A Probabilistic Approach to Navigation in Hypertext

Mark Levene
University College London
Gower Street
London WC1E 6BT, U.K.
email: mlevene@cs.ucl.ac.uk

George Loizou
Birkbeck College
Malet Street
London WC1E 7HX, U.K.
email: george@dcs.bbk.ac.uk

Abstract

One of the main unsolved problems confronting Hypertext is the navigation problem, namely the problem of having to know where you are in the database graph representing the structure of a Hypertext database, and knowing how to get to some other place you are searching for in the database graph. Previously we formalised a Hypertext database in terms of a directed graph whose nodes represent pages of information. The notion of a trail, which is a path in the database graph describing some logical association amongst the pages in the trail, is central to our model. We defined a Hypertext Query Language, HQL, over Hypertext databases and showed that in general the navigation problem, i.e. the problem of finding a trail that satisfies a HQL query (technically known as the model checking problem), is NP-complete. Herein we present a preliminary investigation of using a probabilistic approach in order to enhance the efficiency of model checking. The flavour of our investigation is that if we have some additional statistical information about the Hypertext database then we can utilise such information during query processing. We present two different approaches.

The first approach utilises the theory of probabilistic automata. In this approach we view a Hypertext database as a probabilistic automaton, which we call a Hypertext probabilistic automaton. In such an automaton we assume that the probability of traversing a link is determined by the usage statistics of that link. We exhibit a special case when the number of trails that satisfy a query is always finite and indicate how to give a finite approximation of answering a query in the general case.

The second approach utilises the theory of random Turing machines. In this approach we view a Hypertext database as a probabilistic algorithm, realised via a Hypertext random automaton. In such an automaton we assume that out of a choice of links, traversing any one of them is equally likely. We obtain a lower bound of the probability that a random trail satisfies a query. In principle, by iterating this probabilistic algorithm, associated with the Hypertext database, the probability of finding a trail that satisfies the query can be made arbitrarily large.

Keywords. Hypertext, navigation, trail, probabilistic automata, query language

1 Introduction

Hypertext [NIEL90], or more generally Hypermedia, is text (which may contain multimedia) that can be read nonsequentially, in contrast to traditional text, for example in book form, which has a single linear sequence defining the order in which the text is to be scanned. Hypertext presents several different options to readers, and the individual reader chooses a particular sequence at the time of reading. The inspiration for Hypertext originates from Bush's visionary idea of the *memex* machine [BUSH45].

A *Hypertext database* [LEVE94] (see also [STOT89, FRIS92]) is formalised as a directed graph [BUCK90] (called the *reachability relation*) whose nodes (called *states*) represent textual units of information (called *pages*) and whose arcs (called *links* or *transitions*) allow the reader to navigate from an anchor node to a destination node. The structure of a Hypertext database changes over time, and thus differs from traditional databases, such as relational databases [ULLM88], which have a regular structure defined by a relational database schema. In order to query pages we also associate with a Hypertext database an *information repository*, which maps each state of the reachability relation to a set of *conditions* that are true at that state. In logical terminology a Hypertext database is a *temporal structure* [EMER90] (see also [STOT92] whereby temporal logic is used to formalise the notion of a Hypertext database).

The process of traversing links and following a *trail* of information in a Hypertext database is called *navigation* (or alternatively *link following*). In graph-theoretic terms a trail in a Hypertext database is a *path* in the database graph (we allow a node to occur more than once in a path; paths are called *walks* in [BUCK90]). Navigating through a Hypertext database leads to the problem of getting "lost in hyperspace" [NIEL90], which is the problem of having to know where you are in the database graph and knowing how to get to some other place that you are searching for in the database graph. From now on we will refer to this fundamental problem as the *navigation problem*.

In a more general context of a Hypertext system the process of finding and examining the text pages associated with destination nodes is called *browsing* [NIEL90]. The *browser* is the component of a Hypertext system that helps users search for the information they are interested in by graphically displaying the relevant parts of the database and providing contextual and spatial cues with the use of *navigational aids* such as *maps* and *guides* [FRIS92]. A set of tools that aid navigation by performing a structural analysis of the database graph is described in [RIVL94].

In a previous paper [LEVE94] we studied the navigation problem in Hypertext via a *Hypertext Query Language*, HQL, which is a subset of *propositional linear temporal logic* [EMER90]. The navigation problem can be viewed as the process of finding a trail which satisfies the user's query; in logical terminology this is known as the *model checking problem*. In [LEVE94] we have shown that the navigation problem in Hypertext cannot be solved efficiently, since model checking in HQL is, in general, NP-complete [GARE79]. Despite this negative result we have shown therein that in some special cases model checking can be done in polynomial time. Finally, we concluded that in practice, if we do not wish to restrict the user's query language, approximate solutions to the model checking problem must be sought after.

The aim of this paper is to present a preliminary investigation of using a probabilistic approach in order to enhance the efficiency of model checking. We present two different approaches and give some preliminary results for each approach. The flavour of our investigation is that if we have some additional statistical information about the Hypertext database then we can utilise such information during query processing.

The first approach utilises the theory of *probabilistic automata* [RABI63, PAZ71]. In this approach we view a Hypertext database as a probabilistic automaton, called a *Hypertext Probabilistic Automaton* (HPA). The arcs of the reachability relation of the HPA are labelled with probabilities that are computed from statistical information related to the frequency that users choose to navigate from a state s_i to an adjacent state s_j via the arc (s_i, s_j), when browsing the text associated with s_i. In this case we investigate the language accepted by such a HPA with a *cut-point* of λ in the open interval (0,1) (i.e. a trail, viewed as a string, is accepted by the HPA if the probability of the trail is greater than λ). We show that this language is both a star-free [MCNA71] and an unambiguous [STEA85] regular language. In addition, if the reachability relation is 2total, i.e. there are always at least two choices to navigate from any given state s_i, then this language is also finite.

The second approach utilises the theory of *random Turing machines* [LEEU56, RABI76, GILL77, JOHN90]. In this approach we view a Hypertext database as a probabilistic algorithm, realised via a *Hypertext Random Automaton* (HRA), in which random choices are made during navigation assuming that all states adjacent to a state s_i are equally likely, i.e. they are uniformly distributed. In this case we investigate the probability that a random trail consisting of k states, constructed by k random choices while navigating through the Hypertext database, is a trail that satisfies a given query. In principle, by iterating this probabilistic algorithm m times, where m is an appropriate natural number, the probability of finding a trail that satisfies the query can be made arbitrarily large. For the special case when the reachability relation is μ-regular, i.e. the number of choices to navigate from a state s_i is the constant μ, we give a lower bound on the number of trails in the reachability relation that satisfy a given query, assuming that there exists at least one trail that logically implies the query. This lower bound is used to obtain a lower bound on the probability that a random trail satisfies the query.

In both approaches a Hypertext database can be viewed as a *finite homogeneous Markov chain* (or simply a Markov chain) [KEME60, FELL68]. In the first approach the initial and transition probabilities are determined by statistical information and in the second approach these probabilities are assumed to be uniform.

The following notation will be used throughout the paper.

Definition 1.1 (Notation) The set of all natural numbers is denoted by ω.

The set of all strings of finite length, over a finite nonempty alphabet Σ, is denoted by Σ^* and $\Sigma^+ = \Sigma^* - \{e\}$, where e denotes the empty string. A *language* $\mathcal{L}$ over Σ is a subset of Σ^+.

Concatenation of two strings w and z will be denoted by wz and x^k will denote the concatenation of x with itself k times with $k \geq 0$. In addition, a string y is a substring of a string w if there exist strings x and z such that $w = xyz$; if the symbol $a \in \Sigma$ is a substring of w we write $a \in w$.

We denote the cardinality of a set, S, by |S| and the size of a string, i.e. the number of symbols in w, by $||w||$; in particular $||e|| = 0$. The set of all strings in Σ^* having size $k \in \omega$ is denoted by Σ^k. The finite powerset of S is denoted by $PowerSet(\mathrm{S})$.

