Supplemental Material

ANONYMOUS AUTHOR(S)

Additional Key Words and Phrases: Chemistry, Domain Specific Language, Type system, Compiler, Microfluidics, Laboratory-on-a-Chip (LoC), Elecrowetting-on-Dielectric (EWoD)

CONTENTS

10	Contents		1
11	1	Typing Judgments	3
12	1.1	Syntax	3
13	1.2	Evaluation Rules	4
14	1.3	Type Checking Rules	6
15	1.4	Type Inference Rules	8
16	2	Proofs	10
17	2.1	Helper Lemmas	10
18	2.2	Proof of Progress	14
19	2.3	Proof of Preservation	18
20	2.4	Proof of Soundness	24
21	2.5	Proof of Completeness	27
22	3	Syntax Study: ELISA Protocols	31
23	4	Background	33
24	4.1	Digital Microfluidic Technology	33
25	5	Assay Execution Videos	35
26	6	BioScript's ChemType Tests	35
27	6.1	Real-world Prevention Tests	35
28	6.2	Synthetic Failures	36
29	6.3	Synthetic Passes	37
30	7	BioScript Assays	37
31	7.1	Syntactic Sugar Translation	37
32	7.2	Urine Opiate Hierarchy	37
33	7.3	PCR Droplet Replenishment	41
34	7.4	Probabilistic PCR	41
35	7.5	ChemType Test1	42
36	7.6	ChemType Test2	42
37	7.7	ChemType Test3	43
38	7.8	Broad Spectrum Opiate	43
39	7.9	Cipro ELISA	44
40	7.10	Diazepam ELISA	44
41	7.11	547 Dilution	45
42	7.12	Fentanyl ELISA	45
43	7.13	Morphine ELISA	46
44	7.14	Morphine ELISA with Control Samples	47
45	7.15	Heroin ELISA	48
46	7.16	Oxycodone ELISA	48
47	2017 VVV	VV VVVV/2017/2 ADT 615 00	

	:2		Anon.
50	7.17	Image Probe Synthesis	49
51	7.18	Glucose Detection	49
52	7.19	PCR	49
53	7.20	Neurotransmitter Sensing	50
54	8	AquaCore Assays	50
55	8.1	Glucose Detection	50
56	8.2	PCR	51
57	8.3	Imaging Probe Synthesis	51
58	8.4	Neurotransmitter Sensing	52
59	9	BioCoder Assays	53
60	9.1	PCR	53
61	9.2	Probabilistic PCR	53
62	9.3	PCR Droplet Replenish	55
63	9.4	Glucose Detection	56
64	9.5	Image Probe Synthesis	57
65	9.6	Neurotransmitter Sensing	58
66	10	Antha Assays	59
67	10.1	Glucose Detection	59
68	10.2	Imaging Probe Synthesis	61
69	10.3	PCR	62
70	10.4	Neurotransmitter Sensing	66
71	11	ChemType Results	68
72	11.1	Efficacy of ChemType	68
73	11.2	ChemType Runtime Evaluation	68
74	Refe	References 69	
75			

103104105

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125 126

127

128

129

130

131

132

133 134

135

136

137 138

139

140 141

143

145146147

99

1 TYPING JUDGMENTS

Typing judgments are repeated here for convenience

1.1 Syntax

```
Terms:
t ::=
                x \in \mathcal{X}
                                                    Variable
                t_1 \oplus t_2
                                                    Math operation
                                                   Detect
                detect module on x for t
                v \in \mathcal{V}
                                                    Value
                                                    Values:
v :=
                                                    Material value
                mat
                                                    Real number
                r
                                                   Natural number
                n
module ::=
                module<sub>1</sub>, .., module<sub>n</sub>
                                                    Sensor module(s)
                                                    Statements:
s ::=
                i; s
                                                    Sequencing
                skip
                                                    Skip
i ::=
                                                    instructions:
                x := t
                                                    Assignment
                x := \min x_1 \text{ with } x_2 \text{ for } t
                                                    Mixing
                \langle x_1, ..., x_n \rangle := \text{split } x \text{ into } n
                                                   Splitting
                if t then s_1 else s_2
                                                    Conditional
                while t s
                                                   Loop
T ::=
                                                    Union Types:
                \cup \overline{S}
                                                    Union type
                V
                                                    Type variables
S ::=
                                                    Scalar types:
                Mat_1 \mid ... \mid Mat_n
                                                    Material types
                                                    Real number
                                                    Natural number
                \mathbb{N}
\Gamma ::=
                                                    Context:
                Ø
                                                   Empty context
                \Gamma, x : T
                                                    Variable type binding
X
                                                    Set of variables x
C
                                                    Constraints
```

Fig. 1. Syntax of ChemType.

:4 Anon.

1.2 Evaluation Rules

E-Var
$$\frac{x \in \text{dom}(\sigma)}{(\sigma, x) \to \sigma(x)}$$
E-MathR1
$$\frac{(\sigma, t_1) \to t'_1}{(\sigma, t_1 \oplus t_2) \to t'_1 \oplus t_2}$$

$$\frac{\text{E-MathR2}}{v \in \mathbb{N} \lor v \in \mathbb{R}} \qquad (\sigma, t_2) \to t_2'$$
$$(\sigma, v \oplus t_2) \to v \oplus t_2'$$

$$\frac{\text{E-Math}}{(v_1 \in \mathbb{N} \land v_2 \in \mathbb{N}) \lor (v_1 \in \mathbb{R} \land v_2 \in \mathbb{R})} \qquad v_1 \oplus v_2 = v}{(\sigma, v_1 \oplus v_2) \to v}$$

E-DetectR

$$(\sigma, t) \to t'$$

 $\overline{(\sigma, \text{ detect } module \text{ on } x \text{ for } t) \rightarrow \text{detect } module \text{ on } x \text{ for } t'}$

E-DETECT
$$\frac{\sigma(x) \in Mat_i \qquad r_2 = \text{detect}(\sigma(x), module, r_1)}{(\sigma, \text{detect module on } x \text{ for } r_1) \rightarrow r_2}$$

Fig. 2. Evaluation rules for terms.

```
E-AssignR
197
                \frac{(\sigma, t) \to t' \qquad t \notin X}{(\sigma, x := t; s) \to (\sigma, x := t'; s)} \qquad \frac{\text{E-Assign}}{(\sigma, x := v; s) \to (\sigma[x \mapsto v], s)} \qquad \frac{\sigma' = (\sigma \setminus \{x'\})[x \mapsto \sigma(x')]}{(\sigma, x := x'; s) \to (\sigma', s)}
                E-MixR
                                                                           (\sigma, t) \to t'
                \frac{(\sigma,t)\to t'}{(\sigma,x:=\max x_1 \text{ with } x_2 \text{ for } t;\ s)\to (\sigma,x:=\min x_1 \text{ with } x_2 \text{ for } t';\ s)}
202
204
                E-Mix
205
                                    \sigma(x_1) \in Mat_i \sigma(x_2) \in Mat_i
206
                                     interact(\sigma(x_1), \ \sigma(x_2), \ r) \neq \frac{1}{2}
                \sigma' = (\sigma \setminus \{x_1, x_2\})[x \mapsto \operatorname{interact}(\sigma(x_1), \sigma(x_2), r)]
                       (\sigma, x := \min x_1 \text{ with } x_2 \text{ for } r; s) \rightarrow (\sigma', s)
210
                E-Split
211
                \sigma(x) \in Mat_i \sigma' = (\sigma \setminus \{x\}) \overline{[x_i \mapsto \operatorname{split}(\sigma(x), n)]}
212
                        (\sigma, \langle x_0, ... x_n \rangle := \text{split } x \text{ into } n; s) \rightarrow (\sigma', s)
213
214
215
                E-IFR
216
                \frac{(\sigma, t) \to t'}{(\sigma, \text{ if } t \text{ then } s_1 \text{ else } s_2; s) \to (\sigma, \text{ if } t' \text{ then } s_1 \text{ else } s_2; s)}
217
218
219
                E-IFTRUE
220
221
                (\sigma, \text{ if } n s_1 \text{ else } s_2; s) \rightarrow (\sigma, s_1 \bullet s)
222
223
                E-IfFalse
224
                (\sigma, \text{ if } 0 \ s_1 \text{ else } s_2; \ s) \rightarrow (\sigma, \ s_2 \bullet \ s)
225
226
               E-WHILE
227
                (\sigma, \text{ while } t s_1; s_2) \rightarrow (\sigma, \text{ if } t \text{ then } (s_1 \bullet \text{ while } t s_1; s_2) \text{ else } s_2)
228
229
               skip \bullet s = s
230
               (i; s) \bullet s' = i; (s \bullet s')
231
232
```

Fig. 3. Evaluation rules for statements.

:6 Anon.

1.3 Type Checking Rules

(T-Var)
$$\frac{x:T\in\Gamma \qquad x\in X}{\Gamma,X\vdash x\colon T}$$

$$\text{(T-Math)} \qquad \frac{\Gamma, X \vdash t_1 : T \qquad \Gamma, X \vdash t_2 : T}{T = \mathbb{N} \lor T = \mathbb{R}}$$

$$\text{(T-Detect)} \quad \frac{\Gamma, X \vdash x : \cup \overline{Mat_i} \qquad \Gamma, X \vdash t : \mathbb{R}}{\Gamma, X \vdash \text{detect } \textit{module} \text{ on } x \text{ for } t : \mathbb{R}}$$

(T-MAT)
$$\frac{\overline{mat \in Mat_i}}{\Gamma, X \vdash mat : \cup \overline{Mat_i}}$$

(T-Real)
$$\Gamma, X \vdash r : \mathbb{R}$$

(T-NAT)
$$\Gamma, X \vdash n : \mathbb{N}$$

Fig. 4. Type checking rules for terms.

(T-Inst)
$$\frac{\Gamma, X \vdash i, X' \qquad \Gamma, X' \vdash s, X''}{\Gamma, X \vdash i; \ s, X''}$$

$$\frac{\Gamma, X \vdash i; \ s, X''}{\Gamma, X \vdash skip, X}$$
(T-Assign-1)
$$\frac{x \colon T \in \Gamma \qquad \Gamma, X \vdash v \colon T' \qquad T' \subseteq T}{\Gamma, X \vdash x \coloneqq v, X \cup \{x\}}$$

$$\frac{x \colon T \in \Gamma \qquad \Gamma, X \vdash x' \colon T' \qquad T' \subseteq T}{\Gamma, X \vdash x \coloneqq x', X \setminus \{x'\} \cup \{x\}}$$
(T-Assign-2)
$$\frac{x \colon T \in \Gamma \qquad \Gamma, X \vdash x' \colon T' \qquad T' \subseteq T}{\Gamma, X \vdash x \coloneqq x', X \setminus \{x'\} \cup \{x\}}$$

$$\frac{x \colon T \in \Gamma \qquad t \notin V \cup X \qquad \Gamma, X \vdash t \colon T'}{\Gamma, X \vdash x \coloneqq t, X \cup \{x\}}$$

$$\frac{T, X \vdash x \coloneqq t, X \cup \{x\}}{\Gamma, X \vdash x \coloneqq t, X \cup \{x\}}$$
(T-Mix)
$$\frac{\Gamma, X \vdash x \coloneqq mix \ x_1 \text{ with } x_2 \text{ for } t, X \setminus \{x_1, x_2\} \cup \{x\}}{\Gamma, X \vdash x \coloneqq mix \ x_1 \text{ with } x_2 \text{ for } t, X \setminus \{x_1, x_2\} \cup \{x\}}$$
(T-Split)
$$\frac{\Gamma, X \vdash x \coloneqq mix \ x_1 \text{ with } x_2 \text{ for } t, X \setminus \{x_1, x_2\} \cup \{x\}}{\Gamma, X \vdash x \coloneqq mix \ x_1 \text{ with } x_2 \text{ for } t, X \setminus \{x_1, x_2\} \cup \{x\}}$$
(T-If)
$$\frac{\Gamma, X \vdash x \colon \bigcup \overline{Mat_i} \qquad \Gamma(x) \subseteq \Gamma(x_1), ..., \Gamma(x) \subseteq \Gamma(x_n)}{\Gamma, X \vdash x \colon \bigcup \overline{Mat_i} \qquad \Gamma(x) \subseteq \Gamma(x_1), ..., \Gamma(x) \subseteq \Gamma(x_n)}$$

$$\frac{\Gamma, X \vdash x \colon \bigcup \overline{Mat_i} \qquad \Gamma(x) \subseteq \Gamma(x_1), ..., \Gamma(x) \subseteq \Gamma(x_n)}{\Gamma, X \vdash x \colon \bigcup \overline{Lx} \quad x_1 \text{ with } x_2 \text{ for } t, X \vdash x_2, X''}}$$

$$\frac{\Gamma, X \vdash t \colon \mathbb{N} \qquad \Gamma, X \vdash s_1, X' \qquad \Gamma, X \vdash s_2, X''}{\Gamma, X \vdash t \vdash x} \qquad \Gamma, X \vdash s_1, X' \qquad \Gamma, X \vdash s_2, X''}$$

$$\frac{\Gamma, X \vdash t \colon \mathbb{N} \qquad \Gamma, X \vdash s, X' \qquad X \subseteq X'}{\Gamma, X \vdash while t \ s. X}$$

Fig. 5. Type checking rules for statements.

:8 Anon.

1.4 Type Inference Rules

346
347
(CT-VAR)
$$\frac{x:T\in\Gamma \quad x\in X}{\Gamma,X\vdash x:T\mid\emptyset}$$
348
349
350
(CT-MATH)
$$\frac{\Gamma,X\vdash t_1:T_1\mid C_1 \quad \Gamma,X\vdash t_2:T_2\mid C_2}{\Gamma,X\vdash t_1\oplus t_2:T_1\mid}$$
352
$$C_1\cup C_2\cup \{T_1=T_2=\mathbb{N}\vee T_1=T_2=\mathbb{R}\}$$
353
354
355
(CT-DETECT)
$$\frac{\Gamma,X\vdash x:T_1\mid C_1 \quad \Gamma,X\vdash t:T_2\mid C_2}{\Gamma,X\vdash detect\ module\ on\ x\ for\ t:\mathbb{R}\mid}$$
356
357
358
359
(CT-MAT)
$$\frac{mat\in Mat_i}{\Gamma,X\vdash mat:\cup\overline{Mat_i}\mid\emptyset}$$
361
362
(CT-REAL)
$$\Gamma,X\vdash r:\mathbb{R}\mid\emptyset$$

 $\Gamma, X \vdash n : \mathbb{N} \mid \emptyset$

(CT-NAT)

Fig. 6. Type Inference rules for terms.

$$\begin{array}{lll} & \Gamma, X \vdash i, X' \mid C_1 & \Gamma, X' \vdash s, X'' \mid C_2 \\ \hline \Gamma, X \vdash i; \; s, X'' \mid C_1 \cup C_2 \\ \hline \end{array}$$

Fig. 7. Type Inference rules for statements.

:10 Anon.

2 PROOFS

2.1 Helper Lemmas

LEMMA 1 (NAME EXTENSION).

 $\forall \Gamma, X, i, X', X''$.

$$\Gamma, X_1 \vdash i, X_2 \land X_1 \subseteq X_1' \Rightarrow$$

 $\exists X_2'$

442 443

444

445

446

451

453

455

463

465

466 467

468 469

471

472

473 474

475 476

477

478

479

480

481 482

483

484

485

486

487

488

489 490

$$\Gamma, X_1' \vdash i, X_2' \land X_2 \subseteq X_2'$$

449 450 and

$$\forall \Gamma, X, s, X', X''$$
.

$$\Gamma, X_1 \vdash s, X_2 \land X_1 \subseteq X_1' \Rightarrow$$

 $\exists X_2'$.

$$\Gamma, X_1' \vdash s, X_2' \land X_2 \subseteq X_2'$$

PROOF. Trivial by mutual induction on s and i.

457 LEMMA 2.

For every Γ , X, i, X', s, and X'', if

$$\Gamma, X \vdash i, X', X'' \subseteq X' \text{ and } \Gamma, X'' \vdash s, X'''$$

there exists X'''' such that

$$\Gamma, X \vdash i; s, X'''' \text{ and } X''' \subseteq X''''$$

PROOF. Direct from Lemma 1 and the rule T-INST.

464 LEMMA 3 (TYPING •).

For every Γ , X, s_1 , X', s_2 , and X''

$$\Gamma, X \vdash s_1, X' \land \Gamma, X' \vdash s_2, X'' \Rightarrow \Gamma, X \vdash s_1 \bullet s_2, X''$$

PROOF. Trivial by induction on s_1 .

Lemma 4.

For every Γ , X, s_1 , X', s_2 , and X'', if

$$\Gamma, X \vdash s_1, X', X'' \subseteq X' \text{ and } \Gamma, X'' \vdash s_2, X'''$$

there exists X'''' such that

$$\Gamma, X \vdash s_1 \bullet s_2, X'''' \text{ and } X''' \subseteq X''''$$

PROOF. Direct from Lemma 1 and Lemma 3.

LEMMA 5 (CANONICAL FORMS).

For every Γ , X and v,

- If $\Gamma, X \vdash v : Mat_i$, then $v \in Mat_i$.
- If $\Gamma, X \vdash \upsilon : \mathbb{R}$, then $\upsilon \in \mathbb{R}$.
- If $\Gamma, X \vdash v : \mathbb{N}$, then $v \in \mathbb{N}$.

Proof. Immediate from case analysis on the structure of v and using the inversion lemma, Lemma 6.

LEMMA 6 (INVERSION ON TYPING OF TERMS).

(1) If $\Gamma, X \vdash x : T$,

then $x : T \in \Gamma$ and $x \in X$.

(2) If Γ , $X \vdash t_1 \oplus t_2 : T$,

then $\Gamma, X \vdash t_1 : T \text{ and } \Gamma, X \vdash t_1 : T \text{ and } T = \mathbb{N} \lor T = \mathbb{R}$

, Vol. 1, No. 1, Article . Publication date: February 2017.

```
(3) If \Gamma, X \vdash detect module on x for t : T,
                   then there exists \overline{Mat_i} such that \Gamma, X \vdash x : \bigcup \overline{Mat_i} and \Gamma, X \vdash t : \mathbb{R} and T = \mathbb{R}
             (4) If \Gamma, X \vdash mat : T, then there exists i such that T = Mat_i
             (5) If \Gamma, X \vdash r : T, then T = \mathbb{R}
             (6) If \Gamma, X \vdash n : T, then T = \mathbb{N}
             PROOF. Immediate from case analysis on the type derivation rules.
             Lemma 7 (Inversion on typing of statements and instructions).
             (1) For all \Gamma, X, i, s, X'',
                   Ιf
                       \Gamma, X \vdash i; s, X'',
                   then, there exists X' such that
                       \Gamma, X \vdash i, X' and
504
                       \Gamma, X' \vdash s, X''.
             (2) For all \Gamma, X, x, t, X',
506
                   If
                       \Gamma, X \vdash x := t, X'
                   then, there exists T, T' such that
                       x: T \in \Gamma
                       \Gamma, X \vdash t : T',
                       T' \subseteq T,
                            T' = \mathbb{R} \vee T' = \mathbb{N} \vee t = mat, and
                            X' = X \cup \{x\}
                       or
515
                            t = x' and
516
                            X' = X \setminus \{x'\} \cup \{x\}.
517
             (3) For all \Gamma, X, x, x_1, x_2, t, X',
518
                   If
519
                       \Gamma, X \vdash x := mix \ x_1 \ with \ x_2 \ for \ t, X'
520
                   then, there exist is and js such that
521
                       \Gamma, X \vdash x_1 : \cup \overline{Mat_i},
522
                       \Gamma, X \vdash x_2 : \cup \overline{Mat_i},
523
                       \Gamma, X \vdash t : \mathbb{R},
524
                       interact-abs(Mat_i, Mat_i) \subseteq \Gamma(x), and
525
                       X' = X \setminus \{x_1, x_2\} \cup \{x\}.
526
             (4) For all \Gamma, X, x, x_1, ..., x_n, X',
527
528
                       \Gamma, X \vdash \langle x_1, ..., x_n \rangle := split \ x \ into \ n, X'
529
                   then, there exist is such that
                       \Gamma, X \vdash x : \cup \overline{Mat_i},
531
                       \Gamma(x) \subseteq \Gamma(x_1), ..., \Gamma(x) \subseteq \Gamma(x_n), and
                       X' = X \setminus \{x\} \cup \{x_1, ..., x_n\}.
533
             (5) For all \Gamma, X, t, s_1, s_2, X''',
534
                   If
535
                       \Gamma, X \vdash if t then s_1 else s_2, X'''
                   then, there exists X' and X'' such that
                       \Gamma, X \vdash t : \mathbb{N},
```

:12 Anon.

```
\Gamma, X \vdash s_1, X'
541
                             \Gamma, X \vdash s_2, X'', and
                              X^{\prime\prime\prime\prime}=X^{\prime}\cap X^{\prime\prime}.
542
543
                 (6) For all \Gamma, X, t, s, X',
                        If
                              \Gamma, X \vdash while \ t \ s, X'
                        then,
                              \Gamma, X \vdash t : \mathbb{N}
547
                             \Gamma, X \vdash s, X'',
                             X \subseteq X^{\prime\prime} and
549
                             X' = X.
```

PROOF. Immediate from case analysis on the type derivation rules.

