Monoidal Computer: Towards Cryptography in Pictures

Dusko Pavlovic
Royal Holloway
July 2012

Outline

Problem
Computability and basic monoidal computer
Complexity and comprehensive monoidal computer
Cryptography and randomized monoidal computer
Summary

Abstraction

Pablo Picasso

Abstraction?

Roy Lichtenstein

Gunther von Hagens
Natural numbers

Abstract Specification (Peano)

| sorts: | \( \mathbb{N} \) |
| operations: | \( 0 : 1 \to \mathbb{N} \) |
| | \( \sigma : \mathbb{N} \to \mathbb{N} \) |
| axioms: | \( 0 \neq \sigma x \) |
| | \( \sigma x = \sigma y \implies x = y \) |

Concrete implementation

\[
0 = \{\} \\
1 = \{0\} \\
2 = \{0, 1\} \\
3 = \{0, 1, 2\} \\
\ldots \\
n = \{0, 1, 2, \ldots, n-1\}
\]

Practice requires abstraction

Problem
Do we really have to count to \( m + n \) to add \( m \) and \( n \)?

Solution
Numeral systems: rich data abstractions

Computer

Concrete implementation (von Neumann)

von Neumann computer
Concrete implementation (Turing)

Turing Machine

Concrete implementation (Church)

\[(\lambda x. f x) a = f a\]
\[(\lambda x. f x) = f\]

Lambda Calculus

Concrete implementation (Lambek)

Abacus

Concrete implementation (von Neumann)

Cellular automata

Abstract specification

Requirements:
- simple interface
- no implementation details
- easy access to all implementations
Outline

Problem
Computability and basic monoidal computer
Monoidal computer
Arithmetic and logic
Basic theorems
Fixed points
Kleene’s Theorem
Halting Problem
Rice’s Theorem
Complexity
Cryptography
Summary

Data service

Monoidal category

A
B
C
D

"data types"
"computations"
"data"
"sequential composition"

"data types"
"computations"
"data"
"parallel composition"
**Data service**

**Monoidal category**

- "data types"
- "computations"
- "data"
- "sequential composition"
- "parallel composition"

**Monoidal category with data service**

- "data types"
- "computations"
- "data"
- "sequential composition"
- "parallel composition"
- "copying and deleting"

**Diagrammatic notations**

- **Commutative diagram**
- **String diagram**

**Convolution**

**Remark (cf. Frobenius algebra)**

Data service boils down to a Frobenius algebra structure, such that the induced convolution

\[ f \cap g = f g \]

is idempotent, i.e.

\[ f \cap f = f \]
Definition
A basic monoidal computer is a (strict symmetric) monoidal category with data service, and with
\[(B)\] universal data type \(B\) such that all data are its tuples
\[
\forall A \exists l. A = \bigotimes_i B^m_i
\]

Examples
The classical basic monoidal computer \(C\) consists of
- objects: \(\mathcal{O} = \{1, B, B^2, B^3, \ldots\}\)
  - where \(B = \{0, 1\}\)
- morphisms: partial computable functions
- \(u^B_m\): a universal Turing machine
  - with \(m\) input tapes and \(n\) output tapes
- \(\sigma_{\text{Kleene}}\): a Kleene \(smn\)-function

Definition
A basic monoidal computer is a (strict symmetric) monoidal category with data service, and with
\((u)\) universal evaluators \(u^C_{MN} : B \otimes M \to N\), such that every computation is evaluation of some program
\[
\forall f : M \to N \exists p : I \to B.
\]

Theorem 1
A strict symmetric monoidal category \(C\) is a basic monoidal computer if and only if
- every \(A \in C\) is a finite tensor power of an idempotent Frobenius algebra \(B\), and
- for every \(M, N \in C\) there is a family of surjections
\[
C(X, B) \xrightarrow{\rho^B} C(X \otimes M, N)
\]
natural in \(X\).
Basic monoidal computer

Examples

The extensional basic monoidal computer $\mathcal{E}$ consists of

- objects: $|\mathcal{E}| = \{I, B\}$ for $B = R \cong R^R \cong R \times R$
  - in a category of domains
- morphisms: continuous
- $u_R^I$: the image of $\text{id}_R$ along $R^R \cong R^{R \times R}$
- $s_R^I$: the image of $u_R^I$ along $R^{R \times R} \cong R^{(R \times R) \times R}$

Notation

- Kleene bracket:
  $\{ p \} = p$

Basic monoidal computer

Notation

- notational abuse (when confusion is unlikely)

$\lambda$-calculus

Proposition 2

Every monoidal computer provides a model for untyped nonextensional $\lambda$-calculus.

