

Operational semantics for disintegration

Chung-chieh Shan (Indiana University)

Norman Ramsey (Tufts University)

Mathematical Foundations of Programming Semantics

2016-05-25

What is semantics for?

1. Language for composing generative stories

normal 0 10

(monadic bind, unit)

2. Guarantee that terms denote distributions

(product measure *tricky*)

What is semantics for?

1. Language for composing generative stories

```
do { $x \leftarrow$  normal 0 10;  
     normal  $x$  1}
```

(monadic **bind**, unit)

2. Guarantee that terms denote distributions

(product measure *tricky*)

What is semantics for?

1. Language for composing generative stories

```
do { $x \leftarrow$  normal 0 10;  
     return  $e^x$ }
```

(monadic bind, **unit**)

2. Guarantee that terms denote distributions

(product measure *tricky*)

What is semantics for?

1. Language for composing generative stories

```
do { $x \leftarrow \text{normal } 0\ 10$ ;  
      return  $e^x$ }
```

(monadic bind, unit)

2. Guarantee that terms denote distributions

```
do { $x \leftarrow m$ ;  
      do { $y \leftarrow n$ ;  
          return  $(x, y)$ }}
```

(product measure *tricky*)

What is semantics for?

1. Language for composing generative stories

```
do { $x \leftarrow$  normal 0 10;  
      return  $e^x$ }
```

(monadic bind, unit)

2. Guarantee that terms denote distributions

```
do { $x \leftarrow m$ ;  
      do { $y \leftarrow n$ ;  
          return ( $x, y$ )}} = do { $x \leftarrow m$ ;  
       $y \leftarrow n$ ;  
      return ( $x, y$ )}
```

(product measure *tricky*)

What is semantics for?

3. Equational reasoning by semantics-preserving rewriting!

Fubini

$$\text{do } \{x \leftarrow m; \\ y \leftarrow n; \\ \text{return } (x, y)\} = \text{do } \{y \leftarrow n; \\ x \leftarrow m; \\ \text{return } (x, y)\}$$

Conjugacy

$$\text{do } \{x \leftarrow \text{normal } 0 \ 1; \\ \text{factor } e^{-x^2}; \\ \text{return } x\} = \text{do } \{\text{factor } (1/\sqrt{3}); \\ x \leftarrow \text{normal } 0 \ \sqrt{3}; \\ \text{return } x\}$$

What is semantics for?

3. Equational reasoning by semantics-preserving rewriting!

Fubini

$$\begin{array}{l} \mathbf{do} \{x \leftarrow m; \\ \quad y \leftarrow n; \\ \quad \mathbf{return} (x, y)\} \end{array} = \begin{array}{l} \mathbf{do} \{y \leftarrow n; \\ \quad x \leftarrow m; \\ \quad \mathbf{return} (x, y)\} \end{array}$$

Conjugacy

$$\begin{array}{l} \mathbf{do} \{x \leftarrow \mathbf{normal} 0 1; \\ \quad \mathbf{factor} e^{-x^2}; \\ \quad \mathbf{return} x\} \end{array} = \begin{array}{l} \mathbf{do} \{\mathbf{factor} (1/\sqrt{3}); \\ \quad x \leftarrow \mathbf{normal} 0 \sqrt{3}; \\ \quad \mathbf{return} x\} \end{array}$$

What is semantics for?

3. Equational reasoning by semantics-preserving rewriting!

Fubini

$$\begin{array}{l} \mathbf{do} \{x \leftarrow m; \\ \quad y \leftarrow n; \\ \quad \mathbf{return} (x, y)\} \end{array} = \begin{array}{l} \mathbf{do} \{y \leftarrow n; \\ \quad x \leftarrow m; \\ \quad \mathbf{return} (x, y)\} \end{array}$$

Conjugacy

$$\begin{array}{l} \mathbf{do} \{x \leftarrow \mathbf{normal} 0 1; \\ \quad \mathbf{factor} e^{-x^2}; \\ \quad \mathbf{return} x\} \end{array} = \begin{array}{l} \mathbf{do} \{\mathbf{factor} (1/\sqrt{3}); \\ \quad x \leftarrow \mathbf{normal} 0 \sqrt{3}; \\ \quad \mathbf{return} x\} \end{array}$$