Lemma 8.

For every σ , t and t' if $(\sigma, t) \to t'$ then $t' \notin X$

Proof. Immediate from the term transition rules.

Helper Definitions

```
The conservative property of the abstract interact-abs function:
```

```
\forall mat_i, mat_j, r.
```

```
mat_i \in Mat_i \land mat_j \in Mat_j \Rightarrow

interact(mat_i, mat_j, r) = \mbecause \implies interact-abs(Mat_i, Mat_j) \text{ undefined}

interact(mat_i, mat_j, r) \neq \mbecause \implies interact(mat_i, mat_j, r) \in interact-abs(Mat_i, Mat_j)
```

The type of the functions used for evaluation:

```
detect: Mat_i \to Module \to \mathbb{R} \to \mathbb{R}interact: Mat_i \to Mat_j \to interact-abs(Mat_i, Mat_j)
```

We define the consistency condition between the static typing environment Γ and the runtime store σ as:

```
consistent(\Gamma, X, \sigma) = \forall x, T.(x:T) \in \Gamma \land x \in X \Rightarrow\sigma(x) \in T
```

 $split: Mat_i \to \mathbb{N} \to Mat_i$

:14 Anon.

```
Proof of Progress
638
639
            LEMMA 9 (PROGRESS OF TERMS).
640
         For every \Gamma, X, t or T,
641
         if
642
             \Gamma, X \vdash t : T
643
         then
             \forall \sigma. \ consistent(\Gamma, X, \sigma) \Rightarrow \exists t'. (\sigma, t) \rightarrow t' \ or
645
             t is a value.
647
            Proof.
         Proof by induction on t and case analysis thereafter.
         We assume
649
             (1) \Gamma, X \vdash t : T
         Case for the rule T-VAR:
651
             We have
                  (2) t = x
653
                  (3) x : T \in \Gamma
                  (4) x \in X
655
             We assume
                  (5) consistent(\Gamma, X, \sigma)
657
                  From [3], [4] and [5]:
                       (5) \sigma(x) \in T
659
                  By the rule E-VAR on [5]:
                       (6) (\sigma, x) \rightarrow \sigma(x)
661
                  By the rule E-VAR on [6] and [2]:
662
                       (6) (\sigma, t) \rightarrow \sigma(x)
663
664
            Case for the rule T-MATH:
665
             (2) t = t_1 \oplus t_2
666
             (3) \Gamma \vdash t_1 : T
667
             (4) \Gamma \vdash t_2 : T
668
             (5) T = \mathbb{R} \vee T = \mathbb{N}
669
             By I.H. on t_1:
670
                  Case 1: t_1 is not a value:
671
                       (6) \forall \sigma, X. \ consistent(\sigma, X, \Gamma) \Rightarrow \exists t'_1. \ (\sigma, t_1) \rightarrow t'_1
                       We assume that
673
                           (7) consistent(\sigma, X, \Gamma)
                      and prove that
675
                           \exists t', (\sigma, t) \rightarrow t'
676
                      From [6] and [7], there exists t'_1 such that
677
                           (8) (\sigma, t_1) \rightarrow t'_1
678
                      From the rule E-MATHR1 on [8]:
679
                           (9) (\sigma, t_1 \oplus t_2) \to t_1' \oplus t_2
680
                      From [9] on [2]
681
                           (\sigma, t) \rightarrow t_1' \oplus t_2
682
                  Case 2:
683
                      (10) t_1 is a value v_1:
684
                       By Lemma 5 on [3], [5], [10]
685
686
```

```
(11) v_1 \in \mathbb{N} \lor v_1 \in \mathbb{R}
687
                     By I.H. on t_2:
688
                         Case 2.1: t_2 is not a value:
                             (12) \forall \sigma, X. \ consistent(\sigma, X, \Gamma) \Rightarrow \exists t_2', (\sigma, t_2) \rightarrow t_2'
                                  We assume that
                                      (13) consistent(\sigma, X, \Gamma)
                                  and prove that
                                      \exists t', (\sigma, t) \rightarrow t'
                                  From [12] and [13], there exists t_2' such that
                                      (14) (\sigma, t_2) \rightarrow t_2'
                                  From the rule E-MATHR2 on [14]:
                                      (15) (\sigma, v_1 \oplus t_2) \rightarrow v_1 \oplus t_2'
                                  From [15], [2], [10]:
700
                                      (\sigma, t) \rightarrow v_1 \oplus t_2'
                         Case 2.2:
702
                             (16) t_2 is a value, v_2:
                             By Lemma 5 on [4], [5], [16]
                                  (17) v_2 \in \mathbb{N} \vee v_2 \in \mathbb{R}
704
                             From the rule E-MATH on [11] and [17]:
                                  (18) (\sigma, v_1 \oplus v_2) \rightarrow v_1 \oplus v_2
706
                             From on [17], [10], [16]
708
                                  (\sigma,t) \rightarrow v_1 \oplus v_2
709
710
           Case for the rule T-DETECT:
            t = \text{detect } module_i \text{ on } x \text{ for } t'
711
712
              Similar to the the rule T-MATH rule, by induction hypothesis in t' and then using the rule
713
              E-DETECT and the rule E-DETECTR. The consistency condition is used to derive the first premise
714
              of the rule E-Detect.
715
716
717
           Case for the rule T-MAT:
            mat is a value.
718
719
        Case for the rule T-REAL:
720
            r is a value.
721
722
        Case for the rule T-NAT:
723
            n is a value.
724
                                                                                                                                       725
           Proof.
726
727
        Case analysis on the typing derivation:
            Case for the rule T-Skip:
728
                s = skip
729
            Case for the rule T-INST:
730
                s = i; s'
731
                \Gamma, X \vdash i, X''
732
                Immediate from Lemma 10.
733
```

:16 Anon.

```
736
            LEMMA 10 (PROGRESS FOR INSTRUCTIONS).
         For every \Gamma, X, i, s, X',
737
738
         if
             \Gamma, X \vdash i, X'
739
         then
740
741
             \forall \sigma. consistent(\Gamma, X, \sigma) \Rightarrow
742
                  \exists \sigma', s'. (\sigma, i; s) \rightarrow (\sigma', s')
743
744
            Proof.
745
         We have that
746
             (1) \Gamma, X \vdash i, X'
747
         Case analysis on the typing derivation:
748
         Case for the rule T-Assign-1:
749
             (2) i = (x := v)
750
             (3) x : T \in \Gamma
751
             (4) \Gamma, X \vdash t : T'
752
             (5) T' \subseteq T
753
             From the rule E-Assign
                  (6) (\sigma, x := v; s) \rightarrow (\sigma[x \mapsto v]; s)
755
             From [6], [2]
                  (\sigma, i; s) \rightarrow (\sigma[x \mapsto v]; s)
757
758
            Case for the rule T-Assign-2:
759
760
               Similar to the previous case. The reduction uses the rule E-Assign'
761
762
            Case for the rule T-Assign-3:
763
             (2) i = (x := t)
764
             (3) x : T \in \Gamma
765
             (4) \Gamma, X \vdash t : T'
766
             (5) (T' = \mathbb{R} \vee T' = \mathbb{N}) \wedge t \notin \mathcal{V} \cup \mathcal{X}
767
             (6) T' \subseteq T
768
             By Lemma 9 on [4]:
769
                  Case 1:
                      (7) \forall \sigma. consistent(\Gamma, X, \sigma) \Rightarrow \exists t'. (\sigma, t) \rightarrow t'
771
                       We assume that
772
                           (8) consistent(\Gamma, X, \sigma)
773
                       We prove that
774
                           (\sigma, i; s) \rightarrow (\sigma', s')
775
                      From [7] and [8], there exists t' such that
776
                           (9) (\sigma, t) \rightarrow t'
777
                      From the rule E-AssignR on [9] and [5]:
778
                           (10) (\sigma, (x := t); s) \rightarrow (\sigma', (x := t'); s)
779
                      From [2] on [10]:
780
                           (\sigma, i; s) \rightarrow (\sigma', (x := t'); s)
781
                  Case 2:
782
                      (11) t is a value, v.
783
784
```

787

788

792

798

800

802

804

806

807

808

809 810

811

812 813

814 815

Contradiction with [5].

Case for the rule T-MIX:

```
(2) i = (x := mix x_1 with x_2 for t)
```

The proof is similar to the case for the rule T-Assign. Lemma 9 is applied to the type derivation for t. There are two cases. Case 1: If t steps, the rule E-MixR is applicable. Case 2: If t is a value, the rule E-MIX is applicable. Lemma 6 and the consistency condition is used to show that x_1 and x_2 are both material values $\sigma(x_1)$ and $\sigma(x_2)$ in the store. In addition, from the case analysis on the typing derivation, we have that interact-abs (Mat_i, Mat_i) is defined; thus, by the conservative property of the abstract interaction interact($\sigma(x_1), \sigma(x_2)$) $\neq \frac{1}{2}$.

Case for the rule T-Split:

```
(2) i = (\langle x_1, \dots x_n \rangle = \text{split } x_1 \text{ into } n)
```

The consistency condition is used to show that x is a material value in the store. Then, the rule E-Split is applicable.

Case for the rule T-IF:

```
(2) i = \text{if } t \text{ then } s_1 \text{ else } s_2
```

We apply Lemma 9 to the type derivation for t. There are two cases. Case 1: If t steps, the rule E-IFR is applicable. Case 2: If t is a value, by Lemma 5 on the typing judgement for t, we know that it is a natural number. If it is non-zero, the rule E-IFTRUE is applicable; otherwise the rule E-IFFALSE is applicable.

Case for the rule T-WHILE:

```
(2) i = \text{while } t s
```

The rule E-While is applied without any premise.

816 817 818

819

820

821

822

823

825

826

827

828 829

830

831

832 833

```
LEMMA 11 (PROGRESS FOR STATEMENTS).
For every \Gamma, X, s, X',
if
    \Gamma.X \vdash s.X'
then either s is skip or
    \forall \sigma. consistent(\Gamma, X, \sigma) \Rightarrow
         \exists \sigma', s'. (\sigma, s) \rightarrow (\sigma', s')
   Proof.
```

We have that

(1)
$$\Gamma, X \vdash s, X'$$

Case analysis on s:

```
Case s = skip
```

Conclusion is immediate.

Case

(2)
$$s = i; s'$$

:18 Anon.

```
From [1] and [2],
                     (3) \Gamma, X \vdash i; s', X'
835
                 By Lemma 7 on [3], there exists X'' such that
                     (4) \Gamma, X \vdash i, X''
837
                     (5) \Gamma, X'' \vdash s', X'
                 By Lemma 10 on [4],
                     (6) \forall \sigma. consistent(\Gamma, X, \sigma) \Rightarrow
                          \exists \sigma', s''. (\sigma, i; s') \rightarrow (\sigma', s'')
                 The conclusion is immediate form [2] and [6].
843
                                                                                                                                           Proof of Preservation
        2.3
845
           LEMMA 12 (PRESERVATION OF TERMS).
847
        For every \Gamma, X, t, T and \sigma,
849
            \Gamma, X \vdash t : T \text{ and }
            (\sigma, t) \rightarrow t' and
851
            consistent(\Gamma, X, \sigma)
        then
853
            \Gamma, X \vdash t' : T
           Proof.
855
        We have
856
             (1) \Gamma, X \vdash t : T
857
             (2) (\sigma, t) \rightarrow t'
858
             (3) consistent(\Gamma, X, \sigma)
859
        Straightforward induction on the derivation of \Gamma, X \vdash t : T and then case analysis on the final rule
860
        in the derivation of (\sigma, t) \rightarrow t'
861
862
           Case for the rule T-VAR:
863
            From the rule T-VAR:
                 (4) t = x
865
                 (5) x : T \in \Gamma
                 (6) x \in X
867
             From the rule E-VAR:
                 (7) t' = \sigma(x)
869
             From [3], [5], and [6]
870
                 (7) \ \sigma(x) \in T
871
             By case analysis on the value \sigma(x) and the rule T-MAT,
872
             the rule T-Real and the rule T-Nat
873
                 \Gamma, X \vdash \sigma(x) : T
874
875
           Case for the rule T-MATH:
876
             (4) t = t_1 \oplus t_2
877
             (5) Γ, X \vdash t_1 : T
878
             (6) \Gamma, X \vdash t_2 : T
879
             (7) T = \mathbb{R} \vee T = \mathbb{N}
880
             Case analysis on [2]:
881
882
```

925

926 927

Case for the rule T-REAL:

r is a value and does not step.

(4) t = r

```
Case for the rule E-MATHR1:
                (8) t' = t'_1 \oplus t_2
884
                (9) (\sigma, t_1) \rightarrow t_1'
                By I.H. on [5], [9], and [3]:
                    (10) \Gamma, X \vdash t_1' : T
                From the rule T-MATH on [8], [10], [6], [7]:
                    (11) \Gamma, X \vdash t' : T
            Case for the rule E-MATHR2:
                (12) t' = v_1 \oplus t_2'
                (13) (\sigma, t_2) \rightarrow t_2'
                By I.H. on [6], [13], and [3]:
                    (14) \Gamma, X \vdash t_2' : T
                From the rule T-MATH on [12], [5], [14]:
896
                    (15) \Gamma, X \vdash t' : T
            Case for the rule E-MATH:
898
                (16) t_1 = v_1
                (17) t_2 = v_2
900
                (18) (v_1 \in \mathbb{N} \land v_2 \in \mathbb{N}) \lor (v_1 \in \mathbb{R} \land v_2 \in \mathbb{R})
                (19) v_1 \oplus v_2 = v
                (20) t' = v
902
                From [18], we consider the case:
                    (21) v_1 \in \mathbb{N} \wedge v_2 \in \mathbb{N}
904
                The other case is similar.
906
                From [21], [19]:
                    (22) v \in \mathbb{N}
907
                From [20], [22] and the rule T-NAT:
908
                    (23) \Gamma, X \vdash t' : \mathbb{N}
                From Lemma 6 on [5], [16] and [21]
910
911
                    (24) T = \mathbb{N}
                From [23] and [24]:
912
                    (23) \Gamma, X \vdash t' : T
913
914
           Case for the rule T-DETECT:
915
            (4) t = detect \ module_i \ on \ x \ for \ t'
916
917
             We consider the two cases for [2]: Case for the rule E-DETECTR: Induction hypothesis is applied
918
             to t' and then the rule T-DETECT is applied. Case for the rule E-DETECT: Immediate from the
919
             rule T-REAL.
920
921
           Case for the rule T-MAT:
922
            (4) t = mat
923
            mat is a value and does not step.
924
```

:20 Anon.

```
932
           Case for the rule T-NAT:
            (4) t = n
933
934
            n is a value and does not step.
935
937
           LEMMA 13 (PRESERVATION OF STATEMENTS).
938
        For every \Gamma, X, \sigma, s, X'', \sigma', s',
939
        if
940
            \Gamma, X \vdash s, X'' and
941
            (\sigma, s) \rightarrow (\sigma', s') and
942
            consistent(\Gamma, X, \sigma)
943
        then there exists X' such that
944
            \Gamma, X' \vdash s', X'' and
945
            consistent(\Gamma, X', \sigma')
946
947
           Proof.
948
        We have
            (1) \Gamma, X \vdash s, X''
949
            (2) (\sigma, s) \rightarrow (\sigma', s')
951
            (3) consistent(\Gamma, X, \sigma)
           Case Analysis on [1]:
953
        Case for the rule T-Skip:
954
955
            Contradiction in [2]: The statement skip does not step.
956
           Case for the rule T-Inst:
957
            (4) \Gamma, X \vdash i, X'
958
            (5) \Gamma, X' \vdash s'', X''
959
            (6) s = i; s''
960
961
        Case Analysis on [4]:
962
        Case for the rule T-Assign-1:
963
            (7) i = (x := v)
964
            (8) x : T \in \Gamma
965
            (9) \Gamma, X \vdash t : T'
            (10) X' = X \cup \{x\}
967
            (11) T' \subseteq T
968
            From [6], [7]:
969
                (12) s = (x := v); s''
970
971
            Case analysis on [2]:
                Case for the rule E-Assign:
972
                    (13) t = v
973
                    (14) s' = s''
974
                    Case analysis on v:
975
976
                        Case (15) v = r
                            By Lemma 6 on [9], [13] and [15]
977
                                (16) T' = \mathbb{R}
978
                            From [16], [11]
```

Case for the rule E-Assign:

```
(17) \{ \mathbb{R} \} \subseteq T
981
                             From [15], [17]
982
983
                                 (18) v \in T
                             From [3], [10], [8], [18]
984
                                 (19) consistent(\Gamma, X', \sigma[x \mapsto v])
                             From [5], [14]
                                 (20) \Gamma, X' \vdash s', X''
988
                             The conclusion is [20] and [19].
                         Case T' = \mathbb{N}
                             Similar to the previous case.
                        Case t = mat
                             Similar to the previous case.
992
                Case for the rule E-AssignR:
994
                    (\sigma, v) \rightarrow t'
                    There is no reduction rule for values. Contradiction.
996
           Case for the rule T-Assign-2:
998
              The reduction is with the rule E-Assign'. Similar to the case for the rule T-Assign-2, the
              consistency of \sigma with \Gamma and X and that T' \subseteq T implies the consistency of (\sigma \setminus \{x'\})[x \mapsto \sigma(x')]
1000
              with \Gamma and X \setminus \{x'\} \cup \{x\}.
1001
1002
1003
1004
           Case for the rule T-Assign-3:
            (7) i = (x := t)
1005
            (8) x : T \in \Gamma
1006
            (9) \Gamma, X \vdash t : T'
1007
            (10) (T' = \mathbb{R} \vee T' = \mathbb{N}) \wedge t \notin \mathcal{V} \cup \mathcal{X}
1008
            (11) T' \subseteq T
1009
            (12) X' = X \cup \{x\}
1010
            From [6], [7]:
1011
                (13) s = (x := t); s''
1012
            Case analysis on [2]:
1013
                Case for the rule E-AssignR:
1014
                    (14) s' = (x := t'); s''
1015
                    (15)(\sigma,t) \rightarrow t'
1016
                    By Lemma 12 on [9], [15], and [3]:
1017
                         (16) \Gamma, X \vdash t' : T
1018
                    By Lemma 8 on [15]
1019
1020
                        (17) t' \notin X
                    From [17] and either the rule T-Assign-1 or the rule T-Assign-3 on [8], [16], [10], [11]:
1021
                        (18) \Gamma, X \vdash x := t', X \cup \{x\}
1022
                    From [18] and [12]:
1023
                        (19) \Gamma, X \vdash x := t', X'
1024
1025
                    From the rule T-INST on [19], [5], and then [14]:
                         (20) \Gamma, X \vdash s', X''
1026
                    The conclusion is [20], [3].
1027
```

:22 Anon.

```
1030
                     t = v
                     Contradiction with [10].
1031
1032
                 Case for the rule E-Assign':
1033
                     t = r
                     Contradiction with [10].
1035
            Case for the rule T-MIX:
1037
             (7) i = (x := mix x_1 with x_2 for t)
             (8) \Gamma, X \vdash x_1 : \cup \overline{Mat_i}
             (9) \Gamma, X \vdash x_2 : \cup \overline{Mat_i}
1039
```

(10) $\Gamma, X \vdash t : \mathbb{R}$ (11) interact-abs $(Mat_i, Mat_i) \subseteq \Gamma(x)$

 $(12) X \setminus \{x_1, x_2\} \cup \{x\}$

Case analysis on [2]: There are two cases.

