2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science

Abstract Hidden Markov Models: a monadic account of quantitative information flow

Annabelle McIver
Dept. Computing
Macquarie University
Sydney, Australia
Email: annabelle.mciver@mq.edu.au

Carroll Morgan
School of Comp. Sci. and Eng.
Univ. New South Wales
Sydney, Australia
Email: carrollm@cse.unsw.edu.au

Tahiry Rabehaja
Dept. Computing
Macquarie University
Sydney, Australia
Email: tahiry.rabehaja@mq.edu.au

***Abstract*—Hidden Markov Models, *HMM*'s, are mathematical models of Markov processes whose state is hidden but from which information can leak via channels. They are typically represented as 3-way joint probability distributions.**

We use *HMM*'s as denotations of probabilistic hidden-state sequential programs, after recasting them as "abstract" *HMM*'s, i.e. computations in the Giry monad $\mathbb{D}$, and equipping them with a partial order of increasing security. However to encode the monadic type *with hiding* over state $\mathcal{X}$ we use $\mathbb{D}\mathcal{X}\to\mathbb{D}^2\mathcal{X}$ rather than the conventional $\mathcal{X}\to\mathbb{D}\mathcal{X}$. We illustrate this construction with a very small Haskell prototype.

We then present *uncertainty measures* as a generalisation of the extant diversity of probabilistic entropies, and we propose characteristic analytic properties for them. Based on that, we give a "backwards", uncertainty-transformer semantics for *HMM*'s, dual to the "forwards" abstract *HMM*'s.

Finally, we discuss the Dalenius desideratum for statistical databases as an issue in semantic compositionality, and propose a means for taking it into account.

***Index Terms*—Abstract Hidden Markov Models, Giry Monad, Quantitative Information Flow.**

I. Introduction

A. Setting and overview

Probabilisitic sequential programs with hidden state are effectively Hidden Markov Models, or *HMM*'s,[1] formulated as joint probability distributions over initial state, observations, and final state. We recast *HMM*'s as computations over the Giry monad, suitable for program semantics. Indeed the monadic view of Markov processes in particular is well established [1], [2], using $\mathcal{X}\to\mathbb{D}\mathcal{X}$ where type-constructor $\mathbb{D}$ makes distributions on its base type $\mathcal{X}$; the Kleisli extension is then of type $\mathbb{D}\mathcal{X}\to\mathbb{D}\mathcal{X}$, representing action of multiplying an initial-state-distribution vector by a Markov matrix. But that does not account for hidden state and information flow.

We include hidden state by beginning with $\mathbb{D}\mathcal{X}$ (rather than $\mathcal{X}$): the computation type we obtain is "one level up", of type $\mathbb{D}\mathcal{X}\to\mathbb{D}^2\mathcal{X}$, the extension is $\mathbb{D}^2\mathcal{X}\to\mathbb{D}^2\mathcal{X}$; and we call the double-distribution type *hyper-distributions*.

Although the Giry monad is formulated in terms of general measures [2], we will need only discrete distributions for matrix-based *HMM*'s. Nevertheless we give our constructions and results in more general terms, anticipating e.g. infinite sequences of *HMM*'s, nondeterminism, and iterations for which proper measures will be necessary [3].

In earlier work we have used $\mathbb{D}^2\mathcal{X}$, equipped with a partial order of increasing security, to establish compositionality results [4], to explore the effect of including demonic non-determinism [5] and to give an abstract treatment of probabilistic channels [6], [7]. A second common theme has been the generalisation of entropies (such as Shannon) to a more abstract setting where only their essential properties are preserved [4], [5], [7], [8]. Here we use monads to bring all those separate strands together and go further.

One further step is to show that there is a dual backwards view for abstract *HMM*'s, based on "uncertainty" *transformers* that transform post- uncertainty measures into pre- uncertainty measures where, in turn, *uncertainty measures* generalise probabilistic entropies.

We and others have argued that specific entropies (e.g. Shannon) have limitations in security work generally [4], [9]. Therefore we focus here on their essential properties: continuity and concavity. That view is supported by powerful theorems that such a generalisation supports, and a methodological criterion that uncertainty measures capture contexts in a way that individual styles of entropy cannot.

A second further step is to extend our recent treatment [7] of the *Dalenius Desideratum*, the "collateral" leakage of information due to unknown correlations with third-party data, from merely channels (a "read only" scenario [10], [11], such as access to a statistical database) to programs that might alter the database (thus "read/write" as well). The Dalenius perspective here is the fact that care must be taken wrt. compositionality in a context containing variables to which a program fragment does not explicitly refer [4].

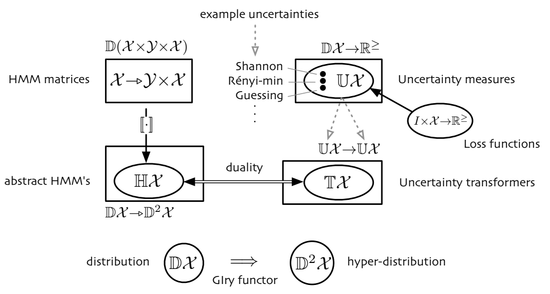
To remain accessible to a broader security community, we do not begin from Giry: rather we first work in elementary terms. In §VI the monadic structures will be seen to have informed our earlier definitions and theorems.

References to the appendix can be resolved at [12].

B. Principal contributions and aims of the paper: summary

Our principal contributions are these, in which the **new constructions and results** are given in bold:

[1]We use apostrophe uniformly for suffixes of acronyms.


example uncertainties
$\mathbb{D}(\mathcal{X}{\times}\mathcal{Y}{\times}\mathcal{X})$
HMM matrices
$\mathcal{X}{\rightarrowtriangle}\mathcal{Y}{\times}\mathcal{X}$
Shannon
Rényi-min
Guessing
$\mathbb{D}\mathcal{X}{\to}\mathbb{R}^{\geq}$
$\mathbb{U}\mathcal{X}$
Uncertainty measures
$[\![\cdot]\!]$
$I{\times}\mathcal{X}{\to}\mathbb{R}^{\geq}$
Loss functions
$\mathbb{U}\mathcal{X}{\to}\mathbb{U}\mathcal{X}$
abstract HMM's
$\mathbb{H}\mathcal{X}$
duality
$\mathbb{T}\mathcal{X}$
Uncertainty transformers
$\mathbb{D}\mathcal{X}{\rightarrowtriangle}\mathbb{D}^2\mathcal{X}$
distribution
$\mathbb{D}\mathcal{X}$
$\Longrightarrow$
Giry functor
$\mathbb{D}^2\mathcal{X}$
hyper-distribution

— Fig. 1. Relationship between the semantic spaces —

— We note that (finite) classical *HMM*'s are a model for straight-line sequential probabilistic programs with hidden state (top-left Fig. 1 shows the result type once applied to a prior of type $\mathbb{D}\mathcal{X}$, §III-A).

— We formulate *abstract HMM's*, that is $\mathbb{H}\mathcal{X}$ as a **monadic model for *HMM*'s** over state $\mathcal{X}$, and **give their characteristic properties** (bottom left: §III-B,V,VII).

— We formulate *uncertainty measures* $\mathbb{U}\mathcal{X}$ as a generalisation of diverse entropies (top centre), and **give their characteristic properties** (top right: §VIII-A).[2]

— We note that uncertainty measures **have a complete representation** as *loss functions* (centre right: §IX).

— We give a dual, uncertainty-transformer semantics $\mathbb{T}\mathcal{X}$ of $\mathbb{H}\mathcal{X}$, stating its characteristic properties (bottom right) and proving that they enable the duality with $\mathbb{H}\mathcal{X}$ (centre: Thm. 29 in §VIII-B).

— We show how all of this is an instance of the general Giry monad as computation, of which (finite) *HMM*'s use a discrete portion (bottom centre: §VI).

— We explain how the "Dalenius effect" is manifested in this framework, and how it can be treated (§XI).

In other sections we review abstract channels (§II-B), hyper-distributions (§II-C) and the security order (§V) on hypers.

We believe that Thm. 29, in particular its assumptions and proof, is a significant new result.

Our principal aims in this paper are these:

— (More abstract) We construct forward- and dual backward semantic spaces for probabilistic sequential computations over hidden state, using monadic computations and partial (refinement) orders in this new context, and formulate and prove the general properties that make them suitable for embedding finite (for the moment) *HMM*'s.

— (More concrete) We want to provide the basis for a source-level reasoning method, analogous to Hoare logic or weakest preconditions, for quantitative non-interference in sequential programs. For this, the dual, transformer semantics for *HMM*'s seems to be a necessary first step, together with a link between the social aspects of security and the mathematical behaviour of a program (§X).

The conclusion §XIII discusses the benefits of doing this.

[2]These were studied, but less extensively, as "disorders", in [5].

C. *General notations (see also App. A in the appendix)*

Application of function f to argument x is written $f.x$, to reduce parentheses. It associates to the left.

Although a matrix M with rows, columns indexed by R, C is a function $R{\times}C \to \mathbb{R}$, we avoid constant reference to the reals $\mathbb{R}$ by writing just $R{\rightarrowtriangle}\overrightarrow{C}$ for that; similarly we write the type of a vector over X as $\overrightarrow{X}$. We write $M_{r,c}$ for the element of matrix M indexed by row r and column c; the r-th row of M is $M_{r,-}$; and the c-th column is $M_{-,c}$, of types $\overrightarrow{Y}, \overrightarrow{X}$ resp. For row- or column vector v: $\overrightarrow{I}$ we write v_i for its i-th element. Thus e.g. we have $(M_{-,c})_r{=}M_{r,c}$.

When multiplying vectors and matrices we assume without comment that the vector has been oriented properly, i.e. as a row or column as required. Thus v acts as a row in $v{\cdot}M$ but as a column in $M{\cdot}v$.[3]

We write for example $x{:}\,X$ when we are introducing x at that point (i.e. a binding occurrence); with $x{\in}X$ we are stating a property of some x and X already introduced.

Other specific notations are explained at first use, and a glossary in order of their occurrence is given in App. A.

II. Abstract channels and hyper-distributions

We review abstract channels as a conceptual stepping-stone to hyper-distributions, or "hypers" for short.

A. *Channels and distributions as matrices and vectors*

A *channel* is a (stochastic) matrix of non-negative reals with 1-summing rows; we use upper-case Roman letters like C. The rows are labelled with elements from some set $\mathcal{X}$; and the columns from some set $\mathcal{Y}$. Thus a channel typically has type $\mathcal{X}{\rightarrowtriangle}\mathcal{Y}$; here, both $\mathcal{X}, \mathcal{Y}$ will be finite.

A distribution in $\mathbb{D}\mathcal{X}$ can be presented as a 1-summing vector $\overrightarrow{\mathcal{X}}$, usually lower-case Greek: especially π for prior; sometimes ρ for posterior; or simply δ for distribution.

Definition 1: **Weight** Let C or π be a matrix or vector resp. Then $\sum C$ or $\sum \pi$ is its *weight*, the sum $\sum_{x,y} C_{x,y}$ or $\sum_x \pi_x$ taken over all its indices. $\square$

Thus e.g. we have $\sum C_{x,-} = \sum_y C_{x,y}$ and that C is stochastic just when $1 = \sum C_{x,-}$ for all x.

