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

2017. XXXX-XXXX/2017/2-ART $15.00
https://doi.org/

1 TYPING JUDGMENTS

Typing judgments are repeated here for convenience

1.1 Syntax

t ::=		Terms:
|	$x \in \mathcal{X}$	Variable
|	$t_1 \oplus t_2$	Math operation
|	detect module on x for t	Detect
|	$v \in \mathcal{V}$	Value
v ::=		Values:
|	mat	Material value
|	r	Real number
|	n	Natural number
$module$ ::=		
|	$\text{module}_1, .., \text{module}_n$	Sensor module(s)
s ::=		Statements:
|	$i;\ s$	Sequencing
|	skip	Skip
i ::=		instructions:
|	$x := t$	Assignment
|	$x :=$ mix x_1 with x_2 for t	Mixing
|	$\langle x_1, .., x_n \rangle :=$ split x 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
|	$\mathbb{R}$	Real number
|	$\mathbb{N}$	Natural number
Γ ::=		Context:
|	$\emptyset$	Empty context
|	$\Gamma, x : T$	Variable type binding
X		Set of variables x
C		Constraints

Fig. 1. Syntax of ChemType.

1.2 Evaluation Rules

E-Var
$$\frac{x \in \mathrm{dom}(\sigma)}{(\sigma, x) \longrightarrow \sigma(x)}$$

E-MathR1
$$\frac{(\sigma, t_1) \longrightarrow t_1'}{(\sigma,\ t_1 \oplus t_2) \longrightarrow t_1' \oplus t_2}$$

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

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

E-DetectR
$$\frac{(\sigma, t) \longrightarrow t'}{(\sigma,\ \text{detect } \mathit{module} \text{ on } x \text{ for } t) \longrightarrow \text{detect } \mathit{module} \text{ on } x \text{ for } t'}$$

E-Detect
$$\frac{\sigma(x) \in \mathit{Mat}_i \qquad r_2 = \text{detect}(\sigma(x),\ \mathit{module},\ r_1)}{(\sigma,\ \text{detect } \mathit{module} \text{ on } x \text{ for } r_1) \longrightarrow r_2}$$

Fig. 2. Evaluation rules for terms.

E-AssignR
$$\frac{(\sigma, t) \longrightarrow t' \qquad t \notin \mathcal{X}}{(\sigma,\ x := t;\ s) \longrightarrow (\sigma,\ x := t';\ s)}$$

E-Assign
$$\frac{}{(\sigma,\ x := v;\ s) \longrightarrow (\sigma[x \mapsto v],\ s)}$$

E-Assign′
$$\frac{\sigma' = (\sigma \setminus \{x'\})[x \mapsto \sigma(x')]}{(\sigma,\ x := x';\ s) \longrightarrow (\sigma', s)}$$

E-MixR
$$\frac{(\sigma, t) \longrightarrow t'}{(\sigma, x := \mathsf{mix}\ x_1\ \mathsf{with}\ x_2\ \mathsf{for}\ t;\ s) \longrightarrow (\sigma, x := \mathsf{mix}\ x_1\ \mathsf{with}\ x_2\ \mathsf{for}\ t';\ s)}$$

E-Mix
$$\frac{\begin{array}{c}\sigma(x_1) \in Mat_i \qquad \sigma(x_2) \in Mat_j \\ \mathsf{interact}(\sigma(x_1),\ \sigma(x_2),\ r) \neq \lightning \\ \sigma' = (\sigma \setminus \{x_1, x_2\})[x \mapsto \mathsf{interact}(\sigma(x_1),\ \sigma(x_2),\ r)]\end{array}}{(\sigma, x := \mathsf{mix}\ x_1\ \mathsf{with}\ x_2\ \mathsf{for}\ r;\ s) \longrightarrow (\sigma',\ s)}$$

E-Split
$$\frac{\sigma(x) \in Mat_i \qquad \sigma' = (\sigma \setminus \{x\})\overline{[x_i \mapsto \mathsf{split}(\sigma(x),\ n)]}}{(\sigma,\ \langle x_0, ..\ x_n\rangle := \mathsf{split}\ x\ \mathsf{into}\ n;\ s) \longrightarrow (\sigma',\ s)}$$

E-IfR
$$\frac{(\sigma,\ t) \longrightarrow t'}{(\sigma,\ \mathsf{if}\ t\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2;\ s) \longrightarrow (\sigma,\ \mathsf{if}\ t'\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2;\ s)}$$

E-IfTrue
$$\frac{n \neq 0}{(\sigma,\ \mathsf{if}\ n\ s_1\ \mathsf{else}\ s_2;\ s) \longrightarrow (\sigma,\ s_1 \bullet\ s)}$$

E-IfFalse
$$\frac{}{(\sigma,\ \mathsf{if}\ 0\ s_1\ \mathsf{else}\ s_2;\ s) \longrightarrow (\sigma,\ s_2 \bullet\ s)}$$

E-While
$$(\sigma,\ \mathsf{while}\ t\ s_1;\ s_2) \longrightarrow (\sigma,\ \mathsf{if}\ t\ \mathsf{then}\ (s_1 \bullet \mathsf{while}\ t\ s_1;\ s_2)\ \mathsf{else}\ s_2)$$

$$\mathsf{skip} \bullet s = s$$
$$(i;\ s) \bullet s' = i;\ (s \bullet s')$$

Fig. 3. Evaluation rules for statements.

1.3 Type Checking Rules

$$\text{(T-VAR)} \quad \frac{x : T \in \Gamma \qquad x \in X}{\Gamma, X \vdash x : T}$$

$$\text{(T-MATH)} \quad \frac{\Gamma, X \vdash t_1 : T \qquad \Gamma, X \vdash t_2 : T \qquad T = \mathbb{N} \vee T = \mathbb{R}}{\Gamma, X \vdash t_1 \oplus t_2 : T}$$

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

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

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

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

Fig. 4. Type checking rules for terms.

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

$$\text{(T-SKIP)} \quad \frac{}{\Gamma, X \vdash \mathsf{skip}, X}$$

$$\text{(T-ASSIGN-1)} \quad \frac{x : T \in \Gamma \qquad \Gamma, X \vdash v : T' \qquad T' \subseteq T}{\Gamma, X \vdash x := v, X \cup \{x\}}$$

$$\text{(T-ASSIGN-2)} \quad \frac{x : T \in \Gamma \qquad \Gamma, X \vdash x' : T' \qquad T' \subseteq T}{\Gamma, X \vdash x := x', X \setminus \{x'\} \cup \{x\}}$$

$$\text{(T-ASSIGN-3)} \quad \frac{\begin{array}{c} x : T \in \Gamma \qquad t \notin \mathcal{V} \cup \mathcal{X} \qquad \Gamma, X \vdash t : T' \\ T' = \mathbb{R} \vee T' = \mathbb{N} \qquad T' \subseteq T \end{array}}{\Gamma, X \vdash x := t, X \cup \{x\}}$$

$$\text{(T-MIX)} \quad \frac{\Gamma, X \vdash t : \mathbb{R} \qquad \dfrac{\Gamma, X \vdash x_1 : \cup\overline{Mat_i} \qquad \Gamma, X \vdash x_2 : \cup\overline{Mat_j}}{\mathsf{interact\text{-}abs}(Mat_i, Mat_j) \subseteq \Gamma(x)}}{\Gamma, X \vdash x := \mathsf{mix}\ x_1\ \mathsf{with}\ x_2\ \mathsf{for}\ t, X \setminus \{x_1, x_2\} \cup \{x\}}$$

$$\text{(T-SPLIT)} \quad \frac{\Gamma, X \vdash x : \cup\overline{Mat_i} \qquad \Gamma(x) \subseteq \Gamma(x_1), .., \Gamma(x) \subseteq \Gamma(x_n)}{\Gamma, X \vdash \langle x_1, .., x_n \rangle := \mathsf{split}\ x\ \mathsf{into}\ n, X \setminus \{x\} \cup \{x_1, .., x_n\}}$$

$$\text{(T-IF)} \quad \frac{\Gamma, X \vdash t : \mathbb{N} \qquad \Gamma, X \vdash s_1, X' \qquad \Gamma, X \vdash s_2, X''}{\Gamma, X \vdash \mathsf{if}\ t\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2, X' \cap X''}$$

$$\text{(T-WHILE)} \quad \frac{\Gamma, X \vdash t : \mathbb{N} \qquad \Gamma, X \vdash s, X' \qquad X \subseteq X'}{\Gamma, X \vdash \mathsf{while}\ t\ s, X}$$

Fig. 5. Type checking rules for statements.

1.4 Type Inference Rules

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

(CT-Math) $$\frac{\Gamma, X \vdash t_1 : T_1 \mid C_1 \qquad \Gamma, X \vdash t_2 : T_2 \mid C_2}{\begin{array}{c}\Gamma, X \vdash t_1 \oplus t_2 : T_1 \mid \\ C_1 \cup C_2 \cup \{T_1 = T_2 = \mathbb{N} \vee T_1 = T_2 = \mathbb{R}\}\end{array}}$$

(CT-Detect) $$\frac{\Gamma, X \vdash x : T_1 \mid C_1 \qquad \Gamma, X \vdash t : T_2 \mid C_2}{\begin{array}{c}\Gamma, X \vdash \text{detect } \mathit{module} \text{ on } x \text{ for } t : \mathbb{R} \mid \\ C_1 \cup C_2 \cup \{T_1 \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset, T_2 = \mathbb{R}\}\end{array}}$$

(CT-Mat) $$\frac{\overline{mat \in Mat_i}}{\Gamma, X \vdash mat : \cup \overline{Mat_i} \mid \emptyset}$$

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

(CT-Nat) $$\Gamma, X \vdash n : \mathbb{N} \mid \emptyset$$

Fig. 6. Type Inference rules for terms.

$$\text{(CT-INST)} \quad \frac{\Gamma, X \vdash i, X' \mid C_1 \qquad \Gamma, X' \vdash s, X'' \mid C_2}{\Gamma, X \vdash i;\ s, X'' \mid C_1 \cup C_2}$$

$$\text{(CT-SKIP)} \quad \frac{}{\Gamma, X \vdash \mathsf{skip}, X \mid \emptyset}$$

$$\text{(CT-ASSGN-1)} \quad \frac{x : T \in \Gamma \qquad \Gamma, X \vdash v : T' \mid C'}{\Gamma, X \vdash x := v, X \cup \{x\} \mid C' \cup \{T' \subseteq T\}}$$

$$\text{(CT-ASSGN-2)} \quad \frac{x : T \in \Gamma \qquad \Gamma, X \vdash x' : T' \mid C'}{\Gamma, X \vdash x := x', X \setminus \{x'\} \cup \{x\} \mid C' \cup \{T' \subseteq T\}}$$

$$\text{(CT-ASSGN-3)} \quad \frac{x : T \in \Gamma \qquad t \notin \mathcal{V} \cup \mathcal{X} \qquad \Gamma, X \vdash t : T' \mid C'}{\Gamma, X \vdash x := t, X \cup \{x\} \mid C' \cup \{T' = \mathbb{R} \vee T' = \mathbb{N}, T' \subseteq T\}}$$

$$\text{(CT-MIX)} \quad \frac{\begin{array}{c}\Gamma, X \vdash x_1 : T \mid C \qquad \Gamma, X \vdash x_2 : T' \mid C' \\ \Gamma, X \vdash t : T'' \mid C''\end{array}}{\begin{array}{c}\Gamma, X \vdash x := \mathsf{mix}\ x_1\ \mathsf{with}\ x_2\ \mathsf{for}\ t, X \setminus \{x_1, x_2\} \cup \{x\} \mid C \cup C' \cup C'' \cup \\ \{T \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset,\ T' \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset,\ T'' = \mathbb{R}, \\ \overline{Mat_i \in T \wedge Mat_j \in T' \Rightarrow \mathsf{interact\text{-}abs}(Mat_i, Mat_j) \subseteq \Gamma(x)\}}\end{array}}$$

$$\text{(CT-SPLIT)} \quad \frac{\Gamma, X \vdash x : T \mid C}{\begin{array}{c}\Gamma, X \vdash \langle x_1, .., x_n \rangle := \mathsf{split}\ x\ \mathsf{into}\ n, X \setminus \{x\} \cup \{x_1, .., x_n\} \mid \\ C \cup \{T \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset, T \subseteq \Gamma(x_1), .., T \subseteq \Gamma(x_n)\}\end{array}}$$

$$\text{(CT-IF)} \quad \frac{\Gamma, X \vdash t : T \mid C \qquad \Gamma, X \vdash s_1, X' \mid C_1 \qquad \Gamma, X \vdash s_2, X'' \mid C_2}{\Gamma, X \vdash \mathsf{if}\ t\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2, X' \cap X'' \mid C \cup C_1 \cup C_2 \cup \{T = \mathbb{N}\}}$$

$$\text{(CT-WHILE)} \quad \frac{\Gamma, X \vdash t : T \mid C \qquad \Gamma, X \vdash s, X' \mid C' \qquad X \subseteq X'}{\Gamma, X \vdash \mathsf{while}\ t\ s, X, C \cup C' \cup \{T = \mathbb{N}\}}$$

Fig. 7. Type Inference rules for statements.

