Towards a Theory of Information Preservation

James Cheney[1], Carl Lagoze[1], Peter Botticelli[2]

[1] Computer Science Department
Cornell University
Ithaca, NY 14850
{jcheney,lagoze}@cs.cornell.edu
[2] Cornell University Library
Cornell University
Ithaca, NY 14850
pkb4@cornell.edu

Abstract. Digital preservation is a pressing challenge to the library community. In this paper, we describe the initial results of our efforts towards understanding digital (as well as traditional) preservation problems from first principles. Our approach is to use the language of mathematics to formalize the concepts that are relevant to preservation. Our theory of *preservation spaces* draws upon ideas from logic and programming language semantics to describe the relationship between concrete objects and their information contents. We also draw on game theory to show how objects change over time as a result of uncontrollable environment effects and directed preservation actions. In the second half of this paper, we show how to use the mathematics of universal algebra as a language for objects whose information content depends on many components. We use this language to describe both migration and emulation strategies for digital preservation.

1 Introduction

Preservation is essential to the mission of libraries and archives. For print materials, we have well-established techniques for protecting and conserving physical artifacts and for reformatting content onto highly durable media (e.g., microfilm). In recent years, however, the growth of digital content has made preservation a critical and vexing problem, one that violates many of the assumptions that govern established preservation programs. Since we do not yet have proven methods or strategies for preserving "born digital" files with no print analogues, there is an urgent need to explore new approaches and to closely examine all potential solutions.

Many in the archives community have begun to tackle digital preservation by returning to first principles, by first asking what essential attributes of digital records ought to be preserved, before considering the technical possibilities for preserving digital objects. It is often argued that digital preservation depends on our ability to preserve not a set of file structures so much as information of enduring value, or the content contained in digital files [16, 8]. In the digital library community, Clifford Lynch has developed this idea by arguing that digital preservation should focus on a "canonicalization" of the content rather than the bits themselves [11]. In essence, Lynch and others have called for the development of a theoretical approach to managing information in an abstract form that can be isolated from particular file formats and display technologies[1].

Nonetheless, many archivists are justifiably skeptical on the question of whether we can preserve digital content in an unstable and rapidly evolving computing environment. Indeed, in the print world, the concreteness and the relative stability of artifacts has enabled us to effectively preserve information by simply preserving its physical container. In other words, the information contained in traditional document formats (e.g. paper with ink, vinyl with groove patterns) is tightly coupled with the physical characteristics of the medium[2]. This is also true of digital formats, which represent the interaction between data (stored as bits) and a finite set of hardware and software artifacts[3]. However, the complexity of the "physical characteristics" of the digital medium—hardware plus software plus data—makes it potentially very costly to preserve information by preserving that complete set of characteristics. Thus, from a theoretical perspective, we argue that practical strategies for preservation depend on our ability to understand and manage the changing relationship between information—the canonical values we ultimately wish to preserve—and the physical environment in which documents are produced and stored.

This paper describes our initial steps toward the development of a mathematical approach to preserving information, one that we hope will eventually enable us to model the behavior of both print and digital documents as information carriers. Our main goal at this stage is to demonstrate the utility of using mathematical models to analyze the outcome of preservation strategies in practical situations. Our modeling approach draws on a number of theoretical bases, including game theory [9, 14], programming language semantics [13], and universal algebra [18].

In applying modeling techniques to the problem of preserving information, we first had to devise a formal way to represent information as a set of relative,

[1] Archivists in Australia, in particular, and also Canada have made considerable progress in exploring the relationship between the record, as a physical or digital object, and the web of contextual information needed to make sense of records—and which in many instances cannot be contained in the artifact itself. See [12, 1, 17]. For a general discussion of archival approaches to preserving the authenticity of digital records, see [6].

[2] This view is strongly reflected in the practice of archival diplomatics, which analyzes the physical structure of documents to determine their authenticity. See [5]. Duranti is also the leading contemporary advocate for a "custodial" approach to digital archives, as a way to maintain control of records by physically controlling the bits used to encode them.

[3] We should note that Jeff Rothenberg has been influential in arguing for emulation as a preservation strategy, though it remains to be seen whether the functionality of computing environments can be effectively preserved over long periods of time. See [15]

and not absolute, values. We thus developed the notion of an *interpretation*, which we define as a mathematical function that describes the complex relationship between the physical manifestation of a document and its accompanying information content.

People naturally form an interpretation whenever they use a document. In most cases, interpretations are expressed only in tacit, informal ways, and yet they have important consequences for preservation. A web page, for example, can have several interpretations, depending on how we use them. A single page can be interpreted as an HTML source text, as a page image displayed on a monitor, or as a browsing experience mediated by a variety of applications and data sources. Different preservation strategies will obviously be needed to preserve the information represented by these different interpretations. We also need to account for the fact that individual interpretations will differ in granularity; i.e., a page-image interpretation might specify a color model to be used for rendering an exact shade of red, while another interpretation might allow for any shade of "red" to be used. In this case, there are two different strategies, with significant differences in cost, for preserving a single web page; the choice of strategy depends upon which interpretation is more appropriate. Thus, our approach to preservation demands that we first transform our otherwise tacit observations into some kind of formal statement about the relationship between physical objects (including software) and the information values we wish to preserve.