Each row $C_{x,-}$ of a channel C is a conditional probability distribution over $\mathcal{Y}$ given that particular $x{:}\,\mathcal{X}$. That is, the y-th element $C_{x,y}$ of $C_{x,-}$ is the probability that C takes input x to output y.

B. *Informal channel semantics:* abstract *channels*

A (1-summing) prior π and (stochastic) channel C together determine a joint distribution as follows.

Definition 2: **Channel applied to prior** Given a prior $\pi{:}\,\overrightarrow{\mathcal{X}}$ and channel $C{:}\,\mathcal{X}{\rightarrowtriangle}\mathcal{Y}$ we write $\pi{\triangleright}C$ for the joint-distribution matrix of type $\mathcal{X}{\rightarrowtriangle}\mathcal{Y}$ resulting from "applying" the channel to the prior, defined $(\pi{\triangleright}C)_{x,y}{:=}\,\pi_x C_{x,y}$.[4] $\square$

Note that matrix $\pi{\triangleright}C$ is not stochastic: rather because C itself is stochastic we have $\sum(\sum(\pi{\triangleright}C)_{x,-}) = \sum \pi_x = 1$.

[3]Thus for v: $\overrightarrow{X}$ and $M{:}\,X{\rightarrowtriangle}Y$ the matrix product $v{\cdot}M$ is in $\overrightarrow{Y}$.

[4]Here juxtaposition is ordinary multiplication of reals.

A non-zero vector is normalised as follows.

Definition 3: **Normalisation** Let $\delta{:}\,\overrightarrow{\mathcal{X}}$ be such that $0{\neq}\sum\delta$. Then the *normalisation* $\mathsf{nrm}.\delta$ of δ is given by $(\mathsf{nrm}.\delta)_x{:=}\,\delta_x/\sum\delta$ for each $x{:}\,\mathcal{X}$. □

Now for some $\pi{:}\,\overrightarrow{\mathcal{X}}$ and channel $C{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}$ define joint distribution $J{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}$ by $J{=}\pi{\triangleright}C$. The (marginal) probability of each output $y{:}\,\mathcal{Y}$ is $\sum J_{-,y}$ and, associated with each, there is a posterior distribution $\mathsf{nrm}.(J_{-,y})$ on $\mathcal{X}$.

Abstracting from the y-values, but retaining the link between the marginal probabilities and the posterior distributions, gives an informal description of our intended "abstract channel" semantics [6]. We make this precise in §II-D.

C. Hypers abstract from joint distributions

The joint-distribution matrix $J{=}\pi{\triangleright}C$ contains "too much" information if we do not need the actual value of y that led to a particular posterior. This is appropriate in security since the information leakage of a channel C wrt. a prior π concerns what an adversary can discover about π, and not the actual observations that led to that discovery. We can abstract from the observations in $\pi{\triangleright}C$ as follows.

If column y of $J=\pi{\triangleright}C$ is all zero, then that y will never occur (for any prior); thus we can omit that column.

And if two columns $y_{1,2}$ of J are proportional to each other, are *similar* (as for triangles), then we can add them since for a given prior the same posterior will be inferred for y_1 as for y_2 and the probability of inferring that posterior will be the sum of the marginal probabilities for $y_{1,2}$.[5]

Finally, a 1-1 renaming of the y-values has no effect on the posteriors and their respective probabilities; so we can remove those names as long as we retain the distinction between separate (non-zero, non-similar) columns.

Abstracting from all that arguably inessential information (about y) leaves only a distribution of posteriors on $\mathcal{X}$ and, for us, this is the semantic view. Writing in general $\mathbb{D}\mathcal{X}$ for 1-summing functions of type $\mathcal{X}{\rightarrow}\mathbb{R}^{\geq}$, a distribution over $\mathcal{X}$ has type $\mathbb{D}\mathcal{X}$ and so a distribution of such distributions has type $\mathbb{D}(\mathbb{D}\mathcal{X})$ that is $\mathbb{D}^2\mathcal{X}$.[6] Those latter are our hypers, and they are our abstraction of joint-distributions $\mathcal{X}{\twoheadrightarrow}\mathcal{Y}$.

D. The semantic function from joints to hypers

In this section we define precisely the denotation $[\![J]\!]$ in $\mathbb{D}^2\mathcal{X}$ of a joint-distribution matrix $J{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}$. The principal tool for that is the "push forward", here in general form:

Definition 4: **Push-forward of a function**
Given sets $\mathcal{Z},\mathcal{Z}'$ and function $f{:}\,\mathcal{Z}{\rightarrow}\mathcal{Z}'$, we write $\mathbb{D}f$ for the *push-forward* of f, a "lifted" function of type $\mathbb{D}\mathcal{Z}{\rightarrow}\mathbb{D}\mathcal{Z}'$ [13]. For $z'{:}\,\mathcal{Z}'$ and $\delta{:}\,\mathbb{D}\mathcal{Z}$ we have [7]

$$\mathbb{D}f.\delta.z' \quad := \quad \sum_{\substack{z:\,\mathcal{Z}\\ f.z=z'}} \delta.z~. \qquad ^{8}$$

□

[5]We write $y_{1,2}$ rather than y_1, y_2 for brevity.

[6]Thus 1-summing vectors δ in $\overrightarrow{\mathcal{X}}$ describe distributions in $\mathbb{D}\mathcal{X}$.

[7]Lifting, as in $\mathbb{D}f$, binds tightest: the conventional notation for $\mathbb{D}f.\delta.z'$ would be $(\mathbb{D}f)(\delta)(z')$, so that $(\mathbb{D}f)(\delta){\in}\mathbb{D}\mathcal{Z}'$.

[8]$\mathbb{D}f$ is the action of functor $\mathbb{D}$ on arrow f: see §VI.

We now define the semantic function itself:

Definition 5: **Reduced joint-distribution denotes hyper**
Let $J{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}$ satisfy $1=\sum J$ so that it describes a discrete joint distribution in $\mathbb{D}(\mathcal{X}{\times}\mathcal{Y})$. Recalling §II-C, define a *reduced* matrix $J^{\downarrow}$ by (1) removing all-zero columns from J and (2) adding any similar columns of J together, retaining the label of only one of them. Let the remaining labels $\mathcal{Y}^{\downarrow}{\subseteq}\mathcal{Y}$ be the column-indices of this reduced matrix $J^{\downarrow}$.[9] [10]

Now define the $\mathcal{Y}^{\downarrow}$-marginal $\delta_y{:=}\,\sum J^{\downarrow}_{-,y}$ of $J^{\downarrow}$, and note from (1) just above that it is nowhere zero (on $\mathcal{Y}^{\downarrow}$). Define function $j{:}\,\mathcal{Y}^{\downarrow}{\rightarrow}\mathbb{D}\mathcal{X}$ by $j.y{:=}\,\mathsf{nrm}.J^{\downarrow}_{-,y}$, i.e. so that $j.y{\in}\mathbb{D}\mathcal{X}$ is the posterior distribution over $\mathcal{X}$ that $J^{\downarrow}$ induces given y. Note from (2) that j is an injection, a fact we use later in Lem. 14. Then $[\![J]\!]$, the hyper in $\mathbb{D}^2\mathcal{X}$ denoted by J in $\mathcal{X}{\twoheadrightarrow}\mathcal{Y}$, is given by $[\![J]\!]{:=}\,(\mathbb{D}j).\delta$. □

An example is given at §III-D below.

E. Abstract channels — review

In earlier work [6] we described an "abstract channel" as a function from prior distributions to hypers. We restate that here in our current denotational style:

Definition 6: **Denotation of channel** Let $C{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}$ be a channel matrix. Its denotation, of type $\mathbb{D}\mathcal{X}{\rightarrow}\mathbb{D}^2\mathcal{X}$, is called an *abstract channel* and is defined for $\pi{:}\,\mathbb{D}\mathcal{X}$ by $[\![C]\!].\pi{:=}\,[\![\pi{\triangleright}C]\!]$, where the $[\![\cdot]\!]$ on the left is what we are defining, and the $[\![\cdot]\!]$ on the right is given by Def. 5.[11] □

In fact the prior π can be recovered from $\pi{\triangleright}C$:

Definition 7: **Average of a hyper** For hyper $\Delta{:}\,\mathbb{D}^2\mathcal{X}$ define its average $\mathsf{avg}.\Delta$ in $\mathbb{D}\mathcal{X}$ by

$$\mathsf{avg}.\Delta.x \quad := \quad \sum_{\delta:\,\mathbb{D}\mathcal{X}} (\Delta.\delta)(\delta.x) \qquad \text{for all } x{:}\,\mathcal{X}, \qquad ^{12}$$

where we use upper-case Greek for hypers. □

We then have

$$(\mathsf{avg}.([\![C]\!].\pi))_x \;=\; (\mathsf{avg}.[\![\pi{\triangleright}C]\!])_x \;=\; \sum(\pi{\triangleright}C)_{x,-} \;=\; \pi_x~,$$

so that $\mathsf{avg}.([\![C]\!].\pi)=\pi$.

III. Classical- vs. abstract *HMM*'s

A. Classical HMM's: single HMM-steps as matrices

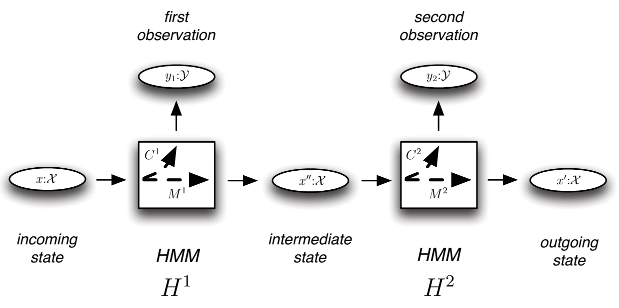
Classically a Hidden Markov Model comprises a set $\mathcal{X}$ of states, a set $\mathcal{Y}$ of observations and two stochastic matrices C,M that give resp. the *emission* probabilities $C_{x,y}$ that x will emit observation y and the *transition* probabilities $M_{x,x'}$ that x will change to x' [15]. Usually, the homogenous case, computation evolves in (probabilistic) steps each determined by the same C,M, with each output state x' becoming the following input x and with the emissions y accumulating: the steps all have the same pair C,M. In our case however,

[9]The reduction is analogous to reduced *channels* in [6]. Although $J^{\downarrow}$ is not unique, the ambiguity does not affect $[\![J]\!]$.

[10]We thank a referee for suggesting that this definition might be simplified by using the converse of stochastic relations, as developed by Doberkat [14]. This is discussed further in §XII.

[11]We use $[\![\cdot]\!]$ uniformly for denotation functions, relying on context instead of e.g. using subscripts like $[\![\cdot]\!]_{\text{chan}}$ and $[\![\cdot]\!]_{\text{joint}}$.

[12]This avg is multiplication μ from the Giry monad: see §VI.



Each step $H^{1,2}$ takes an input- to an output state in $\mathcal{X}$; the observations $y_{1,2}\colon\mathcal{Y}$ are accumulated. In each step $H^{1,2}$ the output state is determined by a markov $M^{1,2}$ on the input to that step, and the observation is determined independently by a channel $C^{1,2}$ on the same input, i.e. *before* application of the markov.