Case of the rule E-MIXR:

By Lemma 12, the type of t is preserved for t'. Thus, the rule T-MIX is applied to the new mix instruction. The assumed consistency condition is preserved since the store σ stays unchanged in the step.

Case of the rule E-MIX:

We already have the typing judgement for the remaining statement s'' in [5]. By Lemma 6 on [8] and the consistency condition, there exists i such that $\sigma(x_1) \in \underline{Mat_i}$ where $\Gamma(x_1) = \cup \overline{Mat_i}$. Similarly, there exists j, $\sigma(x_2) \in Mat_j$ where $\Gamma(x_2) = \cup \overline{Mat_j}$. Since interact-abs(Mat_i, Mat_j) is defined, from the conservative property of the abstract interact-abs function, we have interact($\sigma(x_1), \sigma(x_2), r$) \in interact-abs(Mat_i, Mat_j). Thus, from [11], we have interact ($\sigma(x_1), \sigma(x_2), r$) \in $\Gamma(x)$. From this and the consistency assumption for σ , we have $conistent(\Gamma, X \cup \{x\}, \sigma[x \mapsto interact(\sigma(x_1), \sigma(x_2), r)]$). Therefore, we have $conistent(\Gamma, X \setminus \{x_1, x_2\} \cup \{x\}, \sigma \setminus \{x_1, x_2\} \mid x \mapsto interact(\sigma(x_1), \sigma(x_2), r)]$).

```
Case for the rule T-Split:
```

```
(7) i = (\langle x_1, ... x_n \rangle = \text{split } x \text{ into } n)
(8) \Gamma, X \vdash x : \cup \overline{Mat_i}
```

 $(9) \Gamma(x) \subseteq \Gamma(x_1), ..., \Gamma(x) \subseteq \Gamma(x_n)$

 $(10) X' = X \setminus \{x\} \cup \{x_1, ..., x_n\}$

Case analysis on [2]: There is only one case. Case of the rule E-SPLIT: We already have the typing judgement for the remaining statement s'' in [5]. By Lemma 6 on [8], we have $\Gamma(x) = \cup \overline{Mat_i}$ and $x \in X$. From the consistency condition, we have $\sigma(x) \in \Gamma(x)$. From the type of the split function, we have split($\sigma(x)$, n) $\in \Gamma(x)$. From this and [9], we have that for every $j \in \{1..n\}$, split($\sigma(x)$, n) $\in \Gamma(x_j)$. From this and the consistency assumption for σ , we have $conistent(\Gamma, X \cup \{x_1, ..., x_n\}, \sigma[x_i \mapsto split(\sigma(x), n)])$. Therefore, we have $conistent(\Gamma, X \setminus \{x\} \cup \{x_1, ..., x_n\}, (\sigma \setminus \{x\})[x_i \mapsto split(\sigma(x), n)])$.

, Vol. 1, No. 1, Article . Publication date: February 2017.

```
Case for the rule T-IF
1079
             (7) i = \text{if } t \text{ then } s_1 \text{ else } s_2
1080
1081
             (8) \Gamma, X \vdash t : \mathbb{N}
             (9) \Gamma, X \vdash s_1, X_1
1082
             (10) \Gamma, X \vdash s_2, X_2
1083
             (11) X' = X_1 \cap X_2
             Case analysis on [2]: There are three cases.
                  Case of the rule E-IFR:
1088
                             By Lemma 12, the type of t is preserved for t'. Thus, the rule T-IF is applied to the
                             new if instruction. The assumed consistency condition is preserved since the store
1090
                             \sigma stays unchanged in the step.
1092
                 Case of the rule E-IFTRUE:
                             From [11], we have X_1 \subseteq X'. From this and [9], by Lemma 1, we have \Gamma, X \vdash s_1, X'.
1094
                             From this and [5], by Lemma 3, we have \Gamma, X \vdash s_1 \bullet s'', X''. The assumed consistency
1096
                             condition is preserved since the store \sigma stays unchanged in the step.
                 Case of the rule E-IFFALSE:
1098
                             Similar to the previous case.
1099
1100
            Case for the rule T-WHILE:
1101
             (7) i = \text{while } t s'''
1102
             (8) \Gamma, X \vdash \text{while } t \ s''', X
1103
             (9) \Gamma, X \vdash t : \mathbb{N}
1104
             (10) \Gamma, X \vdash s^{\prime\prime\prime}, X^{\prime\prime\prime\prime}
1105
             (11) X \subseteq X'''
1106
             (12) X' = X
1107
1108
            Case analysis on [2]: There is only one case.
1109
             Case for the rule E-WHILE:
1110
                  (13) s' = \text{if } t \text{ then } (s''' \bullet \text{ while } t s'''; s'') \text{ else } s''.
1111
                 By Lemma 4 on [10], [11] and [8], we have
1112
                      (14) \Gamma, X \vdash (s''' \bullet \text{ while } t \ s'''), X_1
1113
                      (15) X \subseteq X_1
1114
                 From [5] and [12], we have
1115
                      (16) \Gamma, X \vdash s'', X''
1116
                 By Lemma 2 on [14], [15] and [16], we have
1117
                      (17) \Gamma, X \vdash (s''' \bullet \text{ while } t \ s'''; \ s''), X_2
1118
                      (18) X'' \subseteq X_2
1119
                 By the rule T-IF on [9], [17], [16], we have
1120
                      (19) \Gamma, X \vdash \text{if } t \text{ then } (s''' \bullet \text{ while } t \text{ s'''}; \text{ s''}) \text{ else } s'', X_2 \cap X''
1121
```

From [19], [13] and [18], we have

stays unchanged in the step.

The assumed consistency condition [3] is preserved since the store σ

 $\Gamma, X \vdash s', X''$

1122

1123

1124

1125 1126

1127

:24 Anon.

```
Proof of Soundness
1128
1129
           LEMMA 14 (SOUNDNESS OF TYPE INFERENCE FOR TERMS).
1130
        For every \Gamma, X, t, T, C and m,
1131
1132
            \Gamma, X \vdash t : T \mid C
1133
            m is a model for C
            m(\Gamma), X \vdash t : m(T)
1137
           Proof.
1139
        Hypothesis:
            (1) \Gamma, X \vdash t : T \mid C
1141
            (2) m is a model for C
1142
1143
           Structural induction on [1]:
1144
        Case the rule CT-VAR:
1145
            Trivial
1146
1147
           Case the rule CT-MATH:
            (3) \Gamma, X \vdash t_1 : T_1 \mid C_1
1149
            (4) \Gamma, X \vdash t_2 : T_2 \mid C_2
1150
            (5) C = C_1 \cup C_2 \cup \{T_1 = T_2 = \mathbb{N} \lor T_1 = T_2 = \mathbb{R}\}
1151
            (6) T = T_1
1152
            (7) t = t_1 \oplus t_2
1153
            From [2] and [5]:
1154
                (8) m is a model for C_1.
1155
                 (9) m is a model for C_2.
1156
            By I.H. on ([3], [8]), ([4], [9]):
1157
                 (10) m(\Gamma), X \vdash t_1 : m(T_1)
1158
                (11) m(\Gamma), X \vdash t_2 : m(T_2)
1159
            From [5], [2]:
1160
                 m(T_1) = m(T_2) = \mathbb{N} \lor m(T_1) = m(T_2) = \mathbb{R}
1161
            We consider the first disjunct. The case for the second one is similar.
1162
                (12) m(T_1) = m(T_2) = \mathbb{N}
1163
            From [12], [10] and [11]:
1164
                (13) m(\Gamma), X \vdash t_1 : \mathbb{N} \land m(\Gamma), X \vdash t_2 : \mathbb{N}
1165
            By the rule Т-Матн on [13], [14]:
1166
                 (15) m(\Gamma), X \vdash t_1 \oplus t_2 : \mathbb{N}
1167
            From [15], [6], [7] and [12]:
1168
                (16) m(\Gamma), X \vdash t : m(T)
1169
1170
           Case the rule CT-DETECT:
1171
1172
              Similar to the previous case. The only interesting step is that considering the syntax of types
1173
              T, the equality m(T_1) \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset implies m(T_1) = \bigcup \overline{Mat_i}.
1174
1175
```

```
Case the rule CT-MAT:
1177
           Trivial
1178
1179
          Case the rule CT-REAL:
1180
           Trivial
1181
          Case the rule CT-NAT:
           Trivial
1185
1186
          LEMMA 15 (SOUNDNESS OF TYPE INFERENCE FOR STATEMENTS).
1188
       For every \Gamma, X, s, C, X'', and m,
1189
1190
           \Gamma, X \vdash s, X'' \mid C
1191
           m is a model for C
1192
       then
1193
           m(\Gamma), X \vdash s, X''
1194
1195
          Proof.
1196
       Hypothesis:
           (1) \Gamma, X \vdash s \mid C
1197
           (2) m is a model for C
1198
1199
1200
       Structural induction on [1]:
       Case the rule CT-Skip:
1201
           Trivial.
1202
1203
          Case the rule CT-INST:
1204
1205
           (3) s = i; s'
           (4) \Gamma, X \vdash i, X' \mid C_1
1206
           (5) \Gamma, X' \vdash s', X'' \mid C_2
1207
           (6) C = C_1 \cup C_2
1208
           From [2] and [6]:
1209
           (7) m is a model for C_1
1210
1211
           (8) m is a model for C_2
           By I.H. on [5]:
1212
                (9) m(\Gamma), X' \vdash s, X''
1213
           We need to show
1214
               (10) m(\Gamma), X \vdash i, X'
1215
1216
           and then by the rule T-INST on [9], [10]:
               (11) m(\Gamma), X \vdash i; s', X''
1217
           and then by from [10], [11]:
1218
               m(\Gamma), X \vdash s, X''
1219
1220
           Now assuming [4] and [7] that is
1221
               (4) \Gamma, X \vdash i, X' \mid C_1
1222
               (7) m is a model for C_1
1223
           we show that
1224
```

:26 Anon.

```
m(\Gamma), X \vdash i, X'
1226
1227
1228
             Case analysis on [4]:
1229
             Case the rule CT-Assign-1:
                 (12) i = (x := v)
                 (13) x : T \in \Gamma
1233
                 (14) \Gamma, X \vdash v : T' \mid C'
                 (15) X' = X \cup \{x\}
                 (16) C_1 = C' \cup \{T' \subseteq T\}
1235
                 From [7] and [16]:
                     (17) m is a model for C'.
1237
                     (18) m(T') \subseteq m(T)
1239
                 From [13]:
                     (19) x : m(T) \in m(\Gamma)
                 From Lemma 14 on [14] and [17]:
1241
                      (20) m(\Gamma), X \vdash mat : m(T')
1243
                 By the rule T-Assign on [19], [20], and [18]:
                      (21) m(\Gamma), X \vdash x := mat, X \cup \{x\}
                 From [21], [15] and [12]:
1245
                     m(\Gamma), X \vdash i, X'
1247
1248
             Case the rule CT-Assign-2:
1249
1250
                    Similar to the previous case. The the rule T-Assign-2 is used.
1251
             Case the rule CT-Assign-3:
1252
1253
1254
                    Similar to the case the rule CT-Assign-1. To use the rule T-Assign, in contrast to the case
                    the rule CT-Assign-1 that proved t = v, the disjunct T' = \mathbb{R} \vee T' = \mathbb{N} is proved to use the
1255
                    rule T-Assign-3.
1256
1257
             Case the rule CT-MIX:
1258
                 (12) \Gamma, X \vdash x_1 : T \mid C
1259
                 (13) \Gamma, X \vdash x_2 : T' \mid C'
1260
                 (14) \Gamma, X \vdash t : T'' \mid C''
1261
                 (15) i = (x := mix x_1 with x_2 for t)
1262
                 (16) X' = X \setminus \{x_1, x_2\} \cup \{x\}
1263
                 (17) C_1 = C \cup C' \cup C'' \cup \{T \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset, \ T' \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset, \ T'' = \mathbb{R}
1264
                          \overline{Mat_i \in T \land Mat_i \in T'} \Rightarrow \text{interact-abs}(Mat_i, Mat_i) \subseteq \Gamma(x)
1265
1266
                 From [7] and [17]:
1267
                     (18) m is a model for C.
                      (19) m is a model for C'.
1268
1269
                      (20) m is a model for C''.
1270
                      (21) m(T) \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset
1271
                      (22) m(T') \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset
1272
                      (23) m(T^{\prime\prime}) = \mathbb{R}
1273
```

```
(24) Mat_i \in m(T) \land Mat_i \in m(T') \Rightarrow \text{interact-abs}(Mat_i, Mat_i) \subseteq m(\Gamma(x))
1275
               From Lemma 14 on [12] and [18]:
1276
                    (25) m(\Gamma), X \vdash x_1 : m(T)
1277
               From Lemma 14 on [13] and [19]:
1278
                    (26) m(\Gamma), X \vdash x_2 : m(T')
               From Lemma 14 on [14] and [20]:
                    (27) m(\Gamma), X \vdash t : m(T'')
               From [21], there exists is such that:
                    (28) m(T) = \bigcup \overline{Mat_i}
               From [22], there exists js such that:
                   (29) m(T') = \bigcup \overline{Mat_i}
               From [25] and [28]:
                   (30) m(\Gamma), X \vdash x_1 : \cup \overline{Mat_i}
1288
               From [26] and [29]:
                   (31) m(\Gamma), X \vdash x_2 : \cup \overline{Mat_i}
1290
               From [27] and [23]:
                    (32) m(\Gamma), X \vdash t : \mathbb{R}
1292
               From [24], [28], and [29]:
                    (33) interact-abs(Mat_i, Mat_i) \subseteq \Gamma(x)
1294
               By the rule T-MIX on [30], [31], [32], and [33]:
                    (34) \Gamma, X \vdash x := \min x_1 \text{ with } x_2 \text{ for } t, X \setminus \{x_1, x_2\} \cup \{x\}
1296
               From [34], [15] and [16]:
1297
                   (34) \Gamma, X \vdash i, X'
1298
1299
            Case the rule CT-Split:
1300
1301
                  Similar to the case for the rule CT-Mix. The Lemma 14 is applied to the constraint typing
1302
                  judgement for x.
1303
1304
            Case the rule CT-IF:
1305
1306
                  Immediate from applying the Lemma 14 on t and the induction hypothesis on s_1 and s_2.
1307
1308
            Case the rule CT-WHILE:
1309
1310
                  Immediate from applying the Lemma 14 on t and the induction hypothesis on s.
1311
1312
                                                                                                                               1313
1314
       2.5 Proof of Completeness
1315
          LEMMA 16 (COMPLETENESS OF TYPE INFERENCE FOR TERMS).
1316
       For all \Gamma, X, t, T, and C,
1317
1318
           m(\Gamma), X \vdash t : T
1319
           \Gamma, X \vdash t : T' \mid C
1320
1321
           m is a model for C
1322
```

:28 Anon.

```
m(T') = T
1324
1325
1326
          Proof.
1327
       Hypothesis
1328
           (1) m(\Gamma), X \vdash t : T
1329
            (2) \Gamma, X \vdash t : T' \mid C
1331
       Proof by induction on the given constraint typing derivation [2].
1332
1333
          Case for the rule CT-VAR:
            (3) t = x
1335
            (4) x : T \in \Gamma
            (5) x \in X
1337
            (6) C = \emptyset
            By the inversion Lemma 6 on [1]:
1339
               (7) x : T' \in m(\Gamma)
1340
                (8) x \in X
1341
            From [4] and [7]:
               (7) T' = m(T)
1343
            The conclusion is immediate from [6] and [7].
1345
           Case for the rule CT-MATH:
1346
            (3) \Gamma, X \vdash t_1 : T_1 \mid C_1
1347
            (4) \Gamma, X \vdash t_2 : T_2 \mid C_2
1348
            (5) C = C_1 \cup C_2 \cup \{T_1 = T_2 = \mathbb{N} \vee T_1 = T_2 = \mathbb{R}\}
1349
            (6) T' = T_1
1350
            By the inversion Lemma 6 on [1]:
1351
                (7) m(\Gamma), X \vdash t_1 : T
1352
               (8) m(\Gamma), X \vdash t_2 : T
1353
               T=\mathbb{N}\vee T=\mathbb{R}
1354
            We consider the case for the first disjunct:
1355
            (The case for the second one is similar.)
1356
                (9) T = \mathbb{N}
1357
            By induction hypothesis on [7] and [3]:
1358
                (10) m is a model for C_1
1359
                (11) m(T_1) = T
1360
            By induction hypothesis on on [8] and [4]:
1361
               (12) m is a model for C_2
1362
               (13) m(T_2) = T
1363
            From [11], [13] and [9]:
1364
               (14) m(T_1) = m(T_2) = \mathbb{N}
1365
            From [5], [10], [12] and [14]:
1366
                (15) m is a model for C.
1367
            From [6], [11], and [9]:
1368
                (16) m(T') = T
1369
            The conclusion is [15] and [16].
1370
1371
```

```
Case for the rule CT-DETECT:
1373
1374
1375
            Similar to the case for the rule T-MATH. By induction hypothesis on x and t.
1376
          Case for the rule CT-MAT
1377
           Trivial.
          Case for the rule CT-REAL:
1380
1381
           Trivial.
1382
          Case for the rule CT-NAT:
           Trivial.
1384
1386
                                                                                                                           1387
          LEMMA 17 (COMPLETENESS OF TYPE INFERENCE FOR STATEMENTS).
1388
       For all \Gamma, X, s, X', C, and m,
1389
       if
1390
           m(\Gamma), X \vdash s, X'
1391
           \Gamma, X \vdash s, X'' \mid C
1392
       then
1393
           m is a model for C.
1394
1395
1396
          Proof.
       Hypothesis
1397
           (1) m(\Gamma), X \vdash s, X_1
1398
           (2) \Gamma, X \vdash s, X_2 \mid C
1399
       We show that
1400
           m is a model for C.
1401
1402
          Induction on the derivation of [2]:
1403
1404
          Case for the rule CT-Skip:
1405
           Trivial.
1406
1407
          Case for the rule CT-INST:
1408
           (3) s = i; s'
1409
           (4) \Gamma, X \vdash i, X' \mid C_1
1410
           (5) \Gamma, X' \vdash s', X_2 \mid C_2
1411
1412
           (6) C = C_1 \cup C_2
           From [1] and [3]:
1413
               (7) m(\Gamma), X \vdash i; s', X_1
1414
           By the inversion Lemma 7 on [7]:
1415
               (8) m(\Gamma), X \vdash i, X''
1416
               (9) m(\Gamma), X'' \vdash s', X_2
1417
           By induction hypothesis on [9] and [5]:
1418
               (10) m is a model for C_2
1419
           We will show that from [8] and [4]:
1420
```