By allowing for variable interpretations, our goal is to develop mathematical models that will enable us to test the broadest possible range of preservation strategies for both physical and digital objects. To model the effects of particular strategies, we have begun to develop a mathematical structure that we call a *preservation space*. In broad terms, this structure consists of:

1. Physical objects that can represent information.
2. Information content representable by physical objects.
3. Interpretations that link physical objects to information content.
4. Environmental factors that may cause changes to physical objects (and, indirectly, the information they represent) over time.
5. Actions that preservation agents (e.g., individuals, libraries, archives, museums) can take in response to environmental factors.
6. Effects of the interaction of environment and agent actions on physical objects.

Our goal is to use the concept of a preservation space to understand and mathematically model:

- Preservation policies that formally express what information (what interpretation of the physical space) should be preserved over time.
- Preservation plans that define a program-like series of steps that absolutely or probabilistically accomplish a preservation policy.
- Preservation costs that determine the feasibility of policies and plans.

Our theoretical approach to digital preservation co-exists with a number of ongoing research projects. For instance, researchers at the San Diego Supercomputer Center are seeking to design a digital archives as a stable technical environment in which documents can be preserved and used for long periods of time [10]. Work at Stanford focuses on the preservation of documents as bits, and is examining ways to preserve information in spite of hardware failures [3, 4, 2]. The CAMiLEON project is investigating the outcomes of different emulation strategies in preserving digital objects [7]. All of these approaches have distinct merits, and our work is not meant to stand in contrast to them. Rather, our work is intended to provide a framework for understanding these various approaches and evaluating their effectiveness in addressing particular preservation issues.

In the remainder of this paper, we explain our initial theory-building efforts. We begin by developing the basic elements of the preservation space. Next, we use these building blocks to analyze first the preservation of single, self-contained objects, then collections of objects, and finally complex, context-dependent objects. We close by describing some possible future paths for this work.

2 Preservation spaces

In this section we will define the *preservation spaces* that were introduced in the previous section. A preservation space consists of two basic components: a *semantics*, or a way of interpreting objects as information, and a *dynamics*, that is, a description of how objects change over time. Using this framework, we define information preservation as preservation of information content subject to some interpretation, despite changes in representation. We then develop the notions of *plans* and *policies* for preservation.

2.1 Semantics

Our theory distinguishes between an *object state space* $\mathcal{S}$ and an *information content space* $\mathcal{C}$. The object state space consists of all the possible states that objects representing information might assume, and the information space contains the information content representable in the object state space. A partial function $I : \mathcal{S} \rightharpoonup \mathcal{C}$ mapping states to their information content is called an *interpretation*. The structure $(\mathcal{S}, \mathcal{C}, I)$ is a *semantics*.

For example, the object states might be eight-bit character sequences and the information content might be English text. One interpretation is the standard ASCII code, while others include EBCDIC and UNICODE. Another somewhat fuzzier example consists of everyday physical objects whose information content can consist of perceivable properties such as color, shape, size, and so on. Interpretations in this example include "approximate color" (group objects together as being "red", "blue", "green", etc.) and "exact color" (the exact visible light component of the color of an object).

Any interpretation I induces a *partial equivalence relation* $\equiv_I$ over $\mathcal{S}$, namely $x \equiv_I y \iff I(x) = I(y)$. That is to say, given an interpretation I, we can

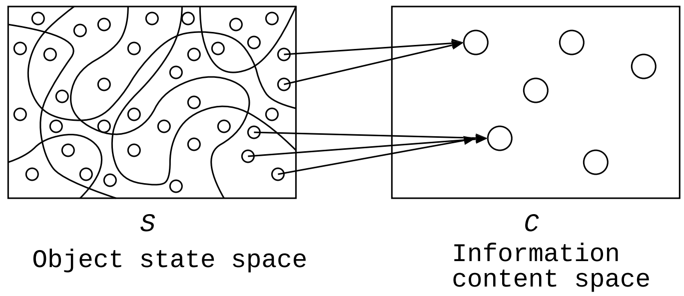

Fig. 1. Object state and information content spaces and interpretation

partition the state space into classes that represent the same information. Figure 1 diagrams a simple object state space and information space. The arrows show part of an interpretation $I : \mathcal{S} \rightharpoonup \mathcal{C}$. The object state space $\mathcal{S}$ is divided into cells of $\equiv_I$-equivalence classes.

The choice of semantics depends on the problem domain. In the case of Web pages, the object state space is the set of all ASCII text files. The corresponding forms of information content include "HTML source", "page image", and "total page interaction experience". The correspondences between the object states and these information content forms are interpretations. For example, two text files are equivalent from the "HTML source" perspective only if they are literally the same text document. However, two source files are equivalent from the "page image" perspective if they render the same (but they may have different markup). Finally, equivalence from the "total page interaction experience" perspective means pages are indistinguishable based on the user's browsing experience. These interpretations are partial functions, since some text files are not legal HTML and thus have no information content in any of the above senses.