Fig. 2. Two successive steps H^1 and H^2 of a heterogeneous *HMM*.

heterogeneous, we can vary the matrices from step to step, each standing for various (different) programs statements.

We show two computations in Fig. 2. If π is the distribution of incoming x, the distribution π'' of intermediate x'' is $\pi{\cdot}M^1$. The distribution of observations y_1 is $\pi{\cdot}C^1$. The second step's input x'' is the output of the first step.

A classical *HMM* hides all of three of x, x'', x', but still the observations $y_{1,2}$ tell us something about each of them provided we know $\pi, M^{1,2}, C^{1,2}$. (This is analogous to knowing the source code of a program, but not being able to observe its variables as it executes.)

From now on we call the emission part of an *HMM* the *channel* and the transition part the *markov* (lower case).

Definition 8: **Single *HMM*-step**
Given channel $C\colon\mathcal{X}{\rightarrowtriangle}\mathcal{Y}$ and markov $M\colon\mathcal{X}{\rightarrowtriangle}\mathcal{X}$, define the *HMM*-matrix $(C{:}M)$ of type $\mathcal{X}{\rightarrowtriangle}\mathcal{Y}{\times}\mathcal{X}$ by

$$(C{:}M)_{x,y,x'} \quad := \quad C_{x,y}M_{x,x'}\ .$$

This (row-1-summing) matrix $(C{:}M)$ becomes a joint distribution of type $\mathbb{D}(\mathcal{X}{\times}\mathcal{Y}{\times}\mathcal{X})$, as top-left in Fig. 1, once the prior is fixed.[13] $\square$

B. Abstract HMM's represent classical HMM's

For abstract channels (§II-E) we focussed on the hyper of posteriors on the *input*; for *HMM*'s we focus on the hyper of posteriors on the *output*, because *HMM*'s are computations and so it is over their outputs we wish to reason.[14]

Definition 9: **Matrix *HMM* denotes abstract *HMM***
Let $H\colon\mathcal{X}{\rightarrowtriangle}\mathcal{Y}{\times}\mathcal{X}$ be an *HMM* presented as a matrix. Its denotation, of type $\mathbb{D}\mathcal{X}{\rightarrow}\mathbb{D}^2\mathcal{X}$, is called an *abstract HMM* and is defined $[\![H]\!].\pi{:=}\,[\![J]\!]$ where $\pi\colon\mathbb{D}\mathcal{X}$, and the joint-distribution matrix $J\colon\mathcal{X}{\rightarrowtriangle}\mathcal{Y}$ is given by $J_{x',y}{:=}\ \sum_x \pi_x H_{x,y,x'}$. $\square$

[13]Although $(C{:}M)$ has the property that for each $x\colon\mathcal{X}$ the (remaining) joint distribution $(C{:}M)_{x,-,-}$ is independent in y, x', this property is not preserved once steps are composed (§IV).

[14]The *prior* on the output would be our calculation from the input prior and the markov of what the output distribution would be, but *before* running the program and making observations.

```
// xs is initialised uniformly at random.
xs:= xs 1/2⊕ -xs
// What does an attacker guess for xs finally?
```

The secret two-bit bit-string xs is set initially from $\{00, 01, 10, 11\}$ with equal probability $^1\!/_4$ for each; the following assignment either leaves xs unchanged (probability $^1\!/_2$) or bit-wise inverts all of it.

— Fig. 3. Pure-markov *HMM* program —

In §XI we discuss the (Dalenius) implications of having abstracted from the *HMM*'s input (with the $\sum_x$ just above).

C. Special cases of HMM's: pure markovs

Markovs are the special case of *HMM* where the channel-part effectively outputs nothing. If an *HMM*-step $(C{:}M)$ has channel C an all-one column vector nc[15] then $\mathcal{Y}$ is a singleton and J becomes a column vector: i.e. $J_{x'} = \sum_x \pi_x M_{x,x'}$, so that in fact J is the usual matrix product $\pi{\cdot}M$.

Definition 10: **One- and two-point distributions**
For $z, z'\colon\mathcal{Z}$ we write $[z]$ for the *point distribution* on z, assigning probability 1 to z and 0 to all other elements of $\mathcal{Z}$.[16] We write $z_p{\oplus}z'$ for the two-point distribution that assigns p to z and $1{-}p$ to z' and 0 to everything else in $\mathcal{Z}$.

Thus $z_1{\oplus}z' = [z]$ and $z_0{\oplus}z' = [z']$. $\square$

Taking nc as the default channel gives $[\![{:}M]\!].\pi = [\![\mathsf{nc}{:}M]\!].\pi = [\pi{\cdot}M]$, the point hyper on $\pi{\cdot}M$. A general H is a markov just when $\sum_{x'} H_{x,y,x'}$ is nc.

Consider the program of Fig. 3 whose single variable is a two-bit string xs. We model it with $\mathcal{X}{=}\{00, 01, 10, 11\}$; prior $\pi\colon\mathbb{D}\mathcal{X}$ is uniform, and its markov M is as just below:

The output distribution is of course $\pi'{=}\pi{\cdot}M{=}\pi$, and so the attacker's guess is optimally any of the four values in $\mathcal{X}$: they are equally good.

$$\begin{array}{r|cccc|} & 00 & 01 & 10 & 11 \\ 00{:} & ^1\!/_2 & 0 & 0 & ^1\!/_2 \\ 01{:} & 0 & ^1\!/_2 & ^1\!/_2 & 0 \\ 10{:} & 0 & ^1\!/_2 & ^1\!/_2 & 0 \\ 11{:} & ^1\!/_2 & 0 & 0 & ^1\!/_2\ . \end{array}$$

This system viewed as an abstract *HMM* would give output *hyper* $\Delta' = [\![{:}M]\!].\pi = [\pi]$, in fact the *point* hyper on π indicating that the attacker is certain (point-probability 1) that the posterior distribution π' on the final value of xs is equal to π in this case, i.e. still uniform.

D. Special cases of HMM's: pure channels

Channels are the special case where the input- and the output state are the same. If $(C{:}M)$ has markov M as the identity id, then it is a "pure channel" with output the same as its input. In that case Def. 9 gives $J_{x',y} = \sum_x \pi_x C_{x,y}\mathsf{id}_{x,x'} = (\pi{\triangleright}C)_{x',y}$, and so $[\![C{:}\mathsf{id}]\!]$ from Def. 9 is just $[\![C]\!]$ from Def. 6.

With id as the default markov, we have $[\![C{:}]\!] = [\![C]\!]$.

Now consider Fig. 4 where some of xs is leaked, but xs itself is not changed. Thus our state $\mathcal{X}$ and prior π are as

[15]for "null channel"

[16]Function $[\cdot]$ is the unit η of the monad: see §VI.

```
// xs is initialised uniformly at random.
print xs[0] 1/2⊕ xs[1]
```

The value of either bit 0 or bit 1 of xs is revealed; the attacker learns that value, but does not know which bit it came from. What should he guess for xs after execution in this case?

— Fig. 4. Pure-channel *HMM* program —

before, the observation space is $\mathcal{Y}{=}\{0,1\}$ and the channel C representing this program is here at left:

$$C = \begin{array}{r} \\ 00: \\ 01: \\ 10: \\ 11: \end{array} \begin{array}{c} \;\;0 \quad\; 1 \\ \begin{pmatrix} 1 & 0 \\ 1/2 & 1/2 \\ 1/2 & 1/2 \\ 0 & 1 \end{pmatrix} \end{array} \qquad J^{\downarrow} = \begin{array}{r} \\ 00: \\ 01: \\ 10: \\ 11: \end{array} \begin{array}{c} \;\;0 \quad\; 1 \\ \begin{pmatrix} 1/4 & 0 \\ 1/8 & 1/8 \\ 1/8 & 1/8 \\ 0 & 1/4 \end{pmatrix} \end{array} .$$

The (reduced) joint distribution based on $J{=}\pi{\triangleright}C$ is above at right. (In fact no reduction was necessary.) The construction of Def. 5 gives us a hyper Δ' as

$$\begin{array}{ll} \text{“inner” distributions} & \text{“outer” distribution} \\ (1/2, 1/4, 1/4, 0) & @\ 1/2 \\ (0, 1/4, 1/4, 1/2) & @\ 1/2\ , \end{array} \tag{1}$$

where in general we write $z_1@p_1,\ z_2@p_2, \cdots$ for the discrete distribution that assigns probability p_1 to z_1 etc. In (1) the values $z_1, z_2, \cdots$ are themselves (inner, posterior) distributions. This hyper shows that with probability $1/2$ the attacker will guess 00 (because he saw a 0 printed, and deduces *a posteriori* that 00 now has the highest probability, twice either of the others; and with probability $1/2$ the attacker will guess 11 (because he saw a 1).

We have abstracted from the printed-out $\mathcal{Y}$-values, i.e. what he saw, concentrating simply on what he deduces.

IV. *HMM* PROGRAMMING: SEQUENTIAL COMPOSITION

A. Classical HMM composition: matrices

Let $H^1, H^2{:}\,\mathcal{X}{\multimap\!\!\!\triangleright}\mathcal{Y}{\times}\mathcal{X}$ be two *HMM*'s. Their sequential composition $H = H^1; H^2$ describes the distribution on x, $y_{1,2}$ together and x' as

$$(H^1;H^2)_{x,(y_1,y_2),x'} \quad := \quad \sum_{x''} H^1_{x,y_1,x''} H^2_{x'',y_2,x'} \ . \tag{2}$$

This can be seen as rewriting H^1 as type $\mathcal{X}{\times}\mathcal{Y}{\multimap\!\!\!\triangleright}\mathcal{X}$, then matrix-multiplying by H^2, and then re-converting the resulting $\mathcal{X}{\times}\mathcal{Y}{\multimap\!\!\!\triangleright}\mathcal{Y}{\times}\mathcal{X}$ back to $\mathcal{X}{\multimap\!\!\!\triangleright}\mathcal{Y}^2{\times}\mathcal{X}$.[17] Note how the set of observables is now $\mathcal{Y}^2$, compounding the observations $\mathcal{Y}$ from each component. (This is why infinite composition of *HMM*'s cannot easily be represented as a finite matrix.)

Remarkably, the action of *HMM*-composition on pure-markovs *HMM*'s is effectively their matrix multiplication, yet its action on pure channels is effectively their “concatenation”: a single definition of composition specialises automatically to the two principal sub-cases. (See App. B.) Thm. 12 shows that the same happens for abstract *HMM*'s.

[17]Lambert Meertens pointed out this nice formulation.

```
// xs is set uniformly at random.
print xs[0] 1/2⊕ xs[1] ;
xs:= xs 1/2⊕ -xs
```

The value of either bit 0 or bit 1 of xs is revealed; the attacker learns that value, but does not know which bit it is. Then xs is either unchanged or inverted, but the attacker does not know which. What's his best guess now for the final value of xs?

