution* on that set, for which all elements of the set are equally likely. If the set of possible inputs is infinite, or if the probability distribution has to be adjusted to fit real data, then the uniform distribution cannot be used anymore, and the probability of each element of the set will depend on some parameters, to be arbitrarily fixed, or statistically estimated from real data. This will be the case in the generation of random Tboxes. Given a set $\mathcal{A}$ of inputs and a probability distribution $(P(A))_{A\in\mathcal{A}}$ on that set, a random sample of size $n$ is a $n-$tuple of random elements of $\mathcal{A}$, each distributed according to the given probability distribution, and *independent* from the others. In other words, it is an element of the cartesian product $\mathcal{A}^n$, distributed according to the product probability distribution, for which the probability of the $n-$tuple $(A_1,\ldots,A_n)$ is $P(A_1)\times\ldots\times P(A_n)$.

Generating uniform random samples is easy when the problem instances have a regular structure whose basic building blocks are not interrelated. It is the case for the SAT problem: the instances of SAT are clauses made of literals which are either propositional variables or their negations. In the so-called *fixed-clause-length* model [9], each clause is generated by randomly selecting $k$ variables among $n$, each of which is negated with probability 0.5. This test model has been adapted to the setting of propositional modal logic and its equivalent DL $\mathcal{ALC}$ ([4; 5], and also [1]) in order to capture hard instances for satisfiability checking. As will be seen in section 2, our goal is to generate random Tboxes in a general and flexible way enabling the samples that are produced to be tuned to possibly adjust a given real distribution.

The problem of generating uniform random samples of Aboxes relative to a fixed Tbox is much more complicated. The reason is that Aboxes are made of facts that may logically depend on each other, because of the definitions of the concepts and roles in the associated Tbox. In section 3, we shall introduce a Markov Chain Monte-Carlo (MCMC) algorithm that generates uniform random samples of Aboxes. Due to their growing number of applications, MCMC methods have received a lot of attention in the past ten years (see Fishman [3] for a general reference). In particular, it has been shown (cf. Sinclair [10]) that some NP-complete problems can be approximately solved in polynomial time by such a method. If a random sample of a probability distribution (uniform or not) is to be generated, the idea is to express that distribution as the equilibrium measure of a Markov chain. A Markov chain is a sequence of random variables, generated by an iterative algorithm resorting at each step to a source of independent calls of a random generator. If that Markov chain is run for long enough, one can consider that the final output has the desired probability distribution.

The paper is organized as follows. Section 2 describes algorithms for the generation of random Tboxes, depending on the DL constructors. In section 3, the generation of random Aboxes related to a given Tbox will be treated by a MCMC algorithm.

## 2 Generating random Tboxes

Let $\mathcal{L}$ be a DL language defined by a set of *constructors*. We define a $\mathcal{L}$-Tbox as a set of *concept expressions* that are constructed on a given set of *basic* concepts and roles, and the set of constructors available in $\mathcal{L}$ .

The various DL languages differ from one another by the set of constructors they allow. In this paper, we consider DL languages whose constructors are contained in the following list: *conjunction* $(C \sqcap D)$, *disjunction* $(C \sqcup D)$, *negation* $(\neg C)$, *universal quantification* $(\forall R.C)$, *existential quantification* $(\exists R.C)$, *number restrictions* $((\geq n\, R)$, $(\leq n\, R)$, $(\geq n\, R\, C)$, $(\leq n\, R\, C))$. In particular, for sake of simplicity, we do not consider role constructors or constructors involving individuals, though most of the notions presented on this paper could easily be extended to them.

The *length* of a concept expression is the number of constructors it contains. It is defined as follows:

$length(A) = 0$ (if $A$ is a basic concept)
$length(\leq n\, R) = length(\geq n\, R) = 1$
$length(C_1 \sqcap C_2) = length(C_1 \sqcup C_2) =$
$length(C_1) + length(C_2) + 1$
$length(\forall\, R.C) = length(\exists\, R.C) = length(\neg C) =$
$length(\leq n\, R\, C) = length(\geq n\, R\, C) = length(C) + 1.$

The *depth* of a Tbox is the maximal length of the concept expressions that it contains.