2 PROOFS

2.1 Helper Lemmas

Lemma 1 (Name extension).
$\forall\, \Gamma, X, i, X', X''.$
$\quad \Gamma, X_1 \vdash i, X_2 \;\wedge\; X_1 \subseteq X_1' \;\Rightarrow$
$\exists X_2'.$
$\quad \Gamma, X_1' \vdash i, X_2' \;\wedge\; X_2 \subseteq X_2'$
and
$\forall\, \Gamma, X, s, X', X''.$
$\quad \Gamma, X_1 \vdash s, X_2 \;\wedge\; X_1 \subseteq X_1' \;\Rightarrow$
$\exists X_2'.$
$\quad \Gamma, X_1' \vdash s, X_2' \;\wedge\; X_2 \subseteq X_2'$

Proof. Trivial by mutual induction on s and i. □

Lemma 2.
For every Γ, X, i, X', s, *and* X'', *if*
$\Gamma, X \vdash i, X'$, $X'' \subseteq X'$ *and* $\Gamma, X'' \vdash s, X'''$
there exists X'''' *such that*
$\Gamma, X \vdash i; s, X''''$ *and* $X''' \subseteq X''''$

Proof. Direct from Lemma 1 and the rule T-Inst. □

Lemma 3 (Typing $\bullet$).
For every Γ, X, s_1, X', s_2, *and* X''
$\Gamma, X \vdash s_1, X' \;\wedge\; \Gamma, X' \vdash s_2, X'' \;\Rightarrow\; \Gamma, X \vdash s_1 \bullet s_2, X''$

Proof. Trivial by induction on s_1. □

Lemma 4.
For every Γ, X, s_1, X', s_2, *and* X'', *if*
$\Gamma, X \vdash s_1, X'$, $X'' \subseteq X'$ *and* $\Gamma, X'' \vdash s_2, X'''$
there exists X'''' *such that*
$\Gamma, X \vdash s_1 \bullet s_2, X''''$ *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* $\Gamma, X \vdash v : \mathbb{R}$, *then* $v \in \mathbb{R}$.
- *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* $\Gamma, X \vdash t_1 \oplus t_2 : T$,
    then $\Gamma, X \vdash t_1 : T$ *and* $\Gamma, X \vdash t_1 : T$ *and* $T = \mathbb{N} \vee T = \mathbb{R}$

(3) *If* $\Gamma, X \vdash$ *detect module on* x *for* $t : T$,
then there exists $\overline{Mat_i}$ *such that* $\Gamma, X \vdash x : \cup\overline{Mat_i}$ *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* Γ, X, i, s, X'',
If
$\Gamma, X \vdash i;\ s, X''$,
then, there exists X' *such that*
$\Gamma, X \vdash i, X'$ *and*
$\Gamma, X' \vdash s, X''$.
(2) *For all* Γ, X, x, t, X',
If
$\Gamma, X \vdash x := t, X'$
then, there exists T, T' *such that*
$x\colon T \in \Gamma$,
$\Gamma, X \vdash t : T'$,
$T' \subseteq T$,
$T' = \mathbb{R} \lor T' = \mathbb{N} \lor t = mat$, *and*
$X' = X \cup \{x\}$
or
$t = x'$ *and*
$X' = X \setminus \{x'\} \cup \{x\}$.
(3) *For all* $\Gamma, X, x, x_1, x_2, t, X'$,
If
$\Gamma, X \vdash x :=$ *mix* x_1 *with* x_2 *for* t, X'
then, there exist i*s and* j*s such that*
$\Gamma, X \vdash x_1 : \cup\overline{Mat_i}$,
$\Gamma, X \vdash x_2 : \cup\overline{Mat_j}$,
$\Gamma, X \vdash t : \mathbb{R}$,
$\overline{\textit{interact-abs}(Mat_i, Mat_j)} \subseteq \Gamma(x)$, *and*
$X' = X \setminus \{x_1, x_2\} \cup \{x\}$.
(4) *For all* $\Gamma, X, x, x_1, .., x_n, X'$,
If
$\Gamma, X \vdash \langle x_1, .., x_n\rangle :=$ *split* x *into* n, X'
then, there exist i*s such that*
$\Gamma, X \vdash x : \cup\overline{Mat_i}$,
$\Gamma(x) \subseteq \Gamma(x_1), .., \Gamma(x) \subseteq \Gamma(x_n)$, *and*
$X' = X \setminus \{x\} \cup \{x_1, .., x_n\}$.
(5) *For all* $\Gamma, X, t, s_1, s_2, X'''$,
If
$\Gamma, X \vdash$ *if* t *then* s_1 *else* s_2, X'''
then, there exists X' *and* X'' *such that*
$\Gamma, X \vdash t : \mathbb{N}$,

$\Gamma, X \vdash s_1, X'$,
$\Gamma, X \vdash s_2, X''$, *and*
$X''' = X' \cap X''$.

(6) *For all* Γ, X, t, s, X',
If
$\Gamma, X \vdash$ *while* $t\ s, X'$
then,
$\Gamma, X \vdash t : \mathbb{N}$
$\Gamma, X \vdash s, X''$,
$X \subseteq X''$ *and*
$X' = X$.

Proof. Immediate from case analysis on the type derivation rules. □

Lemma 8.
For every σ, t *and* t' *if* $(\sigma, t) \longrightarrow t'$ *then* $t' \notin \mathcal{X}$

Proof. Immediate from the term transition rules. □

Helper Definitions

The conservative property of the abstract interact-abs function:

$\forall mat_i, mat_j, r.$
$\quad mat_i \in Mat_i \wedge mat_j \in Mat_j \Rightarrow$
$\quad\quad \text{interact}(mat_i, mat_j, r) = \lightning \Rightarrow \text{interact-abs}(Mat_i, Mat_j)$ undefined
$\quad\quad \text{interact}(mat_i, mat_j, r) \neq \lightning \Rightarrow \text{interact}(mat_i, mat_j, r) \in \text{interact-abs}(Mat_i, Mat_j)$

The type of the functions used for evaluation:

detect : $Mat_i \longrightarrow Module \longrightarrow \mathbb{R} \longrightarrow \mathbb{R}$
interact : $Mat_i \longrightarrow Mat_j \longrightarrow \text{interact-abs}(Mat_i, Mat_j)$
split : $Mat_i \longrightarrow \mathbb{N} \longrightarrow Mat_i$

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

$consistent(\Gamma,\ X,\ \sigma)\ =\ \forall x, T.$
$\quad (x : T) \in \Gamma \wedge x \in X \Rightarrow$
$\quad \sigma(x) \in T$

2.2 Proof of Progress

Lemma 9 (**Progress of Terms**).
For every Γ, X, t *or* T,
if
$\Gamma, X \vdash t : T$
then
$\forall \sigma.\ consistent(\Gamma,\ X,\ \sigma) \Rightarrow \exists t'.\ (\sigma, t) \longrightarrow t'$ *or*
t *is a value.*

Proof.
Proof by induction on t and case analysis thereafter.
We assume
(1) $\Gamma, X \vdash t : T$
Case for the rule T-Var:
We have
(2) $t = x$
(3) $x : T \in \Gamma$
(4) $x \in X$
We assume
(5) $consistent(\Gamma, X, \sigma)$
From [3], [4] and [5]:
(5) $\sigma(x) \in T$
By the rule E-Var on [5]:
(6) $(\sigma, x) \longrightarrow \sigma(x)$
By the rule E-Var on [6] and [2]:
(6) $(\sigma, t) \longrightarrow \sigma(x)$

Case for the rule T-Math:
(2) $t = t_1 \oplus t_2$
(3) $\Gamma \vdash t_1 : T$
(4) $\Gamma \vdash t_2 : T$
(5) $T = \mathbb{R} \vee T = \mathbb{N}$
By I.H. on t_1:
Case 1: t_1 is not a value:
(6) $\forall \sigma, X.\ consistent(\sigma, X, \Gamma) \Rightarrow \exists t_1'.\ (\sigma, t_1) \longrightarrow t_1'$
We assume that
(7) $consistent(\sigma, X, \Gamma)$
and prove that
$\exists t', (\sigma,\ t) \longrightarrow t'$
From [6] and [7], there exists t_1' such that
(8) $(\sigma, t_1) \longrightarrow t_1'$
From the rule E-MathR1 on [8]:
(9) $(\sigma,\ t_1 \oplus t_2) \longrightarrow t_1' \oplus t_2$
From [9] on [2]
$(\sigma,\ t) \longrightarrow t_1' \oplus t_2$
Case 2:
(10) t_1 is a value v_1:
By Lemma 5 on [3], [5], [10]

(11) $v_1 \in \mathbb{N} \lor v_1 \in \mathbb{R}$

By I.H. on t_2:

Case 2.1: t_2 is not a value:

(12) $\forall \sigma, X.\ consistent(\sigma, X, \Gamma) \Rightarrow \exists t_2', (\sigma,\ t_2) \longrightarrow t_2'$

We assume that

(13) $consistent(\sigma, X, \Gamma)$

and prove that

$\exists t', (\sigma,\ t) \longrightarrow t'$

From [12] and [13], there exists t_2' such that

(14) $(\sigma,\ t_2) \longrightarrow t_2'$

From the rule E-MathR2 on [14]:

(15) $(\sigma,\ v_1 \oplus t_2) \longrightarrow v_1 \oplus t_2'$

From [15], [2], [10]:

$(\sigma,\ t) \longrightarrow v_1 \oplus t_2'$

Case 2.2:

(16) t_2 is a value, v_2:

By Lemma 5 on [4], [5], [16]

(17) $v_2 \in \mathbb{N} \lor v_2 \in \mathbb{R}$

From the rule E-Math on [11] and [17]:

(18) $(\sigma, v_1 \oplus v_2) \longrightarrow v_1 \oplus v_2$

From on [17], [10], [16]

$(\sigma, t) \longrightarrow v_1 \oplus v_2$

Case for the rule T-Detect:

$t =$ detect $module_i$ on x for t'

Similar to the the rule T-Math rule, by induction hypothesis in t' and then using the rule E-Detect and the rule E-DetectR. The consistency condition is used to derive the first premise of the rule E-Detect.

Case for the rule T-Mat:

mat is a value.

Case for the rule T-Real:

r is a value.

Case for the rule T-Nat:

n is a value. □

Proof.

Case analysis on the typing derivation:

Case for the rule T-Skip:

$s =$ skip

Case for the rule T-Inst:

$s = i;\ s'$

$\Gamma, X \vdash i, X''$

Immediate from Lemma 10. □

Lemma 10 (**Progress for Instructions**).
For every Γ, X, i, s, X',
if
$\Gamma, X \vdash i, X'$
then
$\forall \sigma.\ consistent(\Gamma, X, \sigma) \Rightarrow$
$\exists \sigma', s'.\ (\sigma,\ i; s) \longrightarrow (\sigma',\ s')$

Proof.
We have that
(1) $\Gamma, X \vdash i, X'$
Case analysis on the typing derivation:
Case for the rule T-Assign-1:
(2) $i = (x := v)$
(3) $x : T \in \Gamma$
(4) $\Gamma, X \vdash t : T'$
(5) $T' \subseteq T$
From the rule E-Assign
(6) $(\sigma, x := v;\ s) \longrightarrow (\sigma[x \mapsto v];\ s)$
From [6], [2]
$(\sigma,\ i; s) \longrightarrow (\sigma[x \mapsto v];\ s)$

Case for the rule T-Assign-2:

Similar to the previous case. The reduction uses the rule E-Assign′

Case for the rule T-Assign-3:
(2) $i = (x := t)$
(3) $x : T \in \Gamma$
(4) $\Gamma, X \vdash t : T'$
(5) $(T' = \mathbb{R} \vee T' = \mathbb{N}) \wedge t \notin \mathcal{V} \cup \mathcal{X}$
(6) $T' \subseteq T$
By Lemma 9 on [4]:
Case 1:
(7) $\forall \sigma.\ consistent(\Gamma,\ X,\ \sigma) \Rightarrow \exists t'.\ (\sigma, t) \longrightarrow t'$
We assume that
(8) $consistent(\Gamma,\ X,\ \sigma)$
We prove that
$(\sigma,\ i; s) \longrightarrow (\sigma',\ s')$
From [7] and [8], there exists t' such that
(9) $(\sigma, t) \longrightarrow t'$
From the rule E-AssignR on [9] and [5]:
(10) $(\sigma, (x := t);\ s) \longrightarrow (\sigma',\ (x := t');\ s)$
From [2] on [10]:
$(\sigma, i;\ s) \longrightarrow (\sigma',\ (x := t');\ s)$
Case 2:
(11) t is a value, v.

Contradiction with [5].

Case for the rule T-Mix:
(2) $i = (x := \text{mix } x_1 \text{ with } x_2 \text{ for } t)$

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_j) is defined; thus, by the conservative property of the abstract interaction interact$(\sigma(x_1), \sigma(x_2)) \neq \lightning$.

Case for the rule T-Split:
(2) $i = (\langle x_1, .. \, x_n \rangle = \text{split } x_1 \text{ into } n)$

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.

□

Lemma 11 (**Progress for Statements**).
For every Γ, X, s, X',
if
$\Gamma, X \vdash s, X'$
then either s *is skip or*
$\forall \sigma.\ consistent(\Gamma, X, \sigma) \Rightarrow$
$\exists \sigma', s'.\ (\sigma, s) \longrightarrow (\sigma', s')$

Proof.
We have that
(1) $\Gamma, X \vdash s, X'$
Case analysis on s:
Case $s = \text{skip}$
Conclusion is immediate.
Case
(2) $s = i; s'$

From [1] and [2],
(3) $\Gamma, X \vdash i; s', X'$
By Lemma 7 on [3], there exists X'' such that
(4) $\Gamma, X \vdash i, X''$
(5) $\Gamma, X'' \vdash s', X'$
By Lemma 10 on [4],
(6) $\forall\sigma.\ consistent(\Gamma, X, \sigma) \Rightarrow$
$\exists\sigma', s''.\ (\sigma, i; s') \longrightarrow (\sigma', s'')$
The conclusion is immediate form [2] and [6].