— Fig. 5. *HMM* program as sequential composition —

B. Abstract HMM's: Kleisli composition

Now we consider $h_1; h_2$ where $h_{1,2}$ are abstract *HMM*'s.[18] Because the components' types $\mathbb{D}\mathcal{X}{\rightarrow}\mathbb{D}^2\mathcal{X}$ do not match directly, i.e. the $\mathbb{D}^2\mathcal{X}$ from the left is not the $\mathbb{D}\mathcal{X}$ required on the right, we use Kleisli composition for that.[19]

Definition 11: **Kleisli composition of abstract *HMM*'s** Given two abstract *HMM*'s $h_{1,2}{:}\,\mathbb{D}\mathcal{X}{\rightarrow}\mathbb{D}^2\mathcal{X}$, their Kleisli composition is defined $(h_1;h_2).\pi{:=}\,\mathsf{avg}.(\mathbb{D}h_2.(h_1.\pi))$ for $\pi{:}\,\mathbb{D}\mathcal{X}$, where $\mathbb{D}h_2$ is the push-forward of h_2 (as given in Def. 4). Equivalently $h_1;h_2{:=}\,\mathsf{avg}\circ\mathbb{D}h_2\circ h_1$.

That is, the lifting inherent in Kleisli-composition applies the right-hand abstract *HMM* h_2 to each inner (i.e. posterior) produced by the left-hand h_1 from prior π, preserving the way in which they are all combined together by the outer distribution. Then the intermediate result, of type $\mathbb{D}^3\mathcal{X}$, is averaged to bring it back to the required type $\mathbb{D}^2\mathcal{X}$. □

C. Proof that composition is faithfully denoted

It is important (though unsurprising) for our interpretation that composition of *HMM*'s as matrices (2) is correctly mapped by $[\![\cdot]\!]$ to their Kleisli composition as abstract *HMM*'s (Def. 11). That is, we have

Theorem 12: **Composition faithfully denoted** Let $H^{1,2}{:}\,\mathcal{X}{\multimap\!\!\!\triangleright}\mathcal{Y}{\times}\mathcal{X}$ be *HMM*'s as matrices. Then we have $[\![H^1;H^2]\!] = [\![H^1]\!];[\![H^2]\!]$, where (2) is used on the left and Def. 11 on the right.

Proof: Given in App. C. □

D. Channel and markov together: example of composition

As an example of sequential composition return to xs and consider Fig. 5 where it is both leaked *and* (possibly) changed. The final hyper Δ' in this case is obtained by applying the markov M to the *inners* generated by C in §III-C while retaining their outers: that gives

$$\begin{array}{ll} (\ 1/2{\times}1/2 + 1/2{\times}0, & (\ 1/2{\times}0 + 1/2{\times}1/2, \\ \;\;1/2{\times}1/4 + 1/2{\times}1/4, & \;\;1/2{\times}1/4 + 1/2{\times}1/4, \\ \;\;1/2{\times}1/4 + 1/2{\times}1/4, & \;\;1/2{\times}1/4 + 1/2{\times}1/4, \\ \;\;1/2{\times}0 + 1/2{\times}1/2\ \ )@\ 1/2 & \;\;1/2{\times}1/2 + 1/2{\times}0\ \ )@\ 1/2 \end{array}$$

which is simplified first to this $\rightarrow$ $\begin{array}{ll}(1/4, 1/4, 1/4, 1/4) & @\ 1/2 \\ (1/4, 1/4, 1/4, 1/4) & @\ 1/2\end{array}$ and then, since the two inners are the same, as a distribution collapses to just $[\pi]$ again.[20] Thus the program of Fig. 5 reveals nothing about the final

[18]We use upper-case for matrices and lower-case for denotations.

[19]This is the usual composition in a Kleisli category. See §VI.

[20]We use an explicit $(\times)$ for multiplication of specific numbers.

value of xs when the initial distribution was uniform. Informally we would explain this by noting that the information about xs released by the print becomes "stale", irrelevant once we do not know whether xs has subsequently been inverted or not. (See §XI however for a discussion of why the initial value of xs might in some cases still be important.)

It would be wrong however to conclude, from $\pi{=}\pi'$ in this case, that the program is secure for xs in general — for when the initial distribution is *not* uniform, the final value of xs *can* be less secure than the initial. This illustrates the danger in assuming something is uniformly distributed simply because we know nothing about it. (See App. D.)

In App. E a small Haskell prototype verifies these calculations.

V. The structure of hyper-space

Our hyper-space $\mathbb{D}^2\mathcal{X}$ has been synthesised by abstraction from the classical "matrix style" description of *HMM*'s. We now recall that there is a partial order $(\sqsubseteq)$ of refinement, where for two hypers $\Delta_{S,I}{:}\,\mathbb{D}^2\mathcal{X}$ we say that Δ_S (a specification) is "refined by" Δ_I (implementation) when, in a sense we make precise below, the implementation Δ_I releases no more information than the specification Δ_S does [3]–[5], [8]. That order lifts pointwise to $\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$, i.e. that $h_S{\sqsubseteq}h_I$ just when $h_S.\pi \sqsubseteq h_I.\pi$ for all $\pi{:}\,\mathbb{D}\mathcal{X}$, thus giving a new *refinement* order for (abstract) *HMM*'s. We write $\Delta_S{\sqsubseteq}\Delta_I$, and call it "uncertainty refinement" if we need to distinguish it from other kinds of refinement. Its ultimate antecedent is the lattice of information [16], which it generalises significantly.

Definition 13: **Uncertainty refinement [3], [5]**
Let $\Delta_{S,I}{:}\,\mathbb{D}^2\mathcal{X}$ be two hypers on $\mathcal{X}$. We say that Δ_S is refined by Δ_I just when there is a distribution $\underline{\Delta}{:}\,\mathbb{D}^3\mathcal{X}$, that is a distribution of *hypers*, such that

$$\Delta_S \;=\; \mathsf{avg}.\underline{\Delta} \quad \text{and} \quad (\mathbb{D}\mathsf{avg}).\underline{\Delta} \;=\; \Delta_I\;.$$

□

Recall that $\mathbb{D}\mathsf{avg}$ is the push-forward of avg (Defs. 4,7).

The advantage of the abstract formulation in Def. 13 is that it is defined on hypers directly, and can be generalised to proper measures, thus extending discrete distributions [5]. But in the case (as here) where we remain discrete, there is an equivalent matrix-style characterisation:

Lemma 14: **Refinement of joint-distributions [4], [8]**
Let $J_S{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}_S$ and $J_I{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}_I$ be joint-distribution matrices, both of them *reduced* in the sense of Def. 5, such that $[\![J_{S,I}]\!]{=}\Delta_{S,I}$ resp.[21] Then

$$\Delta_S \sqsubseteq \Delta_I \quad \text{iff} \quad J_S \cdot R = J_I \tag{3}$$

for some stochastic *refinement matrix* $R{:}\,\mathcal{Y}_S{\twoheadrightarrow}\mathcal{Y}_I$. Note that the state-spaces of $\Delta_{S,I}$ are the same, but their observation spaces $\mathcal{Y}_{S,I}$ can differ.

Proof: Illustrated in App. F; sketch proof in App. G. □

With Lem. 14 the reflexivity and transitivity of relation $(\sqsubseteq)$ is clear from elementary matrix properties. For antisymmetry we refer to [6, Thm 6], whose supporting Lemma 1 there we adapt to suit our purposes here:

Definition 15: **Expected value** For distribution $\delta{:}\,\mathbb{D}\mathcal{Z}$ and function $f{:}\,\mathcal{Z}{\to}V$ for vector space V, the expected value of f on δ is $\mathcal{E}_\delta\, f{:=}\,\sum_{z{:}\,\mathcal{Z}}\, \delta_z{\times}f.z$, where $\sum$ and $(\times)$ are taken in the vector space.[22] □

We will be using $\mathcal{E}$ principally over hypers, i.e. the case $\mathcal{Z} = \mathbb{D}\mathcal{X}$ in the definition.

Lemma 16: **(Strict) monotonicity** Given are two hypers $\Delta_{S,I}{:}\,\mathbb{D}^2\mathcal{X}$ and a strictly concave function $f{:}\,\mathbb{D}\mathcal{X}{\to}\mathbb{R}^{\geq}$.

If $\Delta_S{\sqsubset}\Delta_I$ then $\mathcal{E}_{\Delta_S}\, f < \mathcal{E}_{\Delta_I}\, f$. And if f is (non-strictly) concave, then $\Delta_S{\sqsubseteq}\Delta_I$ implies $\mathcal{E}_{\Delta_S}\, f \leq \mathcal{E}_{\Delta_I}\, f$.

Proof: Proved for abstract channels in [6, Lem 1]; the proof for hypers is essentially identical. □

We now have antisymmetry, because $\Delta_S{\sqsubseteq}\Delta_I{\sqsubseteq}\Delta_S$ and $\Delta_S{\neq}\Delta_I$ implies $\Delta_S{\sqsubset}\Delta_I{\sqsubset}\Delta_S$ whence we have from Lem. 16 the contradiction $\mathcal{E}_{\Delta_S}\, f < \mathcal{E}_{\Delta_I}\, f < \mathcal{E}_{\Delta_S}\, f$ for any strictly concave $f{:}\,\mathbb{D}\mathcal{X}{\to}\mathbb{R}^{\geq}$ of our choice (for example Shannon entropy).

Hyper-space $\mathbb{D}^2\mathcal{X}$ also admits a metric, the Kantorovich metric [17] based on the Manhattan metric on $\mathbb{D}\mathcal{X}$ (§VI). It is used for continuity properties (as we will see §VII-A).

VI. Monads: Giry, Kleisli and Kantorovich

With $\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$ we have given a discrete model of abstract *HMM*'s, suitable for interpreting probabilistic sequential programs with hidden state, together with concrete programming examples (Figs. 3–5). We now show how that embeds into structures based on a Giry monad.

The Giry/Lawvere monad based on the category *Mes* of measurable spaces comprises an endofunctor Π and two natural transformations η (unit) and μ (multiply) [2]; following [1] we take that as a basis for the denotation of computations. We have been using $\mathbb{D}$ as a specialisation of Π to the discrete case, to construct sets of *discrete* probability distributions, with unit-function $[\cdot]$ specialising η that makes a point distribution, and multiply-function avg specialising μ that takes the average of a distribution (of distributions); typically we have $[\cdot]{\in}\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$ and $\mathsf{avg}{\in}\mathbb{D}^2\mathcal{X}{\to}\mathbb{D}\mathcal{X}$.

The functions $\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$ are arrows in the related Kleisli category $(\Pi, \eta, -^\dagger)$, and our Kleisli composition for them (§IV-B) is from there.

Van Breugel compares the Giry *Mes*-monad to a "metric monad" in which a functor $\mathcal{B}$ maps 1-bounded compact metric spaces (S,d) to sets $\mathcal{B}S$ with Borel probability measures generated from the topology of the Kantorovich metric — which itself is derived from the underlying metric d on S [17]. He shows that the Giry- and metric monads are related: if from a metric space (S,d) you generate the Borel algebra $\mathcal{S}$ and thence via Giry the measurable space $(\underline{S}, \underline{\mathcal{S}}) = \Pi(S, \mathcal{S})$, then the Giry-generated σ-algebra $\underline{\mathcal{S}}$ on $\underline{S}$ is the same as the one generated by the Kantorovich metric on $\underline{S}$ derived from the original d on S.

Our use of those metrics is that we begin with a finite space $\mathcal{X}$ and we give it the discrete metric d_1. Our space $\mathbb{D}\mathcal{X}$ of

[21]Recall that the $\mathcal{X}$ here in type $\mathcal{X}{\twoheadrightarrow}\mathcal{Y}_S$ is the final-, not the initial state.