(convenient **factor** expresses non-(sub)probability measures)

Two program transformations specified semantically

1. Simplification

$$m = m'$$

2. Disintegration

$$m = \mathbf{do} \left\{ \begin{array}{l} t \leftarrow m_1; \\ x \leftarrow m_2; \\ \mathbf{return} (t, x) \end{array} \right\}$$

(typically t appears free in m_2)

Two program transformations specified semantically

1. Simplification

$$m = m'$$

2. Disintegration

$$m = \mathbf{do} \left\{ \begin{array}{l} t \leftarrow m_1; \\ x \leftarrow m_2; \\ \mathbf{return} (t, x) \end{array} \right\}$$

(typically t appears free in m_2)

Two program transformations specified semantically

1. Simplification

$$m = m'$$

2. Disintegration

$$m = \mathbf{do} \left\{ \begin{array}{l} t \leftarrow m_1; \\ x \leftarrow m_2; \\ \mathbf{return} (t, x) \end{array} \right\}$$

(typically t appears free in m_2)

Two program transformations specified semantically

1. Simplification

$$m = m'$$

2. Disintegration

$$m = \mathbf{do} \left\{ \begin{array}{l} t \leftarrow m_1; \\ x \leftarrow m_2; \\ \mathbf{return} (t, x) \end{array} \right\} = \mathbf{do} \left\{ \begin{array}{l} t \star m_1; \\ m_2 \end{array} \right\}$$

(typically t appears free in m_2)

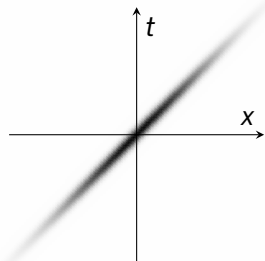
What is disintegration for?

$$m = \mathbf{do} \left\{ \begin{array}{l} t \leftarrow m_1; \\ x \leftarrow m_2; \\ \mathbf{return} (t, x) \end{array} \right\} = \mathbf{do} \left\{ \begin{array}{l} t \star m_1; \\ m_2 \end{array} \right\}$$

What is disintegration for?

```
m = do {t ~ m1;
        x ~ m2;
        return (t, x)}
     = do {t ~ m1;
        m2}
```

Disintegration defines conditional distributions

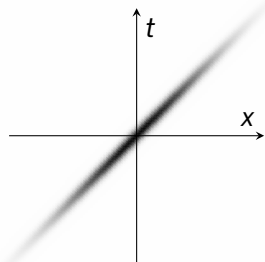


```
do {x ~ normal 0 10; -- latent cause
    t ~ normal x 1;   -- observed effect
    return (t, x)}
```

What is disintegration for?

```
m = do {t ~ m1;
        x ~ m2;
        return (t, x)}
    = do {t ~ m1;
        m2}
```

Disintegration defines conditional distributions

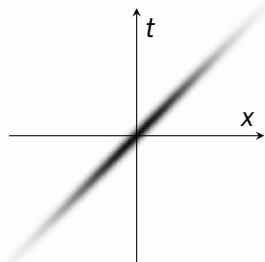


```
do {x ~ normal 0 10; -- latent cause
    t ~ normal x 1;   -- observed effect
    return (t, x)}
= do {t ~ normal 0 sqrt(101);
     x ~ normal (t * 100/101) (10/sqrt(101));
     return (t, x)}
```


What is disintegration for?

```
m = do {t ~ m1;
        x ~ m2;
        return (t, x)} = do {t ~ m1;
                               m2}
```

Disintegration defines conditional distributions



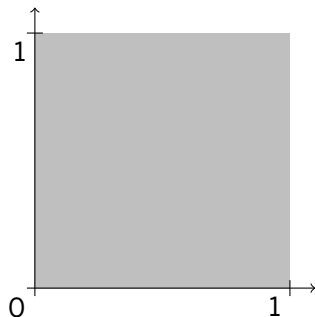
```
do {x ~ normal 0 10; -- latent cause
    t ~ normal x 1;   -- observed effect
    return (t, x)}
```

```
= do {t ~ normal 0 sqrt(101);
      normal (t * 100/101) (10/sqrt(101))}
```

What is disintegration for?