The open interval from 0 to 1 is denoted by (0,1), the closed interval from 0 to 1 is denoted by [0,1] and the half-open interval from 0 to 1, which is open to the right, is denoted by [0,1).

Finally, we abbreviate "if and only if" to iff.

The rest of the paper is organised as follows. In Section 2 we introduce our underlying model for Hypertext that was presented fully in [LEVE94]. In Section 3 we introduce the notion of Hypertext finite automata and give a characterisation of the subclass of regular languages accepted by such automata. In Section 4 we define Hypertext probabilistic automata and characterise the subclass of regular languages accepted by such automata. In Section 5 we define a probabilistic query language for Hypertext and investigate the complexity of model checking in this query language. In Section 6 we define Hypertext random automata based on the concept of random Turing machines and introduce the notion of the success probability of a Hypertext random automaton. In Section 7 we investigate query answering with respect to Hypertext random automata by giving a lower bound on the success probability of a Hypertext random automaton. Finally, in Section 8 we give our concluding remarks.

2 Hypertext Databases and Trail Queries

Herein we give the necessary background from [LEVE94] concerning *Hypertext Databases* and trail queries.

We use the following graph-theoretic terminology where R is a binary relation in a set S (R corresponds to a directed graph in which loops are allowed [BUCK90]). Whenever $(s_i, s_j) \in \mathrm{R}$ we say that s_j is *adjacent* to s_i; we call (s_i, s_j) an *arc*. The *degree* of an element $s_i \in \mathrm{S}$ is the number of states $s_j \in \mathrm{S}$ that are adjacent to s_i. The following special types of binary relations will be used throughout the paper.

Definition 2.1 (A complete relation) A relation R in S is *complete* (cf. complete graph [BUCK90]) if $\forall s_1, s_2 \in$ S, $(s_1, s_2) \in$ R.

Definition 2.2 (A regular relation) A relation R in S is *μ-regular* (or simply regular) if $\exists \mu \in$ {1,. . .,|S|} such that $\forall s_i \in$ S, the degree of s_i is μ (cf. regular graph [BUCK90]).

Definition 2.3 (A total relation) A relation R in S is *total* if $\forall s \in$ S, $\exists t \in$ S such that $(s, t) \in$ R.

Definition 2.4 (A 2total relation) A relation R in S is *2total* if $\forall s \in$ S, $\exists t_1, t_2 \in$ S such that $t_1 \neq t_2$ and $(s, t_1), (s, t_2) \in$ R.

We will now define a Hypertext database and a simplified version of HQL; see [LEVE94] for the full version of HQL.

Definition 2.5 (Hypertext database) Let Φ be a countably infinite set of primitive propositions (also called *conditions*) and S $= \{s_1, \ldots, s_N\}$ be a finite set of *states.*

A Hypertext database H is an ordered pair, <I, R>, where I is a mapping from S to $PowerSet(\Phi)$ and R is a total relation in S.

I is called the *information repository* of H and R is called the *reachability relation* of H.

Intuitively, the information repository, I, represents the conditions that are true at each given state $s \in$ S, where a state represents a *page* of text (or hypermedia). Furthermore, the reachability relation, R, represents the possible trails a user can traverse during the process of navigating through the Hypertext database.

Definition 2.6 (Trail) A *trail* in R is a finite sequence of states (i.e. a string of states) corresponding to a path in R. We let #T denote the number of states of a trail T in R (we call #T the *count* of T) and let T[j], $j \in$ {1,. . .,#T}, denote the jth state of T; if $j < 1$ or $j >$ #T, then T[j] is undefined.

Definition 2.7 (Trail query) A *trail query* (or simply a query) ϕ is an expression of the form

$$c_1 \wedge \ldots \wedge c_n, n \in \omega,$$

where $c_i, i \in \{1, \ldots, n\}$, is a condition.

The semantics of a trail query ϕ with respect to a Hypertext database H = <I, R> (H is omitted if it is understood from context) are specified by

$$\mathrm{T} \models \phi \text{ iff } \forall i \in \{1, \ldots, n\}, \exists j \in \{1, \ldots, \#T\} \text{ such that } c_i \in \mathrm{I}(\mathrm{T}[j]),$$

where T is a trail in R. The *Hypertext Query Language* (HQL) is defined to be the set of all possible trail queries.

In temporal logic [EMER90] terminology each condition c_i in a trail query ϕ is interpreted as asserting that "sometimes" c_i is true. We note that HQL has been simplified for the purpose of this paper; the full version of HQL, as defined in [LEVE94], also supports supports the temporal operators "nexttime" and "finaltime" and general Boolean queries over individual pages.

Definition 2.8 (Model checking) A Hypertext database H = <I, R> is a *model* of a trail query, ϕ, if $\exists$T in R such that T $\models \phi$ with respect to H.

The *model checking* problem is the problem of deciding whether a Hypertext database is a model of a trail query.

Even though HQL, as defined above, is only a restricted subset of the full query language defined in [LEVE94], the following result concerning the intractability of its model checking was shown in [LEVE94].

Theorem 2.1 The model checking problem for HQL trail queries is NP-complete. $\square$

3 Hypertext Databases and Finite Automata

We now give the necessary background from [LEVE94] concerning *Hypertext Finite Automata* (HFA) and investigate the subclass of regular languages accepted by HFA. We assume that the reader is familiar with the basic theory of finite automata [HOPC79, PERR90].

Definition 3.1 (Star-free languages) A regular language is *star-free* if it can be generated from a finite set of strings by repeated applications of the Boolean operations together with concatenation.

A comprehensive treatment of several equivalent definitions of star-free regular languages can be found in [MCNA71].

The following definition is needed in order to state an alternative characterisation of star-free languages.

Definition 3.2 (Aperiodic regular languages) A regular language $\mathcal{L} \subseteq \Sigma^+$ is *aperiodic* if $\exists n \in \omega (n > 0)$ such that $\forall x, y, z \in \Sigma^*$, $xy^nz \in \mathcal{L}$ iff $xy^{n+1}z \in \mathcal{L}$.

The following theorem states the equivalence of star-free and aperiodic regular languages [PERR90, Theorem 6.1].

Theorem 3.1 A regular language is *star-free* iff it is aperiodic. $\square$

Definition 3.3 (Hypertext finite automata) The *Hypertext finite automaton* (HFA) representing a Hypertext database, H = <I, R>, is a quintuple of the form $\mathcal{M}_H$ = ($\mathcal{A}$, Q, Δ, Q^i, Q^f) (or simply $\mathcal{M}$ whenever H is understood from context), where

- $\mathcal{A} = \{1, \ldots, N\}$ is a finite alphabet, with $N =$|S|.
- Q = S = $\{s_1, \ldots, s_N\}$ is the set of states of the HFA.
- $\Delta \subseteq$ Q $\times \mathcal{A} \times$ Q is a transition relation, where $(s_j, \delta(s_j), s_k) \in \Delta$ iff $(s_j, s_k) \in$ R, with δ being a one-to-one and onto mapping from Q to $\mathcal{A}$ such that $\delta(s_i) = i$.
- Q^i = Q and Q^f = Q are the sets of initial and final states, respectively.

If Q = $\emptyset$, then $\mathcal{M}$ is called the *empty Hypertext finite automaton.*

The following result was shown in [LEVE94].

Theorem 3.2 $\mathcal{L}(\mathcal{M}_H)$ is a star-free language, where H = <I, R> is a Hypertext database. $\square$

Definition 3.4 (The language accepted by a HFA) A *path* in the HFA, $\mathcal{M}$ = ($\mathcal{A}$, Q, Δ, Q^i, Q^f) (or simply a path if $\mathcal{M}$ is understood from context) is a nonempty sequence $\{(s_i, \delta(s_i), s_{i+1})\}$ ($i \in \{1, \ldots, n-1\}$) of transitions in Δ, with $n \in \omega$. (We also say that there exists a path from $(s_1, \delta(s_1), s_2)$ to $(s_{n-1}, \delta(s_{n-1}), s_n)$.) If $s_1 = s_n$, then the path is a *cycle.*

The string $w = \delta(s_1)\delta(s_2)\ldots\delta(s_n)$ is called the *label* of the path, s_1 is called its *origin* and s_n is called its *end.* The *length* of the path is n. The sequence of states $< s_1, \ldots, s_n >$ is called a *trace* of w in $\mathcal{M}$ (or simply a trace of w if $\mathcal{M}$ is understood from context).