Our use of the word *semantics* to describe the relationship between objects and their content is close to the that in programming language theory. A program's semantics is an abstract description of its behavior when run (such as the value or function it computes), and a language's semantics is a way of calculating the semantics of programs in the language. Two programs that have the same semantics are considered to be equivalent, even though they might be different in other ways, for example, size or running time. In programming language theory, the choice of semantics implicitly determines what properties of programs (such as running time, storage usage, and long-term behavior) are considered important. Programming language semantics are a special case of preservation semantics, but an important one, especially for digital preservation problems in which the objects of interest include programs.

2.2 Dynamics

Objects are physical entities and so are subject to change over time. In our theory, changes result either from actions taken by the preservation agent, or from interactions with the "environment", that is, circumstances not under the agent's control. We will model this state of affairs as a two-player game, or a graph whose vertices are game states and edges are moves made by the preservation agent or by the environment. In this view, we want to know whether there is some strategy by which the preservation agent can "win" playing against the environment, where winning means preserving the information.

For example, in a preservation space model for a digital preservation problem, the environmental factors would include both media decay, in which bits on magnetic or optical media are no longer readable, and technological obsolescence, in which readers for the media no longer exist. The actions available to a preservation agent would include making duplicate copies of media, refreshing data onto more long-lasting media, migrating data from obsolete media formats to state-of-the-art ones, as well as other, domain-dependent actions. However, in a general preservation space, there are no *a priori* assumptions on the nature of the environment effects and agent actions.

Formally, let $\mathcal{E}$ be a set of *environment moves*, and let $\mathcal{A}$ be a set of *agent actions*. Each object state $s \in \mathcal{S}$ is a game state and the rules of the game are described by $\mathcal{E}$ or $\mathcal{A}$-labeled edges $s \to_e t$, $s \to_a t$ between states. An edge $s \to_a t$ means "in state s, action a results in state t", and similarly for $s \to_e t$. The game is deterministic, so from any state s, there can be at most one edge $s \to_a t$ labeled with a given move a (and similarly for environment moves). The environment and agent take turns making moves; the environment makes its moves randomly, while the agent can decide what moves to make. This is a simplification of a more general form of game in which the players can move simultaneously; the alternating-move game seems adequate for most purposes.

We regard one turn of the game as one time step. Thus, in each time step the environment has some effect on the state and then the agent takes some action that may also change the state. We define a *game sequence* to be a sequence $\omega = s_0 e_1 a_1 s_1 \cdots e_k a_k s_k$ where s_0 is the initial state of the game, each e_i and a_i is the ith move made by the environment or agent, and s_{i+1} is the state after the $i+1$th turn (that is, $s_i \to_{e_i} t \to_{a_i} s_{i+1}$ for some t).

So far we have been using the term "object" informally. In formalizing the concept of object, the central issue is how to model *identity*, that is, whatever it is that makes us believe that *different states* are instantiations of the *same object* at *different times*. It is tempting to regard an object as a sequence $s_1 \dots s_k$ of object states assumed over time. In this approach, two states in this sequence are aspects of the same object. However, this definition is too broad. It requires us to consider too many possible object behaviors, including those which violate the rules of the game. We wish to exclude those state sequences that can never be actual sequences of game states.

Thus, we could instead define an object to be a state sequence $s_1 \dots s_k$ for which a game sequence $s_0 e_1 a_1 s_1 \cdots e_k a_k s_k$ that runs through its states exists. It would also be adequate to define an object simply as a game sequence. Defining objects as game sequences has two advantages over the state-sequence approach. First, it is notationally simpler. Second, it makes analyzing specific objects easier since we need only consider one game sequence at a time, not all game sequences generating a state sequence. It is this approach that we prefer: formally, an *object* is a game sequence ω.

Games are usually characterized as either competitive or cooperative. We might ask which best describes a preservation game. In a competitive game, the environment should always make moves that threaten the integrity of the information, since the preservation agent's goal is to preserve it. In a cooperative game, the environment and agent would share the goal of preservation, and so a trivial winning strategy would be for both to do nothing. Neither of these descriptions matches our intuition about the environment. The environment can be either harmful or helpful, but is not motivated by malice, benevolence, or indeed, in our view, any will at all. Thus, it seems more appropriate to think of the environment as an indifferent player whose moves are chosen according to some unpredictable process that we can describe using some probability distribution P. Nevertheless, cooperation and competition are likely to be useful concepts in situations with several independent preservation agents.

To summarize, a *preservation space* is a structure with the following components:

Semantics	$\mathcal{S}$	object state space
	$\mathcal{C}$	information content space
	$I_1, I_2, \ldots : \mathcal{S} \rightharpoonup \mathcal{C}$	interpretations
Dynamics	$\mathcal{E}$	environment effects
	$\mathcal{A}$	agent actions
	$s \rightarrow_e t, s \rightarrow_a t$	results of actions and effects
	P	probability distribution of environment effects

2.3 Preservation, Plans, and Policies

In this section we employ preservation spaces to analyze relatively simple preservation scenarios. We begin by modeling preservation of single objects; for example, books, web pages, and the like. In reality, information brokers such as libraries and archives are responsible for collections of similar objects or for interdependent objects. Later in the section, we therefore extend our modeling principles to the collection level. This provides the tools for analyzing more complex preservation scenarios in the following section.