$$m = \mathbf{do} \left\{ \begin{array}{l} t \leftarrow m_1; \\ x \leftarrow m_2; \\ \mathbf{return} (t, x) \end{array} \right\} = \mathbf{do} \left\{ \begin{array}{l} t \star m_1; \\ m_2 \end{array} \right\}$$

Disintegration allows an uncountable space of observations

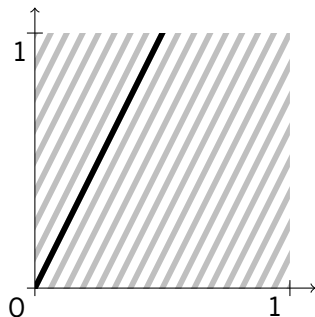


```
do {  
   $x \leftarrow \mathbf{uniform} \ 0 \ 1;$   
   $y \leftarrow \mathbf{uniform} \ 0 \ 1;$   
   $t \leftarrow$  ;  
  return ( $t, (x, y)$ )  
}
```

What is disintegration for?

$$m = \mathbf{do} \left\{ \begin{array}{l} t \leftarrow m_1; \\ x \leftarrow m_2; \\ \mathbf{return} (t, x) \end{array} \right\} = \mathbf{do} \left\{ \begin{array}{l} t \star m_1; \\ m_2 \end{array} \right\}$$

Disintegration allows an uncountable space of observations

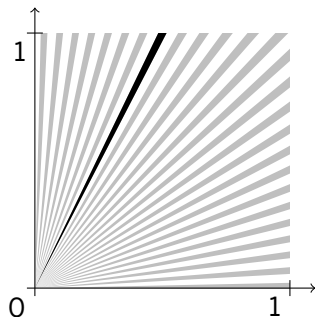


```
do {  
  x  $\leftarrow$  uniform 0 1;  
  y  $\leftarrow$  uniform 0 1;  
  t  $\leftarrow$  return (y - 2 * x);  
  return (t, (x, y))  
}
```

What is disintegration for?

$$m = \mathbf{do} \left\{ \begin{array}{l} t \leftarrow m_1; \\ x \leftarrow m_2; \\ \mathbf{return} (t, x) \end{array} \right\} = \mathbf{do} \left\{ \begin{array}{l} t \star m_1; \\ m_2 \end{array} \right\}$$

Disintegration allows an uncountable space of observations



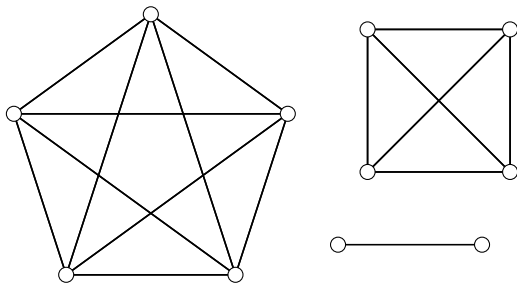
```
do {  
  x  $\leftarrow$  uniform 0 1;  
  y  $\leftarrow$  uniform 0 1;  
  t  $\leftarrow$  return (y/x);  
  return (t, (x, y))  
}
```

From denotation to operation

Denotation: equivalence relation

Operation: directed graph

So denotation justifies soundness of operation

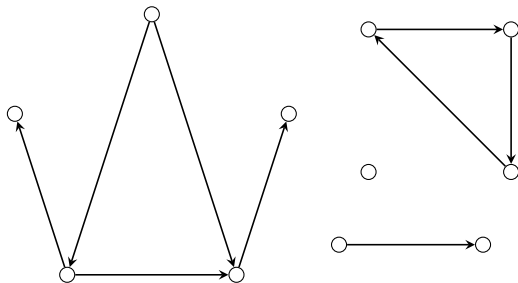


From denotation to operation

Denotation: equivalence relation

Operation: directed graph

So denotation justifies soundness of operation

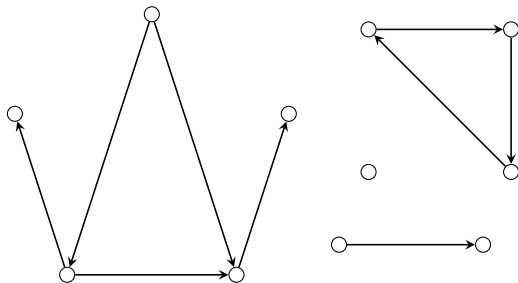


From denotation to operation

Denotation: equivalence relation

Operation: directed graph

So denotation justifies soundness of operation



This setup is typical for execution, but what about disintegration?
Rest of this talk: a disintegrator that justifies its work, step by step
A contextual operational semantics, implemented in PLT Redex