□

2.3 Proof of Preservation

Lemma 12 (**Preservation of Terms**).
For every Γ, X, t, T *and* σ,
if
$\Gamma, X \vdash t : T$ *and*
$(\sigma,\ t) \longrightarrow t'$ *and*
$consistent(\Gamma, X, \sigma)$
then
$\Gamma, X \vdash t' : T$

Proof.
We have
(1) $\Gamma, X \vdash t : T$
(2) $(\sigma,\ t) \longrightarrow t'$
(3) $consistent(\Gamma, X, \sigma)$
Straightforward induction on the derivation of $\Gamma, X \vdash t : T$ and then case analysis on the final rule in the derivation of $(\sigma, t) \longrightarrow t'$

Case for the rule T-Var:
From the rule T-Var:
(4) $t = x$
(5) $x : T \in \Gamma$
(6) $x \in X$
From the rule E-Var:
(7) $t' = \sigma(x)$
From [3], [5], and [6]
(7) $\sigma(x) \in T$
By case analysis on the value $\sigma(x)$ and the rule T-Mat,
the rule T-Real and the rule T-Nat
$\Gamma, X \vdash \sigma(x) : T$

Case for the rule T-Math:
(4) $t = t_1 \oplus t_2$
(5) $\Gamma, X \vdash t_1 : T$
(6) $\Gamma, X \vdash t_2 : T$
(7) $T = \mathbb{R} \vee T = \mathbb{N}$
Case analysis on [2]:

Case for the rule E-MathR1:

(8) $t' = t'_1 \oplus t_2$

(9) $(\sigma, t_1) \longrightarrow t'_1$

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) $(\sigma, t_2) \longrightarrow t'_2$

By I.H. on [6], [13], and [3]:

(14) $\Gamma, X \vdash t'_2 : T$

From the rule T-Math on [12], [5], [14]:

(15) $\Gamma, X \vdash t' : T$

Case for the rule E-Math:

(16) $t_1 = v_1$

(17) $t_2 = v_2$

(18) $(v_1 \in \mathbb{N} \wedge v_2 \in \mathbb{N}) \vee (v_1 \in \mathbb{R} \wedge v_2 \in \mathbb{R})$

(19) $v_1 \oplus v_2 = v$

(20) $t' = v$

From [18], we consider the case:

(21) $v_1 \in \mathbb{N} \wedge v_2 \in \mathbb{N}$

The other case is similar.

From [21], [19]:

(22) $v \in \mathbb{N}$

From [20], [22] and the rule T-Nat:

(23) $\Gamma, X \vdash t' : \mathbb{N}$

From Lemma 6 on [5], [16] and [21]

(24) $T = \mathbb{N}$

From [23] and [24]:

(23) $\Gamma, X \vdash t' : T$

Case for the rule T-Detect:

(4) $t =$ detect $module_i$ on x for t'

We consider the two cases for [2]: Case for the rule E-DetectR: Induction hypothesis is applied to t' and then the rule T-Detect is applied. Case for the rule E-Detect: Immediate from the rule T-Real.

Case for the rule T-Mat:

(4) $t = mat$

mat is a value and does not step.

Case for the rule T-Real:

(4) $t = r$

r is a value and does not step.

Case for the rule T-Nat:
(4) $t = n$
n is a value and does not step.

□

Lemma 13 (**Preservation of Statements**).
For every $\Gamma, X, \sigma, s, X'', \sigma', s'$,
if
$\Gamma, X \vdash s, X''$ *and*
$(\sigma,\ s) \longrightarrow (\sigma',\ s')$ *and*
$consistent(\Gamma, X, \sigma)$
then there exists X' *such that*
$\Gamma, X' \vdash s', X''$ *and*
$consistent(\Gamma, X', \sigma')$

Proof.
We have
(1) $\Gamma, X \vdash s, X''$
(2) $(\sigma,\ s) \longrightarrow (\sigma',\ s')$
(3) $consistent(\Gamma, X, \sigma)$

Case Analysis on [1]:
Case for the rule T-Skip:
Contradiction in [2]: The statement skip does not step.

Case for the rule T-Inst:
(4) $\Gamma, X \vdash i, X'$
(5) $\Gamma, X' \vdash s'', X''$
(6) $s = i;\ s''$

Case Analysis on [4]:
Case for the rule T-Assign-1:
(7) $i = (x := v)$
(8) $x : T \in \Gamma$
(9) $\Gamma, X \vdash t : T'$
(10) $X' = X \cup \{x\}$
(11) $T' \subseteq T$
From [6], [7]:
(12) $s = (x := v);\ s''$
Case analysis on [2]:
Case for the rule E-Assign:
(13) $t = v$
(14) $s' = s''$
Case analysis on v:
Case (15) $v = r$
By Lemma 6 on [9], [13] and [15]
(16) $T' = \mathbb{R}$
From [16], [11]

(17) $\{\mathbb{R}\} \subseteq T$
From [15], [17]
(18) $v \in T$
From [3], [10], [8], [18]
(19) $consistent(\Gamma, X', \sigma[x \mapsto v])$
From [5], [14]
(20) $\Gamma, X' \vdash s', X''$
The conclusion is [20] and [19].
Case $T' = \mathbb{N}$
Similar to the previous case.
Case $t = mat$
Similar to the previous case.
Case for the rule E-AssignR:
$(\sigma, v) \longrightarrow t'$
There is no reduction rule for values. Contradiction.

Case for the rule T-Assign-2:

The reduction is with the the rule E-Assign′. Similar to the case for the rule T-Assign-2, the consistency of σ with Γ and X and that $T' \subseteq T$ implies the consistency of $(\sigma \setminus \{x'\})[x \mapsto \sigma(x')]$ with Γ and $X \setminus \{x'\} \cup \{x\}$.

Case for the rule T-Assign-3:
(7) $i = (x := t)$
(8) $x : T \in \Gamma$
(9) $\Gamma, X \vdash t : T'$
(10) $(T' = \mathbb{R} \lor T' = \mathbb{N}) \land t \notin \mathcal{V} \cup \mathcal{X}$
(11) $T' \subseteq T$
(12) $X' = X \cup \{x\}$
From [6], [7]:
(13) $s = (x := t);\ s''$
Case analysis on [2]:
Case for the rule E-AssignR:
(14) $s' = (x := t');\ s''$
(15) $(\sigma, t) \longrightarrow t'$
By Lemma 12 on [9], [15], and [3]:
(16) $\Gamma, X \vdash t' : T$
By Lemma 8 on [15]
(17) $t' \notin \mathcal{X}$
From [17] and either the rule T-Assign-1 or the rule T-Assign-3 on [8], [16], [10], [11]:
(18) $\Gamma, X \vdash x := t', X \cup \{x\}$
From [18] and [12]:
(19) $\Gamma, X \vdash x := t', X'$
From the rule T-Inst on [19], [5], and then [14]:
(20) $\Gamma, X \vdash s', X''$
The conclusion is [20], [3].
Case for the rule E-Assign:

$t = v$

Contradiction with [10].

Case for the rule E-ASSIGN$'$:

$t = x$

Contradiction with [10].

Case for the rule T-MIX:

(7) $i = (x := \mathsf{mix}\ x_1\ \mathsf{with}\ x_2\ \mathsf{for}\ t)$

(8) $\Gamma, X \vdash x_1 : \cup\overline{Mat_i}$

(9) $\Gamma, X \vdash x_2 : \cup\overline{Mat_j}$

(10) $\Gamma, X \vdash t : \mathbb{R}$

(11) $\overline{\mathsf{interact\text{-}abs}(Mat_i, Mat_j) \subseteq \Gamma(x)}$

(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 $\sigma(x_1) \in \overline{Mat_i}$ 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 $\mathsf{interact\text{-}abs}(Mat_i, Mat_j)$ is defined, from the conservative property of the abstract interact-abs function, we have $\mathsf{interact}(\sigma(x_1), \sigma(x_2), r) \in \mathsf{interact\text{-}abs}(Mat_i, Mat_j)$. 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 $conistent(\Gamma, X \cup \{x\}, \sigma[x \mapsto \mathsf{interact}(\sigma(x_1), \sigma(x_2), r)])$. Therefore, we have $conistent(\Gamma, X \setminus \{x_1, x_2\} \cup \{x\}, \sigma \setminus \{x_1, x_2\}\ [x \mapsto \mathsf{interact}(\sigma(x_1), \sigma(x_2), r)])$.

Case for the rule T-SPLIT:

(7) $i = (\overline{\langle x_1, ..\ x_n \rangle} = \mathsf{split}\ x\ \mathsf{into}\ n)$

(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 $\mathsf{split}(\sigma(x), n) \in \Gamma(x)$. From this and [9], we have that for every $j \in \{1..n\}$, $\mathsf{split}(\sigma(x), n) \in \Gamma(x_j)$. From this and the consistency assumption for σ, we have $conistent(\Gamma, X \cup \{x_1, .., x_n\}, \sigma\overline{[x_i \mapsto \mathsf{split}(\sigma(x),\ n)]})$. Therefore, we have $conistent(\Gamma, X \setminus \{x\} \cup \{x_1, .., x_n\}, (\sigma \setminus \{x\})\overline{[x_i \mapsto \mathsf{split}(\sigma(x),\ n)]})$.

Case for the rule T-IF
(7) $i = \text{if } t \text{ then } s_1 \text{ else } s_2$
(8) $\Gamma, X \vdash t : \mathbb{N}$
(9) $\Gamma, X \vdash s_1, X_1$
(10) $\Gamma, X \vdash s_2, X_2$
(11) $X' = X_1 \cap X_2$

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

Case of the rule E-IFR:
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 σ stays unchanged in the step.

Case of the rule E-IFTRUE:
From [11], we have $X_1 \subseteq X'$. From this and [9], by Lemma 1, we have $\Gamma, X \vdash s_1, X'$. From this and [5], by Lemma 3, we have $\Gamma, X \vdash s_1 \bullet s'', X''$. The assumed consistency condition is preserved since the store σ stays unchanged in the step.

Case of the rule E-IFFALSE:
Similar to the previous case.

Case for the rule T-WHILE:
(7) $i = \text{while } t\ s'''$
(8) $\Gamma, X \vdash \text{while } t\ s''', X$
(9) $\Gamma, X \vdash t : \mathbb{N}$
(10) $\Gamma, X \vdash s''', X'''$
(11) $X \subseteq X'''$
(12) $X' = X$

Case analysis on [2]: There is only one case.
Case for the rule E-WHILE:
(13) $s' = \text{if } t \text{ then } (s''' \bullet \text{while } t\ s''';\ s'') \text{ else } s''$.
By Lemma 4 on [10], [11] and [8], we have
(14) $\Gamma, X \vdash (s''' \bullet \text{while } t\ s'''), X_1$
(15) $X \subseteq X_1$
From [5] and [12], we have
(16) $\Gamma, X \vdash s'', X''$
By Lemma 2 on [14], [15] and [16], we have
(17) $\Gamma, X \vdash (s''' \bullet \text{while } t\ s''';\ s''), X_2$
(18) $X'' \subseteq X_2$
By the rule T-IF on [9], [17], [16], we have
(19) $\Gamma, X \vdash \text{if } t \text{ then } (s''' \bullet \text{while } t\ s''';\ s'') \text{ else } s'', X_2 \cap X''$
From [19], [13] and [18], we have
$\Gamma, X \vdash s', X''$
The assumed consistency condition [3] is preserved since the store σ stays unchanged in the step.

$\square$

2.4 Proof of Soundness

Lemma 14 (**Soundness of Type Inference for Terms**).
For every Γ, X, t, T, C *and* m,
if
$\Gamma, X \vdash t : T \mid C$
m *is a model for* C
then
$m(\Gamma), X \vdash t : m(T)$

Proof.
Hypothesis:
(1) $\Gamma, X \vdash t : T \mid C$
(2) m is a model for C

Structural induction on [1]:
Case the rule CT-Var:
Trivial

Case the rule CT-Math:
(3) $\Gamma, X \vdash t_1 : T_1 \mid C_1$
(4) $\Gamma, X \vdash t_2 : T_2 \mid C_2$
(5) $C = C_1 \cup C_2 \cup \{T_1 = T_2 = \mathbb{N} \vee T_1 = T_2 = \mathbb{R}\}$
(6) $T = T_1$
(7) $t = t_1 \oplus t_2$
From [2] and [5]:
(8) m is a model for C_1.
(9) m is a model for C_2.
By I.H. on ([3], [8]), ([4], [9]):
(10) $m(\Gamma), X \vdash t_1 : m(T_1)$
(11) $m(\Gamma), X \vdash t_2 : m(T_2)$
From [5], [2]:
$m(T_1) = m(T_2) = \mathbb{N} \ \vee \ m(T_1) = m(T_2) = \mathbb{R}$
We consider the first disjunct. The case for the second one is similar.
(12) $m(T_1) = m(T_2) = \mathbb{N}$
From [12], [10] and [11]:
(13) $m(\Gamma), X \vdash t_1 : \mathbb{N} \ \wedge \ m(\Gamma), X \vdash t_2 : \mathbb{N}$
By the rule T-Math on [13], [14]:
(15) $m(\Gamma), X \vdash t_1 \oplus t_2 : \mathbb{N}$
From [15], [6], [7] and [12]:
(16) $m(\Gamma), X \vdash t : m(T)$

Case the rule CT-Detect:

Similar to the previous case. The only interesting step is that considering the syntax of types T, the equality $m(T_1) \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset$ implies $m(T_1) = \cup\overline{Mat_i}$.

Case the rule CT-Mat:
Trivial

Case the rule CT-Real:
Trivial

Case the rule CT-Nat:
Trivial

□

Lemma 15 (**Soundness of Type Inference for Statements**).
For every Γ, X, s, C, X'', *and* m,
if
$\Gamma, X \vdash s, X'' \mid C$
m *is a model for* C
then
$m(\Gamma), X \vdash s, X''$

Proof.
Hypothesis:
(1) $\Gamma, X \vdash s \mid C$
(2) m is a model for C

Structural induction on [1]:
Case the rule CT-Skip:
Trivial.