A path is *successful* if its origin is in Q^i and its end is in Q^f. A string $w \in \mathcal{A}^+$ is said to be *accepted* by $\mathcal{M}$ if it is the label of some successful path. The language accepted by $\mathcal{M}$, denoted by $\mathcal{L}(\mathcal{M})$, is the set of all strings accepted by $\mathcal{M}$.

We note that by Definition 3.3 all paths in a HFA are successful, since $Q^i = Q^f$ = Q.

Definition 3.5 (Unambiguous finite automata) A finite automaton $\mathcal{M}$ is *unambiguous* [STEA85] if $\forall w \in \mathcal{L}(\mathcal{M})$ there exists only one trace of w in $\mathcal{M}$. A regular language $\mathcal{L}$ is unambiguous if there exists an unambiguous finite automaton that accepts $\mathcal{L}$.

The following proposition follows from Definition 3.3, since δ is a one-to-one and onto mapping from the set of states of a HFA to its alphabet.

Proposition 3.3 If $\mathcal{M}$ is a HFA, then $\mathcal{M}$ is unambiguous. $\square$

We next characterise HFA in terms of a subclass of regular sets.

Definition 3.6 (Closure under substrings and join) A regular language $\mathcal{L}$ is closed under substrings if whenever $w \in \mathcal{L}$ and y is a nonempty substring of w, then $y \in \mathcal{L}$.

A regular language $\mathcal{L}$ is closed under join if whenever $xy, yz \in \mathcal{L}$ and $y \neq e$, then $xyz \in \mathcal{L}$.

Theorem 3.4 A regular language $\mathcal{L}$ is closed under substrings and join iff $\mathcal{L} = \mathcal{L}(\mathcal{M})$ for some HFA $\mathcal{M}$.

Proof. The *if part* of the proof follows directly from Definitions 3.3 and 3.4.

For the *only if part* of the proof assume that $\mathcal{L}$ is a regular language that is closed under substrings and join. We need to show that $\mathcal{L} = \mathcal{L}(\mathcal{M})$ for some HFA $\mathcal{M}$. Now, if $\mathcal{L} = \emptyset$, then $\mathcal{M}$ is the empty Hypertext finite automaton for any Hypertext database H = <I, R> with |S| = 0. So, we assume that $\mathcal{L} \neq \emptyset$.

Let $\Sigma = \{a_1, \ldots, a_N\}$, $N \geq 1$, with $\Sigma = \{a_i \mid a_i \in \mathcal{L} \text{ and } ||a_i|| = 1\}$; $\Sigma \neq \emptyset$, since $\mathcal{L} \neq \emptyset$. In addition, let θ be a one-to-one and onto mapping from Σ to $\{1, \ldots, N\}$ such that $\theta(a_i) = i$. We construct $\mathcal{M}$ = ($\mathcal{A}$, Q, Δ, Q^i, Q^f) as follows:

- $\mathcal{A} = \{1, \ldots, N\}$.
- Q = $\{s_1, \ldots, s_N\}$.
- $\Delta \subseteq$ Q $\times \mathcal{A} \times$ Q, where $(s_j, \delta(s_j), s_k) \in \Delta$ iff $a_j a_k \in \mathcal{L}$, with $a_j a_k = \theta^{-1}(\delta(s_j))\theta^{-1}(\delta(s_k)) \in \mathcal{L}$.
- $Q^i = Q^f$ = Q.

The result now follows from Definitions 3.3 and 3.4. $\square$

Definition 3.7 (HFA regular languages) A regular language $\mathcal{L}$ is said to be a HFA language if $\mathcal{L} = \mathcal{L}(\mathcal{M})$ for some HFA $\mathcal{M}$.

We observe that a HFA language $\mathcal{L}(\mathcal{M})$ can be viewed as the set of all navigation paths, through the Hypertext database represented by $\mathcal{M}$, resulting from the composition of two or more paths.

Proposition 3.5 HFA languages are closed under intersection but they are not closed under union, complement or concatenation.

Proof. The fact that HFA languages are closed under intersection follows from Theorem 3.4.

Let $\mathcal{L}_1 = \{$a, b, ab$\}$ and $\mathcal{L}_2 = \{$b, c, bc$\}$. It follows that HFA languages are not closed under union, since abc $\notin \mathcal{L}_1 \cup \mathcal{L}_2$. In addition, HFA languages are not closed under complement, since abc $\in \overline{\mathcal{L}_1}$ but ab $\notin \overline{\mathcal{L}_1}$. Finally, let $\mathcal{L}_3 = \{$c, d, cd$\}$. It follows that HFA languages are not closed under concatenation, since abc $\notin \mathcal{L}_1\mathcal{L}_3$. $\square$

Definition 3.8 (A closure operator) A closure operator $\mathcal{C}$ is defined on Σ^+ as follows: whenever $\mathcal{L} \subseteq \Sigma^+$, $\mathcal{C}(\mathcal{L})$ is the closure of $\mathcal{L}$ under substrings and join.

We say that a language $\mathcal{J}$ *generates* a language $\mathcal{L}$, if $\mathcal{J}$ is the minimal cardinality language such that $\mathcal{J} \subseteq \mathcal{L}$ and $\mathcal{C}(\mathcal{J}) = \mathcal{L}$.

It now follows from Theorem 3.4 that $\mathcal{C}(\mathcal{L}) = \mathcal{L}$ iff $\mathcal{L}$ is a HFA regular language and thus the set of all closed subsets of Σ^+ is exactly the class of HFA regular languages. It can easily be verified that indeed $\mathcal{C}$ is a *closure operator* in the sense of [DAVE90] and thus the set of all closed subsets of Σ^+ is a *complete lattice* when ordered by set inclusion. The greatest lower bound of a set of HFA languages corresponds to their intersection and the least upper bound of a set of HFA languages corresponds to the closure of their union [DAVE90].

Corollary 3.6 *$\mathcal{J}$ generates a HFA regular language $\mathcal{L}$ iff $\mathcal{J} = \Psi \cup (\Upsilon - \{a \mid ab \in \Psi$ or $ba \in \Psi\})$, where $\Upsilon = \{a \mid a \in \mathcal{L}$ and $||a|| = 1\}$ and $\Psi = \{ab \mid a, b \in \Upsilon$ and $ab \in \mathcal{L}\}$* $\square$

An immediate result of the above corollary is that if $\mathcal{J}$ generates a HFA regular language over Σ, then $|\mathcal{J}| \leq |\Sigma| + |\Sigma|^2$. This is to be expected, since the corresponding HFA has at most $|\Sigma|$ states and at most $|\Sigma|^2$ transitions.

The following additional corollary is an immediate consequence of Theorem 3.2 and Proposition 3.3 noting that if, for example, $\Sigma = \{$a$\}$ and $\mathcal{L} = \{$aa$\}$, then $\mathcal{L}$ is both a star-free and unambiguous regular language but $\mathcal{L}$ is not a HFA regular language.

Corollary 3.7 *The class of HFA regular languages is properly included in the intersection of the class of star-free regular languages and the class of unambiguous regular languages.* $\square$

4 Hypertext Probabilistic Automata

Herein we demonstrate how *probabilistic automata* [RABI63, PAZ71] can be used to reduce the search space of a trail query with respect to a Hypertext database. We show that if the reachability relation of a Hypertext database is 2total then the language accepted by the HFA representing the Hypertext database is finite, thus reducing the search space of the model checking problem to a finite set. The accepted finite language is a subset of Σ^k for some $k \geq 1$. In this case the solution can be computed efficiently when k is small.