Proof

- the terms are $t = \frac{B}{B}$

- the application is $pa = \frac{p}{p}$

- the abstraction is $\lambda x. px = \frac{p}{x}$
Natural numbers

Corollary 3

Every monoidal computer contains Church’s presentation of natural numbers:

\[
\begin{align*}
\lambda px.x &= 1
\end{align*}
\]

Total computations

Notation

\[
\begin{align*}
\mathbb{B} &= C(I, \mathcal{B}) \\
\mathbb{N} &= \{0, 1, 2, \ldots n \ldots \} \subseteq \mathbb{B}
\end{align*}
\]

Remark

\[
\begin{align*}
\mathbb{B}^\mathbb{N} &\rightarrow \mathbb{B}^\mathbb{N} \text{ in } \mathcal{C}
\end{align*}
\]

Total computations

Definition

A computation \( L \xrightarrow{f} M \) in \( \mathcal{C} \) is total if it maps numbers to numbers, i.e. restricts

\[
\begin{align*}
\mathbb{B}^\mathbb{N} &\rightarrow \mathbb{B}^\mathbb{N} \text{ in } \text{Set} \\
\mathbb{N}^\mathbb{N} &\rightarrow \mathbb{N}^\mathbb{N} \text{ in } \text{Set}
\end{align*}
\]

Total computations

Definition

A monoidal computer is numeric if

\((\alpha)\) every computation has a numeric program

\[
\begin{align*}
\forall f : L \rightarrow M, \exists p : L \rightarrow \mathcal{N}, f = u^M_{\mathcal{L}}(p \otimes L)
\end{align*}
\]

\((\beta)\) partial evaluation is total

\[
\begin{align*}
\forall N : \mathbb{N} \otimes L \rightarrow \mathbb{N}
\end{align*}
\]

\((\gamma)\) there is a computation \( \downarrow_\mathbb{N} : B \rightarrow B \) such that

\[
\begin{align*}
\downarrow_\mathbb{N} x = \begin{cases} 
1 & \text{ if } x \in \mathbb{N} \\
0 & \text{ otherwise}
\end{cases}
\end{align*}
\]

Remark

In a numeric computer, \( L \xrightarrow{f} M \) is total if and only if

\[
\forall n \in \mathbb{N}, \downarrow_\mathbb{N} fn = 1
\]
### Numeric computers

**Examples**
- numeric: classical, quantum computers
- Kleene’s Fixed Point Theorem
- not numeric: extensional computer

In this talk we study numeric computers.

### Arithmetic

**Theorem 4**

*In any monoidal computer there is a weak Natural Numbers Object*

\[
\begin{align*}
0 & \mapsto B \\
\sigma & \mapsto \lambda px. \rho(px)
\end{align*}
\]

The values of \((g, h)\) are determined uniquely only on \(N \subseteq B\).
Arithmetic

More functions

Logic

Definition
A computation $L \rightarrow B$ is a predicate if its numeric outputs are in $\{0, 1\} \subseteq B$.

More precisely, $L \rightarrow B$ is a predicate if

$\downarrow_{\{0, 1\}} f = 1$ where $\downarrow_{\{0, 1\}} = \neg \rho$
Logic

Remark
\[ \downarrow_{\{0,1\}} \text{ is not uniquely determined, because} \]
\[ \downarrow_{\{0,1\}} x = \begin{cases} 1 & \text{if } \rho x = 0 \\ 0 & \text{if } \rho x = \sigma^{1+10} \\ ? & \text{otherwise} \end{cases} \]

A predicate may thus evaluate to
- numeric values 0 or 1, or to
- any non-numeric values.