:30 Anon.

```
(11) m is a model for C_1
1422
            From [6], [10] and [11]:
1423
1424
                m is a model for C
1425
            Hypothesis:
1426
                (8) m(\Gamma), X \vdash i, X''
                (4) \Gamma, X \vdash i, X' \mid C_1
1429
            Conclusion:
                m is a model for C_1
1431
1432
            Case analysis on derivation of [4]:
1433
                Case for the rule CT-Assign-1:
1435
                    (12) i = (x := v)
                    (13) x : T \in \Gamma
1437
                    (14) \Gamma, X \vdash \upsilon : T' \mid C'
                    (15) X' = X \cup \{x\}
                    (16) C_1 = C' \cup \{T' \subseteq T\}
1439
1440
                    From [8] and [12]:
                        (17) m(\Gamma), X \vdash x := v, X''
1441
                    By the inversion Lemma 7 on [17]:
1442
                        (18) x : T'' \in m(\Gamma)
1443
                        (19) m(\Gamma), X \vdash \upsilon : T'''
1444
                        (22) T^{\prime\prime\prime} \subseteq T^{\prime\prime}
1445
                         (23) X'' = X \cup \{x\}
1446
1447
                    From [18], [13]:
                         (24) m(T) = T^{\prime\prime}
1448
1449
                    By Lemma 16 on [19], [14]:
1450
                         (25) m is a model for C'
                         (26) m(T') = T'''
1451
                    From [22], [24] and [26]
1452
                        (27) m(T') \subseteq m(T)
1453
                    From [16], [24], [25] and [27]
1454
1455
                         m is a model for C_1
1456
                Case for the rule CT-Assign-2:
1457
1458
                         Similar to the case for the rule CT-Assign-1.
1459
1460
                Case for the rule CT-Assign-3:
1461
1462
                         Similar to the case for the rule CT-Assign-1.
1463
1464
                Case for the rule CT-MIX:
1465
                    (12) i = (x := mix x_1 with x_2 for t)
1466
                    (12) \Gamma, X \vdash x_1 : T \mid C
1467
                    (13) \Gamma, X \vdash x_2 : T' \mid C'
1468
                    (14) \Gamma, X \vdash t : T'' \mid C''
1469
1470
```

```
(15) C_1 = C \cup C' \cup C'' \cup
1471
                                \{T \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset, \ T' \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset, \ T'' = \mathbb{R}
1472
                                Mat_k \in T \land Mat_l \in T' \Rightarrow interact-abs(Mat_k, Mat_l) \subseteq \Gamma(x)
1473
1474
                       From [8] and [12]:
                            (16) m(\Gamma), X \vdash x := \min x_1 \text{ with } x_2 \text{ for } t, X''
1475
                       By the inversion Lemma 7 on [16]:
                            (17) m(\Gamma), X \vdash x_1 : \cup Mat_i
                            (18) m(\Gamma), X \vdash x_2 : \cup Mat_i
                           (19) m(\Gamma), X \vdash t : \mathbb{R}
                            (20) interact-abs(Mat_i, Mat_i) \subseteq m(\Gamma(x))
                       By Lemma 16 on [17], [12]:
1482
                            (21) m is a model for C
                           (22) m(T) = \bigcup Mat_i
                       By Lemma 16 on [18], [13]:
                            (23) m is a model for C'
                            (24) m(T') = \bigcup \overline{Mat_i}
                       By Lemma 16 on [19], [14]:
                            (25) m is a model for C''
                            (26) m(T'') = \mathbb{R}
1490
                       From [22], [24], and [20]:
                            (27) Mat_k \in m(T) \land Mat_l \in m(T') \Rightarrow interact-abs(Mat_k, Mat_l) \subseteq m(\Gamma(x))
1492
                       From [22], [24], and [26]:
1493
                            (28) m(T) \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset, \ m(T') \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset, \ m(T'') = \mathbb{R}
1494
                       From [15], [21], [23], [25], [28] and [27]:
1495
                            m is a model for C_1
1496
1497
                  Case for the rule CT-Split:
1498
```

Similar to the case for the rule CT-MIX.

Case for the rule CT-IF:

Immediate from the induction hypothesis on s_1 and s_2 and Lemma 16 on t.

Case for the rule CT-WHILE:

Immediate from the induction hypothesis on s and Lemma 16 on *t*.

3 SYNTAX STUDY: ELISA PROTOCOLS

The enzyme-linked immunosorbent assay (ELISA) is a test that uses antibodies and color changes to identify a substance. ELISA assays are commonly performed by interacting a group of chemicals with an immobile antibody to detect the presence and concentration of particular opiates. A technique was introduced, to allow these assays to be performed on DMFB technology by baking the enzyme directly onto the top plate of the DMFB, requiring further syntactic extensions to declare stationary substances and to explicitly move mobile substances to interact with them.

1499

1500 1501

1502 1503

1504 1505

1506 1507

1508 1509

1510 1511

1512

1513

1514

1515

1516

1517

:32 Anon.

8 shows an example ELISA assay, which is one of the steps of the larger hierarchical opiatedetection decision tree. The operator will need to swap out top plates with different antigens for each ELISA assay, or use a very large electrowetting array to support a top plate to which all necessary antigens for all of the ELISA assays are affixed.

Biologists use many equivalent terms that mean the same thing as Mix. BioScript supports several of these terms, including Tap and Vortex, as shown in 8. The execution engine recognizes these terms as being equivalent and converts them all to its own internal mix operation.

The repeat operation is part of BioScript's core instruction set. It allows the programmer to specify a sequence of instructions that will be repeated a constant number of times, which is far more common in biochemistry than in general computer programming. Thus, instead of the complex, unconstrained loop syntax employed by modern programming languages, we have opted for a simple syntax to make BioScript accessible to scientists with limited programming experience.

The Incubate and Heat operations leverage the external heaters that are integrated into or placed near the DMFB. A DMFB that lacks heating capabilities cannot perform this ELISA assay. Thus, the compiler writer must specialize a language syntax for each DMFB variant based on its integrated peripherals.

The final operation in any ELISA is the detection phase that compares the reading obtained from the sample to a reading obtained from a control. This provides a certain measurement of the presence and concentration of the opiate in the sample. In the hierarchical decision tree immunoassay, the result of this comparison determines which ELISA assay to execute next.

```
1542
     Stationary: Anti-Fentanyl
1543
1544
     Move 20uL of Urine Sample to Anti-Fentanyl
1545
     Move 100uL of Fentanyl-Conjugate to Anti-Fentanyl
1546
     Tap Anti-Fentanyl for 60s
1547
     Incubate Anti-Fentanyl at 23žC for 60min
1548
     Drain Anti-Fentanyl
1549
1550
     Repeat 6 times {
1551
       Move 350uL of Distilled Water to Anti-Fentanyl
1552
       Vortex Anti-Fentanyl for 45s
1553
       Drain Anti-Fentanyl
1554
     }
1555
1556
     Move 100uL of TMB Substrate to Anti-Fentanyl
1557
     Incubate Anti-Fentanyl at 23C for 30min
1558
     Mix Anti-Fentanyl with 100uL of Stop Reagent
1559
                                                   for 60s
1560
     Negative Reading = Measure the fluorescence of
1561
                                 Anti-Fentanyl for 30min
1562
     Drain Anti-Fentanyl
1563
1564
                                        Fig. 8. Fentanyl ELISA
1565
```

1521 1522

1523

1525

1527

1529

1531 1532

1533

15341535

1536

1537

1538

1539 1540 1541

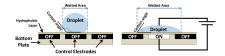


Fig. 9. Depiction of the electrowetting principle [29, 37]: applying an electrostatic potential to a droplet at rest reduces the contact angle with the surface, thereby increasing the surface area in contact with the droplet.

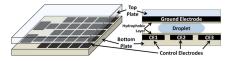


Fig. 10. Left: A digital microfluidic biochip (DMFB) is a planar array of electrodes [15, 19, 36, 39, 41]. Right: Cross-sectional view.

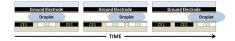


Fig. 11. A droplet is transported from CE2 to CE3 by activating CE3, and then deactivating CE2 (white: activated electrode; black: deactivated electrode).



Fig. 12. Five basic fluidic operations.

4 BACKGROUND

4.1 Digital Microfluidic Technology

As a proof of concept, the current version of the BioScript compiler targets *digital microfluidic biochips (DMFBs)*, which are LoCs based on *electrowetting on dielectric (EWoD)* technology. Fig. 9 illustrates the electrowetting effect: applying an electrical potential to a liquid droplet resting on a hydrophobic surface decreases the contact angle with the surface. This increases the "wet" area of contact between the droplet and the surface [29, 37]. As shown in Fig. 10, a DMFB is a planar array of electrodes which leverages the electrowetting principle to induce droplet transport [15, 19, 36, 39, 41]. Activating and deactivating nearby electrodes in sequence induces droplet transport, as shown in Fig. 11.

A DMFB has a small *instruction set architecture (ISA)* consisting of five primitive operations, as shown in Fig. 12: droplet transport, splitting and merging two droplets, actively mixing two (or more) droplets by first merging them and moving them (typically in a confined space around a pivot), and droplet storage. Droplet input from reservoirs on the perimeter of the chip and output (including waste/disposal) can also be controlled electrically. Since the operations in Fig. 12 can be performed anywhere on the DMFB, a DMFB can be interpreted as a reconfigurable spatial computing device, similar in principle to an FPGA, as the function performed by a (group) of electrode(s) may change over time.

Additional operations can be realized by integrating external devices such as heaters [32], optical detectors [31], and a variety of integrated sensors [27, 38, 45, 46]. In particular, the hierarchical

:34 Anon.

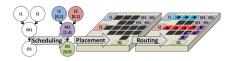


Fig. 13. Illustration of a typical DMFB compiler for assays without control flow (DAGs). The compiler must schedule, place, and route all operations to produce an executable sequence of electrode activations to automatically execute the assay (activated electrodes typically have a droplet on top).

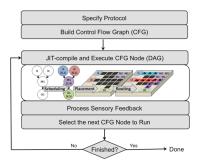


Fig. 14. A dynamic execution model for cyber-physical DMFBs.

decision tree opiate detection immunoassay will require optical sensors for *immunofluorescence* and impedimetric immunosensors utilizing *electrochemical impedance spectroscopy (EIS)* [5, 10, 51].

Despite the lack of high-level language support, compilers for assays without control flow are mature. As shown in Fig. 13 the compiler must solve three interdependent NP-complete problems: operation scheduling [11, 16, 30, 40, 42, 49], reconfigurable module placement [8, 17, 28, 33, 34, 48, 52, 53, 55], and droplet routing [7, 9, 22, 25, 26, 43, 44, 50, 56]. The scheduler must determine the time steps at which each biochemical operation occurs, while satisfying droplet dependency constraints and physical resource constraints of the device. The placer determines the location on the 2D electrode array where each operation is performed as the reaction progresses over time. The router ensures that droplets are transported from their start/stop points at appropriate times during execution of the chemical reaction, while ensuring that droplets undergoing transport do not inadvertently collide with one another or any other ongoing operations on the chip. The router may also introduce wash droplets to remove residue left by droplets that travel over the surface of the chip, to prevent cross-contamination [23, 54, 57].

A *cyber-physical* DMFB leverages integrated sensors [31, 38, 45, 46] and/or real-time video monitoring [6, 13, 21, 47] to send online feedback in real-time to a PC controller. Cyber-physical integration provides the capability of control flow [18]: the assay designer can specify different steps to take based on the sensor feedback.

As control flow cannot be predicted statically, the simple compiler model shown in Fig. 13 no longer suffices. In response, a software-based dynamic execution model, shown in Fig. 14 was introduced [18]. The assay, which is now specified as a *control flow graph* is compiled and executed one basic block at a time. Each basic block other than the CFG exit node terminates with a condition, typically based on sensory feedback. The control software evaluates the condition, determines the next basic block to execute, quickly compiles it on-the-fly, and then initiates execution.

b = Mix a and Nitrogen for 10s

Calcium_Hypo = 50uL of [0-]C1.[0-]C1.[Ca+2]

Dichlor = 50uL CC(=CCl)Cl

1709

1710

1711 1712

1713

17141715

5 ASSAY EXECUTION VIDEOS 1667 1668 Below is a list of the videos generated while gathering data. We are not claiming these as novel or 1669 interesting contributions. They are merely provided to demonstrate how the simulator executes a 1670 given assay. 1671 Urine Opiate Panel with initial positive indication: https://ldrv.ms/v/s!AqwZF7jQGx_IgahNYojptiZr4RpjOg Urine Opiate Panel with initial negative indication: https://ldrv.ms/v/s!AqwZF7jQGx_IhJRtiOKGuSTm0qXX0 PCR Assay: https://ldrv.ms/v/s!AqwZF7jQGx_IhNgWWxzt8GwtfICcFw Image Probe Synthesis: https://ldrv.ms/v/s!AqwZF7jQGx IhNgbKsUIzDGqfyOhbw Neurotransmitter Sensing: https://ldrv.ms/v/s!AqwZF7jQGx_IhNgZjBF4vlsnDr1uCA PCR Droplet Replenishment Assay: https://ldrv.ms/v/s!AqwZF7jQGx_IhJRqfq1dLhbBUIL9RQ Probabilistic PCR Full: https://1drv.ms/v/s!AqwZF7jQGx_IhJRo-xDbVT_Q6UsXPg Probabilistic PCR Early Exit: https://ldrv.ms/v/s!AqwZF7jQGx_IhJRpLXveGI8Qr7BWrw Broad Spectrum Opiate Panel: https://1drv.ms/v/s!AqwZF7jQGx_IhJRs4f6laDxl6T_QJQ 1680 Fentanyl ELISA Assay: https://ldrv.ms/v/s!AqwZF7jQGx_IgahJFK519Id1HCad4g Ciprofloxacin ELISA Assay: https://ldrv.ms/v/s!AqwZF7jQGx_IgahKrD6EDWPbsy9G0g 1682 Heroin ELISA Assay: https://ldrv.ms/v/s!AqwZF7jQGx IgahIrCW0z08KR1111A Morphine ELISA Assay: https://ldrv.ms/v/s!AqwZF7jQGx_IgahG3PHVTctW7u0rOw 1684 Oxycodone ELISA Assay: https://1drv.ms/v/s!AqwZF7jQGx_IgahHIgmGgR43A-l7EA **BIOSCRIPT'S CHEMTYPE TESTS** 1686 1687 6.1 **Real-world Prevention Tests** 1688 AIHA Example1 where ChemType would have prevented an incident 1689 Tetrachloroethylene = 50uL of C(=C(Cl)Cl)(Cl)Cl 1690 Nitric_Acid = 50uL of [N+](=0)(0)[0-]1691 1692 a = Mix Tetrachloroethylene with Nitric_Acid for 40s 1693 ______ 1694 AIHA Example2 where ChemType would have prevented an incident 1695 1696 Methanol = 50uL of CO 1697 Nitric_Acid = 50uL of [N+](=0)(0)[0-]1698 1699 a = Mix Methanol with Nitric_Acid for 40s 1700 ______ 1701 AIHA Example3 where ChemType would have prevented an incident 1702 1703 Nitrogen = N#N 1704 Diaminopropane = 50uL of CC(CN)N 1705 Potassium_Hydride = 50uL of [H-].[K+] 1706 1707 a = Mix Diaminopropane and Potassium_Hydride for 20s

Real world example where ChemType would have prevented an incident

, Vol. 1, No. 1, Article . Publication date: February 2017.

:36 Anon.

```
1716
    Mix Dichlor and Calcium_Hypo for 20s
1717
1718
    ______
1719
1720
    6.2 Synthetic Failures
1721
    Example where ChemType will not allow the mix to occur
1722
    Nitric_Acid = 50uL of [N+](=0)(0)[0-]
1723
    Hydrochloric_Acid = 50uL of Cl
1724
1725
    a = Mix Nitric_Acid with Hydrochloric_Acid for 40s
1726
    ______
1727
      Example where ChemType will not allow the mix to occur
1728
1729
    Isopropanol = 50uL of CC(C)0
    Hydrochloric_Acid = 50uL of Cl
1730
1731
1732
    a = Mix Isopropanol with Hydrochloric_Acid for 40s
1733
    ______
1734
      Example where ChemType will not allow the mix to occur
1735
    Formaldehyde = 50uL of C=0
1736
    Formaldehyde2 = 50uL of C=0
1737
1738
    a = Mix Formaldehyde with Formaldehyde2 for 40s
1739
    ______
1740
      Example where ChemType will not allow the mix to occur
1741
    Benzoin = 50uL of C1=CC=C(C=C1)C(C(=0)C2=CC=CC=C2)0
1742
    Urea = 50uL of C(=0)(N)N
1743
1744
    Formaldehyde = 50uL of C=0
1745
    a = Mix Benzoin with Urea for 40s
1746
    b = Mix Formaldehyde with a for 20s
1747
1748
    ______
1749
      Example where ChemType will not allow the mix to occur
1750
    Benzene = 50uL of C1=CC=CC=C1
1751
    Formaldehvde = 50uL of C=0
1752
    Formaldehyde2 = 50uL of C=0
1753
1754
    a = Mix Benzene with Formaldehyde for 40s
1755
    b = Mix Formaldehyde2 with a for 20s
1756
    ______
1757
      Example where ChemType will not allow the mix to occur
1758
    Benzene = 50uL of C1=CC=CC=C1
1759
    Urea = 50uL of C(=0)(N)N
1760
    Hydrochloric Acid = 50uL of Cl
1761
1762
    a = Mix Benzene with Urea for 40s
1763
1764
```

```
b = Mix Hydrochloric Acid with a for 20s
1766
    ______
1767
     Example where ChemType would have prevented an incident
1768
    Hydrogen_Peroxide = 50uL 00
    Sulfuric\_Acid = 50uL of OS(=0)(=0)0
1770
    Acetone = 50uL of CC(=0)C
1771
1772
    a = Mix Hydrogen_Peroxide and Sulfuric_Acid for 20s
1773
    b = Mix a and Acetone for 10s
1774
    ______
1775
1776
    6.3 Synthetic Passes
1777
    Example where ChemType will allow the mix to occur
1778
    Benzene = 50uL of C1=CC=CC=C1
1779
    Urea = 50uL of C(=0)(N)N
1780
1781
    a = Mix Benzene with Urea for 40s
1782
    ______
1783
     Example where ChemType will allow the mix to occur
1784
    Benzene = 50uL of C1=CC=CC=C1
1785
1786
    Urea = 50uL of C(=0)(N)N
    Benzotrichloride = 50uL of C1=CC=C(C=C1)C(C1)(C1)C1
1787
1788
1789
    a = Mix Benzene with Urea for 40s
1790
    b = Mix Benzotrichloride with a for 20s
1791
    ______
1792
1793
```

7 BIOSCRIPT ASSAYS

7.1 Syntactic Sugar Translation

For the purposes of adhering to scientific assay nomenclature, we have chosen to represent the following assays in the way a biological scientist might. Adhering to the community's "syntax" requires some syntactic sugar. This mapping is described in Table 1.

```
Thus, given:
1800
      move x to y
1801
      tap y
1802
      BioScript would condense into:
1803
1804
      y := mix x with y
```

1794

1795

1796

1797

1798 1799

1805

1806

1807 1808

1809

1811 1812 1813

7.2 Urine Opiate Hierarchy

Two main paths based on outcome of Broad Spectrum Opiate(BSO) Test.