We assume that the reader is familiar with the basic theory of probabilistic automata [RABI63, PAZ71]. (The fundamental theory of finite homogeneous Markov chains (or simply Markov chains) can be found in [KEME60, FELL68].)

Definition 4.1 (Hypertext probabilistic automata) A *Hypertext Probabilistic Automaton* (HPA) with cut-point $\lambda \in [0,1)$ representing a Hypertext database, H = <I, R>, is a sextuple of the form $\mathcal{P}_H^\lambda$ = ($\mathcal{A}$, Q, M, π, Q^i, Q^f) (or simply $\mathcal{P}^\lambda$ whenever H is understood from context), where

- $\mathcal{A} = \{1, \ldots, N\}$, is a finite alphabet, with $N =$|S|.
- Q = S = $\{s_1, \ldots, s_N\}$ is the set of states of the HPA.
- M = $\{p(s_i, s_j)\}(s_i, s_j \in$ Q) is a stochastic $N \times N$ matrix, that is $\forall s_i, s_j \in$ Q, $p(s_i, s_j) \geq 0$ and $\forall s_i \in$ Q, $\sum_{j=1}^{N} p(s_i, s_j) = 1$, where p is a function from S $\times$ S to [0,1]. In addition, M satisfies the constraint that $p(s_i, s_j) > 0$ iff $(s_i, s_j) \in$ R.

  The matrix M is called the *transition matrix* of $\mathcal{P}^\lambda$ and the probabilities $p(s_i, s_j)(s_i, s_j \in$ Q) are called the *transition probabilities* of $\mathcal{P}^\lambda$.

- $\pi = \{p(s_i)\}(s_i \in$ Q) is an $N-$dimensional stochastic row vector, that is $\forall s_i \in$ Q, $p(s_i) \geq 0$ and $\sum_{i=1}^{N} p(s_i) = 1$. In addition, π satisfies the constraint that $p(s_i) > 0$ iff $\exists s_j \in$ Q such that either $(s_i, s_j) \in$ R or $(s_j, s_i) \in$ R.

  The row vector π is called the *initial distribution* of $\mathcal{P}^\lambda$ and the probabilities $p(s_i)(s_i \in$ Q) are called the *initial probabilities* of $\mathcal{P}^\lambda$.

- Q^i = Q and Q^f = Q are the sets of initial and final states, respectively.

If Q = $\emptyset$, then $\mathcal{P}^\lambda$ is called the *empty Hypertext probabilistic automaton.*

We note that the initial distribution of a probabilistic automaton, as defined in [RABI63], is *degenerate*, i.e. it has a single state $s_i \in$ S with $p(s_i) = 1$. By [NASU68, Theorem 4.9] the class of probabilistic automata remains unchanged with this restriction. Thus we can use the results in [RABI63] on assuming that the initial distribution may not be degenerate.

Our interpretation of an initial probability $p(s_i)$ is the probability that a user will start his/her navigation by inspecting the page associated with s_i. Correspondingly, our interpretation of a transition probability $p(s_i, s_j)$ is the probability that a user who is browsing the page associated with s_i will next browse the page associated with the adjacent state s_j. We will assume that the probabilities of the transition matrix and the initial distribution are determined from statistical information collected from past usage of the Hypertext database. Thus we make the (perhaps controversial) assumption that the probabilistic automaton is a Markov chain, implying that when a user is navigating through a Hypertext database his/her next navigation step is solely determined by their current position.

Informally, the probability of a string w is the probability of its trace according to the underlying Markov chain. The language accepted by a HPA with cut-point λ is the set of strings whose probability is greater than λ.

Definition 4.2 (The language accepted by a HPA) Given a HPA, $\mathcal{P}^\lambda$, the probability of a string $w = \delta(s_1)\delta(s_2)\ldots\delta(s_n) \in \mathcal{A}^+$ is denoted by $P(w)$ and is given by

$$P(w) = p(s_1)p(s_1, s_2)\ldots p(s_{n-1}, s_n).$$

The language accepted by $\mathcal{P}^\lambda$, denoted by $\mathcal{L}(\mathcal{P}^\lambda)$, is given by

$$\{w \mid w \in \mathcal{A}^+ \text{ and } P(w) > \lambda\}.$$

We note that $P(e)$ is undefined; thus we assume without loss of generality that $P(e) = 0$. The next lemma investigates the special case of a HPA, whose underlying reachability relation is 2total.

Lemma 4.1 Let $\mathcal{P}^\lambda$ be a HPA representing a Hypertext database H = <I, R>, where R is a 2total reachability relation. Then $\mathcal{L}(\mathcal{P}^\lambda)$ is an unambiguous and star-free regular language.

Proof. To begin with, $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{P}^0)$ is unambiguous and star-free by Proposition 3.3 and Theorem 3.2, respectively, where $\mathcal{M}$ is the HFA representing H.

In order to conclude the proof we show that $\forall\lambda \in (0,1)$, $|\mathcal{L}(\mathcal{P}^\lambda)| \in \omega$, i.e. $|\mathcal{L}(\mathcal{P}^\lambda)|$ is finite. This is sufficient, since all finite languages are star-free and unambiguous regular languages.

Let $w \in \mathcal{L}(\mathcal{M})$. By Proposition 3.3 there is a unique trace $< s_1, \ldots, s_n >$ of w. Furthermore, $P(w) = p(s_1)p(s_1, s_2) \ldots p(s_{n-1}, s_n)$.

Now, let α be the maximal probability $p(s_i)$, where $s_i \in$ Q, and let β be the maximal probability $p(s_i, s_j)$, where $s_i, s_j \in$ Q. Then $\forall x \in \mathcal{L}(\mathcal{M})$, $P(x) \leq \alpha \cdot \beta^k$, where $k = ||x||$.

Furthermore, since R is 2total, it follows that $\forall x, y \in \mathcal{L}(\mathcal{M})$ such that x is a substring of y, $P(x) < P(y)$. It follows that $\alpha \cdot \beta^k$ tends to zero as k tends to infinity and therefore $P(x)$ tends to zero as $||x||$ tends to infinity.

Next, $\forall k \in \omega$ there are only a finite number of strings of length k. It follows that $\exists k \in \omega$ such that $\forall x \in \mathcal{L}(\mathcal{M})$ such that $||x|| \geq k$, $P(x) < \lambda$. The result that $|\mathcal{L}(\mathcal{P}^\lambda)| \in \omega$ follows. □

Note that, in general, $\mathcal{L}(\mathcal{P}^\lambda)$ is not a HFA regular language. For example, suppose that λ = 0.33, $ab, bc \in \mathcal{L}(\mathcal{P}^\lambda)$ and $P(ab) = P(bc) = 0.5$. Then $\mathcal{L}(\mathcal{P}^\lambda)$ is not closed under join, since $P(abc) \leq 0.25$. On the other hand, it can easily be shown that $\mathcal{L}(\mathcal{P}^\lambda)$ is closed under substrings.

The next theorem gives an upper bound on the cardinality of $\mathcal{L}(\mathcal{P}^\lambda)$ which was shown to be finite in the proof of Lemma 4.1.

Theorem 4.2 Let α be the maximal probability $p(s_i)$, where $s_i \in$ Q, and let β be the maximal probability $p(s_i, s_j)$, where $s_i, s_j \in$ Q. If the reachability relation is 2total and $\lambda > 0$, then $|\mathcal{L}(\mathcal{P}^\lambda)| \leq N^k$ for some $k \in \omega$ that satisfies

$$\alpha \cdot \beta^k < \lambda. \quad \square$$

In this case $\mathcal{L}(\mathcal{P}^\lambda) \subseteq \Sigma^k$ for some $k \geq 1$. Thus in this special case model checking can be computed efficiently if k is small. The user can always increase the cut-point λ in order to decrease k. Moreover, if the HPA representing the Hypertext database is assumed to be fixed, then model checking can be carried out in polynomial time of order k, provided the conditions of Theorem 4.2 are satisfied. Obviously, if $\lambda = 0$ then the model checking problem is as hard as in the deterministic case, i.e. it is NP-complete [LEVE94].