Logic

Notation
- denote predicates by Greek letters \( \alpha, \phi, \ldots \)
- abbreviate \( \phi x = 1 \) to \( \phi x \)
  \( \phi x = 0 \) to \( \neg \phi x \)

Fixed points

Proposition 5

Every computation of a monoidal computer has a fixed point.

Fixed points

Definition

Lemma 6

Proof of Proposition 5.
Fixed points

Corollary 7

Monoidal computer where $B = \mathbb{N}$ must be trivial.

Proof sketch.

\[ \bot \quad \bot = \sigma \bot \]

$\bot \in \mathbb{N} \implies \mathbb{N} = B \text{ finite} \implies \mathbb{N} \text{ trivial}

since $\bot \in \mathbb{N}$ but $\bot = \bot \quad \bot \quad \bot \quad \bot \quad \bot$

$\bot \in \mathbb{N}$, $\Phi$ is not total

\[ \square \]

Corollary 8

$\bot$ if fixed by all recursive functions.

More precisely, if $L \xrightarrow{f} M$ is defined by recursion, then

\[ f(\bot, \ldots, \bot) = \bot = \underbrace{\Phi(\bot, \ldots, \bot)}_{\text{where} \cdot} \]

Proof.

Structural induction over the recursion schema using

\[ f(\bot, \bot, \ldots, \bot) = f(\bot, \sigma \bot, \sigma \bot, \ldots, \sigma \bot) \]

\[ \square \]
**Kleene’s Fixed Point Theorem**

**Theorem 9**

For every total computation \( L \rightarrow \mathbb{N} \) and any two types \( L, M \) there is a program \( I : L \rightarrow \mathbb{N} \) which evaluates to the same computation \( L \rightarrow M \) like \( t_p \), i.e.

\[
U^L_I(p, x) = U^M_I(t_p, x)
\]

**Definition**

\[ M \rightarrow \begin{array}{c}
B \rightarrow L \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\end{array} \]

**Proof.**

\[ M \rightarrow \begin{array}{c}
B \rightarrow L \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\end{array} \]

**Corollary 10**

For every computation \( B \rightarrow \mathbb{N} \) there is a program \( I : B \rightarrow \mathbb{N} \) such that

\[
|p| x = f(|p|, x)
\]

**Fixed program**

\[ M \rightarrow \begin{array}{c}
B \rightarrow L \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
\end{array} \]
Fixed program

Proof.
Apply the Fixed Point Theorem to the total computation

Halting Problem

Definition
A program $p$ is said to halt on $n \in \mathbb{N}$ if $(p) n \in \mathbb{N}$.

The halting predicate $H$ is thus defined

Recall
A computation $L \xrightarrow{\varphi} B$ is a predicate if
$$\forall x \in L. \varphi x \in \mathbb{N} \implies \varphi x \in \{0,1\}$$

Definition
A predicate $L \xrightarrow{\varphi} B$ is decidable if it is total, and hence
$$\forall x \in L. x \in \mathbb{N} \implies \varphi x \in \{0,1\}$$

Theorem 11 (Turing)
The halting predicate is undecidable.
Halting Problem

Intuition

- \( L \models B \) is nontrivial if
  \[\begin{align*}
  &\{x \in L \mid \varphi x\} \neq \emptyset \\
  &\{x \in L \mid \neg \varphi x\} \neq \emptyset
  \end{align*}\]
- \( L \models \bot \) then maps
  \[\begin{align*}
  &\{x \in L \mid \varphi x\} \overset{\varphi}{\rightarrow} \{x \in L \mid \neg \varphi x\} \\
  &\{x \in L \mid \neg \varphi x\} \overset{\neg}{\rightarrow} \{x \in L \mid \varphi x\}
  \end{align*}\]

Halting Problem

Lemma 12
The predicate \( B \models \downarrow \) is nontrivial.

Proof.
By the single-case extension of the recursion schema, define \( \sim \), abbreviated to \( \sim \), by
\[\sim x = \begin{cases} 
\bot & \text{if } x \in N \\
1 & \text{if } x = \bot
\end{cases}\]
The other values are arbitrary.

