1-1-1999

Dynamical system representation of open address hash functions

Chaouki T. Abdallah

Gregory L. Heileman

Bernard M. Moret

Bradley J. Smith

Dynamical System Representation of Open Address Hash Functions

Gregory L. Heileman* Chaouki T. Abdallah* Bernard M. E. Moret† Bradley J. Smith*

Abstract

This paper demonstrates how a broad collection of hash function families can be expressed as dynamical systems. We then show that this representation can be useful for analysis. In particular, we provide an analysis which proves that a widely-used family of double hash functions will transform any initial distribution of keys into the uniform distribution over the table space.

1 Introduction.

Number theory and probabilistic analyses have typically been used to justify the use of various probing strategies for open address hashing. For example, Guibas and Szemeredi [1], and later Lueker and Molodowitch [3], have shown that double hashing offers a good approximation to uniform hashing if the hash function is assumed to select table entries with equal probability. In this paper we use an analysis based on nonlinear dynamical systems theory to provide a similar result, without making any probabilistic assumptions about the data distribution. The key to our approach is the ability to express hash functions as dynamical systems.

We consider four important families of hash functions: linear probing, quadratic hashing, linear double hashing, and exponential double hashing. We assume that each of these hashing strategies implements the *Find*(k), *Insert*(x), and *Delete*(k) operations, where k is a key drawn from a totally ordered universe U, and x is a data element that has a key associated with it. We assume that these operations are implemented on a table of size m, and that a table index is computed from the key using a ordinary hash function h, that performs the mapping $h : U \to \mathbb{Z}_m$, where $\mathbb{Z}_m$ denotes the integers modulo m. An open address hash function H makes use of one or more ordinary hash functions to successively probe the table space in the event of collisions.

2 Hashing Functions as Dynamical Systems.

The family of linear hash functions can be expressed as

$$H_{\mathcal{L}_n} = (h(k) + cn) \bmod m, \tag{2.1}$$

where h is an ordinary hash function, c is a constant, and $n = 0, 1, \ldots$ is the probe number. This technique is known as linear hashing because the argument of the modulus operator is linearly dependent on the probe number. In order to construct an equivalent iterator for (2.1), it must be rewritten as a recurrence relation so that it has explicit dependence on previous values of the iteration sequence. Since

$$H_{\mathcal{L}_{n+1}} = (h(k) + cn + c) \bmod m,$$

and using the fact that for $a, b, m \in \mathbb{R}$,

$$(a + b) \bmod m = (a \bmod m + b \bmod m) \bmod m,$$

we may rewrite (2.1) as the following linear time-invariant first-order iterator

$$\begin{aligned} H_{\mathcal{L}_{n+1}} &= (H_{\mathcal{L}_n} + c) \bmod m \\ H_{\mathcal{L}_0} &= h(k). \end{aligned}$$

An equivalent description in one-dimensional state space is given by

$$x_{n+1} = (x_n + c) \bmod m,$$

where $x_0 = h(k)$, and $H_{\mathcal{L}_n} = x_n$ is the output equation used to iterate over the table space. Note that the dependence on k is specified in the initial condition.

Quadratic hashing is an extension of linear probing that makes the probe sequence nonlinearly dependent on the probe number n. For any ordinary hash function h, the family of quadratic hash functions is given by

$$H_{\mathcal{Q}_n} = (h(k) + c_1 n + c_2 n^2) \bmod m,$$

where c_1 and c_2 are constants. Using the technique applied previously, we obtain the following iterator

$$\begin{aligned} H_{\mathcal{Q}_{n+1}} &= (H_{\mathcal{Q}_n} + c_1 + c_2(2n + 1)) \bmod m, \\ H_{\mathcal{Q}_0} &= h(k). \end{aligned}$$

This can be rewritten as the following set of coupled linear (modulo m) time-invariant first-order iterators in two-dimensional state space

$$\begin{aligned} x_{n+1} &= (x_n + 1) \bmod m, \\ y_{n+1} &= (y_n + 2c_2 x_n + c_2 + c_1) \bmod m, \end{aligned}$$

*Department of Electrical and Computer Engineering, †Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, USA.

where $x_0 = 0$, $y_0 = h(k)$, and $H_{\mathcal{Q}_n} = y_n$ is the output equation used to iterate over the table space.

Given two ordinary hash functions g and h, the family $H_{\mathcal{LD}}$ of linear double hash functions is given by

$$H_{\mathcal{LD}_n} = (g(k) + nh(k)) \bmod m, \tag{2.2}$$

where the initial probe $H_{\mathcal{LD}_0} = g(k)$. Thus the probe sequence depends on k through both g and h, and is linear in $g(k)$ and $h(k)$. It can be shown that (2.2) is equivalent to the following iterator

$$\begin{aligned} H_{\mathcal{LD}_{n+1}} &= (H_{\mathcal{LD}_n} + h(k)) \bmod m, \\ H_{\mathcal{LD}_0} &= g(k), \end{aligned}$$

which can be rewritten as a set of coupled linear (modulo m) time-invariant first-order difference equations in two-dimensional state space:

$$x_{n+1} = x_n, \tag{2.3}$$

$$y_{n+1} = (x_n + y_n) \bmod m, \tag{2.4}$$

where $x_0 = h(k)$, $y_0 = g(k)$, and $H_{\mathcal{LD}_n} = y_n$ is the output equation used to iterate over the table space. A widely used member of $H_{\mathcal{LD}}$, proposed by Knuth [2], has $g(k) = k \bmod m$ and $h(k) = k \bmod (m-2)$, where both m and $m-2$ are prime.