Single objects We say that the information of an object $\omega = s_0 e_1 a_1 \cdots s_k$ is *preserved* if the object's state always represents the same information over its lifetime. Formally, preserving ω with respect to an interpretation I means that for each $t, 0 \leq t \leq k$, we have $I(s_t) = I(s_0)$. Schematically, preservation means that the state of ω stays within the same $\equiv_I$-equivalence class. We can visualize this by thinking of the states of ω as a path wandering through $\mathcal{S}$ but never crossing a $\equiv_I$-border (see Figure 2).

The simplest kind of preservation goal, or *policy*, is to preserve an object for n time steps with respect to an interpretation I_i. This policy is written

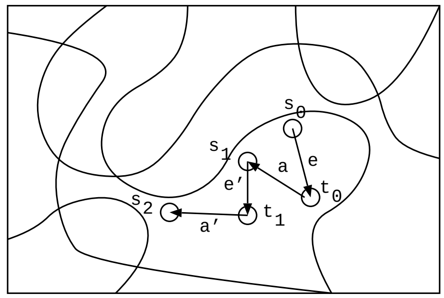

Fig. 2. Object whose information is preserved

$\mathsf{preserve}(i, n)$. If ω is a object whose I_i-information is preserved for n steps, then we write $\omega \models \mathsf{preserve}(i, n)$ (read "ω satisfies policy $\mathsf{preserve}(i, n)$").

Now, suppose we are given a policy A and an initial state s_0, and we want to know if we can accomplish the policy. One might guess that this means determining whether there is an ω starting at s_0 such that $\omega \models A$. This is not the case, because there is no guarantee that the indifferent environment will cooperate in having the effects necessary to construct ω. Also, there may be no way for the agent to determine which actions to take without knowledge of the the future. Consequently, it is not enough to know that it is possible for the information to be preserved; instead, we need something stronger. In game-theoretic terms, we seek a *strategy* that ensures that the information is preserved with high probability.

In game theory, a strategy is a method for deciding what move to make next, given the current game state. The question of interest in cooperative and competitive games is what strategies players should follow to maximize the expected payoff. In an indifferent-environment game, the environment has no payoff and its "strategy" (a random process) is already set. So in an indifferent game, the question of interest is what strategy the preservation agent should follow to maximize the likelihood of accomplishing its policy. We will call strategies for preserving information *preservation plans* (or just *plans*).

Formally, a plan is a function $\varphi : \mathcal{S} \to \mathcal{A}$ that picks an action $\varphi(s)$ to perform if the object is in state s. For example, a reasonable plan might be to attempt emergency recovery if s is an unstable state with a high likelihood of disastrous decay, and otherwise do nothing. An equally valid (but less reasonable) plan might be to do nothing in every state. Plans may or may not be likely to accomplish preservation goals.

If a preservation agent follows a plan φ starting from state s_0, and the environment has effects $e_1 \cdots e_k$, then the resulting object is uniquely determined. This object is denoted $\varphi(s_0, e_1 \cdots e_k) = s_0 e_1 a_1 s_1 \cdots e_k a_k s_k$ such that $s_{i-1} \to_e t_i$, $a_i = \varphi(t_i)$, and $t_i \to_{a_i} s_i$ for each i from 1 to n.

The environment's effects are, of course, not known in advance, but generated at random. Consequently, the actual effects of following plans are probabilistic in nature. Suppose that the agent follows plan φ for s_0, and $E_1 \cdots E_k$ is a sequence of $\mathcal{E}$-valued random variables distributed according to P. Then there is a unique random object (that is, an object-valued random variable) determined by these conditions. This random object is $\varphi(s_0, E_1 \cdots E_k) = s_0 E_1 A_1 S_1 \cdots E_k A_k S_k$.

Put simply, in our theory, being able to preserve information means finding a plan φ for a given state s that maximizes the probability that the resulting random object preserves its information. A *simple preservation problem* $\langle s, \mathsf{preserve}(i, n)\rangle$ consists of an initial state s and a policy $\mathsf{preserve}(i, n)$. A plan φ solves a simple preservation problem $\langle s, \mathsf{preserve}(i, n)\rangle$ with probability p if $p = P(\varphi(s_0, E_1 \cdots E_k) \models \mathsf{preserve}(i, n))$. A problem with only probability-0 solutions is *impossible*. A plan which is a probability-1 solution to a problem is an *absolute solution*.