[22]More generally it is $\int f\, \mathrm{d}\delta$ and requires measurability of f.

discrete distributions on $\mathcal{X}$ inherits the Kantorovich metric based on d_1, which is in fact (exactly $1/2$ times) the Manhattan metric d_M on $\mathbb{D}\mathcal{X}$, that is where $d_M(\delta^1,\delta^2) = \sum_x |\delta^1_x - \delta^2_x|$. And our hyper-space $\mathbb{D}^2\mathcal{X}$ has the Borel algebra generated by Π from $\mathbb{D}\mathcal{X}$, which is determined by the Kantorovich metric derived from d_M. This means for us that Kantorovich continuity implies Giry measurability.

We cannot however use van Breugel's monad $\mathcal{B}$ directly because the denotations $[\![H]\!]$ of *HMM*'s in $\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$ are not 1-Lipschitz in general, and 1-Lipschitz is a condition he imposes. They are however continuous (Lem. 17 to come); and so we use Giry instead, which imposes only measurability (implied by continuity).

Because we continue to use finite $\mathcal{X}$ below, all functions in $\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$ are trivially measurable, thus arrows in the monad. In spite of that, we will occasionally indicate where measurability would apply in a more general treatment. On the other hand, Kantorovich-continuity of functions in $\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$ is not automatic, since $\mathbb{D}\mathcal{X}$ is nondenumerable; and so we cannot assume that the Kleisli composition $f_1; f_2$ of two continuous functions $f_{1,2}{:}\, \mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$ is itself continuous: that must be proved (Lem. 21).

Giry proposes a second probability monad based on the category *Pol* of Polish spaces (rather than measurable spaces, as for *Mes* above). It is possible that this second monad simplifies our development, if the Kantorovich metric metrises the weak topology. In that case, Lem. 21 would follow directly from Giry's construction. [23]

VII. Characteristics of $\mathbb{H}\mathcal{X}$, the abstract *HMM*'s

A. *Continuity and super-linearity*

The semantic function $[\![\cdot]\!]$ (Def. 9) takes *HMM* matrices in $\mathcal{X}{\twoheadrightarrow}\mathcal{Y}{\times}\mathcal{X}$ to functions in $\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$; but not all of those are denotations $[\![H]\!]$ for some H. We now describe two important characteristics satisfied by $[\![H]\!]$ as H ranges over *HMM*'s: they are continuity and super-linearity. We will define abstract *HMM*'s $\mathbb{H}\mathcal{X}$ to be the functions satisfying those conditions.

Our first condition concerns continuity wrt. the Kantorovich metrics on $\mathbb{D}\mathcal{X}$ and $\mathbb{D}^2\mathcal{X}$.

Lemma 17: **Denotations of *HMM*'s are continuous** For all $H{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}{\times}\mathcal{X}$ we have that $[\![H]\!]$ is a continuous function in $\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$ wrt. the Kantorovich metrics.

Proof: Given in App. H. □

Our second condition concerns linear combinations.

Definition 18: **Weighted sum** For $\delta_{1,2}{:}\,\mathbb{D}\mathcal{X}$ we write $\delta_1{}_p{+}\delta_2$ for the weighted sum of the two distributions, so that $(\delta_p{+}\delta')_x = p\delta_x + (1{-}p)\delta'_x$. [24] □

Lemma 19: **Denotations of *HMM*'s are super-linear** For all $H{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}{\times}\mathcal{X}$ we have

$$[\![H]\!].\pi_1\ {}_p{+}\ [\![H]\!].\pi_2 \quad \sqsubseteq \quad [\![H]\!].(\pi_1\ {}_p{+}\ \pi_2)\ , \qquad (4)$$

where $(\sqsubseteq)$ is refinement as defined in Def. 13. [25]

[23] We thank a referee for this observation.

[24] Note that $\delta_p{+}\delta'$ and $\delta_p{\oplus}\delta'$ differ: the former is a single distribution made from merging δ,δ'; the latter is a hyper with support δ,δ'.

[25] Super-linearity can also be seen as a form of monotonicity. See App. I1.

Proof: Given in App. H. □

Motivated by those two lemmas, we now define

Definition 20: **The space $\mathbb{H}\mathcal{X}$ of abstract *HMM*'s** We write $\mathbb{H}\mathcal{X}$ for those h in $\mathbb{D}\mathcal{X}{\to}\mathbb{D}^2\mathcal{X}$ satisfying Lemmas 17,19, i.e. that are Kantorovich-continuous and that satisfy $h.\pi_1{}_p{+}h.\pi_2 \sqsubseteq h.(\pi_1{}_p{+}\pi_2)$. □

Thus our two lemmas above establish that $[\![H]\!]{\in}\mathbb{H}\mathcal{X}$ for any classical *HMM* H.

Since we will therefore be restricting our denotations to $\mathbb{H}\mathcal{X}$, a subset of the arrows in the Giry monad, we expect $\mathbb{H}\mathcal{X}$ to be closed under composition.

Lemma 21: **Abstract *HMM*'s closed under composition** For any two $h_{1,2}{:}\,\mathbb{H}\mathcal{X}$ we have $h_1; h_2 \in \mathbb{H}\mathcal{X}$ as well, where (;) is as in Def. 11.

Proof: Although a direct proof is possible, the result is much easier once we have introduced "uncertainty" transformers (§VIII), then becoming a consequence of Thm. 29 and in particular its Cor. 30, which depends crucially on the dual view we develop in §VIII. □

It is shown in App. I2 that composition in $\mathbb{H}\mathcal{X}$ is monotonic with respect to the refinement order $(\sqsubseteq)$.

This completes our construction of our forward, abstract semantics for *HMM*'s. We now propose a dual view.

VIII. A dual view: uncertainty measures, and their transformers

A. *Uncertainty measures, and their relation to refinement*

"Uncertainty measures" generalise the diversity of entropy measures (e.g. Shannon), the functions from distributions to reals that measure increasing disorder.

Definition 22: **Uncertainty measure** An *uncertainty measure* over $\mathcal{X}$ is a Kantorovich-continuous- and concave function in $\mathbb{D}\mathcal{X}{\to}\mathbb{R}^{\geq}$, i.e. one taking distributions (on $\mathcal{X}$ in this case) to non-negative reals. It is intended that a distribution's greater uncertainty indicates more resilience (less vulnerability) to the distributions's being exploited by an adversary. [26]

We write $\mathbb{U}\mathcal{X}$ for the uncertainty measures over $\mathcal{X}$, and call them "*UM*'s" in the text for brevity. □

A typical example of a *UM* applied to a hyper is as follows. Given prior $\pi{:}\,\mathbb{D}\mathcal{X}$ and channel $C{:}\,\mathcal{X}{\twoheadrightarrow}\mathcal{Y}$, the resulting hyper is $\Delta{:=}[\![\pi{\triangleright}C]\!]$ and the "conditional u uncertainty" of that (compare conditional Shannon entropy) would be $\mathcal{E}_\Delta\, u$. This could be compared to the uncertainty $u.\pi$ of the prior, to give a "u-leakage" of the channel on that prior.

There is a compelling connection between *UM*'s (Def. 22) and refinement (Def. 13, Lem. 14): we have

Lemma 23: **Soundness and completeness of uncertainty measures [6]** For any hypers $\Delta_{1,2}{:}\,\mathbb{D}^2\mathcal{X}$ we have

$$\Delta_1 \sqsubseteq \Delta_2 \quad \text{iff} \quad \mathcal{E}_{\Delta_1} u \leq \mathcal{E}_{\Delta_2} u \quad \text{for all } u{:}\,\mathbb{U}\mathcal{X}.$$

□

We regard "only if" as *soundness* in the sense that if we have a witness to the refinement relation $\Delta_1{\sqsubseteq}\Delta_2$, i.e. either $\underline{\Delta}$ (Def. 13) or R (Lem. 14), then no *UM* can show Δ_2 to be

[26] Smith's "vulnerability measure" based on Bayes Risk [9] is an uncertainty measure except that it goes in the opposite direction.

less uncertain than Δ_1. It is related to the *Data-Processing Inequality*, as explained in [6].

We regard "if" as *completeness* in the sense that if refinement fails, that is $\Delta_1 \not\sqsubseteq \Delta_2$, then there is a *UM* demonstrating the failure [4]–[6], [8].

In App. J is background on the proof of Lem. 23, whose completeness part was originally called "Coriaceous" [8].

B. Abstract HMM's to UM-transformers

In §III we introduced a "forward" denotational view of *HMM*'s that takes initial distributions to final hypers. Here we take the dual view, where an *HMM* takes a "post-uncertainty" to a "pre-uncertainty".

Definition 24: **Uncertainty-measure transformers**
Take $h{:}\,\mathbb{H}\mathcal{X}$ and $u{:}\,\mathbb{U}\mathcal{X}$. Define the *uncertainty transformer* wp.h of type $\mathbb{U}\mathcal{X}{\to}\mathbb{U}\mathcal{X}$ so that for any $u{:}\,\mathbb{U}\mathcal{X}$ and $\pi{:}\,\mathbb{D}\mathcal{X}$ we have

$$\mathrm{wp}.h.u.\pi \quad := \quad \mathcal{E}_{h.\pi}\, u \ ,$$

where on the right we are taking the expected value of u on the hyper $h.\pi$. (Because u is continuous, it is measurable.) By analogy with weakest preconditions for ordinary sequential programming [18], the *UM*-transformer wp.h takes a *UM* to be applied *after* h and produces a *UM* that equivalently can be applied *before* h. (Compare also [19], [20], [21] for probabilistic/demonic sequential programs.) □

In Lem. 25 we show well definedness of Def. 24, that is that wp.$h.u$ is indeed in $\mathbb{U}\mathcal{X}$.

Lemma 25: **Well-definedness of Def. 24**
If $h{:}\,\mathbb{H}\mathcal{X}$ is an abstract *HMM* and $u{:}\,\mathbb{U}\mathcal{X}$ is a *UM*, then wp.$h.u$ is in $\mathbb{U}\mathcal{X}$.

Proof: See App. K. □

C. Characteristic properties of wp.h

For $h{:}\,\mathbb{H}\mathcal{X}$ the *UM*-transformer wp.h has a number of characteristic properties.

Lemma 26: **wp.h is linear and total** For every $h{:}\,\mathbb{H}\mathcal{X}$ and $t = \mathrm{wp}.h$ we have that t is:

1) *linear* so that for $a_{1,2}{:}\,\mathbb{R}^{\geq}$ and $u_{1,2}{:}\,\mathbb{U}\mathcal{X}$ we have

$$t.(a_1 u_1 + a_2 u_2) \quad = \quad a_1 t.u_1 + a_2 t.u_2 \ ;$$

2) *monotonic*, so that $t.u_1.\delta \geq t.u_2.\delta$ for every $u_1{\geq}u_2$ with $u_{1,2}{:}\,\mathbb{U}\mathcal{X}$ and $\delta{:}\,\mathbb{D}\mathcal{X}$; [27] and
3) *total*, so that $t.\mathbf{1}{=}\mathbf{1}$ where $\mathbf{1}.\delta{:=}\,1$ for all $\delta{:}\,\mathbb{D}\mathcal{X}$.