We propose a new family of double hash functions $H_{\mathcal{E}}$, which we call exponential double hashing, that uses two ordinary hash functions g and h to compute a probe sequence according to

$$H_{\mathcal{E}_n} = (g(k) + h(k)^n) \bmod m. \tag{2.5}$$

The exponentiation does not have to be explicitly implemented, but can be computed via successive multiplications during the probing process. For this reason, the number of mathematical operations needed to implement (2.5) is identical to the number needed to implement (2.2). To obtain an iterator for the family $H_{\mathcal{E}}$ described in (2.5), we write

$$\begin{aligned} H_{\mathcal{E}_{n+1}} &= [h(k)h(k)^n + g(k)] \bmod m \\ &= [((h(k)^n + g(k)) \bmod m \cdot h(k) \bmod m) \\ &\qquad \bmod m + ((1-h(k))\, g(k)) \bmod m] \bmod m, \end{aligned}$$

and obtain

$$\begin{aligned} H_{\mathcal{E}_{n+1}} &= (H_{\mathcal{E}_n} h(k) + (1-h(k))g(k)) \bmod m, \\ H_{\mathcal{E}_0} &= g(k) + 1. \end{aligned}$$

This is a first-order time-varying nonlinear system. An equivalent system of first-order time-invariant iterators can be obtained by using a three-dimensional state space:

$$\begin{aligned} x_{n+1} &= x_n, \\ y_{n+1} &= y_n, \\ z_{n+1} &= (y_n z_n + x_n (1 - y_n)) \bmod m, \end{aligned}$$

where $x_0 = g(k)$, $y_0 = h(k)$, $z_0 = x_0 + 1$, and $H_{\mathcal{E}_n} = z_n$ is the output equation used to iterate over the table space. A particular member of $H_{\mathcal{E}}$, recently introduced by Smith et al. [4], has $g(k) = k \bmod m$, and $h(k) = k \bmod (m-2)$, where $m = 2p+1$ is selected so that both m and p are prime. Smith et. al [4] showed experimentally that on average this member of $H_{\mathcal{E}}$ tends to outperform any member of $H_{\mathcal{LD}}$. We suggest here that this may be due to the fact that $H_{\mathcal{E}}$ hash functions operates in a more complicated state space than $H_{\mathcal{LD}}$ hash functions.

3 Analysis.

We now consider an analysis that makes use of the dynamical system representation of $H_{\mathcal{LD}}$. This analysis demonstrates the potential usefulness of representing hash functions as dynamical systems. Given (2.3) and (2.4), the output equation can be rewritten in terms of the initial probes as $y_n = y_0 + nx_0 \pmod m$. Let us denote the joint density function for the initial probes using f_{x_0,y_0}, and without loss of generality (by appropriate scaling) we may assume $m = 2\pi$. In this case the characteristic function of the output equation is given by

$$\begin{aligned} \varphi_{y_n}(t) &= \mathbb{E}e^{ity_n} = \mathbb{E}e^{it(y_0+nx_0)} \\ &= \int_0^{2\pi} e^{it(v+nu)} f_{x_0,y_0}(u,v)\, du\, dv. \end{aligned}$$

By the Riemann-Lebesgue Lemma, it follows that for $t \neq 0$, $\varphi_{y_n}(t) \longrightarrow 0$ weakly as $n \longrightarrow \infty$. Thus, since f_{x_0,y_0} is a density function, for n sufficiently large $\varphi_{y_n}(t) = 1$ when $k = 0$, and $\varphi_{y_n}(t) = 0$ when $k \neq 0$, which is in fact the characteristic function for the uniform distribution on the circle. Since the characteristic function uniquely determines the density function for a given random variable, this demonstrates that for appropriately chosen m, and after a sufficient number of probes, hash functions in the linear double hashing family will transform any initial density into the uniform distribution over the table space. $\diamondsuit$

References

[1] L. Guibas and E. Szemeredi, *The analysis of double hashing*, J. Comput. and Sys. Sci., 16 (1978), pp. 226–274.

[2] D. E. Knuth, *The Art of Computer Programming, volume 3, Searching and Sorting*. Addison-Wesley, Reading, MA, 1973.

[3] G. Lueker and M. Molodowitch, *More analysis of double hashing*, in ACM STOC, 20 (1988), pp. 354–359.

[4] B. J. Smith, G. L. Heileman, and C. T. Abdallah, *The exponential hash function*, ACM J. Exper. Alg., 2(3) (1997), www.jea.acm.org/1997/SmithExponential/.