The following theorem generalises Lemma 4.1 to total reachability relations which may not be 2total.

Theorem 4.3 Let $\mathcal{P}^\lambda$ be a HPA representing a Hypertext database H = <I, R>. Then $\mathcal{L}(\mathcal{P}^\lambda)$ is a star-free and unambiguous regular language.

Proof. This proof utilises the previous theorem and looks at the equivalence classes x/E, of the equivalence relation E, such that xEy, where $x, y \in \mathcal{L}(\mathcal{M})$ iff $P(x) = P(y)$.

It follows that $\mathcal{L}(\mathcal{P}^\lambda)$ is the union of all the equivalence classes x/E such that $P(x) > \lambda$.

We claim that each equivalence class x/E is a star-free and unambiguous regular language. If $|x/\text{E}|$ is finite, then the result follows, so assume that $|x/\text{E}|$ is infinite.

Now, $\forall y \in x/\text{E}$, y is of the form wz^k, with $k \in \omega$ and $k \geq 0$, where either $w \neq e$ or $z \neq e$ and all the symbols in both w and z are distinct. In addition, if $z \neq e$, then it must be the case that $P(z) = 1$ and also that $w, wz, wz^{k+1} \in x/\text{E}$. It therefore follows that there are only a finite number of strings $wz \in x/\text{E}$, since all the symbols in both w and z are distinct.

Moreover, each language $\{y \mid y \in x/\text{E}, y = wz^k$ with $k \in \omega$ and $k \geq 0\}$, which we denote by $\mathcal{L}(y)$, is a regular language, since a finite automaton accepting this language can easily be constructed. Therefore, x/E is a regular language, since it is the union of the finite set of languages,

$\mathcal{L}(y)$, and regular languages are closed under union. Furthermore, it can be verified that x/E is a star-free regular language, since star-free regular languages are closed under union and Theorem 3.1 implies that each $\mathcal{L}(y)$ is a star-free regular language. Moreover, it can also be verified that each regular language $\mathcal{L}(y)$ is unambiguous and that these languages are pairwise disjoint. Therefore, x/E is also an unambiguous regular language, since unambiguous regular languages are closed under union whenever the languages unioned are pairwise disjoint.

Finally, by Theorem 4.2 we can deduce that there are only a finite number of equivalence classes x/E such that $P(x) > \lambda$. Therefore, $\mathcal{L}(\mathcal{P}^\lambda)$ is a star-free and unambiguous regular language, since as mentioned above star-free regular languages are closed under union and, in addition, unambiguous regular languages are closed under union whenever the languages unioned are pairwise disjoint. $\square$

When the reachability relation is not 2total Theorem 4.3 does not guarantee that the cardinality of $\mathcal{L}(\mathcal{P}^\lambda)$ is finite. Still, by inspecting the proof of Theorem 4.3, we can see that the strings in $\mathcal{L}(\mathcal{P}^\lambda)$ all have the simple form wz^k, with $k \geq 1$ and $P(z) = 1$. Thus if $k > 1$, then wz^k is the label of a cycle, which is easily detected as follows. Let $\{s_{i_1}, \ldots s_{i_m}\}$ be the set of states in S having degree one. Furthermore, let $\{s_{j_1}, \ldots s_{j_m}\}$ be the set of states in S such that $\forall q \in \{1, \ldots, m\}$, s_{j_q} is adjacent to s_{i_q}. Now, all we need to do is to test whether there is a cycle in R from s_{j_q} to s_{i_q} such that the degree of each state in this cycle is one. This test is successful iff there exists a string $z^k \in \mathcal{L}(\mathcal{P}^\lambda)$ which is the label of such a cycle. Thus by letting α to be the maximal probability $p(s_i) < 1$, where $s_i \in \mathrm{Q}$, and β to be the maximal probability $p(s_i, s_j) < 1$, where $s_i, s_j \in \mathrm{Q}$, we can utilise Theorem 4.2 to obtain a finite approximation of the cardinality of $\mathcal{L}(\mathcal{P}^\lambda)$, which ignores these cycles when they do exist.

We next consider the concept of an *isolated cut-point* [RABI63, PAZ71], which corresponds to the concept of *bounded error probability* [GILL77].

Definition 4.3 (HPA with isolated cut-point) The cut-point λ of a HPA $\mathcal{P}^\lambda$ is *isolated* (or simply λ is an isolated cut-point whenever $\mathcal{P}^\lambda$ is understood from context) if $\exists \varepsilon > 0$ such that $\forall x \in \Sigma^+$, $|P(x) - \lambda| \geq \varepsilon$, where $|r|$ is the absolute value of a real number, r (cf. [RABI63]).

The concept of isolated cut-point has been of fundamental importance to the theory of probabilistic automata, since it was shown that if a probabilistic automaton has an isolated cut-point, then the language accepted by the probabilistic automaton is regular [RABI63, PAZ71]. On the other hand, a language accepted by a probabilistic automaton can be regular but its cut-point need not be isolated [RABI63, PAZ71].

The following result shows that if $\lambda > 0$, then we can find a HPA $\mathcal{P}^\xi$, whose cut-point ξ is isolated, that is equivalent to $\mathcal{P}^\lambda$ in the sense that the regular languages that $\mathcal{P}^\lambda$ and $\mathcal{P}^\xi$ accept are the same. It can be verified that if $(\mathcal{L} = \mathcal{L}(\mathcal{P}^0)) \subset \Sigma^+$, then $\exists x \in \Sigma^+$ such that $P(x) = 0$, and thus $\lambda = 0$ is not an isolated cut-point unless $\mathcal{L} = \Sigma^+$.

Theorem 4.4 If $\lambda > 0$, then there exists an isolated cut-point $\xi \geq \lambda$ such that $\mathcal{L}(\mathcal{P}^\lambda) = \mathcal{L}(\mathcal{P}^\xi)$.

Proof. If the cut-point λ is already isolated, then take $\xi = \lambda$. So, assume that λ is not an isolated cut-point and thus $\exists x \in \Sigma^+$ such that $P(x) = \lambda$. Let γ be the minimal probability of any string in $y \in \mathcal{L}(\mathcal{P}^\lambda)$, i.e. $\forall y \in \mathcal{L}(\mathcal{P}^\lambda)$, $\lambda < \gamma \leq P(y)$; by the proof of Theorem 4.3 γ exists, since there are only a finite number of equivalence classes y/E such that $P(y) > \lambda$.

Now, by the density of the real numbers we can constrain ξ to satisfy $\lambda < \xi < \gamma$. The result follows by the definition of an isolated cut-point, since ε can be chosen as the minimum of $\gamma - \xi$ and $\xi - \lambda$. $\square$

5 Probabilistic Hypertext Query Language

Using the approach presented in the previous section we extend HQL to be probabilistic and call the resulting query language PHQL. In PHQL a trail, T, probabilistically logically implies a query with cut-point λ iff T logically implies the query and $P(\mathrm{T}) > \lambda$.

Our approach is similar to that taken in [LEHM82] wherein models of logical formulae are Markov chains, and a sequence of states (i.e. a trail) satisfies a logical formula if the formula is true in the model "with probability one". Our approach differs from that in [LEHM82] in that logical PHQL formulae (i.e. trail queries) are satisfied with probability greater than a cut-point $\lambda \in [0,1)$. This is also in contrast to the approach taken in [FAGI90] which allows explicit reasoning about probabilities in logical formulae.