The first observation is that there is an infinity of Tboxes that can be constructed on a given set of basic concepts and roles, and a given set of DL constructors. However, there are only a finite number of concepts with a fixed maximal length. One could consider natural to fix the depth of the desired Tbox, then to choose it uniformly in the finite set of all Tboxes with the same depth. However, on a given set of basic concepts, roles and contructors, the total number of concepts with length $i$ grows exponentially with $i$. Thus a random Tbox uniformly distributed in the set of Tboxes with depth $d$ will contain nearly only concepts of length $d$. For that reason, the uniform distribution cannot reflect the composition of Tboxes that are met in most applications. We propose instead to fix a priori the numbers of concepts of each length that compose the Tbox. This is a way to adjust the features of the artificially generated Tboxes to a real one. We shall denote by $d$ the depth of the Tboxes to be generated, and for $i = 0, \ldots, d$, by $n_i$ the number of concepts with length $i$. Thus the size of the Tboxes will be $n_0 + \cdots + n_d$. We shall define below a procedure $Gen(i)$ that generates a random concept with length $i$. The algorithm for constructing the Tbox will simply consist of calling independently $n_i$ times the procedure $Gen(i)$, for all $i = 0, \ldots, d$.

The procedure $Gen(i)$ is recursive. $Gen(0)$ is simply the random choice with uniform distribution of a basic concept. For $i \geq 1$, the first step of $Gen(i)$ is to choose a contructor. With the same goal of flexibility in the design of the sample, one parametrizes the proportion of each constructor composing the different concepts at a given level. Let $K = \{K_1, \ldots, K_\ell\}$ be the set of available constructors. For each level $i = 1, \ldots, d$, let $\{p_1^{(i)}, \ldots, p_\ell^{(i)}\}$ be a probability distribution on the set $K$. The probability $p_j^{(i)}$ represents the proportion of the constructor $K_j$ used in the definition of concepts at level $i$[1]. The general procedure is as follows.
Choose a constructor, say $K_j$ with probability $p_j^{(i)}$.

- If $K_j$ is the negation constructor, let $C$ be the output of $Gen(i-1)$. Then $Gen(i)$ returns $\neg C$.
- If $K_j$ is a binary constructor $(\sqcap, \sqcup)$, then choose a random integer, uniformly distributed on $\{0, \ldots, i-1\}$. Let $C_1$ be the output of $Gen(i_1)$ and $C_2$ be the output of $Gen(i - i_1 - 1)$. Then $Gen(i)$ returns $K_j(C_1, C_2)$.
- If $K_j$ is a quantification constructor $((\forall R.C)$ or $(\exists R.C))$, then choose a random role $R$ with uniform distribution on the set of basic roles, and let $C$ be the output of $Gen(i-1)$. Then $Gen(i)$ returns $K_j(R, C)$.
- If $K_j$ is a cardinality restriction, the additional difficulty is to choose randomly the integer $n$. Choose first a random role $R$ with uniform distribution on the set of basic roles. Now the integer $n$ can be chosen according to any of the classical distributions

[1] For $i \geq 2$, $p_j^{(i)} = 0$ for the constructors $K_j$ being $(\leq n\, R)$, $(\geq n\, R)$, or the negation restricted to basic concepts.

on the set of integers (Geometric, Poisson, Binomial and so on). It is possible (and even desirable) to have the parameter of the distribution depend on the role $R$. Once $n$ has been chosen,

- if $K_j$ is an unqualified number restriction ($(\leq n\, R)$, or $(\geq n\, R)$), $i$ is necessarily equal to 1 and $Gen(1)$ returns $K_j(n, R)$,
- if $K_j$ is a qualified number restriction ($(\leq n\, R\, C)$, or $(\geq n\, R\, C)$), let $C$ be the output of $Gen(i-1)$. Then $Gen(i)$ returns $K_j(n, R, C)$.