Case the rule CT-Inst:
(3) $s = i; s'$
(4) $\Gamma, X \vdash i, X' \mid C_1$
(5) $\Gamma, X' \vdash s', X'' \mid C_2$
(6) $C = C_1 \cup C_2$
From [2] and [6]:
(7) m is a model for C_1
(8) m is a model for C_2
By I.H. on [5]:
(9) $m(\Gamma), X' \vdash s, X''$
We need to show
(10) $m(\Gamma), X \vdash i, X'$
and then by the rule T-Inst on [9], [10]:
(11) $m(\Gamma), X \vdash i; s', X''$
and then by from [10], [11]:
$m(\Gamma), X \vdash s, X''$

Now assuming [4] and [7] that is
(4) $\Gamma, X \vdash i, X' \mid C_1$
(7) m is a model for C_1
we show that

$m(\Gamma), X \vdash i, X'$

Case analysis on [4]:

Case the rule CT-Assign-1:
(12) $i = (x := v)$
(13) $x : T \in \Gamma$
(14) $\Gamma, X \vdash v : T' \mid C'$
(15) $X' = X \cup \{x\}$
(16) $C_1 = C' \cup \{T' \subseteq T\}$
From [7] and [16]:
(17) m is a model for C'.
(18) $m(T') \subseteq m(T)$
From [13]:
(19) $x : m(T) \in m(\Gamma)$
From Lemma 14 on [14] and [17]:
(20) $m(\Gamma), X \vdash mat : m(T')$
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]:
$m(\Gamma), X \vdash i, X'$

Case the rule CT-Assign-2:

Similar to the previous case. The the rule T-Assign-2 is used.

Case the rule CT-Assign-3:

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 rule T-Assign-3.

Case the rule CT-Mix:
(12) $\Gamma, X \vdash x_1 : T \mid C$
(13) $\Gamma, X \vdash x_2 : T' \mid C'$
(14) $\Gamma, X \vdash t : T'' \mid C''$
(15) $i = (x := \texttt{mix } x_1 \texttt{ with } x_2 \texttt{ for } t)$
(16) $X' = X \setminus \{x_1, x_2\} \cup \{x\}$
(17) $C_1 = C \cup C' \cup C'' \cup \{T \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset,\ T' \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset,\ T'' = \mathbb{R}$
$\overline{Mat_i \in T \wedge Mat_j \in T' \Rightarrow \text{interact-abs}(Mat_i, Mat_j) \subseteq \Gamma(x)\}}$
From [7] and [17]:
(18) m is a model for C.
(19) m is a model for C'.
(20) m is a model for C''.
(21) $m(T) \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset$
(22) $m(T') \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset$
(23) $m(T'') = \mathbb{R}$

(24) $\overline{Mat_i \in m(T) \wedge Mat_j \in m(T') \Rightarrow \text{interact-abs}(Mat_i, Mat_j) \subseteq m(\Gamma(x))}$

From Lemma 14 on [12] and [18]:

(25) $m(\Gamma), X \vdash x_1 : m(T)$

From Lemma 14 on [13] and [19]:

(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) $m(T) = \cup\overline{Mat_i}$

From [22], there exists js such that:

(29) $m(T') = \cup\overline{Mat_j}$

From [25] and [28]:

(30) $m(\Gamma), X \vdash x_1 : \cup\overline{Mat_i}$

From [26] and [29]:

(31) $m(\Gamma), X \vdash x_2 : \cup\overline{Mat_j}$

From [27] and [23]:

(32) $m(\Gamma), X \vdash t : \mathbb{R}$

From [24], [28], and [29]:

(33) $\overline{\text{interact-abs}(Mat_i, Mat_j) \subseteq \Gamma(x)}$

By the rule T-Mix on [30], [31], [32], and [33]:

(34) $\Gamma, X \vdash x := \text{mix } x_1 \text{ with } x_2 \text{ for } t, X \setminus \{x_1, x_2\} \cup \{x\}$

From [34], [15] and [16]:

(34) $\Gamma, X \vdash i, X'$

Case the rule CT-Split:

Similar to the case for the rule CT-Mix. The Lemma 14 is applied to the constraint typing judgement for x.

Case the rule CT-If:

Immediate from applying the Lemma 14 on t and the induction hypothesis on s_1 and s_2.

Case the rule CT-While:

Immediate from applying the Lemma 14 on t and the induction hypothesis on s.

□

2.5 Proof of Completeness

Lemma 16 (**Completeness of Type Inference for Terms**).
For all Γ, X, t, T, *and* C,
if

$m(\Gamma), X \vdash t : T$

$\Gamma, X \vdash t : T' \mid C$

then

m *is a model for* C

$m(T') = T$

Proof.

Hypothesis

(1) $m(\Gamma), X \vdash t : T$

(2) $\Gamma, X \vdash t : T' \mid C$

Proof by induction on the given constraint typing derivation [2].

Case for the rule CT-Var:

(3) $t = x$

(4) $x : T \in \Gamma$

(5) $x \in X$

(6) $C = \emptyset$

By the inversion Lemma 6 on [1]:

(7) $x : T' \in m(\Gamma)$

(8) $x \in X$

From [4] and [7]:

(7) $T' = m(T)$

The conclusion is immediate from [6] and [7].

Case for the rule CT-Math:

(3) $\Gamma, X \vdash t_1 : T_1 \mid C_1$

(4) $\Gamma, X \vdash t_2 : T_2 \mid C_2$

(5) $C = C_1 \cup C_2 \cup \{T_1 = T_2 = \mathbb{N} \vee T_1 = T_2 = \mathbb{R}\}$

(6) $T' = T_1$

By the inversion Lemma 6 on [1]:

(7) $m(\Gamma), X \vdash t_1 : T$

(8) $m(\Gamma), X \vdash t_2 : T$

$T = \mathbb{N} \vee T = \mathbb{R}$

We consider the case for the first disjunct:

(The case for the second one is similar.)

(9) $T = \mathbb{N}$

By induction hypothesis on [7] and [3]:

(10) m is a model for C_1

(11) $m(T_1) = T$

By induction hypothesis on on [8] and [4]:

(12) m is a model for C_2

(13) $m(T_2) = T$

From [11], [13] and [9]:

(14) $m(T_1) = m(T_2) = \mathbb{N}$

From [5], [10], [12] and [14]:

(15) m is a model for C.

From [6], [11], and [9]:

(16) $m(T') = T$

The conclusion is [15] and [16].

Case for the rule CT-Detect:

Similar to the case for the rule T-Math. By induction hypothesis on x and t.

Case for the rule CT-Mat
Trivial.

Case for the rule CT-Real:
Trivial.

Case for the rule CT-Nat:
Trivial.

□

Lemma 17 (**Completeness of Type Inference for Statements**).
For all Γ, X, s, X', C, *and* m,
if
$m(\Gamma), X \vdash s, X'$
$\Gamma, X \vdash s, X'' \mid C$
then
m *is a model for* C.

Proof.
Hypothesis
(1) $m(\Gamma), X \vdash s, X_1$
(2) $\Gamma, X \vdash s, X_2 \mid C$
We show that
m is a model for C.

Induction on the derivation of [2]:

Case for the rule CT-Skip:
Trivial.

Case for the rule CT-Inst:
(3) $s = i; s'$
(4) $\Gamma, X \vdash i, X' \mid C_1$
(5) $\Gamma, X' \vdash s', X_2 \mid C_2$
(6) $C = C_1 \cup C_2$
From [1] and [3]:
(7) $m(\Gamma), X \vdash i; s', X_1$
By the inversion Lemma 7 on [7]:
(8) $m(\Gamma), X \vdash i, X''$
(9) $m(\Gamma), X'' \vdash s', X_2$
By induction hypothesis on [9] and [5]:
(10) m is a model for C_2
We will show that from [8] and [4]:

(11) m is a model for C_1
From [6], [10] and [11]:
m is a model for C

Hypothesis:
(8) $m(\Gamma), X \vdash i, X''$
(4) $\Gamma, X \vdash i, X' \mid C_1$
Conclusion:
m is a model for C_1

Case analysis on derivation of [4]:

Case for the rule CT-Assign-1:
(12) $i = (x := v)$
(13) $x : T \in \Gamma$
(14) $\Gamma, X \vdash v : T' \mid C'$
(15) $X' = X \cup \{x\}$
(16) $C_1 = C' \cup \{T' \subseteq T\}$
From [8] and [12]:
(17) $m(\Gamma), X \vdash x := v, X''$
By the inversion Lemma 7 on [17]:
(18) $x : T'' \in m(\Gamma)$
(19) $m(\Gamma), X \vdash v : T'''$
(22) $T''' \subseteq T''$
(23) $X'' = X \cup \{x\}$
From [18], [13]:
(24) $m(T) = T''$
By Lemma 16 on [19], [14]:
(25) m is a model for C'
(26) $m(T') = T'''$
From [22], [24] and [26]
(27) $m(T') \subseteq m(T)$
From [16], [24], [25] and [27]
m is a model for C_1

Case for the rule CT-Assign-2:

Similar to the case for the rule CT-Assign-1.

Case for the rule CT-Assign-3:

Similar to the case for the rule CT-Assign-1.

Case for the rule CT-Mix:
(12) $i = (x := \mathsf{mix}\ x_1\ \mathsf{with}\ x_2\ \mathsf{for}\ t)$
(12) $\Gamma, X \vdash x_1 : T \mid C$
(13) $\Gamma, X \vdash x_2 : T' \mid C'$
(14) $\Gamma, X \vdash t : T'' \mid C''$

(15) $C_1 = C \cup C' \cup C'' \cup$
$$\left\{\frac{T \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset,\ T' \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset,\ T'' = \mathbb{R}}{Mat_k \in T \land Mat_l \in T' \Rightarrow \text{interact-abs}(Mat_k, Mat_l) \subseteq \Gamma(x)}\right\}$$

From [8] and [12]:
(16) $m(\Gamma), X \vdash x := \text{mix } x_1 \text{ with } x_2 \text{ for } t, X''$

By the inversion Lemma 7 on [16]:
(17) $m(\Gamma), X \vdash x_1 : \cup\overline{Mat_i}$
(18) $m(\Gamma), X \vdash x_2 : \cup\overline{Mat_j}$
(19) $m(\Gamma), X \vdash t : \mathbb{R}$
(20) $\overline{\text{interact-abs}(Mat_i, Mat_j) \subseteq m(\Gamma(x))}$

By Lemma 16 on [17], [12]:
(21) m is a model for C
(22) $m(T) = \cup\overline{Mat_i}$

By Lemma 16 on [18], [13]:
(23) m is a model for C'
(24) $m(T') = \cup\overline{Mat_j}$

By Lemma 16 on [19], [14]:
(25) m is a model for C''
(26) $m(T'') = \mathbb{R}$

From [22], [24], and [20]:
(27) $\overline{Mat_k \in m(T) \land Mat_l \in m(T') \Rightarrow \text{interact-abs}(Mat_k, Mat_l) \subseteq m(\Gamma(x))\}}$

From [22], [24], and [26]:
(28) $m(T) \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset,\ m(T') \cap \{\mathbb{R}, \mathbb{N}\} = \emptyset,\ m(T'') = \mathbb{R}$

From [15], [21], [23], [25], [28] and [27]:
m is a model for C_1

Case for the rule CT-Split:

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.

8 shows an example ELISA assay, which is one of the steps of the larger hierarchical opiate-detection 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.

```
Stationary: Anti-Fentanyl

Move 20uL of Urine Sample to Anti-Fentanyl
Move 100uL of Fentanyl-Conjugate to Anti-Fentanyl
Tap Anti-Fentanyl for 60s
Incubate Anti-Fentanyl at 23ÂžC for 60min
Drain Anti-Fentanyl

Repeat 6 times {
  Move 350uL of Distilled Water to Anti-Fentanyl
  Vortex Anti-Fentanyl for 45s
  Drain Anti-Fentanyl
}

Move 100uL of TMB Substrate to Anti-Fentanyl
Incubate Anti-Fentanyl at 23C for 30min
Mix Anti-Fentanyl with 100uL of Stop Reagent
                                          for 60s
Negative Reading = Measure the fluorescence of
                         Anti-Fentanyl for 30min
Drain Anti-Fentanyl
```

Fig. 8. Fentanyl ELISA

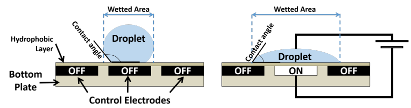

Fig. 9. Depiction of the electrowetting principle [29, 37]: applying an electrostatic potential to a droplet at rest reduces the contact angle with the surface, thereby increasing the surface area in contact with the droplet.

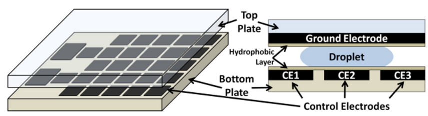

Fig. 10. Left: A digital microfluidic biochip (DMFB) is a planar array of electrodes [15, 19, 36, 39, 41]. Right: Cross-sectional view.

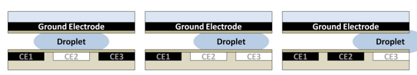

Fig. 11. A droplet is transported from CE2 to CE3 by activating CE3, and then deactivating CE2 (white: activated electrode; black: deactivated electrode).

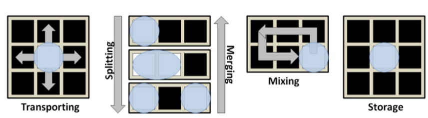

Fig. 12. Five basic fluidic operations.

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

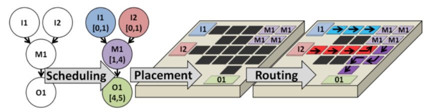

Fig. 13. Illustration of a typical DMFB compiler for assays without control flow (DAGs). The compiler must schedule, place, and route all operations to produce an executable sequence of electrode activations to automatically execute the assay (activated electrodes typically have a droplet on top).

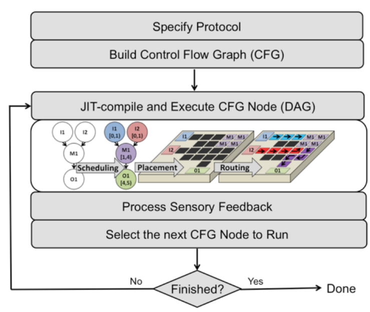

Fig. 14. A dynamic execution model for cyber-physical DMFBs.

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.