Definition 5.1 (Probabilistic logical implication) Let H = <I, R> be a Hypertext database, and T be a trail in R. Then T *probabilistically logically implies* a trail query, ϕ, with respect to H and cut-point $\lambda \in [0,1)$ (or simply T logically implies ϕ, if H and λ are understood from context), written T $\approx^{\lambda}$ ϕ (or simply T $\approx$ ϕ whenever λ is understood from context), if

$$\mathrm{T} \models \phi \text{ and } P(\mathrm{T}) > \lambda.$$

Definition 5.2 (Probabilistic trail query) A *probabilistic trail query* (or simply a probabilistic query or a query whenever no ambiguity arises) is a trail query, ϕ, viewed as a mapping from Hypertext databases to sets of trails such that its inputs are a Hypertext database H = <I, R> and a cut-point $\lambda \in [0,1)$, and its output $\phi(\mathrm{H}, \lambda)$ is defined by

$$\{\mathrm{T} \mid \text{ T is a trail in R and T } \approx^{\lambda} \phi\}.$$

A trail T in this set is called *an answer* of $\phi(\mathrm{H}, \lambda)$.

It follows that $|\phi(\mathrm{H}, \lambda_1)| \leq |\phi(\mathrm{H}, \lambda_2)|$, whenever $\lambda_2 \leq \lambda_1$. Thus, $|\phi(\mathrm{H}, \lambda)|$ is maximal when $\lambda = 0$. When $\lambda = 0$ we abbreviate $\phi(\mathrm{H}, 0)$ to $\phi(\mathrm{H})$.

Definition 5.3 (Probabilistic model checking) A Hypertext database H = <I, R> is a *probabilistic model* of a probabilistic trail query, ϕ, with cut-point $\lambda \in [0,1)$, if $\exists$T in R such that T is an answer of $\phi(\mathrm{H}, \lambda)$.

The *probabilistic model checking* problem is the problem of deciding whether a Hypertext database is a probabilistic model of a probabilistic trail query with a given cut-point.

The following proposition shows that in the general case the probabilistic model checking problem is as hard as model checking.

Proposition 5.1 The probabilistic model checking problem is NP-complete.

Proof. We firstly show that T $\approx^{\lambda}$ ϕ is in NP. In [LEVE94] we have shown that T $\models \phi$ is in NP. The result follows, since the condition $P(\mathrm{T}) > \lambda$ can easily be checked in polynomial time in #T and the sizes of the initial distribution, π, and the transition matrix, M, of $\mathcal{P}^{\lambda}$.

Secondly, T $\approx$ ϕ is NP-hard, since T $\models \phi$ reduces to T $\approx^{0}$ ϕ. then □

Although in general probabilistic model checking is intractable, in the case when R is 2total and $\lambda > 0$ we can utilise the result of Theorem 4.2, implying that $|\phi(\mathrm{H}, \lambda)| \leq |\mathcal{L}(\mathcal{P}^{\lambda})| \leq N^k$, where $k \in \omega$ satisfies $\alpha \cdot \beta^k < \lambda$. If R is not 2total, then we can apply the comment made after Theorem 4.3. As a heuristic users can tune the value of λ in order that the search space of trails be manageable, or alternatively specify k so that only trails whose count is at most k are returned.

6 Hypertext Random Automata

Hereafter we investigate the use of *random Turing machines* [LEEU56, RABI76, GILL77, JOHN90] with the aim of reducing the search time of finding a trail that logically implies a given trail query. Such use of random Turing machines is beneficial, as long as the probability that the returned trail indeed logically implies the trail query is high; normally high means that the probability is greater than one half. This approach is in contrast to the one taken in Section 4, since in this case we will be doing a totally random search without any use of precompiled navigation statistics.

Let H = <I, R> be a Hypertext database and ϕ be a trail query. By a result in [LEVE94], given a trail T in R we can check whether T $\models \phi$ in polynomial time in the sizes of T, ϕ and I. Thus when using a random Turing machine in order to solve the model checking problem, if H is *not* a model of ϕ (i.e. for all T in R, T $\not\models \phi$), then the random Turing machine will always answer correctly with a "no". On the other hand, if H is a model of ϕ, then we are interested in finding the probability that the random Turing machine will answer correctly with a "yes". This probability is determined by the proportion of trails in H that are models of ϕ. In particular, when the probability is greater than one half then this technique is viable.

We now define the notions of a random Hypertext automaton and the languages that are accepted by such an automaton. In the next section we will calculate a lower bound on the probability that such an automaton will randomly find a trail that logically implies a trail query with respect to a Hypertext database.

Definition 6.1 (Hypertext random automata) The *Hypertext Random Automaton* (HRA) representing a Hypertext database, H = <I, R>, is a HPA, $\mathcal{P}_H^0$, denoted by $\mathcal{R}_H$ (or simply $\mathcal{R}$ if H is understood from context), such that both the nonzero initial probabilities $p(s_i)$ and the nonzero transition probabilities $p(s_i, s_j)$ are uniformly distributed. A HRA $\mathcal{R}$ with *success* probability $\epsilon \in$ (0,1) is denoted by $\mathcal{R}^\epsilon$.

The success probability attached to a HRA represents the probability (or informally the *confidence*) that the user would like to have in the sense that a particular trail chosen at random logically implies the trail query the user is interested in.

Definition 6.2 (Languages accepted by HRA) A regular language $\mathcal{L}$ over $\mathcal{A}$ is accepted by a HRA, $\mathcal{R}^\epsilon$, if either $\mathcal{L} = \emptyset$ or $\exists k \in \omega$ which satisfies the following constraints:

1. $\forall i \in \{1, \ldots, n\}, P(w_i) > 0$ (i.e. $w_i \in \mathcal{L}(R_H)$), and
2. $\sum_{i=1}^{n} P(w_i) \geq \epsilon$, where $\mathcal{L} \cap \mathcal{A}^k = \{w_1, \ldots, w_n\}$.

The minimal natural number, k, which satisfies the above two constraints is called the *depth* of $\mathcal{L}$ with respect to $\mathcal{R}^\epsilon$ (or simply the depth of $\mathcal{L}$ whenever $\mathcal{R}^\epsilon$ is understood from context).

We observe that the depth of any language $\mathcal{L} \neq \emptyset$ accepted by $\mathcal{R}^\epsilon$ is greater than or equal to one, since by the definition of a language, $e \notin \mathcal{L}$. The next proposition follows from Definition 6.2 on recalling that by Definition 6.1 the initial and transition probabilities are uniformly distributed.

Proposition 6.1 If $\mathcal{L}$ is accepted by $\mathcal{R}^\epsilon$, then the inequality

$$\frac{\mid \mathcal{L} \cap \mathcal{A}^k \mid}{\mid \mathcal{L}(\mathcal{R}_H) \cap \mathcal{A}^k \mid} \geq \epsilon,$$

obtains. □

A nonempty regular language $\mathcal{L}$ accepted by a HRA $\mathcal{R}^\epsilon$ contains at least one answer to a trail query, say ϕ, that the user is interested in. Obviously, given $w \in \mathcal{A}^+$ we can check in polynomial time in the sizes of w, ϕ and I whether $w \in \mathcal{L}$, since $\mathcal{L}$ is regular. Thus it is always easy to verify whether a particular trail logically implies the trail query ϕ or not.

Next, let $\mathcal{R}^\epsilon$ be a HRA with state set Q $= \{s_1, \ldots, s_N\}$. In addition, let ν_0 denote the number of initial probabilities $p(s_i)$ such that $p(s_i) > 0$ and let ν_i denote the number of transition probabilities $p(s_i, s_j)$ such that s_j is adjacent to s_i. Finally, let adjacent(s_i) denote any sequence $< s_i^1, \ldots, s_i^{\nu_i} >$, where $\{s_i^1, \ldots, s_i^{\nu_i}\}$ is the set of all the states adjacent to s_i.

We assume that a function $random(n)$, where $n \in \omega$, is available and that it returns a natural number uniformly distributed between 1 and n; $random(n)$ corresponds to rolling an unbiased n-sided dice.