Term $e, m ::=$ **do** {**factor** e ; m } | **do** { $x \leftarrow m$; m } | **do** { $x \star m$; m }
| **return** e | **uniform** $e e$ | **normal** $e e$ | **lebesgue**
| (e, e) | **fst** e | **snd** e | *number* | $e - e$ | e/e | \dots | x

(not shown: conditionals)

Stratified syntax during disintegration

State $s ::= \mathbf{do} \{ \mathbf{factor} \ e; \ s \} \mid \mathbf{do} \{ x \leftarrow m; \ s \} \mid \mathbf{do} \{ \mathbf{delimit}; \ t \}$

Thread $t ::= \mathbf{do} \{ \mathbf{factor} \ e; \ t \} \mid \mathbf{do} \{ x \leftarrow m; \ t \} \mid \mathbf{do} \{ x \star m; \ m \}$

Term $e, m ::= \mathbf{do} \{ \mathbf{factor} \ e; \ m \} \mid \mathbf{do} \{ x \leftarrow m; \ m \}$
| **return** e | **uniform** $e \ e$ | **normal** $e \ e$ | **lebesgue**
| (e, e) | **fst** e | **snd** e | *number* | $e - e$ | e/e | \dots | x

(not shown: conditionals)

Example disintegration

```
do { $x \leftarrow$  uniform 0 1;  
     $y \leftarrow$  uniform 0 1;  
    return ( $y/x$ , ( $x, y$ ))}
```

Stage 1: Isolate (evaluate)

Stage 2: Invert (unevaluate)

Example disintegration

```
do {delimit;  
   $r \leftarrow$  do { $x \leftarrow$  uniform 0 1;  
     $y \leftarrow$  uniform 0 1;  
    return ( $y/x$ , ( $x$ ,  $y$ ))};  
   $t \leftarrow$  return (fst  $r$ );  
  return (snd  $r$ )}
```

Example disintegration (step 1)

```
do {delimit;  
  r  $\leftarrow$  do {x  $\leftarrow$  uniform 0 1;  
              y  $\leftarrow$  uniform 0 1;  
              return (y/x, (x,y))};  
  t  $\star$  return (fst r);  
  return (snd r)}
```

$\xrightarrow{\text{bind}}$

```
do {delimit;  
  x  $\leftarrow$  uniform 0 1;  
  r  $\leftarrow$  do {y  $\leftarrow$  uniform 0 1;  
              return (y/x, (x,y))};  
  t  $\star$  return (fst r);  
  return (snd r)}
```

Example disintegration (step 1)

<pre>do {delimit; r \leftarrow do {x \leftarrow uniform 0 1; y \leftarrow uniform 0 1; return (y/x, (x,y))}; t \star return (fst r); return (snd r)}</pre>	$\xrightarrow{\text{bind}}$	<pre>do {delimit; x \leftarrow uniform 0 1; r \leftarrow do {y \leftarrow uniform 0 1; return (y/x, (x,y))}; t \star return (fst r); return (snd r)}</pre>
--	-----------------------------	--

<pre>do {delimit; H; r \leftarrow do {x \leftarrow m₁; m₂}; P[r]}</pre>	$\xrightarrow{\text{bind}}$	<pre>do {delimit; H; x \leftarrow m₁; r \leftarrow m₂; P[r]}</pre>
---	-----------------------------	--

H is a heap of bindings

$P[r]$ is a program that demands r

Example disintegration (step 1)

```
do {delimit;  
   $x \leftarrow$  uniform 0 1;  
   $r \leftarrow$  do { $y \leftarrow$  uniform 0 1;  
    return ( $y/x, (x, y)$ )};  
   $t \star$  return (fst  $r$ );  
  return (snd  $r$ )}
```