5 ASSAY EXECUTION VIDEOS

Below is a list of the videos generated while gathering data. We are not claiming these as novel or interesting contributions. They are merely provided to demonstrate how the simulator executes a given assay.

Urine Opiate Panel with initial positive indication: https://1drv.ms/v/s!AqwZF7jQGx_IgahNYojptiZr4RpjOg
Urine Opiate Panel with initial negative indication: https://1drv.ms/v/s!AqwZF7jQGx_IhJRtiOKGuSTm0qXX0
PCR Assay: https://1drv.ms/v/s!AqwZF7jQGx_IhNgWWxzt8GwtfICcFw
Image Probe Synthesis: https://1drv.ms/v/s!AqwZF7jQGx_IhNgbKsUIzDGqfyOhbw
Neurotransmitter Sensing: https://1drv.ms/v/s!AqwZF7jQGx_IhNgZjBF4vlsnDr1uCA
PCR Droplet Replenishment Assay: https://1drv.ms/v/s!AqwZF7jQGx_IhJRqfq1dLhbBUIL9RQ
Probabilistic PCR Full: https://1drv.ms/v/s!AqwZF7jQGx_IhJRo-xDbVT_Q6UsXPg
Probabilistic PCR Early Exit: https://1drv.ms/v/s!AqwZF7jQGx_IhJRpLXveGI8Qr7BWrw
Broad Spectrum Opiate Panel: https://1drv.ms/v/s!AqwZF7jQGx_IhJRs4f6laDxl6T_QJQ
Fentanyl ELISA Assay: https://1drv.ms/v/s!AqwZF7jQGx_IgahJFK519Id1HCad4g
Ciprofloxacin ELISA Assay: https://1drv.ms/v/s!AqwZF7jQGx_IgahKrD6EDWPbsy9G0g
Heroin ELISA Assay: https://1drv.ms/v/s!AqwZF7jQGx_IgahIrCW0z08KR1111A
Morphine ELISA Assay: https://1drv.ms/v/s!AqwZF7jQGx_IgahG3PHVTctW7u0rOw
Oxycodone ELISA Assay: https://1drv.ms/v/s!AqwZF7jQGx_IgahHIgmGgR43A-l7EA

6 BIOSCRIPT'S CHEMTYPE TESTS

6.1 Real-world Prevention Tests

AIHA Example1 where ChemType would have prevented an incident

```
Tetrachloroethylene = 50uL of C(=C(Cl)Cl)(Cl)Cl
Nitric_Acid = 50uL of [N+](=O)(O)[O-]

a = Mix Tetrachloroethylene with Nitric_Acid for 40s
===========================================================================
```

AIHA Example2 where ChemType would have prevented an incident

```
Methanol = 50uL of CO
Nitric_Acid = 50uL of [N+](=O)(O)[O-]

a = Mix Methanol with Nitric_Acid for 40s
===========================================================================
```

AIHA Example3 where ChemType would have prevented an incident

```
Nitrogen = N#N
Diaminopropane = 50uL of CC(CN)N
Potassium_Hydride = 50uL of [H-].[K+]

a = Mix Diaminopropane and Potassium_Hydride for 20s
b = Mix a and Nitrogen for 10s
===========================================================================
```

Real world example where ChemType would have prevented an incident

```
Dichlor = 50uL CC(=CCl)Cl
Calcium_Hypo = 50uL of [O-]Cl.[O-]Cl.[Ca+2]
```

```
Mix Dichlor and Calcium_Hypo for 20s
==========================================================================
```

6.2 Synthetic Failures

Example where ChemType will not allow the mix to occur

```
Nitric_Acid = 50uL of [N+](=O)(O)[O-]
Hydrochloric_Acid = 50uL of Cl

a = Mix Nitric_Acid with Hydrochloric_Acid for 40s
==========================================================================
```

Example where ChemType will not allow the mix to occur

```
Isopropanol = 50uL of CC(C)O
Hydrochloric_Acid = 50uL of Cl

a = Mix Isopropanol with Hydrochloric_Acid for 40s
==========================================================================
```

Example where ChemType will not allow the mix to occur

```
Formaldehyde = 50uL of C=O
Formaldehyde2 = 50uL of C=O

a = Mix Formaldehyde with Formaldehyde2 for 40s
==========================================================================
```

Example where ChemType will not allow the mix to occur

```
Benzoin = 50uL of C1=CC=C(C=C1)C(C(=O)C2=CC=CC=C2)O
Urea = 50uL of C(=O)(N)N
Formaldehyde = 50uL of C=O

a = Mix Benzoin with Urea for 40s
b = Mix Formaldehyde with a for 20s
==========================================================================
```

Example where ChemType will not allow the mix to occur

```
Benzene = 50uL of C1=CC=CC=C1
Formaldehyde = 50uL of C=O
Formaldehyde2 = 50uL of C=O

a = Mix Benzene with Formaldehyde for 40s
b = Mix Formaldehyde2 with a for 20s
==========================================================================
```

Example where ChemType will not allow the mix to occur

```
Benzene = 50uL of C1=CC=CC=C1
Urea = 50uL of C(=O)(N)N
Hydrochloric Acid = 50uL of Cl

a = Mix Benzene with Urea for 40s
```

```
b = Mix Hydrochloric Acid with a for 20s
===========================================================================
```

Example where ChemType would have prevented an incident

```
Hydrogen_Peroxide = 50uL OO
Sulfuric_Acid = 50uL of OS(=O)(=O)O
Acetone = 50uL of CC(=O)C

a = Mix Hydrogen_Peroxide and Sulfuric_Acid for 20s
b = Mix a and Acetone for 10s
===========================================================================
```

6.3 Synthetic Passes

Example where ChemType will allow the mix to occur

```
Benzene = 50uL of C1=CC=CC=C1
Urea = 50uL of C(=O)(N)N

a = Mix Benzene with Urea for 40s
===========================================================================
```

Example where ChemType will allow the mix to occur

```
Benzene = 50uL of C1=CC=CC=C1
Urea = 50uL of C(=O)(N)N
Benzotrichloride = 50uL of C1=CC=C(C=C1)C(Cl)(Cl)Cl

a = Mix Benzene with Urea for 40s
b = Mix Benzotrichloride with a for 20s
===========================================================================
```

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:

```
move x to y
tap y
```

BioScript would condense into:

```
y := mix x with y
```

7.2 Urine Opiate Hierarchy

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

When(BSO) Test Indicates Positive: https://1drv.ms/v/s!AqwZF7jQGx_IgahNYojptiZr4RpjOg

When(BSO) Test Indicates Negative: https://1drv.ms/v/s!AqwZF7jQGx_IhJRtiOKGuSTm0qXX0g

General Instruction	BioScript Equivalent
stationary Anti-Morphine	Anti-Morphine1 := x µl of Anti-Morphine
heat x at n	Not yet modeling temperature\volume. Left for future work.
incubate x at n	Not yet modeling temperature\volume. Left for future work.
move x to y	y := mix x with y
tap x	x := mix x with x
invert x	x := mix x with x
vortex x	x := mix x with x
drain x	$freshVariable := x$
repeat x times	while $x > 0$
measure x of x'	detect x on x'
perform...	x := detect x on x'

Table 1. Syntactic sugar translations from assay to BioScript.

BioScript Code:

```
Stationary Anti-Morphine
Stationary Anti-Oxy
Stationary Anti-Fentanyl
Stationary Anti-Ciprofloxcin
Stationary Anti-Heroin
Stationary Oxycodone-Enzyme

Stop-Reagent1 = 10uL of Stop-Reagent
Stop-Reagent2 = 10uL of Stop-Reagent
Stop-Reagent3 = 10uL of Stop-Reagent
Stop-Reagent4 = 10uL of Stop-Reagent
Stop-Reagent5 = 10uL of Stop-Reagent

Move 10uL of UrineSample to Anti-Morphine
Move 10uL of UrineSample to Anti-Oxy
Move 10uL of UrineSample to Anti-Fentanyl
Move 10uL of UrineSample to Anti-Ciprofloxcin
Move 10uL of UrineSample to Anti-Heroin

MorphineReading = Measure the fluorescence of Anti-Morphine for 5s
OxyReading = Measure the fluorescence of Anti-Oxy for 5s
FentanylReading = Measure the fluorescence of Anti-Fentanyl for 5s
CiproReading = Measure the fluorescence of Anti-Ciprofloxcin for 5s
HeroinReading = Measure the fluorescence of Anti-Heroin for 5s

Drain Anti-Morphine
Drain Anti-Oxy
Drain Anti-Fentanyl
Drain Anti-Ciprofloxcin
Drain Anti-Heroin
```

```
if (((FentanylReading == true) or (MorphineReading == true)) or (((OxyReading == true) or (

  Move 20uL of Urine-Sample to Anti-Heroin
  Move 100uL of Heroin-Conjugate to Anti-Heroin
  Tap Anti-Heroin for 60s
  Incubate Anti-Heroin at 23C for 60min
  Drain Anti-Heroin

  Repeat 6 times {
    Move 350uL of Distilled-Water to Anti-Heroin
    Vortex Anti-Heroin for 45s
    Drain Anti-Heroin
  }

  Move 100uL of TMB-Substrate to Anti-Heroin
  Incubate Anti-Heroin at 23C for 30min
  Mix Anti-Heroin with 100uL of Stop Reagent1 for 60s

  Heroin Reading = Measure the fluorescence of Anti-Heroin for 30min
  Drain Anti-Heroin

  Move 20uL of Urine-Sample to Oxycodone-Enzyme
  Move 100uL of Oxycodone-Conjugate to Oxycodone-Enzyme
  Tap Oxycodone-Enzyme for 60s
  Incubate Oxycodone-Enzyme at 23C for 60min
  Drain Oxycodone-Enzyme

  Repeat 6 times {
    Move 350uL of Distilled-Water to Oxycodone-Enzyme
    Vortex Oxycodone-Enzyme for 45s
    Drain Oxycodone-Enzyme
  }

  Move 100uL of TMB-Substrate to Oxycodone-Enzyme
  Incubate Oxycodone-Enzyme at 23C for 30min
  Mix Oxycodone-Enzyme with 100uL of Stop-Reagent2 for 60s

  Oxy Reading = Measure the fluorescence of Oxycodone-Enzyme for 30min
  Drain Oxycodone-Enzyme

  if ((Heroin Reading == false) and (Oxy Reading == false)) {
    Move 20uL of UrineSample to Anti-Ciprofloxacin
    Move 100uL of Ciprofloxacin-Conjugate to Anti-Ciprofloxacin
    Tap Anti-Ciprofloxacin for 60s
    Incubate Anti-Ciprofloxacin at 23C for 60min
    Drain Anti-Ciprofloxacin
```

```
    Repeat 5 times {
      Move 250uL of Distilled-Water to Anti-Ciprofloxacin
      Vortex Anti-Ciprofloxacin for 45s
      Drain Anti-Ciprofloxacin
    }

    Move 50uL of TMB-Substrate to Anti-Ciprofloxacin
    Incubate Anti-Ciprofloxacin at 25C for 30min
    Mix Anti-Ciprofloxacin with 100L of Stop Reagent3 for 60s

    Urine Reading = Measure the fluorescence of Anti-Ciprofloxacin for 5min
    Drain Anti-Ciprofloxacin
  }

} else {
  Split Epithelial = Mix UrineSample with Azoxymethane for 5s
  Move Split Epithelial to Anti-Fentanyl
  Move 100uL of Fentanyl-Conjugate to Anti-Fentanyl
  Tap Anti-Fentanyl for 60s
  Incubate Anti-Fentanyl at 23C for 60min
  Drain Anti-Fentanyl

  Repeat 6 times {
    Move 350uL of Distilled-Water to Anti-Fentanyl
    Vortex Anti-Fentanyl for 45s
    Drain Anti-Fentanyl
  }

  Move 100uL of TMB-Substrate to Anti-Fentanyl
  Incubate Anti-Fentanyl at 23C for 30min
  Mix Anti-Fentanyl with 100uL of Stop Reagent4 for 60s
  Urine Reading = Measure the fluorescence of Anti-Fentanyl for 30min
  Drain Anti-Fentanyl

  if (!(Heroin Reading >= FentanylControl)) {
    Move 20uL of Urine-Sample to Oxycodone-Enzyme
    Move 100uL of Oxycodone-Conjugate to Oxycodone-Enzyme
    Tap Oxycodone-Enzyme for 60s
    Incubate Oxycodone-Enzyme at 23C for 60min
    Drain Oxycodone-Enzyme

    Repeat 6 times {
      Move 350uL of Distilled-Water to Oxycodone-Enzyme
      Vortex Oxycodone-Enzyme for 45s
      Drain Oxycodone-Enzyme
    }

    Move 100uL of TMB-Substrate to Oxycodone-Enzyme
```

```
    Incubate Oxycodone-Enzyme at 23C for 30min
    Mix Oxycodone-Enzyme with 100uL of Stop Reagent5 for 60s

    Urine Reading = Measure the fluorescence of Oxycodone-Enzyme for 30min
    Drain Oxycodone-Enzyme
  }
}
```

7.3 PCR Droplet Replenishment

PCR Droplet Replenishment Assay: https://1drv.ms/v/s!AqwZF7jQGx_IhJRqfq1dLhbBUIL9RQ

BioScript Code:

```
PCRMix = Vortex 50 uL of PCRMasterMix with 50 uL of Template for 1s

Repeat 50 times {
  Heat PCRMix at 95C for 20s
  volumeWeight = Weigh PCRMix

  if (volumeWeight <= 50uL) {
    replacement = Vortex 25uL of PCRMasterMix with 25uL of Template for 5s
    Heat replacement at 95C for 45s
    PCRMix = Mix PCRMix with replacement for 5s
  }

  Heat PCRMix at 68C for 30s
  Heat PCRMix at 95C for 45s
}

Heat PCRMix at 68C for 5min
Save PCRMix
```