Now, suppose that $\mathcal{L}$ is accepted by $\mathcal{R}^\epsilon$ and that the depth of $\mathcal{L}$ is k. Then the following algorithm, designated random_trail$(\mathcal{R}^\epsilon, k)$, is a linear random Turing machine which returns a trail, T, of count k such that by Proposition 6.1 the probability that T $\in \mathcal{L}$ is greater than or equal to ϵ. Ultimately we accept the random trail, T, if T $\in \mathcal{L}$, otherwise we reject T.

Algorithm 1 (random_trail$(\mathcal{R}^\epsilon, k)$)

```
1.  begin
2.    $i := random(\nu_0)$;
3.    T[1] := $i$;
4.    cur_state := $\delta^{-1}$(i);
5.    for $j$ = 2 to $k$ do
6.      seq := adjacent(cur_state);
7.      $i := random(\nu_i)$;
8.      cur_state := the ith element of seq;
9.      $i := \delta$(cur_state);
10.     T[$j$] := $i$;
11.   end for
12.   #T = $k$;
13.   return T;
14. end.
```

A HRA $\mathcal{R}^\epsilon$ can also be viewed as inducing a *binomial* distribution [FELL68] with success probability ϵ. If repeated independent *trials* are allowed, then we may ask ourselves how many trials are necessary in order to have confidence $\epsilon \leq \sigma < 1$ that at least one trial results in a success. That is, we must find $m \in \omega$ satisfying the inequality

$$\sigma \geq 1 - (1 - \epsilon)^m.$$

7 Query Answering with Respect to Hypertext Random Automata

We now utilise the results of the previous section in order to discuss possible improvements to the time complexity of model checking.

Let H = <I, R> be a Hypertext database, ϕ be a trail query and let $\mathcal{R}$ be the HRA representing H. Furthermore, let $\mathcal{L} = \phi$(H) be the language induced by the output of the trail query ϕ with respect to H.

Assume the user has acquired the knowledge that $\mathcal{L}$ is accepted by $\mathcal{R}^\epsilon$ and that the depth of $\mathcal{L}$ with respect to $\mathcal{R}^\epsilon$ is k. In this case, the user can invoke Algorithm 1, and assign T to random_trail$(\mathcal{R}^\epsilon, k)$. If T $\models \phi$, then H is a model of ϕ, with T being an answer to ϕ(H), otherwise the conclusion that H is not a model of ϕ can be deduced with error probability of $(1-\epsilon)$. As mentioned at the end of the previous section we can reduce this probability to $(1-\epsilon)^m$ if we invoke random_trail($\mathcal{R}^\epsilon, k$) repetitively m times and return an answer to ϕ(H) if one out of the m invocations is successful.

The rest of this section is concerned with finding a lower bound on the probability of success, ϵ, when a given Hypertext database is a model of a given trail query. The following assumptions are made:

- S is the set of states of $\mathcal{R}$ and $N \geq 1$, with |S|= N.
- H = <I, R> is a Hypertext database and $\mathcal{R}$ is a HRA representing H.
- $k \geq 1$ is the depth of the regular languages accepted by $\mathcal{R}$ with success probability $\epsilon \in$ (0,1).
- ϕ is the trail query $c_1 \wedge \ldots \wedge c_n$, $n \geq 1$, and we assume without loss of generality that $\Phi = \{c_1, \ldots, c_n\}$.
- T is assigned to random_trail($\mathcal{R}^\epsilon, k$).

Theorem 7.1 Assume that R is complete and that $\forall q \in \{1,\ldots,n\}$ there exists exactly one state $s_j, j \in \{1,\ldots,N\}$, such that $c_q \in$ I(s_j). Then a lower bound on the number of trails that logically imply ϕ is given by

$$\begin{aligned} \mid \phi(H) \mid^m &= N^k - \binom{m}{1}(N-1)^k + \binom{m}{2}(N-2)^k - \binom{m}{3}(N-3)^k + \ldots + (-1)^m (N-m)^k \\ &= \sum_{i=0}^{m} (-1)^i \binom{m}{i} (N-i)^k, \qquad (1) \end{aligned}$$

where $m \geq 1$ is the number of states s_j such that I$(s_j) \neq \emptyset$. In addition, $m \leq n$, $m \leq N$ and $m \leq k$.

Proof. Firstly, $m \leq n$, since $\forall q \in \{1,\ldots,n\}$ there exists exactly one state $s_j, j \in \{1,\ldots,N\}$, such that $c_q \in$ I(s_j). Secondly, $m \leq N$, since |S|= N . Thirdly, if $k < m$, then all of the trails in N^k do *not* logically imply ϕ. Thus the maximal value of m is the minimum of the values n, N and k.

The term N^k gives the total number of trails in R whose count is k, assuming that R is complete. The term $\binom{m}{1}(N-1)^k = m(N-1)^k$ gives the number of trails that do not logically imply ϕ, which do not contain at least one of the states s_j such that I$(s_j) \neq \emptyset$. In general, the term $\binom{m}{i}(N-i)^k$ gives the number of trails that do not logically imply ϕ, which do not contain at least i of the states s_j such that I$(s_j) \neq \emptyset$. The result now follows from the principle of *inclusion* and *exclusion* [GRIM89]. $\square$

Definition 7.1 (The minimal number of trails that logically imply ϕ) The minimal value of $\mid \phi(H) \mid^m$ is denoted by min($\mid \phi(H) \mid^m$).

We conjecture that the value min($\mid \phi(H) \mid^m$) is obtained when m is maximal, that is when m is the minimum of the values n, N and k. Using this conjecture we obtain the next corollary.

Corollary 7.2 *A lower bound on the probability of success, ϵ, of a HRA, $\mathcal{R}$, denoted by $P(\phi)$, on assuming that the reachability relation is complete, is given by*

$$P(\phi) = \frac{min(\mid \phi(H) \mid^m)}{N^k}. \quad \square$$

We will now assume that the reachability relation R of the Hypertext database H is μ-regular, $\mu \in \{1, \ldots, N\}$. When μ = N, then R is complete. The following definition is used in Theorem 7.3.

Definition 7.2 (Proper subtraction) The *proper subtraction* of two natural numbers, a and b, denoted by $a \ominus b$, is given by

$$a \ominus b = \begin{cases} a - b & \text{if} \quad a \geq b \\ 0 & \text{if} \quad a < b \end{cases}$$

The following result generalises Theorem 7.1 to μ-regular reachability relations.

Theorem 7.3 The number of trails, $\mid \phi(H) \mid^m$, that logically imply ϕ when the reachability relation R is μ-regular is bounded from above and from below as follows:

$$\sum_{i=0}^{m} (-1)^i \binom{m}{i} (N-i)(\mu \ominus (i \ominus (N-\mu)))^{k-1} \leq \mid \phi(H) \mid^m \leq \sum_{i=0}^{m} (-1)^i \binom{m}{i} (N-i)(\mu \ominus i)^{k-1}. \quad (2)$$

Proof. Firstly, consider the lower bound of $\mid \phi(H) \mid^m$ given by

$$\sum_{i=0}^{m} (-1)^i \binom{m}{i} (N-i)(\mu \ominus (i \ominus (N-\mu)))^{k-1}.$$

Let $s_j \in$ S be any state of $\mathcal{R}$. The expression $(\mu \ominus (i \ominus (N - \mu)))$ assumes that out of the μ states adjacent to s_j, a maximum possible number were chosen to be contained in a trail, say T, in order that T $\not\models \phi$ holds. That is, a minimum out of the i chosen states is assumed to be adjacent to s_j, and thus a maximal number of trails logically implies ϕ.

Secondly, consider the upper bound of $\mid \phi(H) \mid^m$ given by

$$\sum_{i=0}^{m} (-1)^i \binom{m}{i} (N-i)(\mu \ominus i)^{k-1}.$$

Let $s_j \in$ S be any state of $\mathcal{R}$. The expression $(\mu \ominus i)$ assumes that out of the μ states adjacent to s_j, a minimum possible number were chosen to be contained in a trail, say T, in order that T $\not\models \phi$ holds. That is, a maximum out of the i chosen states is assumed to be adjacent to s_j, and thus a minimal number of trails logically implies ϕ.