When(BSO) Test Indicates Positive: https://ldrv.ms/v/s!AqwZF7jQGx IgahNYojptiZr4RpjOg

When(BSO) Test Indicates Negative: https://ldrv.ms/v/s!AqwZF7jQGx IhJRtiOKGuSTm0qXX0g

:38 Anon.

```
General Instruction
                                  BioScript Equivalent
1815
                                  Anti-Morphine 1 := x \mu l of Anti-Morphine
1816
       stationary Anti-Morphine
       heat x at n
                                  Not yet modeling temperature\volume. Left for future work.
                                  Not yet modeling temperature\volume. Left for future work.
       incubate x at n
                                  y := \min x \text{ with } y
       move x to y
1819
                                  x := \min x \text{ with } x
       tap x
       invert x
                                  x := \min x \text{ with } x
1821
       vortex x
                                  x := \min x \text{ with } x
1822
       drain x
                                  freshVariable := x
1823
                                  while x > 0
       repeat x times
       measure x of x'
                                  \det x \text{ on } x'
1825
                                  x := \det x \text{ on } x'
       perform...
                          Table 1. Syntactic sugar translations from assay to BioScript.
1827
1829
1830
        BioScript Code:
1831
1832
      Stationary Anti-Morphine
      Stationary Anti-Oxy
1833
      Stationary Anti-Fentanyl
1834
      Stationary Anti-Ciprofloxcin
1835
      Stationary Anti-Heroin
1836
1837
      Stationary Oxycodone-Enzyme
1838
      Stop-Reagent1 = 10uL of Stop-Reagent
1839
      Stop-Reagent2 = 10uL of Stop-Reagent
1840
      Stop-Reagent3 = 10uL of Stop-Reagent
1841
      Stop-Reagent4 = 10uL of Stop-Reagent
1842
      Stop-Reagent5 = 10uL of Stop-Reagent
1843
1844
      Move 10uL of UrineSample to Anti-Morphine
1845
      Move 10uL of UrineSample to Anti-Oxy
1846
      Move 10uL of UrineSample to Anti-Fentanyl
1847
      Move 10uL of UrineSample to Anti-Ciprofloxcin
1848
      Move 10uL of UrineSample to Anti-Heroin
1849
1850
      MorphineReading = Measure the fluorescence of Anti-Morphine for 5s
1851
      OxyReading = Measure the fluorescence of Anti-Oxy for 5s
1852
      FentanylReading = Measure the fluorescence of Anti-Fentanyl for 5s
1853
      CiproReading = Measure the fluorescence of Anti-Ciprofloxcin for 5s
1854
      HeroinReading = Measure the fluorescence of Anti-Heroin for 5s
1855
1856
      Drain Anti-Morphine
1857
1858
      Drain Anti-Oxy
      Drain Anti-Fentanyl
1859
      Drain Anti-Ciprofloxcin
1860
      Drain Anti-Heroin
1861
1862
```

```
1863
     if (((FentanylReading == true) or (MorphineReading == true)) or (((OxyReading == true) or (
1864
1865
       Move 20uL of Urine-Sample to Anti-Heroin
1866
       Move 100uL of Heroin-Conjugate to Anti-Heroin
       Tap Anti-Heroin for 60s
       Incubate Anti-Heroin at 23C for 60min
1870
       Drain Anti-Heroin
       Repeat 6 times {
1872
         Move 350uL of Distilled-Water to Anti-Heroin
         Vortex Anti-Heroin for 45s
1874
         Drain Anti-Heroin
       }
1876
1878
       Move 100uL of TMB-Substrate to Anti-Heroin
1879
       Incubate Anti-Heroin at 23C for 30min
1880
       Mix Anti-Heroin with 100uL of Stop Reagent1 for 60s
1882
       Heroin Reading = Measure the fluorescence of Anti-Heroin for 30min
       Drain Anti-Heroin
1883
1884
1885
       Move 20uL of Urine-Sample to Oxycodone-Enzyme
1886
       Move 100uL of Oxycodone-Conjugate to Oxycodone-Enzyme
1887
       Tap Oxycodone-Enzyme for 60s
1888
       Incubate Oxycodone-Enzyme at 23C for 60min
1889
       Drain Oxycodone-Enzyme
1890
1891
       Repeat 6 times {
         Move 350uL of Distilled-Water to Oxycodone-Enzyme
1892
         Vortex Oxycodone-Enzyme for 45s
1893
1894
         Drain Oxycodone-Enzyme
1895
       }
1896
       Move 100uL of TMB-Substrate to Oxycodone-Enzyme
1897
       Incubate Oxycodone-Enzyme at 23C for 30min
1898
       Mix Oxycodone-Enzyme with 100uL of Stop-Reagent2 for 60s
1899
1900
1901
       Oxy Reading = Measure the fluorescence of Oxycodone-Enzyme for 30min
1902
       Drain Oxycodone-Enzyme
1903
       if ((Heroin Reading == false) and (Oxy Reading == false)) {
1904
         Move 20uL of UrineSample to Anti-Ciprofloxacin
1905
         Move 100uL of Ciprofloxacin-Conjugate to Anti-Ciprofloxacin
1906
1907
         Tap Anti-Ciprofloxacin for 60s
         Incubate Anti-Ciprofloxacin at 23C for 60min
1908
         Drain Anti-Ciprofloxacin
1909
1910
```

:40 Anon.

```
1912
         Repeat 5 times {
           Move 250uL of Distilled-Water to Anti-Ciprofloxacin
1913
1914
            Vortex Anti-Ciprofloxacin for 45s
            Drain Anti-Ciprofloxacin
1915
          }
1917
         Move 50uL of TMB-Substrate to Anti-Ciprofloxacin
1918
1919
          Incubate Anti-Ciprofloxacin at 25C for 30min
         Mix Anti-Ciprofloxacin with 100L of Stop Reagent3 for 60s
1920
1921
         Urine Reading = Measure the fluorescence of Anti-Ciprofloxacin for 5min
1922
1923
         Drain Anti-Ciprofloxacin
1924
       }
1925
     } else {
1926
1927
       Split Epithelial = Mix UrineSample with Azoxymethane for 5s
1928
       Move Split Epithelial to Anti-Fentanyl
1929
       Move 100uL of Fentanyl-Conjugate to Anti-Fentanyl
1930
       Tap Anti-Fentanyl for 60s
       Incubate Anti-Fentanyl at 23C for 60min
1931
1932
       Drain Anti-Fentanyl
1933
1934
       Repeat 6 times {
1935
         Move 350uL of Distilled-Water to Anti-Fentanyl
1936
         Vortex Anti-Fentanyl for 45s
         Drain Anti-Fentanyl
1937
1938
       }
1939
1940
       Move 100uL of TMB-Substrate to Anti-Fentanyl
       Incubate Anti-Fentanyl at 23C for 30min
1941
1942
       Mix Anti-Fentanyl with 100uL of Stop Reagent4 for 60s
1943
       Urine Reading = Measure the fluorescence of Anti-Fentanyl for 30min
1944
       Drain Anti-Fentanyl
1945
       if (!(Heroin Reading >= FentanylControl)) {
1946
         Move 20uL of Urine-Sample to Oxycodone-Enzyme
1947
         Move 100uL of Oxycodone-Conjugate to Oxycodone-Enzyme
1948
          Tap Oxycodone-Enzyme for 60s
1949
1950
          Incubate Oxycodone-Enzyme at 23C for 60min
1951
         Drain Oxycodone-Enzyme
1952
         Repeat 6 times {
1953
           Move 350uL of Distilled-Water to Oxycodone-Enzyme
1954
            Vortex Oxycodone-Enzyme for 45s
1955
1956
            Drain Oxycodone-Enzyme
1957
          }
1959
         Move 100uL of TMB-Substrate to Oxycodone-Enzyme
1960
```

```
Incubate Oxycodone-Enzyme at 23C for 30min
          Mix Oxycodone-Enzyme with 100uL of Stop Reagent5 for 60s
1962
1963
          Urine Reading = Measure the fluorescence of Oxycodone-Enzyme for 30min
1964
          Drain Oxycodone-Enzyme
       }
     }
1968
     7.3 PCR Droplet Replenishment
1969
1970
     PCR Droplet Replenishment Assay: https://ldrv.ms/v/s!AqwZF7jQGx IhJRqfq1dLhbBUIL9RQ
1971
       BioScript Code:
1972
     PCRMix = Vortex 50 uL of PCRMasterMix with 50 uL of Template for 1s
1973
1974
     Repeat 50 times {
1975
       Heat PCRMix at 95C for 20s
1976
       volumeWeight = Weigh PCRMix
1977
1978
        if (volumeWeight <= 50uL) {
1979
          replacement = Vortex 25uL of PCRMasterMix with 25uL of Template for 5s
1980
          Heat replacement at 95C for 45s
1981
          PCRMix = Mix PCRMix with replacement for 5s
1982
       }
1983
1984
       Heat PCRMix at 68C for 30s
1985
       Heat PCRMix at 95C for 45s
1986
1987
1988
     Heat PCRMix at 68C for 5min
1989
     Save PCRMix
1990
1991
     7.4 Probabilistic PCR
1992
     Probabilistic PCR Full: https://1drv.ms/v/s!AqwZF7jQGx_IhJRo-xDbVT_Q6UsXPg
1993
     Probabilistic PCR Early Exit: https://ldrv.ms/v/s!AqwZF7jQGx_IhJRpLXveGI8Qr7BWrw
1994
     PCR-Master-Mix = Mix 50uL of PCRMix with 50uL of Buffer
1995
1996
     # Initialization Step
1997
     Heat PCR-Master-Mix at 94C for 2min
1998
1999
     # Denaturation
2000
2001
     Repeat 20 times {
       Heat PCR-Master-Mix at 94C for 20s
2002
       Heat PCR-Master-Mix at 50C for 40s
2003
2004
     DNA Sensor = Measure the fluorescence of PCR-Master-Mix for 30s
2005
     if (DNA Sensor <= 85) {
2006
       Drain PCR-Master-Mix
2007
2008
     }
2009
```

:42 Anon.

```
2010
     Repeat 20 times {
2011
2012
       Heat PCR-Master-Mix at 94C for 20s
       Heat PCR-Master-Mix at 50C for 40s
2013
2014
2015
     # Final Elongation
     Heat PCR-Master-Mix at 70C for 5min
2016
2017
2018
     Save PCR-Master-Mix
2019
     7.5 ChemType Test1
2020
2021
     Allyl Ethyl Ether = 50uL of C=CCOC1=CC=CC=C1
2022
2023
     Cycloheptatriene = 50uL of C1C=CC=CC=C1
2024
2025
     Cycloheptatriene2 = 50uL of C1C=CC=CC=C1
2026
2027
     Sulfuric Acid = 50uL of OS(=0)(=0)0
2028
2029
     Chlorotrifluoromethane = 50uL of C(F)(F)(F)Cl
2030
2031
     Ammonium dichromate = 50uL of [NH4+].[NH4+].[0-][Cr](=0)(=0)0[Cr](=0)(=0)[0-]
2032
2033
2034
     a = Mix Allyl Ethyl Ether with Cycloheptatriene for 50s
2035
     b = Vortex a with Sulfuric Acid for 30s
     c = Vortex b with Chlorotrifluoromethane for 45s
2036
     d = Mix Cycloheptatriene2 with Ammonium dichromate
2037
2038
     7.6 ChemType Test2
2039
2040
     Isopropyl-Alcohol = 50uL of CC(C)0
2041
2042
     Isopropyl-Alcohol2 = 50uL of CC(C)0
2043
     Allyl-Ethyl-Ether = 50uL of C=CCOC1=CC=CC=C1
2044
2045
2046
     Cycloheptatriene = 50uL of C1C=CC=CC=C1
2047
     Cycloheptatriene2 = 50uL of C1C=CC=CC=C1
2048
2049
2050
     Cycloheptatriene3 = 50uL of C1C=CC=CC=C1
2051
2052
     Sufluric-Acid = 50uL of OS(=0)(=0)0
2053
2054
     Chlorotrifluoromethane = 50uL of C(F)(F)(F)Cl
2055
2056
     Ammonium-dichromate = 50uL of [NH4+].[NH4+].[0-][Cr](=0)(=0)0[Cr](=0)(=0)[0-]
2057
2058
```

```
2059
     a = Mix Cycloheptatriene with Isopropyl-Alcohol for 30s
2060
2061
     a2 = Mix Cycloheptatriene2 with Isopropyl-Alcohol2 for 30s
2062
2063
     b = Vortex Cycloheptatriene3 with Chlorotrifluoromethane
     c = Tap a with b
     d = Invert c with Allyl-Ethyl-Ether
     e = Mix a2 with Sufluric-Acid
     7.7 ChemType Test3
2068
     Sodium-Hydroxide = 50uL of [OH-].[Na+]
2069
2070
2071
     Isopropyl-Alcohol = 50uL of CC(C)0
2072
2073
     Allyl-Ethyl-Ether = 50uL of C=CCOC1=CC=CC=C1
2074
2075
     Cycloheptatriene = 50uL of C1C=CC=CC=C1
2076
     Cycloheptatriene2 = 50uL of C1C=CC=CC=C1
2077
2078
2079
     Sufluric-Acid = 50uL of OS(=0)(=0)0
2080
2081
     Chlorotrifluoromethane = 50uL of C(F)(F)(F)Cl
2082
2083
     Dibutyl-phosphite = 50uL of CCCCO[P+](=0)OCCCC
2084
2085
     Ammonium-dichromate = 50uL of [NH4+].[NH4+].[O-][Cr](=0)(=0)0[Cr](=0)(=0)[O-]
2086
2087
2088
     a = Mix Sodium-Hydroxide with Chlorotrifluoromethane
2089
     b = Mix a with Isopropyl-Alcohol
2090
     c = Mix Allyl-Ethyl-Ether with Cycloheptatriene
2091
     Mix Cycloheptatriene2 with Dibutyl-phosphite
2092
     7.8 Broad Spectrum Opiate
2093
     Move 10uL of UrineSample to Anti-Morphine
2094
     Move 10uL of UrineSample to Anti-Oxy
2095
     Move 10uL of UrineSample to Anti-Fentanyl
2096
2097
     Move 10uL of UrineSample to Anti-Ciprofloxcin
2098
     Move 10uL of UrineSample to Anti-Heroin
2099
     MorphineReading = Measure the fluorescence of Anti-Morphine for 5s
2100
     OxyReading = Measure the fluorescence of Anti-Oxy for 5s
2101
     FentanylReading = Measure the fluorescence of Anti-Fentanyl for 5s
2102
2103
     CiproReading = Measure the fluorescence of Anti-Ciprofloxcin for 5s
     HeroinReading = Measure the fluorescence of Anti-Heroin for 5s
2104
2105
2106
     Drain Anti-Morphine
```

:44 Anon.

```
2108
     Drain Anti-Oxy
     Drain Anti-Fentanyl
2109
2110
     Drain Anti-Ciprofloxcin
     Drain Anti-Heroin
2111
     7.9 Cipro ELISA
2113
2114
     Move 20uL of UrineSample to Ciprofloxacin-Enzyme
2115
     Move 100uL of Ciprofloxacin-Conjugate to Ciprofloxacin-Enzyme
     Tap Ciprofloxacin-Enzyme for 60s
2116
2117
     Incubate Ciprofloxacin-Enzyme at 23 C for 60min
     Drain Ciprofloxacin-Enzyme
2118
2119
2120
     Repeat 5 times {
2121
       Move 250uL of Distilled-Water to Ciprofloxacin-Enzyme
2122
       Vortex Ciprofloxacin-Enzyme for 45s
2123
       Drain Ciprofloxacin-Enzyme
2124
2125
2126
     Move 50uL of TMB-Substrate to Ciprofloxacin-Enzyme
2127
     Incubate Ciprofloxacin-Enzyme at 25 C for 30min
2128
     Mix Ciprofloxacin-Enzyme with 100uL of Stop Reagent for 60s
2129
2130
     Urine Reading = Measure the fluorescence of Ciprofloxacin-Enzyme for 5min
2131
     Drain Ciprofloxacin-Enzyme
2132
     7.10 Diazepam ELISA
2133
2134
     Move 50uL of UrineSample to Diazepam-Enzyme
     Move 100uL of Diazepam-Antibody to Diazepam-Enzyme
2135
2136
     Tap Diazepam-Enzyme for 60s
2137
     Incubate Diazepam-Enzyme at 23 C for 30min
     Drain Diazepam-Enzyme
2138
2139
2140
     Repeat 3 times {
       Move 250uL of Distilled-Water to Diazepam-Enzyme
2141
       Vortex Diazepam-Enzyme for 45s
2142
       Drain Diazepam-Enzyme
2143
     }
2144
2145
     Move 150uL of HRP-Conjugate to Diazepam-Enzyme
2146
     Incubate Diazepam-Enzyme at 23 C for 15min
2147
2148
     Drain Diazepam-Enzyme
2149
2150
     Repeat 3 times {
       Move 250uL of Distilled-Water to Diazepam-Enzyme
2151
2152
       Vortex Diazepam-Enzyme for 45s
       Drain Diazepam-Enzyme
2153
     }
2154
2155
2156
```

```
Move 100uL of TMB-Substrate to Diazepam-Enzyme
     Incubate Diazepam-Enzyme at 23 C for 15min
2158
2159
     Mix Diazepam-Enzyme with 100uL of Stop Reagent for 60s
     Negative Reading = Measure the fluorescence of Diazepam-Enzyme for 30min
2160
     Drain Diazepam-Enzyme
2161
2162
     7.11 547 Dilution
2163
2164
     First-Dilute = Mix Substance A with Dilutant1
2165
     Split First-Dilute into 2 parts:
       1 part to First-Dilute
2166
       1 part to Waste-Droplet
     Drain Waste-Droplet
2170
     Second-Dilute = Mix First-Dilute with Dilutant2
2171
     Split Second-Dilute into 2 parts:
2172
       1 part to Second-Dilute
2173
       1 part to Waste-Droplet
2174
     Drain Waste-Droplet
2175
     Third Dilute = Mix Second-Dilute with Dilutant3
2176
2177
     Split Third-Dilute into 2 parts:
2178
       1 part to Third-Dilute
2179
       1 part to Waste-Droplet
2180
     Drain Waste-Droplet
2181
     Fourth-Dilute = Mix Substance B with Substance C
2182
2183
     Split Fourth-Dilute into 2 parts:
2184
       1 part to Fourth-Dilute
2185
       1 part to Waste-Droplet
2186
     Drain Waste-Droplet
2187
     Final-Dilute = Mix Third-Dilute with Fourth-Dilute
2188
     Split Final-Dilute into 2 parts:
2189
2190
       1 part to Final-Dilute
       1 part to Waste-Droplet
2191
     Drain Waste-Droplet
2192
2193
     Save Final-Dilute
2194
2195
     7.12 Fentanyl ELISA
2196
2197
2198
     # Step 1 Add Urine-Sample to test against
2199
     Antigen1 = Move 20uL of Urine-Sample to Antigen
2200
2201
     # Step 2 Add Enzyme Conjugate
     Antigen2 = Move 100uL of Fentanyl-Conjugate to Antigen1
2202
2203
2204
     # Step 3 Ensure fully mixed interactions
2205
```

:46 Anon.

```
Tap Antigen2 for 60s
2206
2207
2208
     # Step 4
     Incubate Antigen2 at 23 C for 60min
2209
2210
     # Step 5
2211
2212
     Drain Antigen2
     Repeat 6 times {
2214
       Move 350uL of Distilled-Water to Antigen1
2215
2216
       Vortex Antigen1 for 45s
2217
2218
       Drain Antigen1
2219
     }
2220
     # Step 6
2221
     Antigen1 = Move 100uL of TMB-Substrate to Antigen1
2222
2223
     # Step 7
2224
     Incubate Antigen1 at 23 C for 30min
2225
2226
     # Step 8
2227
     Mix Antigen1 with 100uL of Stop-Reagent for 60s
2228
2229
     # Step 9
2230
     Negative-Reading = Measure the fluorescence of Antigen1 for 30min
2231
2232
     # End
2233
     Drain Antigen1
2234
2235
     7.13 Morphine ELISA
2236
     Stationary Morphine-Enzyme
2237
2238
     Move 20uL of Urine-Sample to Morphine-Enzyme
2239
     Move 100uL of Morphine-Conjugate to Morphine-Enzyme
2240
     Tap Morphine-Enzyme for 60s
2241
     Incubate Morphine-Enzyme at 23 C for 60min
2242
2243
     Repeat 6 times {
2244
       Move 350uL of Distilled-Water to Morphine-Enzyme
       Vortex Morphine-Enzyme for 45s
2246
     }
2247
2248
     Move 100uL of TMB-Substrate to Morphine-Enzyme
2249
     Incubate Morphine-Enzyme at 23 C for 30min
2250
     Mix Morphine-Enzyme with 100uL of Stop Reagent for 60s
2251
2252
     Urine Reading = Measure the fluorescence of Morphine-Enzyme for 30min
2253
2254
     , Vol. 1, No. 1, Article . Publication date: February 2017.
```

```
7.14 Morphine ELISA with Control Samples
2255
2256
     # Step 2
2257
     Move 20uL of Negative-Standard to Antigen1
2258
     Move 20uL of Positive-Standard to Antigen2
2259
     Move 20uL of Diluted-Sample to Antigen3
2260
     # Step 3
2262
     Move 100uL of Morphine-Conjugate to Antigen1
2263
     Move 100uL of Morphine-Conjugate to Antigen2
2264
     Move 100uL of Morphine-Conjugate to Antigen3
2265
2266
     # Step 4
2267
     Tap Antigen1 for 60s
2268
     Tap Antigen2 for 60s
2269
     Tap Antigen3 for 60s
2270
2271
     # Step 5
2272
     Incubate Antigen1 at 23 C for 60min
2273
     Incubate Antigen2 at 23 C for 60min
2274
     Incubate Antigen3 at 23 C for 60min
2275
2276
     # Step 6
2277
     Drain Antigen1
2278
     Drain Antigen2
2279
     Drain Antigen3
2280
     Repeat 6 times {
2281
       Move 350uL of Distilled-Water to Antigen1
2282
       Move 350uL of Distilled-Water to Antigen2
2283
       Move 350uL of Distilled-Water to Antigen3
2284
2285
       Vortex Antigen1 for 45s
2286
       Vortex Antigen2 for 45s
2287
       Vortex Antigen3 for 45s
2288
2289
       Drain Antigen1
2290
       Drain Antigen2
2291
       Drain Antigen3
2292
2293
2294
     # Step 7 Dry...
2295
2296
     # Step 8
2297
     Move 100uL of TMB-Substrate to Antigen1
2298
     Move 100uL of TMB-Substrate to Antigen2
2299
     Move 100uL of TMB-Substrate to Antigen3
2300
2301
     # Step 9
2302
```