7.4 Probabilistic PCR

Probabilistic PCR Full: https://1drv.ms/v/s!AqwZF7jQGx_IhJRo-xDbVT_Q6UsXPg
Probabilistic PCR Early Exit: https://1drv.ms/v/s!AqwZF7jQGx_IhJRpLXveGI8Qr7BWrw

```
PCR-Master-Mix = Mix 50uL of PCRMix with 50uL of Buffer

# Initialization Step
Heat PCR-Master-Mix at 94C for 2min

# Denaturation
Repeat 20 times {
  Heat PCR-Master-Mix at 94C for 20s
  Heat PCR-Master-Mix at 50C for 40s
}
DNA Sensor = Measure the fluorescence of PCR-Master-Mix for 30s
if (DNA Sensor <= 85) {
  Drain PCR-Master-Mix
}
```

```
Repeat 20 times {
  Heat PCR-Master-Mix at 94C for 20s
  Heat PCR-Master-Mix at 50C for 40s
}
# Final Elongation
Heat PCR-Master-Mix at 70C for 5min

Save PCR-Master-Mix
```

7.5 ChemType Test1

```
Allyl Ethyl Ether = 50uL of C=CCOC1=CC=CC=C1

Cycloheptatriene = 50uL of C1C=CC=CC=C1

Cycloheptatriene2 = 50uL of C1C=CC=CC=C1

Sulfuric Acid = 50uL of OS(=O)(=O)O

Chlorotrifluoromethane = 50uL of C(F)(F)(F)Cl

Ammonium dichromate = 50uL of [NH4+].[NH4+].[O-][Cr](=O)(=O)O[Cr](=O)(=O)[O-]


a = Mix Allyl Ethyl Ether with Cycloheptatriene for 50s
b = Vortex a with Sulfuric Acid for 30s
c = Vortex b with Chlorotrifluoromethane for 45s
d = Mix Cycloheptatriene2 with Ammonium dichromate
```

7.6 ChemType Test2

```
Isopropyl-Alcohol = 50uL of CC(C)O

Isopropyl-Alcohol2 = 50uL of CC(C)O

Allyl-Ethyl-Ether = 50uL of C=CCOC1=CC=CC=C1

Cycloheptatriene = 50uL of C1C=CC=CC=C1

Cycloheptatriene2 = 50uL of C1C=CC=CC=C1

Cycloheptatriene3 = 50uL of C1C=CC=CC=C1

Sufluric-Acid = 50uL of OS(=O)(=O)O

Chlorotrifluoromethane = 50uL of C(F)(F)(F)Cl

Ammonium-dichromate = 50uL of [NH4+].[NH4+].[O-][Cr](=O)(=O)O[Cr](=O)(=O)[O-]
```

```
a = Mix Cycloheptatriene with Isopropyl-Alcohol for 30s
a2 = Mix Cycloheptatriene2 with Isopropyl-Alcohol2 for 30s

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

```
Sodium-Hydroxide = 50uL of [OH-].[Na+]

Isopropyl-Alcohol = 50uL of CC(C)O

Allyl-Ethyl-Ether = 50uL of C=CCOC1=CC=CC=C1

Cycloheptatriene = 50uL of C1C=CC=CC=C1

Cycloheptatriene2 = 50uL of C1C=CC=CC=C1

Sufluric-Acid = 50uL of OS(=O)(=O)O

Chlorotrifluoromethane = 50uL of C(F)(F)(F)Cl

Dibutyl-phosphite = 50uL of CCCCO[P+](=O)OCCCC

Ammonium-dichromate = 50uL of [NH4+].[NH4+].[O-][Cr](=O)(=O)O[Cr](=O)(=O)[O-]


a = Mix Sodium-Hydroxide with Chlorotrifluoromethane
b = Mix a with Isopropyl-Alcohol
c = Mix Allyl-Ethyl-Ether with Cycloheptatriene
Mix Cycloheptatriene2 with Dibutyl-phosphite
```

7.8 Broad Spectrum Opiate

```
Move 10uL of UrineSample to Anti-Morphine
Move 10uL of UrineSample to Anti-Oxy
Move 10uL of UrineSample to Anti-Fentanyl
Move 10uL of UrineSample to Anti-Ciprofloxcin
Move 10uL of UrineSample to Anti-Heroin

MorphineReading = Measure the fluorescence of Anti-Morphine for 5s
OxyReading = Measure the fluorescence of Anti-Oxy for 5s
FentanylReading = Measure the fluorescence of Anti-Fentanyl for 5s
CiproReading = Measure the fluorescence of Anti-Ciprofloxcin for 5s
HeroinReading = Measure the fluorescence of Anti-Heroin for 5s

Drain Anti-Morphine
```

```
Drain Anti-Oxy
Drain Anti-Fentanyl
Drain Anti-Ciprofloxcin
Drain Anti-Heroin
```

7.9 Cipro ELISA

```
Move 20uL of UrineSample to Ciprofloxacin-Enzyme
Move 100uL of Ciprofloxacin-Conjugate to Ciprofloxacin-Enzyme
Tap Ciprofloxacin-Enzyme for 60s
Incubate Ciprofloxacin-Enzyme at 23 C for 60min
Drain Ciprofloxacin-Enzyme

Repeat 5 times {
  Move 250uL of Distilled-Water to Ciprofloxacin-Enzyme
  Vortex Ciprofloxacin-Enzyme for 45s
  Drain Ciprofloxacin-Enzyme
}

Move 50uL of TMB-Substrate to Ciprofloxacin-Enzyme
Incubate Ciprofloxacin-Enzyme at 25 C for 30min
Mix Ciprofloxacin-Enzyme with 100uL of Stop Reagent for 60s

Urine Reading = Measure the fluorescence of Ciprofloxacin-Enzyme for 5min
Drain Ciprofloxacin-Enzyme
```

7.10 Diazepam ELISA

```
Move 50uL of UrineSample to Diazepam-Enzyme
Move 100uL of Diazepam-Antibody to Diazepam-Enzyme
Tap Diazepam-Enzyme for 60s
Incubate Diazepam-Enzyme at 23 C for 30min
Drain Diazepam-Enzyme

Repeat 3 times {
  Move 250uL of Distilled-Water to Diazepam-Enzyme
  Vortex Diazepam-Enzyme for 45s
  Drain Diazepam-Enzyme
}

Move 150uL of HRP-Conjugate to Diazepam-Enzyme
Incubate Diazepam-Enzyme at 23 C for 15min
Drain Diazepam-Enzyme

Repeat 3 times {
  Move 250uL of Distilled-Water to Diazepam-Enzyme
  Vortex Diazepam-Enzyme for 45s
  Drain Diazepam-Enzyme
}
```

```
Move 100uL of TMB-Substrate to Diazepam-Enzyme
Incubate Diazepam-Enzyme at 23 C for 15min
Mix Diazepam-Enzyme with 100uL of Stop Reagent for 60s
Negative Reading = Measure the fluorescence of Diazepam-Enzyme for 30min
Drain Diazepam-Enzyme
```

7.11 547 Dilution

```
First-Dilute = Mix Substance A with Dilutant1
Split First-Dilute into 2 parts:
  1 part to First-Dilute
  1 part to Waste-Droplet
Drain Waste-Droplet

Second-Dilute = Mix First-Dilute with Dilutant2
Split Second-Dilute into 2 parts:
  1 part to Second-Dilute
  1 part to Waste-Droplet
Drain Waste-Droplet

Third Dilute = Mix Second-Dilute with Dilutant3
Split Third-Dilute into 2 parts:
  1 part to Third-Dilute
  1 part to Waste-Droplet
Drain Waste-Droplet

Fourth-Dilute = Mix Substance B with Substance C
Split Fourth-Dilute into 2 parts:
  1 part to Fourth-Dilute
  1 part to Waste-Droplet
Drain Waste-Droplet

Final-Dilute = Mix Third-Dilute with Fourth-Dilute
Split Final-Dilute into 2 parts:
  1 part to Final-Dilute
  1 part to Waste-Droplet
Drain Waste-Droplet

Save Final-Dilute
```

7.12 Fentanyl ELISA

```
# Step 1 Add Urine-Sample to test against
Antigen1 = Move 20uL of Urine-Sample to Antigen

# Step 2 Add Enzyme Conjugate
Antigen2 = Move 100uL of Fentanyl-Conjugate to Antigen1

# Step 3 Ensure fully mixed interactions
```

```
Tap Antigen2 for 60s

# Step 4
Incubate Antigen2 at 23 C for 60min

# Step 5
Drain Antigen2
Repeat 6 times {
  Move 350uL of Distilled-Water to Antigen1

  Vortex Antigen1 for 45s

  Drain Antigen1
}
# Step 6
Antigen1 = Move 100uL of TMB-Substrate to Antigen1

# Step 7
Incubate Antigen1 at 23 C for 30min

# Step 8
Mix Antigen1 with 100uL of Stop-Reagent for 60s

# Step 9
Negative-Reading = Measure the fluorescence of Antigen1 for 30min

# End
Drain Antigen1
```

7.13 Morphine ELISA

```
Stationary Morphine-Enzyme

Move 20uL of Urine-Sample to Morphine-Enzyme
Move 100uL of Morphine-Conjugate to Morphine-Enzyme
Tap Morphine-Enzyme for 60s
Incubate Morphine-Enzyme at 23 C for 60min

Repeat 6 times {
  Move 350uL of Distilled-Water to Morphine-Enzyme
  Vortex Morphine-Enzyme for 45s
}

Move 100uL of TMB-Substrate to Morphine-Enzyme
Incubate Morphine-Enzyme at 23 C for 30min
Mix Morphine-Enzyme with 100uL of Stop Reagent for 60s

Urine Reading = Measure the fluorescence of Morphine-Enzyme for 30min
```

7.14 Morphine ELISA with Control Samples

```
# Step 2
Move 20uL of Negative-Standard to Antigen1
Move 20uL of Positive-Standard to Antigen2
Move 20uL of Diluted-Sample to Antigen3

# Step 3
Move 100uL of Morphine-Conjugate to Antigen1
Move 100uL of Morphine-Conjugate to Antigen2
Move 100uL of Morphine-Conjugate to Antigen3

# Step 4
Tap Antigen1 for 60s
Tap Antigen2 for 60s
Tap Antigen3 for 60s

# Step 5
Incubate Antigen1 at 23 C for 60min
Incubate Antigen2 at 23 C for 60min
Incubate Antigen3 at 23 C for 60min

# Step 6
Drain Antigen1
Drain Antigen2
Drain Antigen3
Repeat 6 times {
  Move 350uL of Distilled-Water to Antigen1
  Move 350uL of Distilled-Water to Antigen2
  Move 350uL of Distilled-Water to Antigen3

  Vortex Antigen1 for 45s
  Vortex Antigen2 for 45s
  Vortex Antigen3 for 45s

  Drain Antigen1
  Drain Antigen2
  Drain Antigen3

}
# Step 7 Dry...

# Step 8
Move 100uL of TMB-Substrate to Antigen1
Move 100uL of TMB-Substrate to Antigen2
Move 100uL of TMB-Substrate to Antigen3

# Step 9
```

```
Incubate Antigen1 at 23 C for 30min
Incubate Antigen2 at 23 C for 30min
Incubate Antigen3 at 23 C for 30min

# Step 10
Mix Antigen1 with 100uL of Stop Reagent for 60s
Mix Antigen2 with 100uL of Stop Reagent for 60s
Mix Antigen3 with 100uL of Stop Reagent for 60s

# Step 11
Negative Reading = Measure the fluorescence of Antigen1 for 30min
Positive Reading = Measure the fluorescence of Antigen2 for 30min
Sample Reading = Measure the fluorescence of Antigen3 for 30min

# End
Drain Antigen1
Drain Antigen2
Drain Antigen3
```

7.15 Heroin ELISA

```
Move 20uL of Urine-Sample to Heroin-Enzyme
Move 100uL of Heroin-Conjugate to Heroin-Enzyme
Tap Heroin-Enzyme for 60s
Incubate Heroin-Enzyme at 23 C for 60min
Drain Heroin-Enzyme

Repeat 6 times {
  Move 350uL of Distilled-Water to Heroin-Enzyme
  Vortex Heroin-Enzyme for 45s
  Drain Heroin-Enzyme
}

Move 100uL of TMB-Substrate to Heroin-Enzyme
Incubate Heroin-Enzyme at 23 C for 30min
Mix Heroin-Enzyme with 100uL of Stop Reagent for 60s

Urine-Reading = Measure the fluorescence of Heroin-Enzyme for 30min
Drain Heroin-Enzyme
```

7.16 Oxycodone ELISA

```
Move 20uL of Urine-Sample to Oxycodone-Enzyme
Move 100uL of Oxycodone-Conjugate to Oxycodone-Enzyme
Tap Oxycodone-Enzyme for 60s
Incubate Oxycodone-Enzyme at 23 C for 60min
Drain Oxycodone-Enzyme

Repeat 6 times {
  Move 350uL of Distilled-Water to Oxycodone-Enzyme
```

```
  Vortex Oxycodone-Enzyme for 45s
  Drain Oxycodone-Enzyme
}

Move 100uL of TMB-Substrate to Oxycodone-Enzyme
Incubate Oxycodone-Enzyme at 23 C for 30min
Mix Oxycodone-Enzyme with 100uL of Stop Reagent for 60s

Urine Reading = Measure the fluorescence of Oxycodone-Enzyme for 30min
Drain Oxycodone-Enzyme
```

7.17 Image Probe Synthesis

```
Mixture = Mix 10uL of Ion exchange beads with 10uL of Fluoride ions F- for 30s

Heat Mixture at 100 C for 30s
Heat Mixture at 120 C for 30s
Heat Mixture at 135 C for 3min

Mixture = Mix Mixture with 10uL of MeCN solution for 30s

Incubate Mixture at 100 C for 30s
Incubate Mixture at 120 C for 50s

Mixture = Mix Mixture with 10uL of HCl for 60s
Heat Mixture at 60 C for 60s
```

7.18 Glucose Detection