Multiple objects In the real world situations we would like to model, preservation agents are responsible for large, interdependent collections of objects, not just single, self-contained ones. A simplistic solution to this problem would be to consider collections of object states to be indexed sets $S = \{s_1, \ldots, s_k\}$, collection policies to be sets $\{A_1, \ldots, A_k\}$ of statements $A_j = \mathsf{preserve}(i_j, t_j)$, and collection plans to be sets $\Phi = \{\varphi_1, \ldots, \varphi_k\}$. In this approach, a preservation problem is a pair $\langle S, A\rangle$ of a state collection and collection policy, and a probability-p solution is a collection plan Φ such that the probabilility that each φ_i solves each subproblem $\langle s_i, A_i\rangle$ is p.

There are two reasons why this approach is too simplistic. First, it requires considering each object in a collection on its own, which makes reasoning about large collections very complex. For example, in a million-object collection of very similar objects, we would like to be able to treat the many similar objects the same way rather than on a case-by-case basis. If all the objects are essentially the same from a preservation point of view, then it suffices to find a solution for the single-object model and apply it to all the objects. If instead there are many different kinds of objects, then we can represent each kind as a distinct object in the simple manner described above. In the million-object collection, suppose 50% of the items are images, 25% are text documents, and 25% are binary data files in the same format. Instead of naively considering each of the million objects, policies, and plans separately, we can get away with only three, representing generic text, image, and binary documents. The document kind representatives can be either a single typical document, or a representative sample.

Second, the simplistic approach does not make any allowance for preservation dependences among objects. It only allows preservation policies that are logical conjunctions of single-object policies. Real-world policies can be more complex. For example, it might be acceptable to preserve at least one of several duplicates of a document rather than all of them. Also, the question of whether one object is worth preserving may depend on whether another has been successfully preserved; thus, we might only want to bother with policy B if policy A succeeded. Finally, in some situations it is important to ensure that information is *not* been preserved; this is why offices have paper shredders.

To address this problem, instead of collection policies $\{A_1, \ldots, A_k\}$, policies are logical formulas A, B of the form j : $\mathsf{preserve}(i, n)$ (where $1 \leq j \leq k$), A AND B, A OR B, IF A THEN B, and NOTA. The collection policy

$$\{\mathsf{preserve}(i_1, t_1), \ldots, \mathsf{preserve}(i_k, t_k)\}$$

is equivalent to

$$1 : \mathsf{preserve}(i_1, t_1) \text{ AND } \cdots \text{ AND } k : \mathsf{preserve}(i_k, t_k)$$

The other policy forms OR , IF THEN , and NOT correspond to the "preserve at least one", "conditionally preserve", and "ensure not preserved" policies.

Suppose $\Omega = \{\omega_1, \ldots, \omega_k\}$ is a collection of objects. The j in the basic policy j : $\mathsf{preserve}(i, n)$ refers to the object of Ω that is to be preserved; that is, j : $\mathsf{preserve}(i, n)$ says that ω_j is to be preserved under interpretation I_i for t time steps. In that case, we write $\Omega \models j : \mathsf{preserve}(i, n)$. The meanings of the other forms of policies are the standard meanings defined in propositional logic. For example, $\Omega \models A$ AND B means that $\Omega \models A$ and $\Omega \models B$.

Suppose $S = \{s_1, \ldots, s_k\}$ is the set of initial states of the objects under consideration, and A is a policy expressed as a logical formula. A *preservation problem* is a pair $\langle S, A \rangle$. A collection plan Φ is a probability-p solution to $\langle S, A \rangle$ if the object collection Ω constructed by following the plans in Φ starting with the states in S satisfies A with probability p.

3 Compound Object Preservation Spaces

As noted earlier, much of the information available in digital form results from a complex interaction of hardware and software components. For example, a substantial portion of Web information is now the result of the interaction of databases, server side and browser side scripting, and style sheet based rendering. A realistic preservation strategy must account for this interaction, rather than focusing only on the "document" as an independent object.

In this section we show how to enrich our theory to describe how components of information systems interact to provide access to information through multiple layers of technology. To understand object states whose information content depends on shared components, we need additional mathematical machinery. In this section we will outline an approach to understanding compound objects using a branch of symbolic mathematics called *universal algebra*. Then we will explore the implications of compound object preservation theory for understanding the preservation strategies of emulation and migration.

We will outline how to use universal algebra to describe compound objects with sharing. Universal algebra is concerned with relating formal, syntactic expressions called *terms* with their real, semantic meanings in some mathematical structure. The language of arithmetic is a familiar example, containing constant symbols $0, 1, 2, \ldots$ for numbers, as well as function symbols $+, -, \cdot, \div$ that represent arithmetic operations. These symbols can be combined with each other and with *variables* $x, y, z, \ldots$ to form *terms* such as

$$x + 1, (y \cdot 2) \div 3, x + y - z$$

It is usually more convenient to use prefix notation rather than infix notation for terms, in which case the above terms are written

$$+(x, 1), \div(\cdot(y, 2), 3), -(+(x, y), z)$$

The rules which govern how terms can be formed are called *types*. Type rules relate constant and function symbols with *sort* symbols representing sets. In arithmetic, there is only one sort symbol, N, representing the set $\mathbb{N} = \{0, 1, 2, \ldots\}$ of all natural numbers. The type rule for constant symbols is that $n = 0, 1, 2, \ldots$ is a term of sort N (written $n : N$); for function symbols $op = +, -, \cdot, \div$, the rule is that if t_1, t_2 are terms of sort N, then $op(t_1, t_2)$ is a term of sort N (written $op : N \times N \to N$). Variables can have any sort, so long as a variable's sort is consistent within a term. For example, if $f : T \times U \to T$, then $f(x, x)$ would not be a legal term because it demands that x have both sort T and sort U. In arithmetic, there is only one sort symbol, but in general there might be more than one. The collection of sort, constant, and function symbols and their type rules is called a *signature*, Σ.