**Remark 2.1:** It has to be noted that, in contrast with the random generation process described in [8], we do not force the subexpressions of the generated concepts of length $i$ to appear explicitly in the Tbox.

**Remark 2.2:** During the process of generating a Tbox, even if the same concept might theoretically be generated several times, this is extremely unlikely considering the orders of magnitude. However, it is necessary to check that each generated concept is consistent.

**Remark 2.3:** We have supposed that the basic concepts and roles were randomly chosen with a uniform distribution. They could be chosen with another distribution in order to favor some basic concepts and roles or to adjust a given real distribution.

# 3 Generating random Aboxes

Let $\mathcal{T}$ be a Tbox and let $\mathcal{O}$ be a set of individuals. An Abox relative to $\mathcal{T}$ and $\mathcal{O}$ is a set of facts of the form $C(a)$, or $R(a, b)$, where $C$ and $R$ are respectively a concept expression and a basic role appearing in $\mathcal{T}$, and $a$, $b$ are individuals of $\mathcal{O}$.

In this section, we address the problem of generating uniform samples of *admissible* Aboxes relative to a given Tbox and a set of individuals. The notion of admissibility of an Abox varies depending on the purpose for which the benchmarks are constructed. Indeed, it reflects the kinds of Aboxes that are significant as inputs of the algorithm or the analysis under consideration. For a probabilistic analysis of satisfiability checking, one can consider that any Abox is admissible, while for analyzing a query-answering algorithm even if it consists of checking the unsatisfiability of the Abox together with the negation of the query., only satisfiable Aboxes will be considered as admissible inputs. In other cases, the admissible Aboxes will be satisfiable and not redundant (i.e. not containing facts of the form $C(a)$ and $D(a)$ where $D$ subsumes $C$).

Let $\mathcal{A}$ be the set of admissible Aboxes relative to $\mathcal{T}$ and $\mathcal{O}$. Let $c$ be the number of concept expressions in $\mathcal{T}$, $r$ the number of basic roles appearing in $\mathcal{T}$, and $k$ the number of individuals in $\mathcal{O}$. The set $\mathcal{A}$ is finite, with a potentially large cardinality, actually bounded above by $2^{ck+rk^2}$. Enumerating that set in order to produce a uniformly distributed random sample is not feasible. In such cases, one has to resort to a MCMC algorithm. The idea is to define an undirected graph structure, the vertices of which are the elements of the set to be sampled. In our case, the vertices will be Aboxes relative to a given Tbox and a set of objects, and the edges will connect pairs of Aboxes differing by a single fact. A natural Markov chain (see e.g. [10]) on a graph is the *symmetric random walk* that explores the vertex set by choosing randomly at each step the next state among the nearest neighbors of the current state. It can easily be proved that the symmetric random walk on an undirected connected graph admits the uniform distribution on the vertex set as its unique asymptotic distribution. Thus the output of that random walk after a sufficient running time can be considered as a uniformly distributed random variable on the vertex set. The problem is to decide what 'sufficient' means, i.e. to define an explicit stopping test. A method for generating a random sample of Markov chains, with an explicit stopping test has been proposed in [11]. The counter-intuitive result is that the limiting uniform distribution for the sample can be reached long before the elements have spanned the whole state space. Thus, the method can be actually implemented in spite of the combinatorial explosion in the size of the state space.