```
Result1 = Mix 10uL of Glucose with 10uL of Reagent for 10s
Reading1 = Measure the fluorescence of Result1 for 30s

Result2 = Mix 10uL of Glucose with 20uL of Reagent for 10s
Reading2 = Measure the fluorescence of Result2 for 30s

Result3 = Mix 10uL of Glucose with 40uL of Reagent for 10s
Reading3 = Measure the fluorescence of Result3 for 30s

Result4 = Mix 10uL of Glucose with 80uL of Reagent for 10s
Reading4 = Measure the fluorescence of Result4 for 30s

Result5 = Mix 10uL of Sample with 10uL of Reagent for 10s
Reading5 = Measure the fluorescence of Result5 for 30s
```

7.19 PCR

```
PCR-Mixture = 10uL of PCR

Heat PCR-Mixture at 95 C for 5s

Repeat 20 times {
```

```
  Heat PCR-Mixture at 53 C for 15s
  Heat PCR-Mixture at 72 C for 10s
}
Perform Capillary Electrophoresis ( 5 cm at 236 V/cm) on PCR-Mixture Separate with Separati

Measure the fluorescence of PCR-Mixture for 3min
```

7.20 Neurotransmitter Sensing

```
Mixture = Mix Sample with Reagent for 50s
Perform Capillary Electrophoresis ( 9 cm at 223 V/cm) on Mixture Seperate with electrophore
Measure the fluorescence of Mixture for 10s
```

8 AQUACORE ASSAYS

8.1 Glucose Detection

```
Input port ip1 ;Standard glucose
Input port ip2 ;Reagent
Input port ip3 ;Sample
RESULT[1..5] ; dry array for final results
glucose-detection {
input s1, ip1
input s2, ip2
input s3, ip3
move mixer1, s1, 1;5s
move mixer1, s2, 1;5s
mix mixer1, 10;10s
move sensor1, mixer1;5s
sense.OD sensor1, RESULT[1] ;30s

move mixer1, s1, 1 ;5s
move mixer1, s2, 2;5s
mix mixer1, 10;10s
move sensor1, mixer1;5s
sense.OD sensor1, RESULT[2] ;30s

move mixer1, s1, 1 ;5s
move mixer1, s2, 4 ;5s
mix mixer1, 10 ;10s
move sensor1, mixer1 ;5s
sense.OD sensor1, RESULT[3] ;30s

move mixer1, s1, 1 ;5s
move mixer1, s2, 8 ;5s
mix mixer1, 10 ;10s
move sensor1, mixer1 ;5s
sense.OD sensor1, RESULT[4] ;30s

<dry routine to get best line fit for RESULT[1..4]> move mixer1, s3, 1 ;5s
```

```
move mixer1, s2, 1 ;5s
mix mixer1, 10 ;10s
move sensor1, mixer1 ;5s
sense.OD sensor1, RESULT[5] ;30s

<dry routine to get concentration from line given RESULT[5]>
}
Total time = 275s
#reservoirs = 3
ASLoC
```

8.2 PCR

```
Input port ip1 ;PCR mixture
Input port ip2 ;CE separation medium

RESULT[] ;dry array for final results

PCR {
input s1, ip1
input s2, ip2

move heater1, s1 ;5s
dry-mov r1, 20
dry-label loop:
incubate heater1, 95, 5 ;6s
incubate heater1, 53, 15 ;17s
incubate heater1, 72, 10 ;12s
dry-dec r1
dry-bgt loop

move separator1.buf, s2 ;5s
move separator1, heater1 ;5s

separate.CE separator1, 236, 5, 180
sense.FL sensor1, RESULT ;180s
}
Total time = 895s
#reservoirs = 2
ASLoC
```

8.3 Imaging Probe Synthesis

```
Input port ip1 ;Ion exchange beads (in buffer)
Input port ip2 ;Fluoride ions F-
Input port ip3 ;MeCN solution
Input port ip4 ;HCl

Imaging-Probe-Synthesis {
input s1, ip1
```

```
input s2, ip2
input s3, ip3
input s4, ip4

move mixer1, s1
move mixer1, s2
mix mixer1, 30

move heater1, mixer1
concentrate.EV heater1, 100, 30
concentrate.EV heater1, 120, 30
concentrate.EV heater1, 135, 180

move mixer1, s3
move mixer1, heater1
mix mixer1, 30

move heater1, mixer1
incubate heater1, 100, 30
incubate heater1, 120, 50
move mixer1, heater1

move mixer1, s4
mix mixer1, 60
move heater1, mixer1
concentrate.EV heater1, 60, 60
}
Total time = 548s #reservoirs = 4
ASLoC area = 13mm x 9mm = 117 mm2
```

8.4 Neurotransmitter Sensing

```
Input port ip1 ;Sample
Input port ip2 ;Reagent(OPA)
Input port ip3 ;Elecrtrophoresis buffer

RESULT[] ;dry array for final results

neurotransmitter-sensing {
input s1, ip1
input s2, ip2
input s3, ip3

move mixer1, s1 ;5s
move mixer1, s2 ;5s
mix mixer1, 50 ;50s

move separator1.buf, s3 ;5s
move separator1, mixer1 ;5s
```

```
separate.CE separator1, 223, 9, 22
sense.FL sensor1, RESULT ;22s

}
Total time = 92s #reservoirs = 3
ASLoC area = 2.5cm x 1.5cm = 3.75cm2 + CE column
FIGURE 3: Separations-based sensing of neurotransmitters (immunoassay, biochemistry): (a)
tio
```

9 BIOCODER ASSAYS

9.1 PCR

```
void PCR(){
BioSystem bioCoder;

Fluid *PCRMix = bioCoder.new_fluid("PCRMasterMix", Volume(MICRO_LITER,10));
Fluid *SeperationMedium = bioCoder.new_fluid("Seperation Medium",Volume(MICRO_LITER,10));
Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);

bioCoder.first_step();
bioCoder.measure_fluid(PCRMix,tube);

bioCoder.next_step();
bioCoder.incubate(tube,95,Time(5,SECS));

bioCoder.next_step();
bioCoder.LOOP(20);

bioCoder.next_step();
bioCoder.incubate(tube,53,Time(15,SECS));

bioCoder.next_step();
bioCoder.incubate(tube,72,Time(10,SECS));
bioCoder.END_LOOP();

bioCoder.next_step();
bioCoder.ce_detect(tube,5,236,SeperationMedium);

bioCoder.next_step();
bioCoder.measure_fluorescence(tube,Time(3,MINS));

bioCoder.next_step();
bioCoder.end_protocol();
}
```

9.2 Probabilistic PCR

```
void ProbablisticPCR()
```

```
{
BioSystem bioCoder;

Fluid *PCRMix = bioCoder.new_fluid("PCRMasterMix", Volume(MICRO_LITER,10));

Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);

bioCoder.first_step();
bioCoder.measure_fluid(PCRMix,tube);

for(int i = 0 ; i < initial; ++i) {
bioCoder.next_step();
bioCoder.store_for(tube,94,Time(SECS,45));

bioCoder.next_step();
bioCoder.store_for(tube,65,Time(SECS,45));
}

for(int i = initial; i <= Threshold; ++i) {
std::cout <<i<<std::endl;
bioCoder.next_step();
bioCoder.store_for(tube,94,Time(SECS,45));

bioCoder.next_step();
bioCoder.store_for(tube,65,Time(SECS,45));

bioCoder.next_step();
bioCoder.measure_fluorescence(tube,Time(SECS,5),"DNASensor");

bioCoder.IF("DNASensor",GREATER_THAN, .85);
for(int j = i; j < Total+(Threshold-i); ++j) {
bioCoder.next_step();
bioCoder.store_for(tube,94,Time(SECS,45));

bioCoder.next_step();
bioCoder.store_for(tube,65,Time(SECS,45));
}

bioCoder.next_step();
bioCoder.drain(tube,"Amplified PCR");
bioCoder.END_IF();

}

bioCoder.drain(tube,"waste");
bioCoder.end_protocol();

bioCoder.PrintLeveledProtocol();
```

```
bioCoder.PrintTree();
bioCoder.PrintTreeVisualization("ProbablisticPCR");
}
```

9.3 PCR Droplet Replenish

```
void PCRDropletReplacement()
{
int TotalThermo = 9;
BioSystem bioCoder;

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);
//Container* tube2 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);

bioCoder.first_step();
bioCoder.measure_fluid(PCRMix,tube);

bioCoder.next_step();
bioCoder.vortex(tube,Time(SECS,1));
bioCoder.measure_fluid(Template,tube);

bioCoder.next_step();
bioCoder.vortex(tube, Time(SECS,1));

bioCoder.next_step();
bioCoder.store_for(tube,95,Time(SECS,45));

bioCoder.next_step();
bioCoder.LOOP(TotalThermo);

std::cout<<"Debug statement2"<<std::endl;
bioCoder.next_step();
bioCoder.store_for(tube,95,Time(SECS,20));

bioCoder.next_step();
bioCoder.weigh(tube,"weightSensor");

bioCoder.next_step();
bioCoder.IF("WieghtSensor",LESS_THAN, 3.57);
bioCoder.next_step();
bioCoder.measure_fluid(PCRMix, tube);

bioCoder.next_step();
bioCoder.store_for(tube, 95,Time(SECS,45));

bioCoder.next_step();
```

```
bioCoder.vortex(tube, Time(SECS,1));
bioCoder.END_IF();

bioCoder.next_step();
bioCoder.store_for(tube,50,Time(SECS,30));

bioCoder.next_step();
bioCoder.store_for(tube,68,Time(SECS,45));
std::cout<<"Debug statement3"<<std::endl;
bioCoder.END_LOOP();
std::cout<<"Debug statement4"<<std::endl;


bioCoder.next_step();
bioCoder.store_for(tube,68,Time(MINS,5));
std::cout<<"Debug statement5"<<std::endl;
bioCoder.next_step();
bioCoder.drain(tube,"PCR");
std::cout<<"Debug statement6"<<std::endl;
bioCoder.end_protocol();


std::cout<<"Debug statemen7"<<std::endl;
bioCoder.PrintLeveledProtocol();
bioCoder.PrintTree();
bioCoder.PrintTreeVisualization("PCRReplacement");

}
```

9.4 Glucose Detection

```
void GlucoseDetection(){
BioSystem bioCoder;

Fluid *Glucose = bioCoder.new_fluid("Ion exchange beads", Volume(MICRO_LITER,160));
Fluid *Reagent = bioCoder.new_fluid("Fluoride ions",Volume(MICRO_LITER,50));
Fluid *Sample = bioCoder.new_fluid("HCL",Volume(MICRO_LITER,10));

Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
Container* tube2 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
Container* tube3 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
Container* tube4 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);
Container* tube5 = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);

bioCoder.first_step();
bioCoder.measure_fluid(Glucose,Volume(MICRO_LITER,10),tube);
bioCoder.measure_fluid(Glucose,Volume(MICRO_LITER,10),tube2);
bioCoder.measure_fluid(Glucose,Volume(MICRO_LITER,10),tube3);
bioCoder.measure_fluid(Glucose,Volume(MICRO_LITER,10),tube4);
```

```
bioCoder.measure_fluid(Glucose,Volume(MICRO_LITER,10),tube5);

bioCoder.measure_fluid(Reagent,Volume(MICRO_LITER,10),tube);
bioCoder.measure_fluid(Reagent,Volume(MICRO_LITER,20),tube2);
bioCoder.measure_fluid(Reagent,Volume(MICRO_LITER,40),tube3);
bioCoder.measure_fluid(Reagent,Volume(MICRO_LITER,80),tube4);

bioCoder.measure_fluid(Sample,Volume(MICRO_LITER,80),tube5);

bioCoder.next_step();
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));
bioCoder.measure_fluorescence(tube5,Time(5,SECS));

bioCoder.next_step();
bioCoder.end_protocol();
}
```

9.5 Image Probe Synthesis

```
void ImageProbSynthesis(){
BioSystem bioCoder;

Fluid *IonBeads = bioCoder.new_fluid("Ion exchange beads", Volume(MICRO_LITER,10));
Fluid *Fluoride = bioCoder.new_fluid("Fluoride ions",Volume(MICRO_LITER,10));
Fluid *HCL = bioCoder.new_fluid("HCL",Volume(MICRO_LITER,10));
Fluid *MeCNSolution = bioCoder.new_fluid("MeCN solution",Volume(MICRO_LITER,10));

Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);

bioCoder.first_step();
bioCoder.measure_fluid(IonBeads,tube);
bioCoder.measure_fluid(Fluoride,tube);

bioCoder.next_step();
bioCoder.vortex(tube,Time(30,SECS));

bioCoder.next_step();
bioCoder.incubate(tube,100,Time(30,SECS));

bioCoder.next_step();
bioCoder.incubate(tube,120,Time(30,SECS));

bioCoder.next_step();
bioCoder.incubate(tube,135, Time(3,MINS));

bioCoder.next_step();
```

```
bioCoder.measure_fluid(MeCNSolution,tube);

bioCoder.next_step();
bioCoder.vortex(tube,Time(30,SECS));

bioCoder.next_step();
bioCoder.incubate(tube,100,Time(30,SECS));

bioCoder.next_step();
bioCoder.incubate(tube,120,Time(50,SECS));

bioCoder.next_step();
bioCoder.measure_fluid(HCL,tube);

bioCoder.next_step();
bioCoder.vortex(tube,Time(60,SECS));

bioCoder.next_step();
bioCoder.incubate(tube,60,Time(60,SECS));

bioCoder.next_step();
bioCoder.end_protocol();
}
```

9.6 Neurotransmitter Sensing

```
void neurotransmitterSensing(){
BioSystem bioCoder;

Fluid *Sample = bioCoder.new_fluid("Sample", Volume(MICRO_LITER,10));
Fluid *Reagent = bioCoder.new_fluid("Reagent",Volume(MICRO_LITER,10));
Fluid *SeperationMedium = bioCoder.new_fluid("Seperation Medium",Volume(MICRO_LITER,10));