Proof: These properties are immediate from Def. 24. □

A further property of *UM*-transformers is that they are 1-Lipshitz in a certain sense:

Lemma 27: **wp.h is 1-Lipschitz** Take $h{:}\,\mathbb{H}\mathcal{X}$ and define $t{:=}\,\mathrm{wp}.h$. Let $|\cdot|$ be absolute value. Then t is 1-Lipschitz in the sense that

$$\sup_{\delta:\,\mathbb{D}\mathcal{X}} |t.u_1.\delta - t.u_2.\delta| \quad \leq \quad \sup_{\delta:\,\mathbb{D}\mathcal{X}} |u_1.\delta - u_2.\delta| \ .$$

Proof: See App. M. □

[27]We lift $\geq$ pointwise.

Motivated by those lemmas, we define uncertainty transformers to be exactly the functions in $\mathbb{U}\mathcal{X}{\to}\mathbb{U}\mathcal{X}$ that satisfy the properties listed.

Definition 28: **The uncertainty transformers $\mathbb{T}\mathcal{X}$**
The uncertainty transformers $\mathbb{T}\mathcal{X}$ are the functions in $\mathbb{U}\mathcal{X}{\to}\mathbb{U}\mathcal{X}$ that satisfy Lems. 26,27. □

We note that transformers $\mathbb{T}\mathcal{X}$ are closed under composition. In App. I3 we show that refinement for $\mathbb{T}\mathcal{X}$ is pointwise $(\leq)$.

D. UM-transformers back to abstract HMM's

The function wp.$(\cdot)$ has been shown to be of type $\mathbb{H}\mathcal{X}{\to}\mathbb{T}\mathcal{X}$. Here we show that this correspondence is exact, i.e. that for every $t{:}\,\mathbb{T}\mathcal{X}$ there is an $h{:}\,\mathbb{H}\mathcal{X}$ such that $t{=}\mathrm{wp}.h$ and, moreover, that h is unique.

The following theorem thus establishes the exact correspondence between $\mathbb{H}\mathcal{X}$ and $\mathbb{T}\mathcal{X}$, giving an analog for hidden-state probabilistic programs to the well known correspondence between demonic relations and conjunctive predicate transformers [18], or between demonic/probabilistic programs and super-linear expectation transformers [20], [22].

Theorem 29: **Characterisation of transformers** For any $t{:}\,\mathbb{T}\mathcal{X}$ there is a unique $h{:}\,\mathbb{H}\mathcal{X}$ such that $t{=}\mathrm{wp}.h$.

Proof: Details are given in App. N. □

With these characterisations, we can prove two technical facts. In the discrete case (as earlier) they seem self-evident. In the more general setting, however, the work is mainly in ensuring well definedness (e.g. that only measurable functions are integrated, etc.) The first establishes the usual connection between composition, this time between the forwards- and backwards semantics; the second confirms that $\mathbb{H}\mathcal{X}$ is closed under composition (i.e. preserves continuity and super-linearity, Lem. 21).

Corollary 30: **Transformer composition**
For any $h_{1,2}{:}\,\mathbb{H}\mathcal{X}$ we have that also $h_1;h_2{\in}\mathbb{H}\mathcal{X}$, and furthermore that $\mathrm{wp}.(h_1;h_2) = \mathrm{wp}.h_1 \circ \mathrm{wp}.h_2$.

Proof: Direct calculation shows that $\mathrm{wp}.(h_1;h_2) = \mathrm{wp}.h_1 {\circ} \mathrm{wp}.h_2$, although the working is intricate in the general (Giry) case. Well definedness of $h_1;h_2$ itself uses the simpler properties of (functional) composition on the transformer side. See App. O. □

Also, transformer composition respects refinement (App. I4).

IX. Gain- and loss functions define uncertainty measures

A. Gain- and loss functions

Although Def. 22 of uncertainty measures is abstract, they can be made concrete via gain functions [8] or equivalently "loss functions" [4, Eqn. (5)] that encode an attacker's (e.g.) economic interest in the secrets and the cost of obtaining them. We use loss functions here.

Definition 31: **Loss function determines uncertainty measure** A *loss function* ℓ is of type $I{\to}\mathcal{X}{\to}\mathbb{R}^{\geq}$ for some index set I, with the intuitive meaning that $\ell.i.x$ is the cost to the attacker of using "attack strategy" i when the hidden value turns out actually to be x. His expected cost for an attack planned but not yet carried out is then $\mathcal{E}_\delta\,(\ell.i)$ if δ is the distribution in $\mathbb{D}\mathcal{X}$ he believes to be governing x currently.

From such an ℓ we define an uncertainty measure

$$U_\ell.\rho \quad := \quad \inf_{i:\,I}\ \mathcal{E}_\rho\,(\ell.i)\ . \tag{5}$$

When I is finite, the inf can be replaced by min. $\square$
The inf represents a rational strategy of minimising cost or risk, and a typical attacker will act as follows: he chooses the attack strategy (i.e. he chooses i) whose expected cost $\mathcal{E}_\rho\,(\ell.i)$ to him, where ρ is the posterior in $\mathbb{D}\mathcal{X}$ he infers from his observations in $\mathcal{Y}$, will be the least.

Lemma 32: **Well-definedness for Def. 31** For any loss function $\ell{:}\,I{\to}\mathcal{X}{\to}\mathbb{R}^{\geq}$ the function U_ℓ in Def. 31 is continuous and concave.

Proof: We give here the proof for the finite-I case. (The infinite case is considered in [23]; it might require further assumptions on I.) Let ℓ be a loss function and U_ℓ be the associated uncertainty measure.

$\underline{U_\ell \text{ is concave}}$: Take $\rho_{1,2}{:}\,\mathbb{D}\mathcal{X}$ and $p{:}\,[0,1]$. We have

$$\begin{array}{lll}
 & U_\ell.(\rho_1{}_p{+}\rho_2) & \\
= & \min_{i:\,I}\,(\mathcal{E}_{\rho_1{}_p+\rho_2}\,\ell.i) & \text{“definition } U_\ell\text{”} \\
= & \min_{i:\,I}\,(\mathcal{E}_{\rho_1}\,\ell.i_p{+}\mathcal{E}_{\rho_2}\,\ell.i) & \text{“}\lambda\delta\cdot\mathcal{E}_\delta\,u \text{ is linear”} \\
\geq & \quad(\min_{i:\,I}\mathcal{E}_{\rho_1}\,\ell.i) & \text{“}(\min f)_p{+}(\min g) \\
 & {}_p{+}\quad(\min_{i:\,I}\mathcal{E}_{\rho_2}\,\ell.i) & \leq \min(f_p{+}g)\text{”} \\
= & U_\ell.\rho_1{}_p{+}U_\ell.\rho_2\ . & \text{“definition } U_\ell\text{”}
\end{array}$$

$\underline{U_l \text{ is continuous}}$: Since I is finite and each function $(\lambda\rho\cdot\mathcal{E}_\rho\,\ell.i)$ is continuous (Lem. 37 in App. K), applied to $\mathbb{D}\mathcal{X}$ instead of $\mathbb{D}^2\mathcal{X}$), the function U_ℓ is also continuous. $\square$
Remarkably, loss functions are *complete* for uncertainty measures: any uncertainty measure in $\mathbb{U}\mathcal{X}$ can be expressed as U_ℓ for some loss function ℓ in $I{\to}\mathcal{X}{\to}\mathbb{R}^{\geq}$, but possibly requiring I to be infinite [23]. Roughly speaking, this is because of the way concave functions can be expressed as the minimum of their tangential hyperplanes: the coefficients of the hyperplanes' normals are the loss functions.[28]

It is compellingly shown elsewhere how versatile loss (equiv. gain) functions are [8]. Of particular interest is that Lem. 23 applies, both in the discrete [4] and the continuous cases [5], even when uncertainties are restricted to those generated by loss functions: the “distinguishing witness” constructed for completeness is in fact a loss function [4].

X. A *UM*-TRANSFORMER EXAMPLE FOR §IV-D

A. *Profiling an attacker with a loss function*

In the context of Fig. 5 we imagine an attacker whose livelihood depends on his guessing whether `xs[0]=xs[1]` or not, finally. If he guesses incorrectly he loses \$1; if correctly, he breaks even (loses \$0). This is as much a mathematical- as a *social* issue: attacks will be discouraged if they are not worthwhile for the attacker in terms of his own criteria. (See also App. D for this social aspect.)

[28]For example Shannon entropy requires infinite I, and the encoding is then related to minimising the Kullback-Leibler divergence.

In this example, following §IX, we express the attacker's criteria as two strategies “guess same” and “guess different” (thus $I = \{\mathsf{same}, \mathsf{diff}\}$) and a loss function ℓ defined

$$\begin{array}{clclclcl}
 & \ell.\mathsf{same}.(\mathtt{00}) & = & 0 & \ell.\mathsf{diff}.(\mathtt{00}) & = & 1 & \\
\dagger & \ell.\mathsf{same}.(\mathtt{01}) & = & 1 & \ell.\mathsf{diff}.(\mathtt{01}) & = & 0 & \ddagger \\
 & \ell.\mathsf{same}.(\mathtt{10}) & = & 1 & \ell.\mathsf{diff}.(\mathtt{10}) & = & 0 & \\
 & \ell.\mathsf{same}.(\mathtt{11}) & = & 0 & \ell.\mathsf{diff}.(\mathtt{11}) & = & 1\ , &
\end{array}$$

based on the informal description just above: for example if $\mathtt{xs}{=}\mathtt{01}$ but he guesses same, the case indicated by †, then he loses \$1; but if he guesses diff, he breaks even ‡. Using (5) we define our *UM* as $u.\delta = U_\ell.\delta =$

$$\begin{array}{rccc}
 & \ell.\mathsf{same}.\delta & \min & \ell.\mathsf{diff}.\delta \\
= & \mathcal{E}_\delta\,(\ell.\mathsf{same}) & \min & \mathcal{E}_\delta\,(\ell.\mathsf{diff}) \\
= & (\delta_{00}{+}\delta_{11}) & \min & (\delta_{01}{+}\delta_{10})\ .
\end{array}$$

B. *Using UM's and transformers to plan an attack*

We can use our transformer semantics to answer u-dependent questions about Fig. 5 over *all* priors: we use the two we chose earlier in §IV-D as examples.

Writing $[\![P]\!]$ for the abstract *HMM* denoted by the two lines of code in Fig. 5, we have for any π that

$$\begin{array}{rcl}
\mathrm{wp}.[\![P]\!].u.\pi & = & \pi_{00}\ \min\ (\pi_{01}{+}\pi_{10})/2 \\
 & + & \pi_{11}\ \min\ (\pi_{01}{+}\pi_{10})/2\ .
\end{array} \tag{6}$$

(See App. P below for how this $\mathrm{wp}.(\cdot)$ is calculated.)

Now let π^5 be the prior described by the initial comment in Fig. 5. The attacker's (expected) uncertainty wrt. the *final* hyper $[\![P]\!].\pi^5$ is $\mathrm{wp}.[\![P]\!].u$ applied to that *initial* (uniform) prior π^5, that is $\mathrm{wp}.[\![P]\!].u.\pi^5 = 1/2$ directly from (6). Since $u.\pi^5$ is also $1/2$, he is indifferent wrt. whether he should attack before or after P has been allowed to run.