## 3.1 Markov Chain Monte-Carlo method

**Definition 3.1:** *Let $G = (V, E)$ be an undirected connected graph with (finite) vertex set $V$ and edge set $E$. For each $x \in V$, let $\mathcal{N}(x)$ be the set of neighbors of $x$.*

$$\mathcal{N}(x) = \{\, y \in V \;:\; \{x, y\} \in E \,\}\,.$$

*Let $d(x)$ be the degree of vertex $x$ (cardinality of $\mathcal{N}(x)$), and $d$ be the maximal degree of the graph (maximal value of $d(x)$). The* symmetric random walk *on the graph $G$ is the sequence $\{X_m\,,\ m \geq 0\}$ generated by the following algorithm, in which the successive random choices are assumed to be independent.*

$m \longleftarrow 0$
*Initialize* $X_0 \in V$
*Repeat*
  *Choose* $X_{m+1} \in \mathcal{N}(X_m)$ `with probability` $1/d$
    *or* $X_{m+1} = X_m$ `with probability` $(1 - d(x))/d$.
  $m \longleftarrow m + 1$
*Until* `stopping condition.`

**Theorem 3.1:** *Under the hypotheses of definition 3.1, the uniform distribution on $V$ is the unique stationary distribution of the symmetric random walk $\{X_m\,,\ m \geq 0\}$ on the graph $G$. As $m$ tends to infinity, $X_m$ converges in distribution to that stationary distribution.*

In the classical presentation of MCMC algorithms, a sample of size $n$ is obtained by extracting $n$ regularly spaced values of $\{X_m\,,\ m \geq 0\}$:

$$(X_{m_0}, X_{m_0+m_1}, \ldots, X_{m_0+nm_1})\ .$$

Even though several heuristics have been proposed, no rigourous result at this day permits a clear choice for $m_0$ and $m_1$. An alternative was proposed in [11], based on running $n$ independent versions of the algorithm, all starting with the same initialization. This generates a Markov chain on the product of $n$ copies of the initial state space. It has been proved that a *cutoff phenomenon* occurs for this product Markov chain, in the sense that it remains far from equilibrium before a certain cutoff time, and very close after (propositions 3.1 and 3.2 of [11]). The advantage of this approach is that the cutoff phenomenon yields a natural stopping condition for the algorithm: the simulation should be run at least until cutoff, while it is useless to run it for much longer after. It should be noted that since the cutoff time grows only as $\log(n)$, the method is reasonable from the point of view of computing time. The practical interest of this approach is that the cutoff can be detected algorithmically. Let $W$ be a subset of $V$ and assume that its probability for the uniform distribution on $V$, $|W|/|V|$, is known or has been statistically estimated. Let $\{Z_m\,,\ m \geq 0\}$ be the product chain of $n$ symmetric random walks on $G$.

$$Z_m = (X_m^{(1)}, \ldots, X_m^{(n)})\ ,$$

where the $X_m^{(i)}$'s are independent copies of the random walk of definition 3.1, all starting from identical states. Consider the proportion of elements in the sample that are in $W$ at time $m$:

$$S_m(W) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_W(X_m^{(i)})\ ,$$

where $\mathbb{1}_W(x)$ is the indicator function for $x$ belonging to $W$: 1 if it does, 0 else. As $m$ tends to infinity, the distribution of the sample tends to the product of $n$ copies of the uniform distribution by itself. Thus $S_m(W)$ should be asymptotically close to the limit probability $|W|/|V|$. Under some technical assumptions, the first instant $m$ at which $S_m(W)$ reaches the limit value is a converging estimate of the cutoff time (Proposition 4.1 of [11]). This determines the desired stopping condition for the generating algorithm.

## 3.2 Uniform samples of Aboxes

In the graph structure that we define on $V = \mathcal{A}$, edges connect pairs of admissible Aboxes differing by a single fact. A priori, the number of possible neighbors of a given Abox is $ck + rk^2$. It is indeed the number of neighbors of the empty Abox. Thus the maximal degree of the graph structure is $d = ck + rk^2$. Due to possible incompatibilities or redundancies, the degree of a general Abox can be notably smaller. This is reflected in the algorithm, through the admissibility test.

In order to apply theorem 3.1, one has to check that the graph structure so defined is connected. Let $A$ be an admissible Abox. Then any Abox obtained by removing a fact from $A$, is still admissible. Thus there exists a path connecting any admissible Abox to the empty one. Hence the connectedness of the graph.