Container* tube = bioCoder.new_container(STERILE_MICROFUGE_TUBE2ML);

bioCoder.first_step();
bioCoder.measure_fluid(Sample,tube);
bioCoder.measure_fluid(Reagent,tube);

bioCoder.next_step();
bioCoder.vortex(tube,Time(50,SECS));

bioCoder.next_step();
bioCoder.ce_detect(tube,9,223,SeperationMedium);

bioCoder.next_step();
bioCoder.measure_fluorescence(tube,Time(10,SECS));

bioCoder.next_step();
```

```
bioCoder.end_protocol();
}
```

10 ANTHA ASSAYS

10.1 Glucose Detection

```
protocol Glucose_Detection


import (
"github.com/antha-lang/antha/antha/anthalib/wtype"
    // LHComponent type
"github.com/antha-lang/antha/antha/anthalib/wutil"
"github.com/antha-lang/antha/antha/anthalib/mixer"
    //sample function is imported from mixed

// Input parameters
Parameters (
Reagent_volume1 Volume  // 10ul
Reagent_volume2 Volume  // 20ul
Reagent_volume3 Volume  // 40ul
Reagent_volume4 Volume  // 80ul
Glucose_volume Volume   // 10ul
Sample_volume Volume    // 10ul
)


// Data which is returned from this protocol, and data types
Data (

)

// Physical Inputs to this protocol with types
Inputs (
Glucose *wtype.LHComponent
Reagent *wtype.LHComponent
Sample *wtype.LHComponent
)



// Physical outputs from this protocol with types
Outputs (
Result1 *wtype.LHComponent
Result2 *wtype.LHComponent
Result3 *wtype.LHComponent
Result4 *wtype.LHComponent
Result5 *wtype.LHComponent
```

```
)


Requirements {
}

// Conditions to run on startup
Setup {

}

// The core process for this protocol, with the steps to be performed
// for every input
Steps {

glucose := mixer.Sample(Glucose, Glucose_volume)
reagent := mixer.Sample(Reagent, Reagent_volume1)

Result1 = mixer.Mix(glucose, reagent) // cannot specify duration of mixture
//cannot measure fluorescence to get a reading

glucose := mixer.Sample(Glucose, Glucose_volume)  //get new sample of Glucose
reagent := mixer.Sample(Reagent, Reagent_volume2) //get new sample of Reagent

Result2 = mixer.Mix(glucose, reagent) // mix new samples
// would measure fluorescence here

glucose := mixer.Sample(Glucose, Glucose_volume)  //get new sample of Glucose
reagent := mixer.Sample(Reagent, Reagent_volume3) //get new sample of Reagent

Result3 = mixer.Mix(glucose, reagent) // mix new samples
// would measure fluorescence here

glucose := mixer.Sample(Glucose, Glucose_volume)  //get new sample of Glucose
reagent := mixer.Sample(Reagent, Reagent_volume4) //get new sample of Reagent

Result4 = mixer.Mix(glucose, reagent) // mix new samples
// would measure fluorescence here

sample := mixer.Sample(Sample, Sample_volume)  //get new sample of Sample
reagent := mixer.Sample(Reagent, Reagent_volume1) //get new sample of Reagent

Result5 = mixer.Mix(sample, reagent) // mix new samples
// would measure fluorescence here

}

Analysis {

```

```
}

Validation {
}
```

10.2 Imaging Probe Synthesis

```
protocol Imaging_Probe_Synthesis


import (
"github.com/antha-lang/antha/antha/anthalib/wtype" // LHComponent type
"github.com/antha-lang/antha/antha/anthalib/wutil"
"github.com/antha-lang/antha/antha/anthalib/mixer"
    //sample function is imported from mixed

// Input parameters
Parameters (
Ion_volume Volume
Flouride_volume Volume
MeCN_volume Volume
HCl_volume Volume
)


// Data which is returned from this protocol, and data types
Data (

)

// Physical Inputs to this protocol with types
Inputs (
Ion_exchange_beads *wtype.LHComponent
Fluoride_ions_F- *wtype.LHComponent
MeCN *wtype.LHComponent
HCl *wtype.LHComponent
)


// Physical outputs from this protocol with types
Outputs (
Mixture *wtype.LHComponent
)

Requirements {
}

// Conditions to run on startup
```

```
Setup {

}

// The core process for this protocol, with the steps to be performed
// for every input
Steps {

ibv := mixer.Sample(Ion_exchange_beads, Ion_volume)
fif := mixer.Sample(Flourid_ions_F-, Flouride_volume)

mixture := mixer.Mix(ibv, fif) //cannot specify time for mixture

mixture = Incubate(mixture, 100, 30, false) //heat 100 for 30s
mixture = Incubate(mixture, 120, 30, false) //heat 120 for 30s
mixture = Incubate(mixture, 135, 180, false) //heat 135 for 3 minutes

mecn := mixer.Sample(MeCN, MeCN_volume)

mixture = mixer.Mix(mixture, mecn)

mixture = Incubate(mixture, 100, 30, false) //incubate 100 for 30s
mixture = Incubate(mixture, 120, 50, false) //incubuate 120 for 50s

hcl := mixer.Sample(HCl, HCl_volume)

mixture = mixer.Mix(mixture, hcl)

mixture = Incubate(mixture, 60, 60, false) // heat 60 for 60s

Mixture = mixture
}

Analysis {

}

Validation {
}
```

10.3 PCR

```
protocol PCR

import (
"github.com/antha-lang/antha/antha/anthalib/wtype"
"github.com/antha-lang/antha/antha/anthalib/mixer"
"github.com/antha-lang/antha/antha/AnthaStandardLibrary/Packages/enzymes"
"fmt"
```

```
"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"
)

/*type Polymerase struct {
wtype.LHComponent
Rate_BPpers float64
Fidelity_errorrate float64 // could dictate how many colonies are checked in validation!
Extensiontemp Temperature
Hotstart bool
StockConcentration Concentration // this is normally in U?
TargetConcentration Concentration
// this is also a glycerol solution rather than a watersolution!
}
*/


// Input parameters for this protocol (data)
Parameters (
// PCRprep parameters:
ReactionVolume Volume
FwdPrimerConc Concentration
RevPrimerConc Concentration
Additiveconc Concentration
TargetpolymeraseConcentration Concentration
Templatevolume Volume
DNTPconc Concentration
/*
// let's be ambitious and try this as part of type polymerase Polymeraseconc Volume

//Templatetype string  // e.g. colony, genomic, pure plasmid...
    //will effect efficiency. We could get more sophisticated here later on...
//FullTemplatesequence string
    // better to use Sid's type system here after proof of concept
//FullTemplatelength int
    // clearly could be calculated from the sequence...
    //Sid will have a method to do this already so check!
//TargetTemplatesequence string
    // better to use Sid's type system here after proof of concept
//TargetTemplatelengthinBP int
*/
// Reaction parameters: (could be a entered as thermocycle parameters type possibly?)
Numberofcycles int
InitDenaturationtime Time
Denaturationtime Time
//Denaturationtemp Temperature
Annealingtime Time
```

```
AnnealingTemp Temperature // Should be calculated from primer and template binding
Extensiontime Time // should be calculated from template length and polymerase rate
Extensiontemp Temperature
Finalextensiontime Time
Targetsequence string
FwdPrimerSeq string
RevPrimerSeq string
)

// Data which is returned from this protocol, and data types
Data (
FwdPrimerSites []search.Thingfound
RevPrimerSites []search.Thingfound
)


// Physical Inputs to this protocol with types
Inputs (
FwdPrimer *wtype.LHComponent
RevPrimer *wtype.LHComponent
DNTPS *wtype.LHComponent
PCRPolymerase *wtype.LHComponent
Buffer *wtype.LHComponent
Template *wtype.LHComponent
Additives []*wtype.LHComponent // e.g. DMSO
OutPlate *wtype.LHPlate
)

// Physical outputs from this protocol with types
Outputs (
Reaction *wtype.LHComponent
)

Requirements {
}

// Conditions to run on startup
Setup {
}

// The core process for this protocol, with the steps to be performed
// for every input
Steps {

FwdPrimerSites = sequences.FindSeqsinSeqs(Targetsequence, []string{FwdPrimerSeq})

RevPrimerSites = sequences.FindSeqsinSeqs(Targetsequence, []string{RevPrimerSeq})
```

```
if len(FwdPrimerSites)==0 || len(RevPrimerSites)==0{

errordescription := fmt.Sprint(
text.Print("FwdPrimerSitesfound:", fmt.Sprint(FwdPrimerSites)),
text.Print("RevPrimerSitesfound:", fmt.Sprint(RevPrimerSites)),
)

Errorf(errordescription)
}



// Mix components
samples := make([]*wtype.LHComponent, 0)
bufferSample := mixer.SampleForTotalVolume(Buffer, ReactionVolume)
samples = append(samples, bufferSample)
templateSample := mixer.Sample(Template, Templatevolume)
samples = append(samples, templateSample)
dntpSample := mixer.SampleForConcentration(DNTPS, DNTPconc)
samples = append(samples, dntpSample)
FwdPrimerSample := mixer.SampleForConcentration(FwdPrimer, FwdPrimerConc)
samples = append(samples, FwdPrimerSample)
RevPrimerSample := mixer.SampleForConcentration(RevPrimer, RevPrimerConc)
samples = append(samples, RevPrimerSample)

for _, additive := range Additives {
additiveSample := mixer.SampleForConcentration(additive, Additiveconc)
samples = append(samples, additiveSample)
}

polySample := mixer.SampleForConcentration(PCRPolymerase, TargetpolymeraseConcentration)
samples = append(samples, polySample)
reaction := MixInto(OutPlate, "", samples...)

// thermocycle parameters called from enzyme lookup:

polymerase := PCRPolymerase.CName

extensionTemp := enzymes.DNApolymerasetemps[polymerase]["extensiontemp"]
meltingTemp := enzymes.DNApolymerasetemps[polymerase]["meltingtemp"]


// initial Denaturation

r1 := Incubate(reaction, meltingTemp, InitDenaturationtime, false)

for i:=0; i < Numberofcycles; i++ {

```

```
// Denature

r1 = Incubate(r1, meltingTemp, Denaturationtime, false)

// Anneal
r1 = Incubate(r1, AnnealingTemp, Annealingtime, false)

//extensiontime := TargetTemplatelengthinBP/PCRPolymerase.RateBPpers
    // we'll get type issues here so leave it out for now

// Extend
r1 = Incubate(r1, extensionTemp, Extensiontime, false)

}
// Final Extension
r1 = Incubate(r1, extensionTemp, Finalextensiontime, false)


// all done
Reaction = r1
}

// Run after controls and a steps block are completed to
// post process any data and provide downstream results
Analysis {
}

// A block of tests to perform to validate that the sample was processed correctly
// Optionally, destructive tests can be performed to validate results on a
// dipstick basis
Validation {
}
```

10.4 Neurotransmitter Sensing

```
protocol Neurotransmitter_Sensing


import (
"github.com/antha-lang/antha/antha/anthalib/wtype" // LHComponent type
"github.com/antha-lang/antha/antha/anthalib/wutil"
"github.com/antha-lang/antha/antha/anthalib/mixer" //sample function is imported from mixe

// Input parameters
Parameters (
Sample_volume //no val specified
Reagent_volume //no val specified
```

```
electrophoresis_buffer_volume //no val specified
)


// Data which is returned from this protocol, and data types
Data (

)

// Physical Inputs to this protocol with types
Inputs (
Sample *wtype.LHComponent
Reagent *wtype.LHComponent
electrophoresis_buffer *wtype.LHComponent
)



// Physical outputs from this protocol with types
Outputs (
Mixture *wtype.LHComponent
)


Requirements {
}

// Conditions to run on startup
Setup {

}

// The core process for this protocol, with the steps to be performed
// for every input
Steps {

sample := mixer.Sample(Sample, Sample_volume)
reagent := mixer.Sample(Reagent, Reagent_volume)

Mixture = mixer.Mix(sample, reagent) // cannot specify duration of mixture
//cannot perform capillary electrophoresis
//cannot measure fluorescence to get a reading
}

Analysis {

}
```

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.

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

```
Validation {
}
```

11 CHEMTYPE RESULTS

11.1 Efficacy of ChemType

11.2 ChemType Runtime Evaluation

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

Benchmark	Compilation Time (sec)	Constraint Solving Time (sec)	Gathered Constraints
AIHA 1 [2]	0.012	0.936	70
AIHA 2 [2]	0.012	1.648	68
AIHA 3 [2]	0.014	1.214	17
Broad Spectrum Opiate [4, 24, 35]	0.011	0.887	11
Ciprofloxacin [24]	0.023	1.722	14
Diazepam [20]	0.024	1.007	14
Dilution [20]	0.014	0.892	9
Fentanyl [35]	0.018	0.900	13
Full Morphine [20]	0.048	4.188	19
Glucose Detection [3]	0.012	1.633	14
Heroine [20]	0.020	1.553	13
Image Probe Synthesis [3]	0.015	2.181	13
Morphine [20]	0.018	1.026	13
Mustard Gas [1]	0.015	1.433	83
Neurotransmitter Sensing [3]	0.029	3.845	6
Oxycodone [4]	0.026	0.959	13
PCR [3]	0.032	3.534	8
Safety Zone [12]	0.013	1.341	76
Synthetic 1	0.031	1.507	196
Synthetic 2	0.014	1.505	274
Synthetic 3	0.012	0.944	95
Synthetic 4	0.014	0.623	1
Synthetic 5	0.026	1.350	10
Synthetic 6	0.013	0.855	8

REFERENCES

[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

[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

[17] 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

[26] 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

[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=IkqVNwAACAAJ

[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.

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.

[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.2011.11.006

[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

[47] Yong Jun Shin and Jeong Bong Lee. 2010. Machine vision for digital microfluidics. *Review of Scientific Instruments* 81, 1 (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.

[52] 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

[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