In universal algebra, there is a solid boundary between symbolic expressions and actual mathematical constants and functions; terms are a *language* for describing the behavior of mathematical structures. This distinction can be confusing, so it is common to use notation to distinguish between symbols and the sets, constants and functions they represent. We will use the following conventions: If T is a sort symbol of Σ, then T^σ is a set (called the *carrier set* of T). If c is a constant symbol of sort T then c^σ is a value in T^σ. If f is a function symbol of type $T_1 \times \cdots \times T_k \to T$ then f^σ is a function mapping the cartesian product space $T_1^\sigma \times \cdots \times T_k^\sigma$ to T^σ. (Here we are overloading the notation $\times$ as both type syntax and set-theoretic cartesian product; this is standard practice). Any mathematical structure σ containing carrier sets T^σ, constants c^σ, and functions f^σ that obey these rules is called a Σ-algebra.

Intuitively, in Web preservation, sort symbols such as Text, $\mathsf{Browser}$, Html, Image are the names of specific kinds of information content. Constants are the names of specific elements of these information content spaces; for example, $Netscape4.75$ might be a constant symbol of sort $\mathsf{Browser}$. Similarly, function symbols are the names of ways of combining information to get new information, for example we might introduce a function symbol $render : \mathsf{Browser} \times \mathsf{Html} \to \mathsf{Image}$ to represent a function that produces the page image a browser produces for a Web page.

The distinction between term syntax and mathematical semantics parallels that between object states and information content. Indeed, our approach will be to enrich object states with symbolic expressions that describe how to combine the object's basic components to access the object's information content. We define compound object states to be pairs $\langle \eta, t \rangle$, where $\eta = \{x_1 = s_1, \ldots, x_k =$

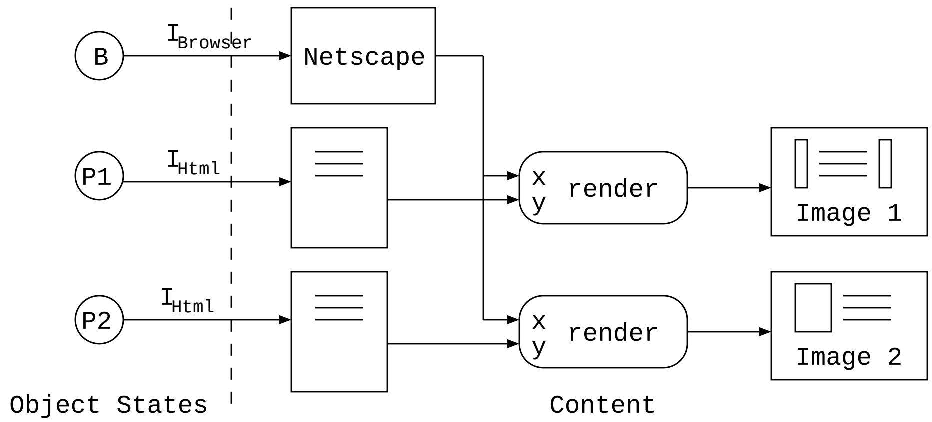

Fig. 3. Compound objects

$s_k\}$ is a *state environment* that maps variables x_i to basic states s_i, and t is a term using variables $x_1, \ldots, x_k$. For example,

$$\langle \{x = B, y = p\}, render(x, y) \rangle$$

is a compound object whose term says "Render page y with browser x" and whose state environment says "x is B and y is p". B might be the executable code for Netscape 4.75, and p might be the contents of `http://www.microsoft.com`. Over time, x might be changed to B', the executable for Internet Explorer 5.5, without affecting the web page component, and we would have

$$\langle \{x = B', y = p\}, render(x, y) \rangle$$

Figure 3 shows a related example, two different compound objects that share the same browser. In this case, changing a single basic object (the browser) could change the content of many compound objects.

Informally, to interpret a compound object state, we first interpret the content of each of its components, then interpret the object's symbolic expression as a function that maps the information contents of the components to the information content of the whole, and finally apply this function to the component contents. This implies that the information content space $\mathcal{C}$ is $\bigcup_T T^\sigma$, the union of all the sort carrier sets of some Σ-algebra σ. In the above examples, the compound object states represent the image information resulting from running a browser on a web page. The variables are associated with basic object states, which are interpreted as browsers or web pages. Then *render* combines them to produce a page image.