:48 Anon.

```
Incubate Antigen1 at 23 C for 30min
2304
     Incubate Antigen2 at 23 C for 30min
2305
2306
     Incubate Antigen3 at 23 C for 30min
2307
     # Step 10
2308
     Mix Antigen1 with 100uL of Stop Reagent for 60s
2309
     Mix Antigen2 with 100uL of Stop Reagent for 60s
2310
2311
     Mix Antigen3 with 100uL of Stop Reagent for 60s
2312
2313
     # Step 11
2314
     Negative Reading = Measure the fluorescence of Antigen1 for 30min
2315
     Positive Reading = Measure the fluorescence of Antigen2 for 30min
     Sample Reading = Measure the fluorescence of Antigen3 for 30min
2317
2318
     # End
2319
     Drain Antigen1
2320
     Drain Antigen2
2321
     Drain Antigen3
2322
     7.15 Heroin ELISA
2323
2324
     Move 20uL of Urine-Sample to Heroin-Enzyme
     Move 100uL of Heroin-Conjugate to Heroin-Enzyme
2325
2326
     Tap Heroin-Enzyme for 60s
2327
     Incubate Heroin-Enzyme at 23 C for 60min
2328
     Drain Heroin-Enzyme
2329
2330
     Repeat 6 times {
2331
       Move 350uL of Distilled-Water to Heroin-Enzyme
2332
       Vortex Heroin-Enzyme for 45s
2333
       Drain Heroin-Enzyme
2334
     }
2335
     Move 100uL of TMB-Substrate to Heroin-Enzyme
2336
2337
     Incubate Heroin-Enzyme at 23 C for 30min
     Mix Heroin-Enzyme with 100uL of Stop Reagent for 60s
2338
2339
     Urine-Reading = Measure the fluorescence of Heroin-Enzyme for 30min
2340
     Drain Heroin-Enzyme
2341
2342
     7.16 Oxycodone ELISA
2343
2344
     Move 20uL of Urine-Sample to Oxycodone-Enzyme
     Move 100uL of Oxycodone-Conjugate to Oxycodone-Enzyme
2345
     Tap Oxycodone-Enzyme for 60s
2346
     Incubate Oxycodone-Enzyme at 23 C for 60min
2347
2348
     Drain Oxycodone-Enzyme
2349
2350
     Repeat 6 times {
2351
       Move 350uL of Distilled-Water to Oxycodone-Enzyme
2352
```

```
Vortex Oxycodone-Enzyme for 45s
2353
       Drain Oxycodone-Enzyme
2354
2355
     }
2356
     Move 100uL of TMB-Substrate to Oxycodone-Enzyme
2357
     Incubate Oxycodone-Enzyme at 23 C for 30min
2358
     Mix Oxycodone-Enzyme with 100uL of Stop Reagent for 60s
2360
     Urine Reading = Measure the fluorescence of Oxycodone-Enzyme for 30min
2361
     Drain Oxycodone-Enzyme
2362
     7.17 Image Probe Synthesis
2364
2365
     Mixture = Mix 10uL of Ion exchange beads with 10uL of Fluoride ions F- for 30s
2366
2367
     Heat Mixture at 100 C for 30s
2368
     Heat Mixture at 120 C for 30s
2369
     Heat Mixture at 135 C for 3min
2370
2371
     Mixture = Mix Mixture with 10uL of MeCN solution for 30s
2372
     Incubate Mixture at 100 C for 30s
2373
2374
     Incubate Mixture at 120 C for 50s
2375
2376
     Mixture = Mix Mixture with 10uL of HCl for 60s
     Heat Mixture at 60 C for 60s
2377
2378
     7.18 Glucose Detection
2379
     Result1 = Mix 10uL of Glucose with 10uL of Reagent for 10s
2380
2381
     Reading1 = Measure the fluorescence of Result1 for 30s
2382
2383
     Result2 = Mix 10uL of Glucose with 20uL of Reagent for 10s
     Reading2 = Measure the fluorescence of Result2 for 30s
2384
2385
     Result3 = Mix 10uL of Glucose with 40uL of Reagent for 10s
2386
     Reading3 = Measure the fluorescence of Result3 for 30s
2387
2388
     Result4 = Mix 10uL of Glucose with 80uL of Reagent for 10s
2389
     Reading4 = Measure the fluorescence of Result4 for 30s
2390
2391
2392
     Result5 = Mix 10uL of Sample with 10uL of Reagent for 10s
2393
     Reading5 = Measure the fluorescence of Result5 for 30s
2394
     7.19 PCR
2395
2396
     PCR-Mixture = 10uL of PCR
2397
     Heat PCR-Mixture at 95 C for 5s
2398
2399
2400
     Repeat 20 times {
2401
```

:50 Anon.

```
Heat PCR-Mixture at 53 C for 15s
       Heat PCR-Mixture at 72 C for 10s
2403
2404
     }
     Perform Capillary Electrophoresis ( 5 cm at 236 V/cm) on PCR-Mixture Separate with Separati
2405
2406
     Measure the fluorescence of PCR-Mixture for 3min
2407
2408
2409
     7.20 Neurotransmitter Sensing
2410
     Mixture = Mix Sample with Reagent for 50s
2411
     Perform Capillary Electrophoresis (9 cm at 223 V/cm) on Mixture Seperate with electrophore
2412
     Measure the fluorescence of Mixture for 10s
2413
2414
     8 AQUACORE ASSAYS
2415
     8.1 Glucose Detection
2416
     Input port ip1 ;Standard glucose
2417
     Input port ip2 ;Reagent
2418
     Input port ip3 ;Sample
2419
     RESULT[1..5]; dry array for final results
2420
     glucose-detection {
2421
     input s1, ip1
2422
     input s2, ip2
2423
     input s3, ip3
2424
     move mixer1, s1, 1;5s
2425
     move mixer1, s2, 1;5s
2426
     mix mixer1, 10;10s
2427
     move sensor1, mixer1;5s
2428
     sense.OD sensor1, RESULT[1];30s
2429
2430
     move mixer1, s1, 1;5s
2431
     move mixer1, s2, 2;5s
2432
     mix mixer1, 10;10s
2433
     move sensor1, mixer1;5s
2434
     sense.OD sensor1, RESULT[2];30s
2435
2436
     move mixer1, s1, 1;5s
2437
     move mixer1, s2, 4;5s
2438
     mix mixer1, 10;10s
2439
     move sensor1, mixer1;5s
2440
     sense.OD sensor1, RESULT[3] ;30s
2441
2442
     move mixer1, s1, 1;5s
2443
     move mixer1, s2, 8;5s
2444
     mix mixer1, 10;10s
2445
     move sensor1, mixer1;5s
2446
     sense.OD sensor1, RESULT[4];30s
2447
2448
     <dry routine to get best line fit for RESULT[1..4]> move mixer1, s3, 1 ;5s
2449
2450
     , Vol. 1, No. 1, Article . Publication date: February 2017.
```

```
move mixer1, s2, 1;5s
2451
     mix mixer1, 10;10s
2452
2453
     move sensor1, mixer1;5s
     sense.OD sensor1, RESULT[5];30s
2454
2455
     <dry routine to get concentration from line given RESULT[5]>
2456
2457
2458
     Total time = 275s
2459
     \#reservoirs = 3
     ASLoC
2460
2461
     8.2 PCR
2462
2463
     Input port ip1 ;PCR mixture
2464
     Input port ip2 ;CE separation medium
2465
2466
     RESULT[]; dry array for final results
2467
2468
     PCR {
2469
     input s1, ip1
2470
     input s2, ip2
2471
2472
     move heater1, s1;5s
2473
     dry-mov r1, 20
2474
     dry-label loop:
     incubate heater1, 95, 5;6s
2475
2476
     incubate heater1, 53, 15;17s
     incubate heater1, 72, 10;12s
2477
     dry-dec r1
2478
2479
     dry-bgt loop
2480
2481
     move separator1.buf, s2;5s
2482
     move separator1, heater1;5s
2483
2484
     separate.CE separator1, 236, 5, 180
     sense.FL sensor1, RESULT; 180s
2485
2486
     Total time = 895s
2487
     \#reservoirs = 2
2488
2489
     ASLoC
2490
     8.3 Imaging Probe Synthesis
2491
     Input port ip1 ;Ion exchange beads (in buffer)
2492
     Input port ip2 ;Fluoride ions F-
2493
     Input port ip3 ;MeCN solution
2494
2495
     Input port ip4 ;HCl
2496
2497
     Imaging-Probe-Synthesis {
2498
     input s1, ip1
```

:52 Anon.

```
2500
     input s2, ip2
2501
     input s3, ip3
2502
     input s4, ip4
2503
2504
     move mixer1, s1
2505
     move mixer1, s2
     mix mixer1, 30
2507
2508
     move heater1, mixer1
2509
     concentrate.EV heater1, 100, 30
2510
     concentrate.EV heater1, 120, 30
2511
     concentrate.EV heater1, 135, 180
2512
2513
     move mixer1, s3
2514
     move mixer1, heater1
2515
     mix mixer1, 30
2516
     move heater1, mixer1
2517
2518
     incubate heater1, 100, 30
2519
     incubate heater1, 120, 50
2520
     move mixer1, heater1
2521
2522
     move mixer1, s4
2523
     mix mixer1, 60
2524
     move heater1, mixer1
2525
     concentrate.EV heater1, 60, 60
2526
2527
     Total time = 548s #reservoirs = 4
2528
     ASLoC area = 13mm \times 9mm = 117 mm2
2529
     8.4 Neurotransmitter Sensing
2530
     Input port ip1 ;Sample
2531
     Input port ip2 ;Reagent(OPA)
2532
2533
     Input port ip3 ;Elecrtrophoresis buffer
2534
     RESULT[]; dry array for final results
2535
2536
     neurotransmitter-sensing {
2537
     input s1, ip1
2538
2539
     input s2, ip2
     input s3, ip3
2540
2541
     move mixer1, s1;5s
2542
     move mixer1, s2;5s
2543
2544
     mix mixer1, 50 ;50s
2545
2546
     move separator1.buf, s3;5s
2547
     move separator1, mixer1 ;5s
2548
```

, Vol. 1, No. 1, Article . Publication date: February 2017.

```
2550
     separate.CE separator1, 223, 9, 22
2551
     sense.FL sensor1, RESULT;22s
2552
2553
     }
     Total time = 92s #reservoirs = 3
2554
     ASLoC area = 2.5 \text{cm} \times 1.5 \text{cm} = 3.75 \text{cm} + CE \text{ column}
     FIGURE 3: Separations-based sensing of neurotransmitters (immunoassay, biochemistry): (a)
2558
     9 BIOCODER ASSAYS
     9.1 PCR
     void PCR(){
2562
     BioSystem bioCoder;
2563
2564
     Fluid *PCRMix = bioCoder.new_fluid("PCRMasterMix", Volume(MICRO_LITER,10));
2565
     Fluid *SeperationMedium = bioCoder.new_fluid("Seperation Medium", Volume(MICRO_LITER, 10));
2566
     Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2567
2568
     bioCoder.first_step();
     bioCoder.measure_fluid(PCRMix, tube);
2570
2571
     bioCoder.next_step();
2572
     bioCoder.incubate(tube,95,Time(5,SECS));
2573
2574
     bioCoder.next_step();
2575
     bioCoder.LOOP(20);
2576
2577
     bioCoder.next_step();
2578
     bioCoder.incubate(tube,53,Time(15,SECS));
2579
2580
     bioCoder.next_step();
2581
     bioCoder.incubate(tube,72,Time(10,SECS));
2582
     bioCoder.END_LOOP();
2583
2584
     bioCoder.next_step();
2585
     bioCoder.ce_detect(tube, 5, 236, SeperationMedium);
2586
2587
     bioCoder.next_step();
2588
     bioCoder.measure_fluorescence(tube,Time(3,MINS));
2589
2590
     bioCoder.next_step();
2591
     bioCoder.end_protocol();
2592
     }
2593
2594
     9.2 Probabilistic PCR
2595
     void ProbablisticPCR()
2596
```

:54 Anon.

```
2598
2599
     BioSystem bioCoder;
2600
     Fluid *PCRMix = bioCoder.new_fluid("PCRMasterMix", Volume(MICRO_LITER,10));
2601
2602
     Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2603
2604
2605
     bioCoder.first_step();
     bioCoder.measure_fluid(PCRMix, tube);
2606
2607
     for(int i = 0; i < initial; ++i) {
2608
2609
     bioCoder.next_step();
     bioCoder.store_for(tube,94,Time(SECS,45));
2611
2612
     bioCoder.next_step();
2613
     bioCoder.store_for(tube,65,Time(SECS,45));
2614
2615
2616
     for(int i = initial; i <= Threshold; ++i) {</pre>
     std::cout <<i<<std::endl;</pre>
2617
2618
     bioCoder.next_step();
2619
     bioCoder.store_for(tube,94,Time(SECS,45));
2620
2621
     bioCoder.next_step();
2622
     bioCoder.store_for(tube,65,Time(SECS,45));
2623
2624
     bioCoder.next_step();
2625
     bioCoder.measure_fluorescence(tube,Time(SECS,5),"DNASensor");
2626
     bioCoder.IF("DNASensor", GREATER_THAN, .85);
2627
     for(int j = i; j < Total+(Threshold-i); ++j) {</pre>
2628
2629
     bioCoder.next_step();
2630
     bioCoder.store_for(tube, 94, Time(SECS, 45));
2631
2632
     bioCoder.next_step();
     bioCoder.store_for(tube,65,Time(SECS,45));
2633
2634
2635
2636
     bioCoder.next_step();
     bioCoder.drain(tube, "Amplified PCR");
2637
2638
     bioCoder.END_IF();
2639
2640
     }
2641
     bioCoder.drain(tube, "waste");
2642
     bioCoder.end_protocol();
2643
2644
     bioCoder.PrintLeveledProtocol();
2645
2646
```

, Vol. 1, No. 1, Article . Publication date: February 2017.

```
2647
     bioCoder.PrintTree();
     bioCoder.PrintTreeVisualization("ProbablisticPCR");
2648
2649
2650
     9.3 PCR Droplet Replenish
     void PCRDropletReplacement()
2652
     int TotalThermo = 9;
     BioSystem bioCoder;
2656
     Fluid *PCRMix = bioCoder.new_fluid("PCRMasterMix", Volume(MICRO_LITER,10));
     Fluid *Template = bioCoder.new_fluid("Template", Volume(MICRO_LITER,10));
     Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2660
     //Container* tube2 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2662
     bioCoder.first_step();
2663
2664
     bioCoder.measure_fluid(PCRMix, tube);
     bioCoder.next_step();
2666
     bioCoder.vortex(tube, Time(SECS, 1));
2668
     bioCoder.measure_fluid(Template, tube);
2669
2670
     bioCoder.next_step();
2671
     bioCoder.vortex(tube, Time(SECS,1));
2672
2673
     bioCoder.next_step();
     bioCoder.store_for(tube,95,Time(SECS,45));
2674
2675
2676
     bioCoder.next_step();
2677
     bioCoder.LOOP(TotalThermo);
2678
2679
     std::cout<<"Debug statement2"<<std::endl;</pre>
2680
     bioCoder.next_step();
     bioCoder.store_for(tube,95,Time(SECS,20));
2681
2682
2683
     bioCoder.next_step();
     bioCoder.weigh(tube, "weightSensor");
2684
2685
2686
     bioCoder.next_step();
     bioCoder.IF("WieghtSensor", LESS_THAN, 3.57);
2687
2688
     bioCoder.next_step();
2689
     bioCoder.measure_fluid(PCRMix, tube);
2690
2691
     bioCoder.next_step();
     bioCoder.store_for(tube, 95,Time(SECS,45));
2692
2693
2694
     bioCoder.next_step();
2695
```

:56 Anon.

```
bioCoder.vortex(tube, Time(SECS,1));
2696
2697
     bioCoder.END_IF();
2698
2699
     bioCoder.next_step();
2700
     bioCoder.store_for(tube,50,Time(SECS,30));
2701
2702
     bioCoder.next_step();
2703
     bioCoder.store_for(tube,68,Time(SECS,45));
     std::cout<<"Debug statement3"<<std::endl;</pre>
2704
     bioCoder.END_LOOP();
2705
     std::cout<<"Debug statement4"<<std::endl;</pre>
2707
2709
     bioCoder.next_step();
     bioCoder.store_for(tube,68,Time(MINS,5));
2710
     std::cout<<"Debug statement5"<<std::endl;</pre>
2711
     bioCoder.next_step();
2712
2713
     bioCoder.drain(tube, "PCR");
     std::cout<<"Debug statement6"<<std::endl;</pre>
2714
2715
     bioCoder.end_protocol();
2716
2717
2718
     std::cout<<"Debug statemen7"<<std::endl;</pre>
2719
     bioCoder.PrintLeveledProtocol();
2720
     bioCoder.PrintTree();
2721
     bioCoder.PrintTreeVisualization("PCRReplacement");
2722
2723
     }
2724
     9.4 Glucose Detection
2725
     void GlucoseDetection(){
2726
2727
     BioSystem bioCoder;
2728
     Fluid *Glucose = bioCoder.new_fluid("Ion exchange beads", Volume(MICRO_LITER,160));
2729
     Fluid *Reagent = bioCoder.new_fluid("Fluoride ions", Volume(MICRO_LITER, 50));
2730
     Fluid *Sample = bioCoder.new_fluid("HCL", Volume(MICRO_LITER, 10));
2731
2732
     Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2733
2734
     Container* tube2 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2735
     Container* tube3 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2736
     Container* tube4 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2737
     Container* tube5 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2738
2739
     bioCoder.first_step();
2740
     bioCoder.measure_fluid(Glucose, Volume(MICRO_LITER, 10), tube);
     bioCoder.measure_fluid(Glucose, Volume(MICRO_LITER, 10), tube2);
2741
     bioCoder.measure_fluid(Glucose, Volume(MICRO_LITER, 10), tube3);
2742
     bioCoder.measure_fluid(Glucose, Volume(MICRO_LITER, 10), tube4);
2743
2744
```

```
bioCoder.measure_fluid(Glucose, Volume(MICRO_LITER, 10), tube5);
2745
2746
2747
     bioCoder.measure_fluid(Reagent, Volume(MICRO_LITER, 10), tube);
     bioCoder.measure_fluid(Reagent, Volume(MICRO_LITER, 20), tube2);
2748
     bioCoder.measure_fluid(Reagent, Volume(MICRO_LITER, 40), tube3);
2749
     bioCoder.measure_fluid(Reagent, Volume(MICRO_LITER, 80), tube4);
2752
     bioCoder.measure_fluid(Sample, Volume(MICRO_LITER, 80), tube5);
2753
     bioCoder.next_step();
2754
     bioCoder.measure_fluorescence(tube,Time(5,SECS));
     bioCoder.measure_fluorescence(tube2,Time(5,SECS));
     bioCoder.measure_fluorescence(tube3, Time(5, SECS));
     bioCoder.measure_fluorescence(tube4,Time(5,SECS));
2758
     bioCoder.measure_fluorescence(tube5,Time(5,SECS));
2759
2760
     bioCoder.next_step();
2761
2762
     bioCoder.end_protocol();
2763
2764
         Image Probe Synthesis
2765
2766
     void ImageProbSynthesis(){
2767
     BioSystem bioCoder;
2768
2769
     Fluid *IonBeads = bioCoder.new_fluid("Ion exchange beads", Volume(MICRO_LITER,10));
     Fluid *Fluoride = bioCoder.new_fluid("Fluoride ions", Volume(MICRO_LITER, 10));
2770
     Fluid *HCL = bioCoder.new_fluid("HCL", Volume(MICRO_LITER, 10));
2771
2772
     Fluid *MeCNSolution = bioCoder.new_fluid("MeCN solution", Volume(MICRO_LITER, 10));
2773
2774
     Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2775
2776
     bioCoder.first_step();
2777
     bioCoder.measure_fluid(IonBeads, tube);
2778
     bioCoder.measure_fluid(Fluoride, tube);
2779
2780
     bioCoder.next_step();
     bioCoder.vortex(tube,Time(30,SECS));
2781
2782
2783
     bioCoder.next_step();
2784
     bioCoder.incubate(tube,100,Time(30,SECS));
2785
2786
     bioCoder.next_step();
     bioCoder.incubate(tube, 120, Time(30, SECS));
2787
2788
2789
     bioCoder.next_step();
     bioCoder.incubate(tube, 135, Time(3, MINS));
2790
2791
2792
     bioCoder.next_step();
2793
```