Now suppose that $\mathtt{xs}[0]{=}1$ is known initially, thus with prior π being $(0, 0, 1/2, 1/2)$ so that u applied initially gives $u.\pi{=}1/2$. But u applied finally would give $\mathrm{wp}.[\![P]\!].u.\pi = (0 \min 1/4) + (1/2 \min 1/4) = 1/4 < 1/2$, so that it is better to attack later even though `xs` might have been altered by P. This scenario confirms that in fact for some priors, the program in Fig. 5 cannot be regarded as secure.

XI. *HMM*'S AND THE DALENIUS DESIDERATUM

Our abstracting from initial-state correlations allows a semantics for programs' *final* states alone. Sometimes however leakage from the *initial* state is important, even if that state is overwritten by the markov part of the *HMM*: what the initial state *was* might reveal information about what some other correlated state still *is*, even if that other state is not mentioned in the program at all. This general concern was raised wrt. statistical databases by Dalenius [10] who argued that it is inescapable; Dwork later gave a proof of this [11].

With our constructions here, we are able to see the Dalenius effect in programming terms. A “classical” sequential program does not affect variables to which it does not refer; for example `x:= E` does not affect some other variable `y` in any way. But the program `print x` (recalling the notation of Fig. 4) can affect *what we know* about variable y even though it does not refer to y at all.

Consider for example an input distribution $(0,0)@{}^1\!/\!_2, (1,1)@{}^1\!/\!_2$ on two variables (x,y). Its y-marginal distribution is uniform on $\{0,1\}$. But the output hyper of that program, projected onto y, is $[0]@{}^1\!/\!_2, [1]@{}^1\!/\!_2$, showing that the distribution on y is now a point, no longer uniform:[29] with probability $^1\!/\!_2$ that point will be [0], and with probability $^1\!/\!_2$ that point will be [1]. Reading the printout of x tells us which point distribution on y we have got, and we have essentially the Dalenius effect between "database" x, "query" `print x` and "third-party data" y.

This effect is exacerbated when we include state updates, as we have done with our abstract *HMM*'s here. (Updates were not considered originally by Dalenius or by Dwork.) For then the program `print x; x:= 0` and `x:= 0` have the same abstract-*HMM* semantics on state-space (just) x, but different semantics on state-space x, y.[30] The Dalenius effect has become, in programming terms, a failure of compositionality wrt. unreferenced global variables.

We show in this section how to deal with that: in brief, we include both the initial- and the final values of the state in our semantics. The crucial point is that we do not have to do more than that, in particular that we do not have to consider "all possible third-party data y of any type".

We now address the details. Consider a "constant" markov $M^{\mathsf{x}}_{x,x'} = \mathit{1\ if\ } x'{=}\mathsf{x}\ \mathit{else\ 0}$ for some fixed $\mathsf{x}{:}\mathcal{X}$. Then $[\![C{:}M^{\mathsf{x}}]\!] = [\![{:}M^{\mathsf{x}}]\!]$ for any channel C — any leaking by C of the initial state is ignored, because that state is overwritten by x. We now adapt our framework so that it is *not* ignored.

Let $C{:}\,\mathcal{X}{\rightarrow\!\triangleright}\mathcal{Y}$ be a channel and $M{:}\,\mathcal{X}{\rightarrow\!\triangleright}\mathcal{X}$ a markov, with $\mathcal{Z}$ fresh. Write $C_{\times\mathcal{Z}}$ in $(\mathcal{X}{\times}\mathcal{Z}){\rightarrow\!\triangleright}\mathcal{Y}$ for the expanded channel

$$(C_{\times\mathcal{Z}})_{(x,z),y} \quad := \quad C_{x,y} \ .$$

Similarly $M_{\times\mathcal{Z}}{:}\,(\mathcal{X}{\times}\mathcal{Z}){\rightarrow\!\triangleright}(\mathcal{X}{\times}\mathcal{Z})$ is given by

$$(M_{\times\mathcal{Z}})_{(x,z),(x',z')} \quad := \quad M_{x,x'} \text{ if } z{=}z' \text{ else } 0 \ .$$

These definitions ensure that for $\pi{:}\,\mathbb{D}(\mathcal{X}{\times}\mathcal{Z})$ neither $\pi{\triangleright}(C_{\times\mathcal{Z}})$ nor $\pi{\triangleright}(M_{\times\mathcal{Z}})$ depends on the $\mathcal{Z}$ component. Take for example $\mathcal{Z}{:=}\{z_0,z_1\}$ consider C, M as below:

$$C \;=\; \begin{array}{r} \\ x_0{:} \\ x_1{:} \end{array}\!\!\begin{array}{c} \begin{array}{cc} y_0 & y_1 \end{array} \\ \begin{pmatrix} 1 & 0 \\ ^1\!/\!_4 & ^3\!/\!_4 \end{pmatrix} \end{array} \qquad M \;=\; \begin{array}{r} \\ x_0{:} \\ x_1{:} \end{array}\!\!\begin{array}{c} \begin{array}{cc} x_0 & x_1 \end{array} \\ \begin{pmatrix} ^1\!/\!_2 & ^1\!/\!_2 \\ ^1\!/\!_2 & ^1\!/\!_2 \end{pmatrix} \end{array}$$

$$C_{\times\mathcal{Z}} \quad = \quad \begin{array}{r} \\ (x_0,z_0){:} \\ (x_0,z_1){:} \\ (x_1,z_0){:} \\ (x_1,z_1){:} \end{array}\!\!\begin{array}{c} \begin{array}{cc} y_0 & y_1 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ ^1\!/\!_4 & ^3\!/\!_4 \\ ^1\!/\!_4 & ^3\!/\!_4 \end{pmatrix} \end{array}$$

[29]Since the output is a hyper, if knowledge of y were unaffected we would have the point hyper on the uniform distribution, that is $[0@{}^1\!/\!_2, 1@{}^1\!/\!_2]$.

[30]On state-space x, both programs produce the output hyper [[0]] that denotes "x is certainly 0." On x, y however, the first might reveal something about y while the second cannot.

$$M_{\times\mathcal{Z}} \quad = \quad \begin{array}{r} \\ x_0z_0{:} \\ x_0z_1{:} \\ x_1z_0{:} \\ x_1z_1{:} \end{array}\!\!\begin{array}{c} \begin{array}{cccc} x_0z_0 & x_0z_1 & x_1z_0 & x_1z_1 \end{array} \\ \begin{pmatrix} ^1\!/\!_2 & 0 & ^1\!/\!_2 & 0 \\ 0 & ^1\!/\!_2 & 0 & ^1\!/\!_2 \\ ^1\!/\!_2 & 0 & ^1\!/\!_2 & 0 \\ 0 & ^1\!/\!_2 & 0 & ^1\!/\!_2 \end{pmatrix} \end{array} .$$

The definitions above show that in $C_{\times\mathcal{Z}}$ the rows of the original C are each repeated $2 = \#\mathcal{Z}$ times; and the subsequent update by $M_{\times\mathcal{Z}}$ leaves $\mathcal{Z}$ unchanged. Observe that these definitions now account for information flows with respect to initial distributions $\mathbb{D}(\mathcal{X}{\times}\mathcal{Z})$ where, crucially, the $\mathcal{Z}$ component is merely "carried along". But it captures the Dalenius effect mentioned, as in the following scenario.

Consider an initial distribution $\pi{:}\,\mathbb{D}(\mathcal{X}{\times}\mathcal{Z})$ such that $\pi_{x_i,z_j} = 1$ if and only if $i{=}j$. We see that, even though $\mathcal{Z}$ is not accessed by the program at all, if ever y_1 is observed then the $\mathcal{Z}$ component must certainly be z_1, and if y_0 is observed then it is 4 times more likely to be z_0 than z_1.

Although $\mathcal{Z}$ is arbitrary, it can be shown that this Dalenius effect on any $\mathcal{Z}$ can be determined by the *HMM* semantics *specifically in the case where* $\mathcal{X}{=}\mathcal{Z}$. That is, we do not have to consider "all $\mathcal{Z}$'s", which would be impractical. By projecting onto the $\mathcal{Z}$ component, $[\![C_{\times\mathcal{Z}}{:}M_{\times\mathcal{Z}}]\!]$ as a whole acts as a pure channel, and if $\#\mathcal{Z}{\geq}\#\mathcal{X}$ then the component matrices C and M can be completely recovered from observations made only on the composition $[\![C_{\times\mathcal{Z}}{:}M_{\times\mathcal{Z}}]\!]$.

XII. Related work

There is great diversity in approaches to information flow in (probabilistic) programs, which we have surveyed in our own earlier work [4]–[7]. Here we concentrate on general techniques for semantic constructions, in particular those based on monads, duality and refinement.

Refinement of probabilistic programs appeared in [24] where evaluations were used to construct a powerdomain for probabilistic but possibly non-terminating computations; this was extended to include demonic choice in the discrete case in [20], [22], and was significantly generalised in [25]. Our "uncertainty refinement" that combines information flow with functional properties first appeared for information flow in straight-line programs in [4], was extended to general measure spaces [5] and appeared independently for the specific case of channels [8]. Whereas Jones and Plotkin began with an underlying partial order over which to construct a probability space, our uncertainty-refinement order begins "one level up", using hyper-distributions $\mathbb{D}^2\mathcal{X}$ to encode an "attack model" that accounts for information flow.

Doberkat defines *stochastic relations* that correspond to forward-semantic functions of type $\mathcal{X}{\rightarrow}\mathbb{D}\mathcal{X}$ for Markov processes: these are what we generalise by going "one level up". The converse of those stochastic relations [14] might improve the presentation of our Def. 5, where a hyper is extracted from a channel and a prior, i.e. from a joint distribution.

Dual models for program semantics include [18], then for probabilistic programs [19], [21] in the purely probabilistic case. Subsequently [20] added demonic choice. And [25], [26] study dual models for probability and nondeterminism using a version of Riesz's representation theorem.

In particular, Goubault-Larrecq's approach [26] to combining probability and nondeterminism differs from our earlier work [20]. It uses general denotations for probabilistic programs in which nondeterminism is introduced at the level of measures (by weakening the modularity law) rather than as healthy sets of measures [20], [22], [25]. That leads naturally to a backward semantics of probabilistic demonic programs because nondeterminism is captured within integration. There is thus a strong analogy between our *UM*-transformers and Goubault-Larrecq's "previsions" because both are continuous functionals that act on some set of tests (bounded continuous functions). The main difference is that our *UM*-transformers are specifically tailored to capture security semantics, which is what leads to concavity on our set of uncertainty measures. Notice moreover that Goubault-Larrecq encounters a difficulty similar to our composition of *HMM*'s, that the decomposition $[\![C{:}M]\!]$ (resp. collinearity) is not preserved by Giry composition. Indeed, both difficulties are resolved by working in a larger space, namely, the space of abstract *HMM*'s (resp. not-necessarily-collinear continuous previsions).

In [27] a dual model for Markov processes is used to prove properties about approximations of finite behaviours, and in [28] it is shown how expectation transformers relate to explicit program models described by Markov processes.