Halting Problem

Proof that \( H \) is undecidable

If \( H \) is decidable then

\[\begin{align*}
B \models \sim \Phi & \iff \sim \Phi \\
\downarrow & \iff \downarrow
\end{align*}\]

is decidable too. But this is impossible, because...
Rice’s Theorem

Definition
A predicate \( B \xrightarrow{\alpha} B \) is extensional (i.e., over computations) if for all \( p, q \in B \)
\[ \{p\} = \{q\} \implies \alpha p = \alpha q \]

Theorem 13 (Rice)
Every nontrivial predicate over computations is undecidable.

Proof
Let \( B \xrightarrow{\sim} B \) be a nontrivial extensional predicate. Since \( B \xrightarrow{\sim} B \) is nontrivial, there is \( B \xrightarrow{\sim} B \), again abbreviated to \( \sim \) such that \( \sim p = \sim p \).

Define
\[
\begin{align*}
\begin{array}{c}
\text{if } \alpha \text{ is by assumption extensional,} \\
\{ p \} = \{ \neg p \} & \implies \alpha(p) = \alpha(\neg p) \\
\text{and thus}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Outline}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Problem}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Computability and basic monoidal computer}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Complexity and comprehensive monoidal computer}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Length}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Complexity}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Comprehension}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Ensembles}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Cryptography and randomized monoidal computer}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{Summary}
\end{array}
\end{align*}
\]
Upto relations

**Definition**

For \( f, g : A \rightarrow \mathbb{N} \), we define the \( \leq \) relations as follows:

\[
\begin{align*}
 f & \leq g \iff \exists c \in \mathbb{N} \forall n \in \mathbb{N}. \ f(n) \leq c + g(n) \\
 f & \equiv g \iff f \leq g \land g \leq f
\end{align*}
\]

Length

**Definition**

The length is a total computation \( f : B \rightarrow B \) satisfying:

\[
\begin{align*}
 f & \leq n \\
 f(p; q) & \equiv f(p) + f(q) \equiv f(p \otimes q)
\end{align*}
\]

Complexity measures

**Definition**

A complexity measure is a computation \( cx_L : B \otimes L \rightarrow B \), such that:

- \( cx_L(p, x) \) is defined iff \( (p)x \) is defined
- \( cx_L(f, x) \equiv f(x) \)

**Proposition 14**

For every complexity measure \( cx_L \), there is a decidable predicate \( T_{cx_L} : B \otimes L \otimes B \rightarrow 2 \) such that:

\[
T_{cx_L}(p, x, n) \iff cx_L(p, x) \leq n
\]

Proposition 14

For every complexity measure \( cx_L \), there is a decidable predicate \( T_{cx_L} : B \otimes L \otimes B \rightarrow 2 \) such that:

\[
T_{cx_L}(p, x, n) \iff cx_L(p, x) \leq n
\]

**Definition**

A time complexity measure is a computation \( tm_L : B \otimes L \rightarrow B \), such that:

\[
\begin{align*}
 tm_L(p, q, x) & \equiv tm_L(p, q) + tm_L(q, x) \\
 tm_{L\otimes M}(p \otimes q, x ; y) & \equiv \max \{ tm_L(p, x), tm_M(q, y) \}
\end{align*}
\]
Complexity measures

Definition
A space complexity measure is a computation \( sp_L : B \otimes L \rightarrow B \), such that

\[
sp_L(p \circ q, x) = \max \{ sp_L(p, qx), sp_L(q, x) \}
\]

\[
sp_{\text{data}}(p \otimes q, x \otimes y) = sp_L(p, x) + sp_L(q, y)
\]

Remark
The crucial feature of the quantum computer is that superposition allows a nonstandard form of parallel composition of exponentially many computations (where the outputs cannot be completely separated in the end), but at a linear cost in space.