The technical definition of the interpretation of a compound object $\langle \eta, t \rangle$ is as follows. We assume that there are preexisting basic object state interpretations $I_T : \mathcal{S} \to T^\sigma$, one for each sort. First, suppose that each variable x_i is used with sort T_i in t and has value s_i in η. Now define $I\langle \eta, t \rangle$ as follows:

1. (Variable) If $t = x_i$, then $I\langle \eta, x_i \rangle = I_{T_i}(s_i)$ (if it exists).
2. (Constant) If $t = c$, then $I\langle \eta, c \rangle = c^\sigma$.
3. (Function) If $t = f(t_1, \ldots, t_k)$, then $I\langle \eta, f(t_1, \ldots, t_k) \rangle = f^\sigma(I\langle \eta, t_1 \rangle, \ldots, I\langle \eta, t_k \rangle)$ (if all the subterm interpretations exist).

The *compound object space* is a preservation space where $\mathcal{S}$ is the set of compound object states $\langle \eta, t \rangle$, $\mathcal{C}$ contains all the sets T^σ, and the interpretation $I : \mathcal{S} \rightharpoonup \mathcal{C}$, defined above, maps compound objects $\langle \eta, t \rangle$ to their compound object interpretations. The dynamics for compound objects can also be built up from the dynamics for basic objects.

The compound object theory enjoys several advantages over plain preservation spaces. First, it treats compound object states as state environments of "primitive" states plus instructions for combining them to produce the information of the overall object. Second, it includes a level of indirection (variables and state environments) that describes sharing of frequently used objects such as programs. Third, it explains how to interpret compound objects in terms of the interpretations of the components and the instructions. The compound object theory of preservation is descriptive enough to model both migration and emulation, as we will now demonstrate.

3.1 Modeling migration and emulation

Suppose we have a program p which runs on machine A using documents d, and our policy is to preserve access to the information in the documents, but p is obsolete and being replaced with a new, different version, q. One way to preserve access to these documents is through *migration* [16]. In migration, documents of old formats are replaced with "equivalent" documents of new formats.

We can model migration using sorts

$$\mathsf{Program}, \mathsf{Data}, \mathsf{Machine}, \mathsf{Behavior}$$

and a single operation $run : \mathsf{Machine} \times \mathsf{Program} \times \mathsf{Data} \to \mathsf{Behavior}$; running p on A with input d means evaluating $run(A, p, d)$. A migrator is represented by a function symbol $m : \mathsf{Data} \to \mathsf{Data}$ such that

$$\begin{aligned} & run^\sigma(I_{\mathsf{Machine}}(A), I_{\mathsf{Program}}(q), m^\sigma(I_{\mathsf{Data}}(d))) \\ = {} & run^\sigma(I_{\mathsf{Machine}}(A), I_{\mathsf{Program}}(p), I_{\mathsf{Data}}(d)) \end{aligned}$$

for each d of interest. Changing the object state from

$$\langle \{A = A_0; p = p_0; d = d_0\}, run(A, p, d) \rangle$$

to

$$\langle \{A = A_0; q = q_0; d = d_0\}, run(A, q, m(d)) \rangle$$

corresponds to *late migration*: keeping the document in the old format but translating it to the new format whenever it is accessed. An alternative is *early migration*, in which we replace the old version with the translation. Early migration corresponds to changing the state to

$$\langle \{A = A_0; q = q_0; d = d'\}, run(A, q, d) \rangle$$

where d' is some Data object such that $I_{\mathsf{Data}}(d') = m^\sigma(I_{\mathsf{Data}}(d_0))$. Depending on the circumstances, early migration, late migration, or some combination might be the better choice.

Now instead suppose that machine A is obsolete and B is its replacement. One way to preserve access to programs p that run on A is through *emulation* [15]. In emulation, the behavior of old machines is reproduced on new machines by a program, called an *emulator*, that interprets the machine instructions for the old machine and translates them into "equivalent" ones for the new machine.

We can also model this state of affairs using the sorted algebra above. An emulator for A on B would correspond to a program e such that

$$\begin{aligned} & run(I_{\mathsf{Machine}}(B), I_{\mathsf{Program}}(e), pair(as\_data(I_{\mathsf{Program}}(p)), I_{\mathsf{Data}}(d))) \\ = & run(I_{\mathsf{Machine}}(A), I_{\mathsf{Program}}(p), I_{\mathsf{Data}}(d)) \end{aligned}$$

for all (interesting) programs p and inputs d. Note that for this to work we need to add functions $pair : \mathsf{Data} \times \mathsf{Data} \to \mathsf{Data}$ that pairs two Data elements together, and $as\_data : \mathsf{Program} \to \mathsf{Data}$ that looks at a program as an element of Data. Preserving an object

$$\langle\{A = A_0; p = p_0; d = d_0\}, run(A, p, d)\rangle$$

by emulation means replacing it with another object

$$\langle\{B = B_0; e = e_0; p = p_0, d = d_0\} run(B, e, pair(as\_data(p), d))\rangle \ .$$

4 Future directions

There are many important issues in preservation that our theory, as presented, does not yet address. In real situations, agents have limited resources with which to enforce preservation policies. We believe our theory can readily include preservation costs, and should therefore be useful for analyzing preservation problems involving constrained resources. Another issue is that the value (or utility, as in economics) of different kinds of information varies and can change over time. As with cost, we believe that it is possible to add information utility to our theory and to consider preservation as a utility-maximization problem. By adding these factors to our theory, we should be able to formulate optimal preservation strategies in which agents seek the best tradeoff between cost and utility.