Recently Jacobs and Hasuo have explored a general categorical construction of a backward transformer semantics from a forward monadic model of probabilistic computations (discrete, continuous and quantum) [29], [30]. Their construction uses measurability as the underlying feature of "predicates", while the stronger condition of continuity is crucial for our uncertainty measures. It would be interesting to see whether an instantiation of that categorical derivation can provide more structure for what we have done.

The concave functions advocated here for analysing information-flow properties have appeared in [5], [8] and have been identified in [31] as an ingredient in privacy analysis.

XIII. Conclusions and prospects

Our principal objective was to provide an abstract setting for *HMM*'s based on well understood principles of semantic spaces. We did that using Giry's general monadic framework applied at the level of $\mathbb{D}\mathcal{X}$ (rather than $\mathcal{X}$); the resulting structures include a refinement order which is sensitive to both functional *and* information-flow properties, and they lead to a dual, transformer space supported by theorems demonstrating the duality.

More abstractly (recall §I-B), we aimed to profit by joining two ideas: the established use of *HMM*'s as descriptions of probabilistic mechanisms having hidden state, and the established use of monads for modelling computations. Our novel use of $\mathbb{D}\mathcal{X}$ in the monad, rather than the state $\mathcal{X}$ itself, is the principal innovation that allowed this; and the synthesised hyper-distribution space that results leads to other advantages (†'s below).

An immediate benefit accrues because, in monad-enabled programming languages, probabilistic-programming packages can be built very quickly and e.g. [32] is just one of many examples. Indeed the translation into real programs is almost elementary because of the powerful and general structures available: a prototype has very recently been constructed by Schrijvers [33], independently verifying the examples in Figs. 3–6. (See App. E for an overview.)

More importantly, any monad brings with it both general equational properties and specific properties applying to the monad in question (such as those in [2]). These conceptual tools allow reasoning about the structures modelled (*HMM*'s in this case) in ways that would be obscured by their more direct operational representation (e.g. as matrices).

† The other advantages of hypers are several: one is that they abstract from differences between entropies in a way that allows all of the entropies to be used uniformly. For example, a hyper contains all the information necessary to calculate the information leakage of a particular program fragment (typically, in the security literature, a pure channel §III-D), as shown in [6], and furthermore the Kantorovich-metric structure of $\mathbb{D}\mathcal{X}$ we used earlier for channels [7] now carries over to *HMM*'s.

† Another advantage of hypers is that their partial-order enables semantics for "looping *HMM*'s" in the standard way (least fixed-point) for computer science, rather than a direct ad-hoc definition based on matrices. Indeed a typical use of *HMM*'s is to run a single *HMM*-step (§III-A) repeatedly and then to make statistical deductions about its hidden features: sophisticated mathematical tools are available for this special case [15]. Via abstract *HMM*'s we can however, in principle, handle complex, heterogeneous systems beyond (what amounts to, in the special case just above) a single loop containing just a single statement.

Our more concrete aim (again §I-B) was to allow source-level reasoning about probabilistic programs with hidden state. Historically *at the source level* this works best with backwards reasoning based on predicates (or similar) that can be embedded between program statements rather than forwards reasoning which, here, would be calculations using $\mathbb{D}\mathcal{X}{\rightarrow}\mathbb{D}^2\mathcal{X}$ directly.

Here our "predicates" are *UM*'s, which in this paper however are mathematical objects unsuitable for embedding directly in program texts (see App. P, last paragraph) As remarked in §IX-A, however, any *UM* can be expressed as U_l for some loss-function l which function –crucially– is indeed an expression based on program variables [23]. The added complexity introduced by the hidden state is that the program-logic based on that observation must represent the index-set (I) of the loss function; that would most likely be done by adding a special-purpose quantifier (since the loss-function index must be a bound variable within the assertion, not appearing in the program proper).

Exploiting this opportunity for a source-level quantitative logic of probabilistic hidden state is planned for future work.

ACKNOWLEDGEMENTS We're grateful for advice from Franck van Breugel, James Worrell, Tom Schrijvers and other members of IFIP WG 2.1, and for inspiration and insight from the INRIA Princess team. And we thank the referees particularly for their suggestions of related work and possible improvements. We acknowledge support from the Australian Research Council's grant DP120101413 and the INRIA équipe associée Princess; and Morgan acknowledges the support of NICTA, which is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre-of-Excellence Program.

REFERENCES

[1] E. Moggi, "Computational lambda-calculus and monads," in *Proc. 4th IEEE Symp. LiCS*, 1989, pp. 14–23.
[2] M. Giry, "A categorical approach to probability theory," in *Categorical Aspects of Topology and Analysis*, ser. Lecture Notes in Mathematics. Springer, 1981, vol. 915, pp. 68–85.
[3] A. McIver, L. Meinicke, and C. Morgan, "Hidden-Markov program algebra with iteration," *Mathematical Structures in Computer Science*, 2014.
[4] ——, "Compositional closure for Bayes risk in probabilistic noninterference," in *Proc. 37th Int. Colloq. ICALP 2010, Part II*, 2010, pp. 223–235.
[5] ——, "A Kantorovich-monadic powerdomain for information hiding, with probability and nondeterminism," in *Proc. 27th Symp. LiCS*, 2012, pp. 460–70.
[6] A. McIver, C. Morgan, G. Smith, B. Espinoza, and L. Meinicke, "Abstract channels and their robust information-leakage ordering," in *Proc. 3rd Conf. PoST (ETAPS)*, ser. Lecture Notes in Computer Science, M. Abadi and S. Kremer, Eds., vol. 8414. Springer, 2014, pp. 83–102.
[7] M. S. Alvim, K. Chatzikokolakis, A. McIver, C. Morgan, C. Palamidessi, and G. Smith, "Additive and multiplicative notions of leakage, and their capacities," in *Proc 27th IEEE Symp. CSF*. IEEE, 2014, pp. 308–322.
[8] M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith, "Measuring information leakage using generalized gain functions," in *Proc. 25th IEEE Symp. CSF*, Jun. 2012, pp. 265–79.
[9] G. Smith, "On the foundations of quantitative information flow," in *Proc. 12th Conf. FoSSaCS (ETAPS)*, ser. Lecture Notes in Computer Science, L. de Alfaro, Ed., vol. 5504, 2009, pp. 288–302.
[10] T. Dalenius, "Towards a methodology for statistical disclosure control," *Statistik Tidskrift*, vol. 15, pp. 429–44, 1977.
[11] C. Dwork, "Differential privacy," in *Proc. 33rd Int. Colloq. ICALP*, 2006, pp. 1–12.
[12] A. McIver, C. Morgan, and T. Rabehaja, "Abstract hidden Markov models: a monadic account of quantitative information flow," 2015, includes draft appendices. At `www.cse.unsw.edu.au/~carrollm/probs/bibliographyBody.html#lics15`.
[13] D. Fremlin, *Measure Theory*. Torres Fremlin, 2000.
[14] E. Doberkat, "The converse of a stochastic relation," in *Proc. 6th Conf. FoSSaCS (ETAPS)*, ser. LNCS, A. Gordon, Ed., vol. 2620. Springer-Verlag, 2003, pp. 233–49.
[15] D. Jurafsky and J. Martin, *Speech and Language Processing*. Prentice Hall International, 2000.
[16] J. Landauer and T. Redmond, "A lattice of information," in *Proc. 6th IEEE CSFW'93*, Jun. 1993, pp. 65–70.
[17] F. van Breugel, "The metric monad for probabilistic nondeterminism," 2005, `www.cse.yorku.ca/~franck/research/drafts/monad.pdf`.
[18] E. Dijkstra, *A Discipline of Programming*. Prentice-Hall, 1976.
[19] D. Kozen, "A probabilistic PDL," in *Proc. 15th ACM Symp. Theory of Computing*. ACM, 1983, pp. 291–7.
[20] C. Morgan, A. McIver, and K. Seidel, "Probabilistic predicate transformers," *ACM Trans Prog Lang Sys*, vol. 18, no. 3, pp. 325–53, 1996.
[21] C. Jones, "Probabilistic nondeterminism," Edinburgh University, Monograph ECS-LFCS-90-105, 1990, (Ph.D. Thesis).
[22] A. McIver and C. Morgan, *Abstraction, Refinement and Proof for Probabilistic Systems*, ser. Tech Mono Comp Sci. Springer, 2005.
[23] K. Chatzikokolakis, Private communications, 2014.
[24] C. Jones and G. Plotkin, "A probabilistic powerdomain of evaluations," in *Proc. 4th IEEE Symp. LiCS*, 1989, pp. 186–95.
[25] R. Tix, K. Keimel, and G. Plotkin, "Semantic domains for combining probability and non-determinism," *Electron. Notes Theor. Comput. Sci.*, vol. 222, pp. 3–99, 2009.
[26] J. Goubault-Larrecq, "Continuous previsions," in *Proc. 16th EACSL*, ser. Lecture Notes in Computer Science, vol. 4646. Springer, 2007, pp. 542–57.
[27] P. Chaput, V. Danos, P. Panangaden, and G. D. Plotkin, "Approximating Markov processes by averaging," *J. ACM*, vol. 61, no. 1, 2014.
[28] F. Gretz, J. Katoen, and A. McIver, "Operational versus weakest pre-expectation semantics for the probabilistic guarded command language," *Perform. Eval.*, vol. 73, pp. 110–132, 2014.
[29] B. Jacobs, "Measurable spaces and their effect logic," in *Proc. 28th LiCS*, 2013, pp. 83–92.
[30] I. Hasuo, "Generic weakest precondition semantics from monads enriched with order," in *Proc. CMCS*, ser. LNCS, M. Bonsangue, Ed., vol. 8446. Springer, 2014, pp. 10–32.
[31] D. Kifer and B.-R. Lin, "Towards an axiomatization of statistical privacy and utility," 2010, Penn State Technical report: CSE-10-002.
[32] M. Erwig and S. Kollmansberger, "Probabilistic functional programming in Haskell," *Journal of Functional Programming*, vol. 16, pp. 21–34, 2006.
[33] T. Schrijvers, "A monadic model for computations that leak secrets," 2015, `www.cse.unsw.edu.au/~carrollm/LiCS-TS.html`.
[34] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. John Wiley & Sons, Inc., 2006.
[35] D. Blackwell, "The comparison of experiments," in *Proc. 2nd Berkely Symp. Mathematical Statistics and Probability*. Univ. California Press, 1951, pp. 93–102.
[36] R. G. Bartle, *The Elements of Integration*. John Wiley & Sons, Inc., 1966.
[37] J. R. Munkres, *Topology*. Prentice Hall, 1999.
[38] M. H. Stone, "The generalized Weierstrass approximation theorem," *Math Magazine*, vol. 21, no. 4, pp. 167–184, March 1948.
[39] M. Bačák and J. M. Browein, "On difference convexity of locally Lipschitz functions," *Optimization: A Journal of Math Prog and Oper Research*, vol. 60, no. 8-9, pp. 961–978, 2011.
[40] L. H. Loomis and S. Sternberg, *Advanced Calculus*. Jones and Bartlett Publishers, 1990.
[41] K. R. Parthasarathy, *Probability Measures on Metric Spaces*. AMS Chelsea Publishing, 1967.
[42] R. Ranga Rao, "Relations between weak and uniform convergence of measures with applications," *Annals of Mathematical Statistics*, vol. 33, no. 2, pp. 659–680, January 1962.

APPENDIX

Appendices can be found at [12].