Example disintegration (step 2)

```
do {delimit;  
   $x \leftarrow$  uniform 0 1;  
   $r \leftarrow$  do { $y \leftarrow$  uniform 0 1;  
    return ( $y/x, (x,y)$ )};  
   $t \leftarrow$  return (fst  $r$ );  
  return (snd  $r$ )}
```

Example disintegration (step 2)

```
do {delimit;  
   $x \leftarrow \text{uniform } 0\ 1;$   
   $r \leftarrow \text{do } \{y \leftarrow \text{uniform } 0\ 1;$   
     $\text{return } (y/x, (x, y))\};$   
   $t \star \text{return } (\text{fst } r);$   
   $\text{return } (\text{snd } r)\}$ 
```

$\xrightarrow{\text{bind}}$

```
do {delimit;  
   $x \leftarrow \text{uniform } 0\ 1;$   
   $y \leftarrow \text{uniform } 0\ 1;$   
   $r \leftarrow \text{return } (y/x, (x, y));$   
   $t \star \text{return } (\text{fst } r);$   
   $\text{return } (\text{snd } r)\}$ 
```

```
do {delimit;  
   $H;$   
   $r \leftarrow \text{do } \{x \leftarrow m_1;$   
     $m_2\};$   
   $P[r]\}$ 
```

$\xrightarrow{\text{bind}}$

```
do {delimit;  
   $H;$   
   $x \leftarrow m_1;$   
   $r \leftarrow m_2;$   
   $P[r]\}$ 
```


Example disintegration (step 3)

do { delimit ; $x \leftarrow \text{uniform } 0\ 1$; $y \leftarrow \text{uniform } 0\ 1$; $r \leftarrow \text{return } (y/x, (x, y))$; $t \star \text{return } (\text{fst } r)$; return (snd r)}	$\xrightarrow{\text{return}}$	do { delimit ; $x \leftarrow \text{uniform } 0\ 1$; $y \leftarrow \text{uniform } 0\ 1$; $r \leftarrow \text{return } (y/x, (x, y))$; $t \star \text{return } (\text{fst } (y/x, (x, y)))$; return (snd r)}
--	-------------------------------	--

do { delimit ; H ; $r \leftarrow \text{return } v$; $P[r]$ }	$\xrightarrow{\text{return}}$	do { delimit ; H ; $r \leftarrow \text{return } v$; $P[v]$ }
--	-------------------------------	--

v is head normal form for lazy evaluation

Example disintegration (step 4)

<pre>do {delimit; x \leftarrow uniform 0 1; y \leftarrow uniform 0 1; r \leftarrow return (y/x, (x, y)); t \leftarrow return (fst (y/x, (x, y))); return (snd r)}</pre>	$\xrightarrow{\text{fst}}$	<pre>do {delimit; x \leftarrow uniform 0 1; y \leftarrow uniform 0 1; r \leftarrow return (y/x, (x, y)); t \leftarrow return (y/x); return (snd r)}</pre>
<pre>do {delimit; P[fst (e₁, e₂)]}</pre>	$\xrightarrow{\text{fst}}$	<pre>do {delimit; P[e₁]}</pre>

Example disintegration (step 5)

<pre>do { delimit; x \leftarrow uniform 0 1; y \leftarrow uniform 0 1; r \leftarrow return (y/x, (x,y)); t \star return (y/x); return (snd r)}</pre>	$\xrightarrow{\text{Fubini}}$	<pre>do {x' \leftarrow uniform 0 1; delimit; x \leftarrow return x'; y \leftarrow uniform 0 1; r \leftarrow return (y/x, (x,y)); t \star return (y/x); return (snd r)}</pre>
--	-------------------------------	---

<pre>do { delimit; H; x \leftarrow uniform v1 v2; P[x]}</pre>	$\xrightarrow{\text{Fubini}}$	<pre>do {x' \leftarrow uniform v1 v2; delimit; H; x \leftarrow return x'; P[x]}</pre>
--	-------------------------------	---