:58 Anon.

```
bioCoder.measure_fluid(MeCNSolution, tube);
2795
2796
     bioCoder.next_step();
2797
     bioCoder.vortex(tube,Time(30,SECS));
2798
2799
     bioCoder.next_step();
     bioCoder.incubate(tube, 100, Time(30, SECS));
2802
     bioCoder.next_step();
     bioCoder.incubate(tube, 120, Time(50, SECS));
2803
2804
2805
     bioCoder.next_step();
     bioCoder.measure_fluid(HCL, tube);
2807
2808
     bioCoder.next_step();
2809
     bioCoder.vortex(tube,Time(60,SECS));
2810
2811
     bioCoder.next_step();
2812
     bioCoder.incubate(tube,60,Time(60,SECS));
2813
     bioCoder.next_step();
2814
2815
     bioCoder.end_protocol();
2816
2817
     9.6 Neurotransmitter Sensing
2818
2819
     void neurotransmitterSensing(){
2820
     BioSystem bioCoder;
2821
     Fluid *Sample = bioCoder.new_fluid("Sample", Volume(MICRO_LITER,10));
2822
     Fluid *Reagent = bioCoder.new_fluid("Reagent", Volume(MICRO_LITER, 10));
2823
     Fluid *SeperationMedium = bioCoder.new_fluid("Seperation Medium", Volume(MICRO_LITER, 10));
2824
2825
2826
     Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
2827
     bioCoder.first_step();
2828
     bioCoder.measure_fluid(Sample, tube);
2829
     bioCoder.measure_fluid(Reagent, tube);
2830
2831
2832
     bioCoder.next_step();
2833
     bioCoder.vortex(tube,Time(50,SECS));
2834
2835
     bioCoder.next_step();
     bioCoder.ce_detect(tube,9,223,SeperationMedium);
2836
2837
2838
     bioCoder.next_step();
2839
     bioCoder.measure_fluorescence(tube, Time(10, SECS));
2840
2841
     bioCoder.next_step();
2842
     , Vol. 1, No. 1, Article . Publication date: February 2017.
```

```
bioCoder.end_protocol();
2843
2844
     }
2845
2846
     10 ANTHA ASSAYS
     10.1 Glucose Detection
2848
     protocol Glucose_Detection
2849
2850
2851
     import (
2852
     "github.com/antha-lang/antha/anthalib/wtype"
2853
          // LHComponent type
2854
     "github.com/antha-lang/antha/antha/anthalib/wutil"
2855
      "github.com/antha-lang/antha/antha/anthalib/mixer"
2856
          //sample function is imported from mixed
2857
2858
     // Input parameters
2859
     Parameters (
2860
     Reagent_volume1 Volume // 10ul
2861
     Reagent_volume2 Volume // 20ul
2862
     Reagent_volume3 Volume
                              // 40ul
2863
     Reagent_volume4 Volume // 80ul
2864
     Glucose_volume Volume
                                // 10ul
2865
     Sample_volume Volume
                                // 10ul
2866
     )
2867
2868
2869
     // Data which is returned from this protocol, and data types
2870
     Data (
2871
2872
     )
2873
2874
     // Physical Inputs to this protocol with types
2875
     Inputs (
2876
     Glucose *wtype.LHComponent
2877
     Reagent *wtype.LHComponent
2878
     Sample *wtype.LHComponent
2879
     )
2880
2881
2882
2883
     // Physical outputs from this protocol with types
2884
     Outputs (
2885
     Result1 *wtype.LHComponent
2886
     Result2 *wtype.LHComponent
2887
     Result3 *wtype.LHComponent
2888
     Result4 *wtype.LHComponent
2889
     Result5 *wtype.LHComponent
2891
```

:60 Anon.

```
2892
     )
2893
     Requirements {
2895
2896
     }
2897
2898
     // Conditions to run on startup
2899
     Setup {
2900
2901
     }
2902
2903
     // The core process for this protocol, with the steps to be performed
2904
     // for every input
2905
     Steps {
2906
2907
     glucose := mixer.Sample(Glucose, Glucose_volume)
2908
     reagent := mixer.Sample(Reagent, Reagent_volume1)
2909
2910
     Result1 = mixer.Mix(glucose, reagent) // cannot specify duration of mixture
2911
     //cannot measure fluorescence to get a reading
2912
2913
     glucose := mixer.Sample(Glucose, Glucose_volume) //get new sample of Glucose
     reagent := mixer.Sample(Reagent, Reagent_volume2) //get new sample of Reagent
2914
2915
2916
     Result2 = mixer.Mix(glucose, reagent) // mix new samples
     // would measure fluorescence here
2917
2918
2919
     glucose := mixer.Sample(Glucose, Glucose_volume) //get new sample of Glucose
2920
     reagent := mixer.Sample(Reagent, Reagent_volume3) //get new sample of Reagent
2921
     Result3 = mixer.Mix(glucose, reagent) // mix new samples
2922
     // would measure fluorescence here
2923
2924
2925
     glucose := mixer.Sample(Glucose, Glucose_volume) //get new sample of Glucose
     reagent := mixer.Sample(Reagent, Reagent_volume4) //get new sample of Reagent
2926
2927
     Result4 = mixer.Mix(glucose, reagent) // mix new samples
2928
     // would measure fluorescence here
2929
2930
2931
     sample := mixer.Sample(Sample, Sample_volume) //get new sample of Sample
     reagent := mixer.Sample(Reagent, Reagent_volume1) //get new sample of Reagent
2932
2933
     Result5 = mixer.Mix(sample, reagent) // mix new samples
2934
     // would measure fluorescence here
2935
2936
2937
     }
2938
2939
     Analysis {
2940
```

, Vol. 1, No. 1, Article . Publication date: February 2017.

```
}
2942
2943
     Validation {
2944
2945
     10.2 Imaging Probe Synthesis
2947
2948
     protocol Imaging_Probe_Synthesis
2949
2950
2951
     import (
2952
     "github.com/antha-lang/antha/antha/anthalib/wtype" // LHComponent type
2953
     "github.com/antha-lang/antha/antha/anthalib/wutil"
2954
     "github.com/antha-lang/antha/anthalib/mixer"
2955
          //sample function is imported from mixed
2956
2957
     // Input parameters
2958
     Parameters (
2959
     Ion_volume Volume
     Flouride_volume Volume
2960
     MeCN_volume Volume
2961
     HCl_volume Volume
2962
2963
2964
2965
     // Data which is returned from this protocol, and data types
2966
     Data (
2967
2968
2969
     )
2970
2971
     // Physical Inputs to this protocol with types
     Inputs (
2972
2973
     Ion_exchange_beads *wtype.LHComponent
     Fluoride_ions_F- *wtype.LHComponent
2974
2975
     MeCN *wtype.LHComponent
     HCl *wtype.LHComponent
2976
     )
2977
2978
2979
2980
     // Physical outputs from this protocol with types
2981
     Outputs (
     Mixture *wtype.LHComponent
2982
2983
     )
2984
2985
     Requirements {
2986
     }
2987
2988
     // Conditions to run on startup
2989
```

:62 Anon.

```
Setup {
2990
2991
2992
     }
2993
     // The core process for this protocol, with the steps to be performed
2994
     // for every input
2995
     Steps {
2996
2997
2998
     ibv := mixer.Sample(Ion_exchange_beads, Ion_volume)
2999
     fif := mixer.Sample(Flourid_ions_F-, Flouride_volume)
3000
3001
     mixture := mixer.Mix(ibv, fif) //cannot specify time for mixture
3002
3003
     mixture = Incubate(mixture, 100, 30, false) //heat 100 for 30s
     mixture = Incubate(mixture, 120, 30, false) //heat 120 for 30s
3004
3005
     mixture = Incubate(mixture, 135, 180, false) //heat 135 for 3 minutes
3006
3007
     mecn := mixer.Sample(MeCN, MeCN_volume)
3008
3009
     mixture = mixer.Mix(mixture, mecn)
3010
3011
     mixture = Incubate(mixture, 100, 30, false) //incubate 100 for 30s
     mixture = Incubate(mixture, 120, 50, false) //incubuate 120 for 50s
3012
3013
3014
     hcl := mixer.Sample(HCl, HCl_volume)
3015
3016
     mixture = mixer.Mix(mixture, hcl)
3017
3018
     mixture = Incubate(mixture, 60, 60, false) // heat 60 for 60s
3019
3020
     Mixture = mixture
3021
3022
3023
     Analysis {
3024
     }
3025
3026
     Validation {
3027
3028
3029
     10.3 PCR
3030
     protocol PCR
3031
3032
3033
     import (
3034
     "github.com/antha-lang/antha/anthalib/wtype"
3035
     "github.com/antha-lang/antha/antha/anthalib/mixer"
3036
     "github.com/antha-lang/antha/antha/AnthaStandardLibrary/Packages/enzymes"
     "fmt"
3037
3038
```

```
"github.com/antha-lang/antha/antha/AnthaStandardLibrary/Packages/text"
     "github.com/antha-lang/antha/antha/AnthaStandardLibrary/Packages/search"
     "github.com/antha-lang/antha/antha/AnthaStandardLibrary/Packages/sequences"
3042
     /*type Polymerase struct {
3044
     wtype.LHComponent
     Rate_BPpers float64
     Fidelity_errorrate float64 // could dictate how many colonies are checked in validation!
3047
     Extensiontemp Temperature
3048
     Hotstart bool
3049
3050
     StockConcentration Concentration // this is normally in U?
3051
     TargetConcentration Concentration
3052
     // this is also a glycerol solution rather than a watersolution!
3053
3054
     */
3055
3056
3057
     // Input parameters for this protocol (data)
3058
     Parameters (
3059
     // PCRprep parameters:
3060
     ReactionVolume Volume
3061
     FwdPrimerConc Concentration
3062
     RevPrimerConc Concentration
3063
     Additiveconc Concentration
     TargetpolymeraseConcentration Concentration
3064
3065
     Templatevolume Volume
     DNTPconc Concentration
3066
3067
     /*
3068
     // let's be ambitious and try this as part of type polymerase Polymeraseconc Volume
3069
3070
     //Templatetype string // e.g. colony, genomic, pure plasmid...
          //will effect efficiency. We could get more sophisticated here later on...
3071
3072
     //FullTemplatesequence string
         // better to use Sid's type system here after proof of concept
3073
     //FullTemplatelength int
3074
         // clearly could be calculated from the sequence...
3075
          //Sid will have a method to do this already so check!
3076
3077
     //TargetTemplatesequence string
3078
          // better to use Sid's type system here after proof of concept
3079
     //TargetTemplatelengthinBP int
     */
3080
     // Reaction parameters: (could be a entered as thermocycle parameters type possibly?)
3081
     Numberofcycles int
3082
3083
     InitDenaturationtime Time
     Denaturationtime Time
3084
3085
     //Denaturationtemp Temperature
3086
     Annealingtime Time
3087
```

:64 Anon.

```
AnnealingTemp Temperature // Should be calculated from primer and template binding
3088
     Extensiontime Time // should be calculated from template length and polymerase rate
3089
3090
     Extensiontemp Temperature
     Finalextensiontime Time
3091
3092
     Targetsequence string
3093
     FwdPrimerSeq string
3094
     RevPrimerSeq string
3095
     )
3096
3097
     // Data which is returned from this protocol, and data types
     Data (
3098
3099
     FwdPrimerSites []search.Thingfound
3100
     RevPrimerSites []search.Thingfound
3101
     )
3102
3103
3104
     // Physical Inputs to this protocol with types
3105
     Inputs (
3106
     FwdPrimer *wtype.LHComponent
3107
     RevPrimer *wtype.LHComponent
3108
     DNTPS *wtype.LHComponent
3109
     PCRPolymerase *wtype.LHComponent
3110
     Buffer *wtype.LHComponent
3111
     Template *wtype.LHComponent
3112
     Additives []*wtype.LHComponent // e.g. DMSO
3113
     OutPlate *wtype.LHPlate
3114
3115
3116
     // Physical outputs from this protocol with types
3117
     Outputs (
3118
     Reaction *wtype.LHComponent
3119
     )
3120
3121
     Requirements {
3122
3123
     // Conditions to run on startup
3124
     Setup {
3125
3126
     }
3127
     // The core process for this protocol, with the steps to be performed
3128
     // for every input
3129
     Steps {
3130
3131
3132
     FwdPrimerSites = sequences.FindSeqsinSeqs(Targetsequence, []string{FwdPrimerSeq})
3133
3134
     RevPrimerSites = sequences.FindSeqsinSeqs(Targetsequence, []string{RevPrimerSeq})
3135
3136
```

```
3137
     if len(FwdPrimerSites)==0 || len(RevPrimerSites)==0{
3138
3139
     errordescription := fmt.Sprint(
3140
     text.Print("FwdPrimerSitesfound:", fmt.Sprint(FwdPrimerSites)),
3141
     text.Print("RevPrimerSitesfound:", fmt.Sprint(RevPrimerSites)),
     Errorf(errordescription)
3149
     // Mix components
3150
     samples := make([]*wtype.LHComponent, 0)
3151
     bufferSample := mixer.SampleForTotalVolume(Buffer, ReactionVolume)
3152
     samples = append(samples, bufferSample)
3153
3154
     templateSample := mixer.Sample(Template, Templatevolume)
     samples = append(samples, templateSample)
3155
     dntpSample := mixer.SampleForConcentration(DNTPS, DNTPconc)
3156
     samples = append(samples, dntpSample)
3157
3158
     FwdPrimerSample := mixer.SampleForConcentration(FwdPrimer, FwdPrimerConc)
3159
     samples = append(samples, FwdPrimerSample)
3160
     RevPrimerSample := mixer.SampleForConcentration(RevPrimer, RevPrimerConc)
3161
     samples = append(samples, RevPrimerSample)
3162
3163
     for _, additive := range Additives {
3164
     additiveSample := mixer.SampleForConcentration(additive, Additiveconc)
3165
     samples = append(samples, additiveSample)
3166
     }
3167
     polySample := mixer.SampleForConcentration(PCRPolymerase, TargetpolymeraseConcentration)
3168
3169
     samples = append(samples, polySample)
3170
     reaction := MixInto(OutPlate, "", samples...)
3171
3172
     // thermocycle parameters called from enzyme lookup:
3173
     polymerase := PCRPolymerase.CName
3174
3175
     extensionTemp := enzymes.DNApolymerasetemps[polymerase]["extensiontemp"]
3176
3177
     meltingTemp := enzymes.DNApolymerasetemps[polymerase]["meltingtemp"]
3178
3179
     // initial Denaturation
3180
3181
     r1 := Incubate(reaction, meltingTemp, InitDenaturationtime, false)
3182
3183
     for i:=0; i < Numberofcycles; i++ {</pre>
3184
3185
```

:66 Anon.

```
3186
     // Denature
3187
3188
     r1 = Incubate(r1, meltingTemp, Denaturationtime, false)
3189
3190
     // Anneal
     r1 = Incubate(r1, AnnealingTemp, Annealingtime, false)
3192
3193
3194
     //extensiontime := TargetTemplatelengthinBP/PCRPolymerase.RateBPpers
          // we'll get type issues here so leave it out for now
3195
3197
     // Extend
     r1 = Incubate(r1, extensionTemp, Extensiontime, false)
3199
3200
3201
     // Final Extension
3202
     r1 = Incubate(r1, extensionTemp, Finalextensiontime, false)
3203
3204
3205
     // all done
3206
     Reaction = r1
3207
     }
3208
3209
     // Run after controls and a steps block are completed to
     // post process any data and provide downstream results
3210
3211
     Analysis {
3212
3213
3214
     // A block of tests to perform to validate that the sample was processed correctly
3215
     // Optionally, destructive tests can be performed to validate results on a
3216
     // dipstick basis
3217
     Validation {
3218
3219
3220
     10.4 Neurotransmitter Sensing
3221
     protocol Neurotransmitter_Sensing
3222
3223
3224
3225
     import (
3226
     "github.com/antha-lang/antha/antha/anthalib/wtype" // LHComponent type
     "github.com/antha-lang/antha/anthalib/wutil"
3227
     "github.com/antha-lang/antha/antha/anthalib/mixer" //sample function is imported from mixe
3228
3229
3230
     // Input parameters
3231
     Parameters (
     Sample_volume //no val specified
3232
3233
     Reagent_volume //no val specified
3234
     , Vol. 1, No. 1, Article . Publication date: February 2017.
```

```
electrophoresis_buffer_volume //no val specified
3235
3236
     )
3237
3238
     // Data which is returned from this protocol, and data types
     Data (
3242
     )
3243
3244
     // Physical Inputs to this protocol with types
     Inputs (
     Sample *wtype.LHComponent
     Reagent *wtype.LHComponent
3248
     electrophoresis_buffer *wtype.LHComponent
3249
3250
3251
3252
3253
     // Physical outputs from this protocol with types
3254
     Outputs (
     Mixture *wtype.LHComponent
3255
3256
     )
3257
3258
3259
     Requirements {
3260
     }
3261
3262
     // Conditions to run on startup
3263
     Setup {
3264
3265
3266
     // The core process for this protocol, with the steps to be performed
3267
3268
     // for every input
     Steps {
3269
3270
     sample := mixer.Sample(Sample, Sample_volume)
3271
     reagent := mixer.Sample(Reagent, Reagent_volume)
3272
3273
3274
     Mixture = mixer.Mix(sample, reagent) // cannot specify duration of mixture
     //cannot perform capillary electrophoresis
3275
     //cannot measure fluorescence to get a reading
3276
     }
3277
3278
3279
     Analysis {
3280
3281
     }
3282
3283
```

:68 Anon.

Table 2. Experimental tests that validate ChemType; parentheses denote reactive group numbers assigned to chemical names. The AIHA and Safey Zone tests are real-world documented incidents that ChemType could have prevented. I & C respectively denote Incompatible errors (known to be dangerous) and Caution errors (likely to be dangerous) based on the EPA/NOAA reactive groups. This table includes both synthetic and real-world results.