A third issue is that all our interpretations are discrete, in the sense that information content must be preserved exactly. There are many situations (e.g. image preservation using lossy compression) in which this view is too strict; instead information degrades gradually and continuously until the distortion is no longer acceptable. We believe that it is possible to enrich our theory to handle gradual degradation gracefully, possibly using many interpretations I_α, $0 \leq \alpha \leq 1$, that capture information equivalence up to distortion level α, or a "distortion function" $d : \mathcal{C} \times \mathcal{C} \to [0, 1]$, that indicates how acceptable a substitute one informational entity is for another.

A fourth issue is that of multiple agents. In the real world, there are many independent organizations and individuals, each with different preservation goals (as well as other goals, such as security). These agents may cooperate, compete, or behave indifferently towards one another. Again, we believe that it is possible to augment our theory to describe multi-agent situations and to formulate and solve preservation problems involving many agents.

5 Conclusions

We have presented a mathematical framework for analyzing preservation problems involving print and digital media. We also showed how to enrich this framework using universal algebra in order to reason about preserving information carried by composing objects together.

Our approach develops the concept of a preservation space, in which we can apply modeling techniques in ways that are general enough to help us understand a wide variety of preservation problems and potential strategies. We are seeking a preservation theory that will unify the seemingly incompatible views of information as either analog or digital, by abstracting out the commonalities and showing where the differences arise. Our eventual goal is to provide a firm theoretical basis on which we can evaluate the effectiveness of technical solutions to preserving information, and to contribute to our understanding of how information can be preserved in highly complex forms where purely intuitive approaches to preservation are unlikely to succeed.

Acknowldegements We thank Bill Arms for many thoughtful discussions and criticisms of this paper. The work described in this paper is supported by the National Science Foundation (Grant No. IIS-9905955, The PRISM Project).

References

1. Terry Cook. Electronic records, paper minds: The revolution in information management and archvies in the post-custodial and post-modernist era. *Archives and Manuscripts*, 22(2), November 1992.
2. Brian Cooper, Arturo Crespo, and Hector Garcia-Molina. Implementing a reliable digital object archive. In *Fourth European Conference on Digital Libraries*, 2000.
3. Arturo Crespo and Hector Garcia-Molina. Archival storage for digital libraries. In *Third ACM Conference on Digital Libraries*, 1998.
4. Arturo Crespo and Hector Garcia-Molina. Modeling archival repositories. In *Fourth European Conference on Digital Libraries*, 2000.
5. Luciana Duranti. *Diplomatics: New Uses for an Old Science.* Scarecrow Press, 1998.
6. Anne J. Gilliland-Swetland. Enduring paradigm, new opportunities: The value of the archival perspective in the digital environment. Technical Report 89, Council on Library and Information Resources, 2000. http://www.clir.org/pubs/reports/pub89/contents.html.

7. Stewart Granger. Digital preservation and the CAMiLEON project. http://ds.dial.pipex.com/stewartg/cam-london.html.
8. Margaret Hedstrom. Digital preservation: A time bomb for digital libraries. http://www.uky.edu/~kiernan/DL/hedstrom.html, 1997.
9. A. J. Jones. *Game Theory: Mathematical Models of Conflict*. Ellis Horwood, 1980.
10. Bertram Ludascher, Richard Marciano, and Reagan Moore. Towards self-validating knowledge-based archives. http://www.sdsc.edu/~ludaesch/Paper/ride01.html, 2001.
11. Clifford Lynch. Canonicalization: A fundamental tool to facilitate preservation and management of digital information. *D-LIb Magazine*, 5(9), September 1999. http://www.dlib.org/dlib/september99/09lynch.html.
12. Sue McKemmish. Are records ever actual. Technical report, Monash University, 1998.
13. John C. Mitchell. *Foundations for Programming Languages*. MIT Press, 1996.
14. Peter Morris. *Introduction to Game Theory*. Springer-Verlag, 1994.
15. Jeff Rothenberg. Ensuring the longevity of digital documents. *Scientific American*, 272(1):42–7, January 1995.
16. Donald Waters and John Garrett. Preserving digital information, report of the task force on archiving of digital information. Technical Report 63, Council on Library and Information Resources, 1996. http://www.clir.org/pubs/reports/pub63/contents.html.
17. Andrew Waugh, Ross Wilkinson, Brendan Hills, and Jon Dell'oro. Preserving digital information forever. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*, 2000.
18. Wolfgang Wechler. *Universal Algebra for Computer Scientists*. Springer, Berlin, 1992.