The algorithm for generating a uniform sample of $n$ admissible Aboxes relative to a given Tbox and a given set of individuals is the following:

$[A_1, \ldots, A_n] \longleftarrow [\emptyset, \ldots, \emptyset]$
*Repeat*
  *For* $i = 1$ `to` $N$:
    *Choose* `concept fact`
      `with probability` $ck/(ck + rk^2)$
    *or* `role fact`
      `with probability` $rk^2/(ck + rk^2)$
    *If* `concept fact` *Then*
      *Choose* `a concept expression` $C$
        `with probability` $1/c$, *and*
      *Choose* `an individual` $o$
        `with probability` $1/k$
      *If* $C(o) \in A_i$ *Then* $A_i \longleftarrow A_i \setminus \{C(o)\}$
        *Else If* $A_i \cup \{C(o)\}$ `admissible`
        *Then* $A_i \longleftarrow A_i \cup \{C(o)\}$
    *If* `role fact` *Then*
      *Choose* `a role` $R$
        `with probability` $1/r$ *and*
      *Choose* `a couple of individuals` $(o, o')$
        `with probability` $1/k^2$
      *If* $R(o, o') \in A_i$ *Then* $A_i \longleftarrow A_i \setminus \{R(o, o')\}$
        *Else If* $A_i \cup \{R(o, o')\}$ `admissible`
        *Then* $A_i \longleftarrow A_i \cup \{R(o, o')\}$
  *EndFor*
*Until* $|\{A_i\ :\ F \in A_i\}| \geq np[F]$.

In the stopping condition, $F$ is a given fact, and $p[F]$ is the probability that a random Abox contains $F$: $p[F]$ can be estimated on a smaller sample.

It has to be noted that this algorithm calls several times the admissibility test. In theory, this test might be very costly since it consists of checking satisfiability or redundancy of Aboxes. However, in the setting of our algorithm, this admissibility test is always checked on Aboxes that differ from an admissible Abox by one single fact. It can be expected that checking the admissibility of such Aboxes is less costly than checking admissibility in general without a priori knowledge about the Aboxes. It is indeed the case when the admissibility test can be focused on the fact that is added to an admissible Abox. In the setting of the DL language (referred to as core-CLASSIC) having the constructors $\sqcap, \forall, (\geq\ nR), (\leq\ nR)$ and $\neg$ (on basic concepts only), we have implemented the following algorithm for testing whether satisfiability is preserved when a new fact is added to a satisfiable Abox.

- If the fact $F$ to be added to $\mathcal{A}$ is $C(o)$,
  - if $C$ has the form $(\forall R_1 \ldots \forall R_n.D)$, for each $R_1$-...-$R_n$-successor [2] $s$ of $o$, if it exists $D'(s) \in \mathcal{A}$ such that $D$ and $D'$ are disjoint, or if $D$ has the form $(\leq\ nR)$ and $s$ has more than $n$ R-successors in $\mathcal{A}$, then $\mathcal{A} \cup F$ is not admissible,
  - for each $R_1$-...-$R_n$-predecessor $p$ of $o$ such that $(\forall R_1 \ldots \forall R_n.D)(p) \in \mathcal{A}$, if $C$ and $D$ are disjoint then $\mathcal{A} \cup F$ is not admissible.
- If the fact $F$ to be added to $\mathcal{A}$ is $R(o, o')$,
  - for each $R_1$-...-$R_n$-predecessor [3] $p$ in $\mathcal{A}$ of $o$ such that $(\forall R_1 \ldots \forall R_n.(\leq\ nR))(p) \in \mathcal{A}$, if $o$ has $n$ R-successors in $\mathcal{A}$ then $\mathcal{A} \cup F$ is not admissible,
  - for each $(\forall R.\forall R_1 \ldots \forall R_n.C)(o) \in \mathcal{A}$, for each $R_1$-...-$R_i$-successor $s$ of $o'$ such that $(\forall R_{i+1} \ldots \forall R_n.D)(s) \in \mathcal{A}$, if $C$ and $D$ are disjoint then $\mathcal{A} \cup F$ is not admissible.