Comprehension

Definition
Comprehension is a family of type constructors

\[
\lambda : C(L, B) \rightarrow \text{Sub}(L)
\]

\[
\left( L \rightarrow 2 \subseteq B \right) \rightarrow \left( \lambda \mid x \right) L
\]

such that for every \( X \xrightarrow{\phi} L \) with \( \phi x = 1 \), there is a unique \( X \xrightarrow{\psi} \left( \lambda \mid x \right) L \) through which \( x \) factors.
Comprehensive monoidal computer

Definition
A comprehensive monoidal computer is a (strict symmetric) monoidal category with data service, and with

\( (B) \) universal data type \( B \) such that all data types are included into its tensors

\[ \forall A \exists i. A \mapsto B^i \]

Comprehensive monoidal computer

Definition
A comprehensive monoidal computer is a (strict symmetric) monoidal category with data service, and with

\( (\mathcal{E}) \) comprehension

\[ i_L : C(L, B) \to \text{Sub}(L) \]

\[ L \to 2 \subseteq B \to \{ x \mid x \in L \} \]

Comprehensive monoidal computer

Definition
A comprehensive monoidal computer is a (strict symmetric) monoidal category with data service, and with

\( (u) \) universal evaluators \( u^N_M : B \otimes M \to N \), such that every computation is evaluation of some program

\[ \forall f : M \to N \exists p : I \to B. N^f_M = N^u_p \]

Comprehensive monoidal computer

Example
The comprehensive classical monoidal computer \( C \) consists of

- objects: \( \{C\} = \) recursively enumerable subsets of \( \{0, 1\}^* \), including
  - finite types
    \[ \{0, 1\}^n = \{ x \in \{0, 1\}^* \mid |x| = n \} \]
  - complexity classes
    \[ \text{tm}_n = \{ p \mid \forall n \in \mathbb{N}. \forall x, p, x \leq fn \} \]
  - ...
- morphisms: ...

Comprehensive monoidal computer

Hierarchy Theorem
Use the Halting Problem construction to prove that there is a program which is in \( cx(f^{i+1}) \setminus cx(f^i) \).
Ensembles of computations

Definition
An ensemble of types in a comprehensive monoidal computer \( C \) is a pair \( \langle A, \alpha \rangle \) where
- \( A \) is a type and
- \( \alpha : \mathbb{N} \to \mathbb{N} \) is a total function with
  \[ m < n \implies a_m < a_n \]
which together represent the tower
\[
A_{\alpha_0} \quad A_{\alpha_1} \quad A_{\alpha_2} \quad \ldots \quad A
\]
where \( A_n = \{ x \in A \mid f(x) = n \} \).

Ensembles of computations

Definition
An ensemble of computations in a comprehensive monoidal computer \( C \) consists of
- computation \( f \in C(L, M) \)
- ensembles of types \( \langle L_{\alpha_0} \rangle_{\alpha_0} \) and \( \langle M_{\alpha_0} \rangle_{\alpha_0} \) such that the restrictions \( f_j = f \upharpoonright L_j \) land in \( M_j \).

Stable computations

Definition
Computations \( f : L \to M \) such that for all \( x, y \in \mathbb{N} \) holds
\[ \ell x = \ell y \land \ell x \uparrow \ell y \implies \ell(f(x)) = \ell(f(y)) \]
are called stable.

Lemma 16
Every stable computation in a monoidal computer extends to a unique ensemble of computations.
Monoidal computer of ensembles

Theorem 17

Let $C$ be a comprehensive monoidal computer. Then the induced ensembles of types and computations form a comprehensive monoidal computer $C_{\text{ens}}$.

Ensembles

Proof idea.

The monoidal computer structure of $C_{\text{ens}}$ is constructed from the monoidal computer structure of $C$ as follows:

- the Frobenius structures in $C_{\text{ens}}$ are ensembles of the Frobenius structures in $C$.
- universal evaluator $u^{\alpha^{(\lambda)}}_{\lambda} \in C_{\text{ens}}$ inputs not only a program $p$ and the data $x$, but also the programs for $\lambda$ and $\mu$ and the parameter $i$, and then it
  - verifies $(x = \lambda)$
  - computes $px = u^\lambda(p,x)$
  - verifies $(i px) = \mu i$
  - outputs $px$
- partial evaluator $s(L,i;M^{(\alpha)})$ computes similarly.