3289	Experiment Name	Outcome	Experiment Description
3290	AIHA 1 [2]	FAIL - I	Mix Nitric Acid (2) and Tetrachloroethylene (17,28)
3291	AIHA 2 [2]	FAIL - I	Mix Nitric Acid (2) and Methanol (4)
3292	AIHA 3 [2]	FAIL - I	Mix Potassium Hydride (35, 21) and Diaminopropane (7)
3293	Mustard Gas [1]	FAIL - I	Mix Calcium Hypo (1) and Dichlor (17)
3294	Safety Zone [12]	FAIL - I	Mix Hydrogen Peroxide (44) and Sulfuric Acid (2), then mix
3295			Acetone (19)
3296 3297	Sythn - Fail 1	FAIL - I	Mix Nitric Acid (2) and Hydrochloric Acid (1)
3297	Sythn - Fail 2	FAIL - C	Mix Hydrochloric Acid (1) and Isopropanol (4)
3298	Sythn - Fail 3	FAIL - C	Mix Benzoin (4, 16, 19) and Urea (6), then mix formaldehyde
3300			(5, 76)
3301	Sythn - Pass 1	PASS	Mix Benzene (16) and Urea (6)
3302	Sythn - Pass 2	PASS	Mix Benzene (16) and Urea (6), then mix Tetraethoxysilane
3303			(58)
3303		•	•

```
Validation {
3305
      }
3306
```

CHEMTYPE RESULTS

Efficacy of ChemType

ChemType Runtime Evaluation

Table 3. Execution time and number of constraints for synthetic and real-world bioassays.

3334				
3335	Benchmark	Compilation Time	Constraint Solving	Gathered Con-
3336		(sec)	Time (sec)	straints
3337	AIHA 1 [2]	0.012	0.936	70
3338	AIHA 2 [2]	0.012	1.648	68
3339	AIHA 3 [2]	0.014	1.214	17
3340	Broad Spectrum Opi-	0.011	0.887	11
3341	ate [4, 24, 35]			
3342	Ciprofloxacin [24]	0.023	1.722	14
3343	Diazepam [20]	0.024	1.007	14
3344	Dilution [20]	0.014	0.892	9
3345	Fentanyl [35]	0.018	0.900	13
3346	Full Morphine [20]	0.048	4.188	19
3347	Glucose Detection [3]	0.012	1.633	14
3348	Heroine [20]	0.020	1.553	13
3349	Image Probe Synthe-	0.015	2.181	13
3350	sis [3]			
3351	Morphine [20]	0.018	1.026	13
3352	Mustard Gas [1]	0.015	1.433	83
3353	Neurotransmitter	0.029	3.845	6
3354	Sensing [3]			
3355	Oxycodone [4]	0.026	0.959	13
3356	PCR [3]	0.032	3.534	8
3357	Safety Zone [12]	0.013	1.341	76
3358	Synthetic 1	0.031	1.507	196
3359	Synthetic 2	0.014	1.505	274
3360	Synthetic 3	0.012	0.944	95
3361	Synthetic 4	0.014	0.623	1
3362	Synthetic 5	0.026	1.350	10
3363	Synthetic 6	0.013	0.855	8

REFERENCES

3364 3365

3366

3367

3368

3369

3370

3371

3372

3373

3374

3375

3376

3377

3378

3379

3380

- [1] 2016. Swimming pool chemical incident. http://bit.ly/2gghGZI. (2016). accessed: 2016-11-01.
- [2] American Industrial Hygiene Association. 2016. http://bit.ly/2eZtf1m. (2016). Accessed: 2016-11-08.
- [3] Ahmed M. Amin, Mithuna Thottethodi, T. N. Vijaykumar, Steven Wereley, and Stephen C. Jacobson. 2007. Aquacore: a programmable architecture for microfluidics. In 34th International Symposium on Computer Architecture (ISCA 2007), June 9-13, 2007, San Diego, California, USA. 254–265. https://doi.org/10.1145/1250662.1250694
- [4] Ronald C Backer, Joseph R Monforte, and Alphonse Poklis. 2005. Evaluation of the DRI® oxycodone immunoassay for the detection of oxycodone in urine. *Journal of analytical toxicology* 29, 7 (2005), 675–677.
- [5] E. Barsoukov and J.R. Macdonald. 2005. Impedence Spectroscopy: Theory, Experiments, and Applications. Wiley-Interscience (2005).
- [6] Amar S. Basu. 2013. Droplet morphometry and velocimetry (DMV): a video processing software for time-resolved, label-free tracking of droplet parameters. Lab Chip 13 (2013), 1892–1901. Issue 10. https://doi.org/10.1039/C3LC50074H
- [7] Karl-Friedrich Böhringer. 2006. Modeling and Controlling Parallel Tasks in Droplet-Based Microfluidic Systems. IEEE Trans. on CAD of Integrated Circuits and Systems 25, 2 (2006), 334–344. https://doi.org/10.1109/TCAD.2005.855958
- [8] Ying-Han Chen, Chung-Lun Hsu, Li-Chen Tsai, Tsung-Wei Huang, and Tsung-Yi Ho. 2013. A Reliability-Oriented Placement Algorithm for Reconfigurable Digital Microfluidic Biochips Using 3-D Deferred Decision Making Technique. IEEE Trans. on CAD of Integrated Circuits and Systems 32, 8 (2013), 1151–1162. https://doi.org/10.1109/TCAD.2013. 2249558

:70 Anon.

[9] Minsik Cho and David Z. Pan. 2008. A High-Performance Droplet Routing Algorithm for Digital Microfluidic Biochips.
 IEEE Trans. on CAD of Integrated Circuits and Systems 27, 10 (2008), 1714–1724. https://doi.org/10.1109/TCAD.2008.
 2003282

- [10] P. Cooreman, R. Thoelen, J. Manca, M. vandeVen, V. Vermeeren, L. Michiels, M. Ameloot, and P. Wagner. 2005. Impedimetric immunosensors based on the conjugated polymer PPV. *Biosens. Bioelectron.* 20 (2005), 2151âĂŞ2156. Issue 10.
- [11] Jie Ding, Krishnendu Chakrabarty, and Richard B. Fair. 2001. Scheduling of microfluidic operations for reconfigurabletwo-dimensional electrowetting arrays. IEEE Trans. on CAD of Integrated Circuits and Systems 20, 12 (2001), 1463–1468. https://doi.org/10.1109/43.969439
 - [12] DA Dobbs, RG Bergman, and KH Theopold. 1990. Piranha solution explosion. (1990), 2-2 pages.
- [13] Ryan Fobel, Christian Fobel, and Aaron R. Wheeler. 2013. DropBot: An open-source digital microfluidic control system with precise control of electrostatic driving force and instantaneous drop velocity measurement. *Applied Physics Letters* 102, 19, Article 193513 (2013). https://doi.org/10.1063/1.4807118
- [14] Georges G. E. Gielen (Ed.). 2006. Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2006, Munich, Germany, March 6-10, 2006. European Design and Automation Association, Leuven, Belgium. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=11014
 - [15] J. Gong and C.J. Kim. 2008. Direct-referencing two-dimensional-array digital microfluidics using multilayer printed circuit board. J. Microelectromech. Syst. 17 (2008), 257–264. Issue 2.
 - [16] Daniel Grissom and Philip Brisk. 2012. Path scheduling on digital microfluidic biochips. In The 49th Annual Design Automation Conference 2012, DAC '12, San Francisco, CA, USA, June 3-7, 2012, Patrick Groeneveld, Donatella Sciuto, and Soha Hassoun (Eds.). ACM, 26–35. https://doi.org/10.1145/2228360.2228367
- and Sona Flassouri (Eds.). ACM, 26–35. https://doi.org/10.1145/2228360.2228367
 Daniel Grissom and Philip Brisk. 2014. Fast Online Synthesis of Digital Microfluidic Biochips. IEEE Trans. on CAD of Integrated Circuits and Systems 33, 3 (2014), 356–369. https://doi.org/10.1109/TCAD.2013.2290582
- [18] Daniel Grissom, Christopher Curtis, and Philip Brisk. 2014. Interpreting Assays with Control Flow on Digital
 Microfluidic Biochips. ACM Journal on Emerging Technologies (JETC) in Computing Systems 10 (April 2014). Issue 3.
- [19] B. Hadwen, G. R. Broder, D. Morganti, A. Jacobs, C. Brown, J. R. Hector, Y. Kubota, and H. Morgan. 2012. Programmable large area digital microfluidic array with integrated droplet sensing for bioassays. *Lab Chip* 12, 18 (Sep 2012), 3305–3313.
- [20] Peter V Hornbeck. 1991. Enzyme-linked immunosorbent assays. Current protocols in immunology (1991), 2–1.
 - [21] Kai Hu, Bang-Ning Hsu, Andrew Madison, Krishnendu Chakrabarty, and Richard B. Fair. 2013. Fault detection, real-time error recovery, and experimental demonstration for digital microfluidic biochips. In *Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013*, Enrico Macii (Ed.). EDA Consortium San Jose, CA, USA / ACM DL, 559–564. https://doi.org/10.7873/DATE.2013.124
- [22] Tsung-Wei Huang and Tsung-Yi Ho. 2009. A fast routability- and performance-driven droplet routing algorithm for digital microfluidic biochips. In 27th International Conference on Computer Design, ICCD 2009, Lake Tahoe, CA, USA, October 4-7, 2009. 445–450. https://doi.org/10.1109/ICCD.2009.5413119
- [23] Tsung-Wei Huang, Chun-Hsien Lin, and Tsung-Yi Ho. 2010. A Contamination Aware Droplet Routing Algorithm for
 the Synthesis of Digital Microfluidic Biochips. IEEE Trans. on CAD of Integrated Circuits and Systems 29, 11 (2010),
 1682–1695. https://doi.org/10.1109/TCAD.2010.2062770
- [24] Yousheng Jiang, Xuanyun Huang, Kun Hu, Wenjuan Yu, Xianle Yang, and Liqun Lv. 2011. Production and characterization of monoclonal antibodies against small hapten-ciprofloxacin. African Journal of Biotechnology 10, 65 (2011), 14342–14347.
- [25] Oliver Keszocze, Robert Wille, Krishnendu Chakrabarty, and Rolf Drechsler. 2015. A General and Exact Routing Methodology for Digital Microfluidic Biochips. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2015, Austin, TX, USA, November 2-6, 2015, Diana Marculescu and Frank Liu (Eds.). IEEE, 874–881. https://doi.org/10.1109/ICCAD.2015.7372663
- Oliver Keszocze, Robert Wille, and Rolf Drechsler. 2014. Exact routing for digital microfluidic biochips with temporary blockages. In *The IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2014, San Jose, CA, USA, November 3-6, 2014*, Yao-Wen Chang (Ed.). IEEE, 405–410. https://doi.org/10.1109/ICCAD.2014.7001383
- 3422 [27] Y. Li, H. Li, and R.J. Baker. 2014. Volume and concentration identification by using an electrowetting on dielectric device. *IEEE DCAS* (2014), 1–4.
- [28] C. Liao and S. Hu. 2011. Multiscale variation-aware techniques for high-performance digital microfluidic lab-on-a-chip component placement. *IEEE Trans Nanobioscience* 10, 1 (Mar 2011), 51–58.
- [29] G. Lippmann. 1875. Relations entre les phénomènes électriques et capillaires. Gauthier-Villars. https://books.google.ch/books?id=IkqVNwAACAA]
- [30] Chia-Hung Liu, Kuang-Cheng Liu, and Juinn-Dar Huang. 2013. Latency-optimization synthesis with module selection
 for digital microfluidic biochips. In 2013 IEEE International SOC Conference, Erlangen, Germany, September 4-6, 2013,
 Norbert Schuhmann, Kaijian Shi, and Nagi Naganathan (Eds.). IEEE, 159–164. https://doi.org/10.1109/SOCC.2013.

3396

3397

3406

3407

3408

- 3431 6749681
- [31] L. Luan, R.D. Evans, N.M. Jokerst, and R.B. Fair. 2008. Integrated optical sensor in a digital microfluidic platform. *IEEE Sensors* 8 (2008), 628–635. Issue 5.
- [32] Yan Luo, Bhargab B. Bhattacharya, Tsung-Yi Ho, and Krishnendu Chakrabarty. 2015. Design and Optimization of a Cyberphysical Digital-Microfluidic Biochip for the Polymerase Chain Reaction. *IEEE Trans. on CAD of Integrated Circuits and Systems* 34, 1 (2015), 29–42. https://doi.org/10.1109/TCAD.2014.2363396
- [33] Elena Maftei, Paul Pop, and Jan Madsen. 2010. Tabu search-based synthesis of digital microfluidic biochips with dynamically reconfigurable non-rectangular devices. *Design Autom. for Emb. Sys.* 14, 3 (2010), 287–307. https://doi.org/10.1007/s10617-010-9059-x
- [34] Elena Maftei, Paul Pop, and Jan Madsen. 2013. Module-Based Synthesis of Digital Microfluidic Biochips with Droplet-Aware Operation Execution. JETC 9, 1 (2013), 2. https://doi.org/10.1145/2422094.2422096
- [35] Chi-Liang Mao, Keith D Zientek, Patrick T Colahan, Mei-Yueh Kuo, Chi-Ho Liu, Kuo-Ming Lee, and Chi-Chung Chou. 2006. Development of an enzyme-linked immunosorbent assay for fentanyl and applications of fentanyl antibody-coated nanoparticles for sample preparation. *Journal of pharmaceutical and biomedical analysis* 41, 4 (2006), 1332–1341.
- 3444 [36] H. Moon, S.K. Cho, and C.J. Garrell, R.and Kim. 2002. Low voltage electrowetting-on-dielectric. J. Appl. Phys. 92 (2002), 4080âĂŞ4087. Issue 7.
- [37] Frieder Mugele and Jeanchristophe Baret. 2005. Electrowetting: from basics to applications. Journal of Physics:
 Condensed Matter 17 (2005), R705-R774.
- [38] Miguel Angel Murran and Homayoun Najjaran. 2012. Capacitance-based droplet position estimator for digital
 microfluidic devices. Lab Chip 12 (2012), 2053–2059. Issue 11. https://doi.org/10.1039/C2LC21241B
- [39] J. H. Noh, J. Noh, E. Kreit, J. Heikenfeld, and P. D. Rack. 2012. Toward active-matrix lab-on-a-chip: programmable electrofluidic control enabled by arrayed oxide thin film transistors. *Lab Chip* 12, 2 (Jan 2012), 353–360.
- [40] Kenneth O'Neal, Daniel Grissom, and Philip Brisk. 2012. Force-Directed List Scheduling for Digital Microfluidic
 Biochips. In 20th IEEE/IFIP International Conference on VLSI and System-on-Chip, VLSI-SoC 2012, Santa Cruz, CA, USA,
 October 7-10, 2012, Srinivas Katkoori, Matthew R. Guthaus, Ayse Kivilcim Coskun, Andreas Burg, and Ricardo Reis
 (Eds.). IEEE, 7-11. https://doi.org/10.1109/VLSI-SoC.2012.6378997
- [41] MG Pollack, AD Shenderov, and RB Fair. 2002. Electrowetting-based actuation of droplets for integrated microfluidics. *Lab on a Chip* 2, 2 (2002), 96–101.
- [42] Andrew J. Ricketts, Kevin M. Irick, Narayanan Vijaykrishnan, and Mary Jane Irwin. 2006. Priority scheduling in digital
 microfluidics-based biochips, See [14], 329–334. https://doi.org/10.1109/DATE.2006.244178
- [43] Pranab Roy, Hafizur Rahaman, and Parthasarathi Dasgupta. 2010. A novel droplet routing algorithm for digital microfluidic biochips. In Proceedings of the 20th ACM Great Lakes Symposium on VLSI 2009, Providence, Rhode Island, USA, May 16-18 2010, R. Iris Bahar, Fabrizio Lombardi, David Atienza, and Erik Brunvand (Eds.). ACM, 441-446. https://doi.org/10.1145/1785481.1785583
- [44] Pranab Roy, Hafizur Rahaman, and Parthasarathi Dasgupta. 2012. Two-level clustering-based techniques for intelligent droplet routing in digital microfluidic biochips. *Integration* 45, 3 (2012), 316–330. https://doi.org/10.1016/j.vlsi.2011.11.
 3462
 906
- [45] S. C. Shih, I. Barbulovic-Nad, X. Yang, R. Fobel, and A. R. Wheeler. 2013. Digital microfluidics with impedance sensing for integrated cell culture and analysis. *Biosens Bioelectron* 42 (Apr 2013), 314–320.
- [46] Steve C. C. Shih, Ryan Fobel, Paresh Kumar, and Aaron R. Wheeler. 2011. A feedback control system for high-fidelity digital microfluidics. *Lab Chip* 11 (2011), 535–540. Issue 3. https://doi.org/10.1039/C0LC00223B
- 3466 [47] Yong Jun Shin and Jeong Bong Lee. 2010. Machine vision for digital microfluidics. Review of Scientific Instruments 81, 1
 3467 (2 2010). https://doi.org/10.1063/1.3274673
- [48] Fei Su and Krishnendu Chakrabarty. 2006. Module placement for fault-tolerant microfluidics-based biochips. ACM
 Trans. Design Autom. Electr. Syst. 11, 3 (2006), 682–710. https://doi.org/10.1145/1142980.1142987
- [49] Fei Su and Krishnendu Chakrabarty. 2008. High-level synthesis of digital microfluidic biochips. JETC 3, 4 (2008). https://doi.org/10.1145/1324177.1324178
- [50] Fei Su, William L. Hwang, and Krishnendu Chakrabarty. 2006. Droplet routing in the synthesis of digital microfluidic
 biochips, See [14], 323–328. https://doi.org/10.1109/DATE.2006.244177
- [51] Ian I. Suni. 2008. Impedance methods for electrochemical sensors using nanomaterials. *TrAC Trends in Analytical Chemistry* 27, 7 (2008), 604 611. https://doi.org/10.1016/j.trac.2008.03.012 Electroanalysis Based on Nanomaterials.
- Tao Xu and Krishnendu Chakrabarty. 2008. Integrated droplet routing and defect tolerance in the synthesis of digital microfluidic biochips. JETC 4, 3 (2008). https://doi.org/10.1145/1389089.1389091
- [53] Tao Xu, Krishnendu Chakrabarty, and Fei Su. 2008. Defect-Aware High-Level Synthesis and Module Placement for
 Microfluidic Biochips. IEEE Trans. Biomed. Circuits and Systems 2, 1 (2008), 50–62. https://doi.org/10.1109/TBCAS.2008.
 918283

:72 Anon.

[54] Hailong Yao, Qin Wang, Yiren Shen, Tsung-Yi Ho, and Yici Cai. 2016. Integrated Functional and Washing Routing
 Optimization for Cross-Contamination Removal in Digital Microfluidic Biochips. *IEEE Trans. on CAD of Integrated Circuits and Systems* 35, 8 (2016), 1283–1296. https://doi.org/10.1109/TCAD.2015.2504397

[55] Ping-Hung Yuh, Chia-Lin Yang, and Yao-Wen Chang. 2007. Placement of defect-tolerant digital microfluidic biochips using the T-tree formulation. JETC 3, 3 (2007). https://doi.org/10.1145/1295231.1295234

- [56] Ping-Hung Yuh, Chia-Lin Yang, and Yao-Wen Chang. 2008. BioRoute: A Network-Flow-Based Routing Algorithm for the Synthesis of Digital Microfluidic Biochips. IEEE Trans. on CAD of Integrated Circuits and Systems 27, 11 (2008), 1928–1941. https://doi.org/10.1109/TCAD.2008.2006140
- [57] Yang Zhao and Krishnendu Chakrabarty. 2012. Cross-Contamination Avoidance for Droplet Routing in Digital Microfluidic Biochips. IEEE Trans. on CAD of Integrated Circuits and Systems 31, 6 (2012), 817–830. https://doi.org/10. 1109/TCAD.2012.2183369