It enabled us to implement the above MCMC generation algorithm in order to obtain random uniform benchmarks to evaluate a saturation algorithm of databases encoding core-CLASSIC-Aboxes [2].

## 4 Conclusion and Perspectives

In this paper, we have presented the foundations for generating random Tboxes and Aboxes. In particular, we have shown the usefulness of MCMC methods for generating uniform samples of Aboxes relative to a given Tbox and a set of individuals. In the setting of the core-CLASSIC language, we have implemented an algorithm for generating random Tboxes and a MCMC simulation algorithm for generating random satisfiable Aboxes. Those algorithms have been run to generate, first, a Tbox having 10 roles and 100 concepts (including 80 % of basic concepts or negation of basic concepts), second, a sample of 100 Aboxes relative to that Tbox and to 100 individuals.

This preliminary work is the beginning of a broad program which will consist of performing a probabilistic analysis of the structure of the random Tboxes and Aboxes, and of the DL inference algorithms. Two levels will have to be distinguished in this probabilistic analysis. The first one will consider random Aboxes relative to a fixed Tbox, the second one will analyze random Aboxes relative to Tboxes that will themselves be randomly generated.

It will be interesting to check whether zero-one laws are observed in the DL setting as they occur for random graphs or boolean satisfiability. Zero-one laws are a universal feature of asymptotic random phenomenons. If a random event is defined on a structure of very large size with a high degree of built-in independence, then it can be expected in general that its probability is either close to 0 or close to 1. Such phenomenons are common in random graphs.

In an analogous way, one should expect concentration results for many variables of practical interest. The interest of a true probabilistic analysis is to give not only an average for the quantities of interest (costs, size ...), but also a precision, or confidence interval around those average values.

[2] If the role chain is empty, $(\forall R_1 \ldots \forall R_n.D)$ is $D$ and $o$ is its own and only $R_1$-...-$R_n$-successor.

[3] If the role chain is empty, $o$ is its own and only $R_1$-...-$R_n$-predecessor.

## References


[1] P. Bresciani, E.Franconi, S.Tessaris. Implementing and testing expressive Description Logics: a preliminary report. In *Proceedings of DL-95*, 1995.

[2] Q. Elhaik and M-C. Rousset. Making an Abox persistent. In *Proceedings of DL'98*, 1998.

[3] G.S. Fishman. *Monte-Carlo concepts algorithms and applications.* Springer-Verlag, New York, 1996.

[4] F. Giunchiglia, R.Sebastiani. A SAT-based decision procedure for ALC. In *Proceedings of KR'96*, 1996.

[5] F. Giunchiglia, M. Roveri, R.Sebastiani. A new method for testing decision procedures in modal and terminological logics. In *Proceedings of DL'96*, 1996.

[6] D. Nardi F.M. Donini, M. Lenzerini, and A. Schaerf. Reasoning in description logics. In *Principles of Artificial Intelligence.* G.Brewska (ed.), Springer-Verlag, New York, 1995.

[7] M. Hofri. *Probabilistic analysis of algorithms.* Springer-Verlag, New York, 1987.

[8] B. Nebel, J. Heinsohn, D. Kudenko, and H.J. Profitlich. An empirical analysis of terminological representation system. *Artificial Intelligence*, 1994.

[9] B. Selman. Stochastic search and phase transitions: AI meets physics. In *Proceedings of AAAI-94*, 1994.

[10] A. Sinclair. *Algorithms for random generation and counting: a Markov chain approach.* Birkhäuser, Boston, 1993.

[11] B. Ycart. Cutoff for samples of Markov chains. Research report MAI number 51, available on web site http://www-lmc.imag.fr/MAI, 1998.