Outline

Problem

Computability and basic monoidal computer

Complexity and comprehensive monoidal computer

Cryptography and randomized monoidal computer

Randomized categories

Randomized computations

Indistinguishability

One-way functions

Summary

Finitary distributions

Definition

A distribution over a set $S$ is a map $\Phi : S \to [0, 1]$ which is

- zero at all but finitely many points

$$\# \{ x \in S \mid \Phi x \neq 0 \} < \infty$$
- subunitary

$$\sum_{x \in S} \Phi x \leq 1$$

$\mathcal{D} S$ denotes the set of all distributions over $S$.

Randomized categories

Definition

The randomized version $S_{\text{r}}$ of a category $S$ has

- objects: $|S_{\text{r}}| = |S|$.
- morphisms: $S_{\text{r}}(A, B) = \mathcal{D}(A, B)$
- composition:

$$\Phi : S(A, B) \to [0, 1] \quad \psi : S(B, C) \to [0, 1]$$

$$\langle \psi \circ \Phi \rangle : S(A, C) \to [0, 1]$$

$$\langle \psi \circ \Phi \rangle_h = \sum_{g|f = h} \psi_g \cdot \Phi_f$$

Randomized monoidal categories

Definition

The randomized version $S_{\text{r}}$ of a monoidal category $S$ has

- $\ldots$
- tensor:

$$\Phi : S(A, B) \to [0, 1] \quad \Theta : S(C, D) \to [0, 1]$$

$$\langle \Phi \otimes \Theta \rangle : S(A \otimes C, B \otimes D) \to [0, 1]$$

$$\langle \Phi \otimes \Theta \rangle_h = \sum_{f \otimes g = h} \Phi_f \cdot \Theta_g$$
Randomized monoidal categories

Remark
The identity-on-the-objects functor
\[ S \xrightarrow{\cdot} S \]
\[ f \mapsto U_f = \begin{cases} 1 & \text{if } f = g \\ 0 & \text{otherwise} \end{cases} \]
creates
- identities
- any other equationally defined structure
  - data service (Frobenius algebras)
  - universal evaluators
  - partial evaluators...

Randomized computations

Definition
Let \( C \) be a monoidal computer, \( S \in |C| \) and \( S = C(I, S) \) a finite type.

A randomized computation \( f(P) : L \to M \) is the distribution constructed as follows
\[ f \in C(S \otimes L, M) \]
\[ P \in \Omega \]
\[ f(P) \in DC(L, M) \]
\[ f(P)_{\phi} = \sum_{s \in \{ f(p) = g \}} P_p \]
where
- \( f : S \otimes L \to M \) is the underlying computation
- \( P : C(I, S) \to [0,1] \) is the distribution of random seeds

Randomized computations (continued)

Definition
Let \( C \) be a monoidal computer, \( S \in |C| \) and \( S = C(I, S) \) a finite type.

A randomized computation \( f(P) : L \to M \) is the distribution constructed as follows
\[ f \in C(S \otimes L, M) \]
\[ P \in \Omega \]
\[ f(P) \in DC(L, M) \]
\[ f(P)_{\phi} = \sum_{s \in \{ f(p) = g \}} P_p \]
where
- \( f : S \otimes L \to M \) is the underlying computation
- \( P : C(I, S) \to [0,1] \) is the distribution of random seeds

Randomized computations (continued)

Idea

- random variable \( P : C(I, S) \to [0,1] \) supplying the random seeds can be represented by
- formal variable \( I \xrightarrow{r} S \) in the polynomial category \( C[x : S] \)

Randomized computations (continued)

Idea (continued)

- \( \ldots \) since \( I \xrightarrow{r} S \) can represent an arbitrary computation \( I \xrightarrow{r} S \)

Randomized computations (continued)

Idea (continued)

- \( \ldots \) or \( I \xrightarrow{r} S \) can represent an arbitrary distribution \( I \xrightarrow{p} S \)
Randomized monoidal computer

Definition

A randomized monoidal computer is a polynomial extension $C[x,y,z,... : S]$ of a monoidal computer $C$.