The result now follows by the same argument that was used in the proof of Theorem 7.1, since $(\mu \ominus i) \leq (\mu \ominus (i \ominus (N - \mu)))$. $\square$

In the special case when $\mu = N$, the lower and upper bounds given in (2) of Theorem 7.3 coincide with the lower bound given in (1) of Theorem 7.1. The next corollary which generalises Corollary 7.2 to μ-regular reachability relations is evident.

Corollary 7.4 *A lower bound on the probability of success, ϵ, of a HRA, $\mathcal{R}$, denoted by $P(\phi)$, on assuming that the reachability relation is μ-regular, is given by*

$$P(\phi) = \frac{min(\mid \phi(H) \mid^m)}{N\mu^{k-1}}. \quad \square$$

In the general case when the reachability relation may not be μ-regular, we may be able to approximate $\mid \phi(H) \mid^m$ by considering the degree of regularity to be the average number of states adjacent to any other state in $\mathcal{R}$.

Although the Hypertext random automata approach is attractive, it is only viable if the probability of success, ϵ, is not overly small. Obviously, if the number of states, N, is very large then $P(\phi)$ may be very small, thus rendering this approach impractical. In many situations we may be able to utilise additional knowledge that will allow us to obtain a better approximation of ϵ, which is higher than the lower bound $P(\phi)$. For example, in the statement of Theorem 7.1 we have assumed that $\forall q \in \{1, \ldots, n\}$ there exists exactly one state $s_j, j \in \{1, \ldots, N\}$, such that $c_q \in \mathrm{I}(s_j)$. In practice we would expect that there will be several states s_j such that $c_q \in \mathrm{I}(s_j)$. This information will give us a more accurate estimate of ϵ, which is greater than the lower bound $P(\phi)$. It is an open problem how best to incorporate such additional knowledge into the calculation of $\mid \phi(H) \mid^m$.

8 Concluding Remarks

In a previous paper the navigation problem in Hypertext was shown to be intractable in the general case [LEVE94]. Herein we have investigated two probabilistic approaches in an attempt to solve the navigation problem in Hypertext more efficiently.

In the first approach we have utilised the theory of probabilistic automata by defining a HPA with cut-point $\lambda \in [0,1)$, which uses precompiled statistics in order to form the initial and transition probabilities. The HPA approach reduces the search space when answering trail queries and by Theorem 4.2, in the special case when the reachability relation is 2total, the search space is always finite. We extended HQL to PHQL in order to take HPA into account and, although by Proposition 5.1 probabilistic model checking is NP-complete in the general case, we can make use of Theorem 4.3 in order to tune λ so that the search space of trails is manageable.

In the second approach we have utilised the theory of random Turing machines defining a HRA with success probability ϵ; the HRA uses the random algorithm, random_trail, to return a trail. The HRA approach reduces the search time when answering trail queries, since in order to find a random trail, only k rolls of an unbiased dice are required, where k is the count of the random trail returned. In Corollaries 7.2 and 7.4 we gave a lower bound on the probability of success when using HRA, with respect to complete and μ-regular reachability relations, respectively, in order to answer trail queries.

The investigation carried out in this paper is preliminary and both the approaches need to be further developed. It would be interesting to combine the two approaches in order to gain the benefits of both of them. We are currently investigating heuristic algorithms based on the approaches presented herein, which utilise knowledge about the relevancy of pages, in order to navigate through a Hypertext database, in a best-first manner.

References

[BUCK90] F. Buckley and F. Harary, *Distance in Graphs.* Redwood City, Ca., Addison-Wesley, 1990.

[BUSH45] V. Bush, As we may think. *Atlantic Monthly*, **76**, (1945), 101-108.

[DAVE90] B.A. Davey and H.A. Priestly, *Introduction to Lattices and Order.* Cambridge, U.K., Cambridge University Press, 1990.

[EMER90] E.A. Emerson, Temporal and modal logic. In: *Handbook of Theoretical Computer Science, Volume B*, Ed. J. van Leeuwen, Amsterdam, Elsevier Science Publishers B.V., 1990, Chapter 16, pp. 997-1072.

[FAGI90] R. Fagin, J.Y. Halpern and N. Megiddo, A logic for reasoning about probabilities. *Information and Computation*, **87**, (1990), 78-128.

[FELL68] W. Feller, *An Introduction to Probability Theory and its Applications, Third Edition.* NY, John Wiley and Sons, 1968.

[FRIS92] M.F. Frisse and S.B. Cousins, Models for Hypertext. *Journal of the American Society for Information Science*, **43**, (1992), 182-192.

[GARE79] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness.* New York, Freeman, 1979.

[GILL77] J. Gill, Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, **6**, (1977), 675-695.

[GRIM89] R.P. Grimaldi, *Discrete and Combinatorial Mathematics, Second Edition.* Reading, Ma., Addison-Wesley, 1989.

[HOPC79] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation.* Reading, Ma., Addison-Wesley, 1979.

[JOHN90] D.S. Johnson, A catalog of complexity classes. In: *Handbook of Theoretical Computer Science, Volume A*, Ed. J. van Leeuwen, Amsterdam, Elsevier Science Publishers B.V., 1990, Chapter 2, pp. 67-161.

[KEME60] J.G. Kemeny and J.L. Snell, *Finite Markov Chains.* Princeton, D. van Nostrand, 1960.

[LEEU56] K. de Leeuw, E.F. Moore, C.E. Shannon and N. Shapiro, Computability by probabilistic Turing machines. In: *Automata Studies*, Eds. C.E. Shannon and J. McCarthy, Princeton, New Jersey, Princeton University Press, 1956, pp. 183-212.

[LEHM82] D. Lehman and S. Shelah, Reasoning with time and chance. *Information and Control*, **53**, (1982), 165-198.

[LEVE94] M. Levene and G. Loizou, Navigation in Hypertext is easy only sometimes. Research Note RN/94/43, Department of Computer Science, University College London.

[MCNA71] R. McNaughton and S. Pappert, *Counter-Free Automata.* Cambridge, Ma., MIT Press, 1971.

[NASU68] M. Nasu and N. Honda, Fuzzy events realized by finite probabilistic automata. *Information and Control*, **12**, (1968), 284-303.

[NIEL90] J. Nielsen, *Hypertext and Hypermedia.* San Diego, Academic Press, 1990.

[PAZ71] A. Paz, *Introduction to Probabilistic Automata.* New York, Academic Press, 1971.

[PERR90] D. Perrin, Finite automata. In: *Handbook of Theoretical Computer Science, Volume B*, Ed. J. van Leeuwen, Amsterdam, Elsevier Science Publishers B.V., 1990, Chapter 1, pp. 1-57.

[RABI63] M.O. Rabin, Probabilistic automata, *Information and Control*, **6**, (1963), 230-245.

[RABI76] M.O. Rabin, Probabilistic algorithms. In: *Algorithms and Complexity, Recent Results and New Directions*, Ed. J.F. Traub, New York, Academic Press, 1976, pp. 21-39.

[RIVL94] E. Rivlin, R. Botafogo and B. Shneiderman, Navigating in hyperspace: Designing a structure-based toolbox. *Communications of the ACM*, **37**, (1994), 87-96.

[STEA85] R.E. Stearns and H.B. Hunt, On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, **14**, (1985), 598-611.

[STOT89] P.D. Stotts and R. Furata, Petri-net-based Hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems*, **7**, (1999), 3-29.

[STOT92] P.D. Stotts, R. Furata and J.C. Ruiz, Hyperdocuments as automata: Trace-based browsing property verification. In: *Proceedings of the ACM Conference on Hypertext, ECHT'92*, pp. 272-281, 1992.

[ULLM88] J.D. Ullman, *Principles of Database and Knowledge-Base Systems, Volume I.* Rockville, Md., Computer Science Press, 1988.