Example disintegration (step 6)

do { $x' \leftarrow$ uniform 0 1; delimit ; $x \leftarrow$ return x' ; $y \leftarrow$ uniform 0 1; $r \leftarrow$ return ($y/x, (x, y)$); $t \star$ return (y/x); return (snd r)}	$\xrightarrow{\text{return}}$	do { $x' \leftarrow$ uniform 0 1; delimit ; $x \leftarrow$ return x' ; $y \leftarrow$ uniform 0 1; $r \leftarrow$ return ($y/x, (x, y)$); $t \star$ return (y/x'); return (snd r)}
--	-------------------------------	---

do { delimit ; H ; $r \leftarrow$ return v ; $P[r]$ }	$\xrightarrow{\text{return}}$	do { delimit ; H ; $r \leftarrow$ return v ; $P[v]$ }
---	-------------------------------	---

Example disintegration (step 7)

<pre>do {$x' \leftarrow$ uniform 0 1; delimit; $x \leftarrow$ return x'; $y \leftarrow$ uniform 0 1; $r \leftarrow$ return ($y/x, (x, y)$); $t \star$ return (y/x'); return (snd r)}</pre>	$\xrightarrow{\text{un/}}$	<pre>do {$x' \leftarrow$ uniform 0 1; delimit; $z \leftarrow$ uniform ($0/x'$) ($1/x'$); $y \leftarrow$ return ($z \times x'$); $x \leftarrow$ return x'; $r \leftarrow$ return ($y/x, (x, y)$); $t \star$ return (z); return (snd r)}</pre>
--	----------------------------	---

<pre>do {delimit; H_1; $y \leftarrow$ uniform $v_1 v_2$; H_2; $t \star$ return $E[y/v]$; m}</pre>	$\xrightarrow{\text{un/}}$	<pre>do {delimit; $z \leftarrow$ uniform (v_1/v) (v_2/v); $y \leftarrow$ return ($z \times v$); H_1; H_2; $t \star$ return $E[z]$; m}</pre>
---	----------------------------	---

Example disintegration (step 8)

```
do { $x' \leftarrow$  uniform 0 1;  
  delimit;  
   $z \leftarrow$  uniform (0/ $x'$ ) (1/ $x'$ );  
   $y \leftarrow$  return ( $z \times x'$ );  
   $x \leftarrow$  return  $x'$ ;  
   $r \leftarrow$  return ( $y/x, (x, y)$ );  
   $t \star$  return  $z$ ;  
  return (snd  $r$ )}
```

```
 $\xrightarrow{\text{unreturn}}$  do { $x' \leftarrow$  uniform 0 1;  
  delimit;  
   $t \star$  uniform (0/ $x'$ ) (1/ $x'$ );  
   $z \leftarrow$  return  $t$ ;  
   $y \leftarrow$  return ( $z \times x'$ );  
   $x \leftarrow$  return  $x'$ ;  
   $r \leftarrow$  return ( $y/x, (x, y)$ );  
  return (snd  $r$ )}
```

```
do {delimit;  
   $H_1$ ;  
   $z \leftarrow m_1$ ;  
   $H_2$ ;  
   $t \star$  return  $z$ ;  
   $m_2$ }
```

```
 $\xrightarrow{\text{unreturn}}$  do {delimit;  
   $H_1$ ;  
   $t \star m_1$ ;  
   $z \leftarrow$  return  $t$ ;  
   $H_2$ ;  
   $m_2$ }
```

Example disintegration (step 9)

do {	$\xrightarrow{\text{done}}$	do { $t \star$ lebesgue ;
$x' \leftarrow$ uniform 0 1;		$x' \leftarrow$ uniform 0 1;
delimit ;		factor if t is between $0/x'$ and $1/x'$
$t \star$ uniform $(0/x')$ $(1/x')$;		then $ x $ else 0;
$z \leftarrow$ return t ;		$z \leftarrow$ return t ;
$y \leftarrow$ return $(z \times x')$;		$y \leftarrow$ return $(z \times x')$;
$x \leftarrow$ return x' ;		$x \leftarrow$ return x' ;
$r \leftarrow$ return $(y/x, (x, y))$;		$r \leftarrow$ return $(y/x, (x, y))$;
return $(\text{snd } r)$ }		return $(\text{snd } r)$ }

Disintegration is useful:

- ▶ defines conditional distributions
- ▶ allows an uncountable space of observations

Disintegrator is useful:

- ▶ generates steps by operational semantics
- ▶ justifies steps by denotational semantics
- ▶ resembles lazy evaluator + change of variables
- ▶ leaves verification conditions behind for exchanging integrals