Randomized monoidal computer

Comments

- interpreting variables as distributions induces the randomized computations

\[(x : S) \mapsto (P : DE) \quad (y : S) \mapsto (Q : DE) \quad \ldots\]

\[C[x,y,z,... : S] \rightarrow C_D\]

Randomized monoidal computer

Equality of randomized computations

Question

How do we estimate the probability that two randomized computations will produce the same output on a given input?

Equality of randomized computations

- $f \cap g$ produces an output for $x$

  iff $fx = gx$

- $f = g$ is thus an internal proposition in $C[x]$
Equality of randomized computations

- $f \cap g$ produces an output for $x$ iff $fx = gx$
- $f = g$ is thus an internal proposition in $C[x]$
- interpreting $x \mapsto P$ yields
  \[
  \downarrow = (f \cap g) \circ P = \Pr(f(x) = g(x) \mid x \not\in P)
  \]

Equality of randomized computations

- $f \cap g$ produces an output for $x$ iff $fx = gx$
- $f = g$ is thus an internal proposition in $C[x]$
- interpreting $x \mapsto P$ yields
  \[
  \downarrow = (f \cap g) \circ P = \Pr(f(x) = g(x) \mid x \not\in P)
  \]
  truth values are probabilities
  (i.e. distributions over 1)

Randomized ensembles

Definition

A randomized ensemble $f_i(P) : L_i \rightarrow M_i$ is the distribution constructed as follows

\[
\begin{aligned}
  &\exists \alpha \in C(S_i @ L_i, M_i) & P_i \in D S_i, \\
  & f_i(P_i) \in D(C(L_i, M_i)) \\
  & f_i(P)_g = \sum_{s_{i,j}(x_i) = g} P_i \rho_i
\end{aligned}
\]

Equality of randomized ensembles

- $f_i \cap g_i$ produces an output for $x_i$ iff $f_i x_i = g_i x_i$
- $f_i = g_i$ is thus an internal proposition in $C[x_i]$
- interpreting $x_i \mapsto P_i$ yields
  \[
  \downarrow = (f_i \cap g_i) \circ P_i = \Pr(f_i x_i = g_i x_i \mid x_i \not\in P_i)
  \]

Indistinguishability

Idea

An ensemble of propositions $\alpha_i : 1 \rightarrow 2$ consists of
- a computation $\alpha : R \rightarrow 2$
- total increasing computation $\rho : N \rightarrow N$, such that
\[
\alpha_i = \frac{\# \{ r \in R_1 \mid \alpha r = \alpha \}}{|R_1|}
\]
- both $\alpha$ and $\rho$ are feasibly computable
### Categorization

**Indistinguishability**

**Idea**
- An ensemble of propositions \(\alpha : 1 \leftrightarrow 2\) consists of:
  - a computation \(\alpha : R \rightarrow 2\)
  - total increasing computation \(\rho : N \rightarrow N\), such that
    \[\alpha_i = \frac{1}{|R|} \sum_{\forall r \in R_i} \mathbf{1}(\alpha_r = 1)\]
- both \(\alpha\) and \(\rho\) are feasibly computable
- Two ensembles of propositions \(\alpha_i, \beta_i : 1 \leftrightarrow 2\) are indistinguishable if finding the different outputs of \(\alpha, \beta\) requires unfeasible computational resources
  - e.g., going exponentially high up the ensemble \(|R|\)

**One-way function**

**Definition**
A computable ensemble \(f : L \rightarrow M\) is one-way if for all feasible ensembles \(h : M \rightarrow L\) holds

\[
\sim : L_i \otimes L_i \rightarrow 2
\]

**Trapdoor function**

**Definition**
A computable ensemble \(f : L \rightarrow M\) is trapdoor if for all feasible ensembles \(h : D \otimes M \rightarrow L\) holds

\[
\sim : L_i \otimes L_i \rightarrow 2
\]

**Outline**

- Problem
  - Computability and basic monoidal computer
  - Complexity and comprehensive monoidal computer
  - Cryptography and randomized monoidal computer

**Summary**