Input-Output Disjointness for Forward Expressions in the Logic of Information Flows

Heba Aamer
a joint work with **Jan Van den Bussche**

Hasselt University , Belgium

SmP Workshop, 27^{th} of July 2021

Logic of Information Flows

Logic of Information Flows (LIF) [Ternovska, FroCoS2019]:

Pupose: model how information propagates in complex systems (complex systems = **modules** connected together)

Syntax: algebraic (algebraization of FOL)

Semantics: dynamic

Logic of Information Flows

Logic of Information Flows (LIF) [Ternovska, FroCoS2019]:

Pupose: model how information propagates in complex systems (complex systems = **modules** connected together)

Syntax: algebraic (algebraization of FOL)

Semantics: dynamic

Expressiveness of the logic depends on

1. the set of operations in the language
2. the logic expressing the atomic modules

Module is a relation with input arguments and output arguments.

Dynamic Semantics and the "Law of Inertia"

Consider a binary relation *Increment*

- 1st argument: **input**
- 2nd argument: **output**
- Example: $Increment(x; y)$

Dynamic Semantics and the "Law of Inertia"

Consider a binary relation *Increment*

- 1st argument: **input**
- 2nd argument: **output**
- Example: $\mathit{Increment}(x; y)$

Standard (static) semantics, valuation ν:

$$\nu \models \mathit{Increment}(x; y) \quad \Leftrightarrow \quad \nu(y) = \nu(x) + 1$$

Dynamic Semantics and the "Law of Inertia"

Consider a binary relation *Increment*

- 1st argument: **input**
- 2nd argument: **output**
- Example: $\mathit{Increment}(x; y)$

Standard (static) semantics, valuation ν:

$$\nu \models \mathit{Increment}(x; y) \quad \Leftrightarrow \quad \nu(y) = \nu(x) + 1$$

Dynamic semantics, pair of valuations (ν_1, ν_2):

$$(\nu_1, \nu_2) \models \mathit{Increment}(x; y) \quad \Leftrightarrow \quad \nu_2(y) = \nu_1(x) + 1$$

$$\boxed{\text{and } \nu_2 = \nu_1 \text{ elsewhere}}$$

Facebook Example

Consider a database D with a binary relation:

Friends

alice	bob
alice	carol
carol	dave

All pairs (ν_1, ν_2) such that $D, (\nu_1, \nu_2) \models \mathit{Friends}(x; y)$

ν_1			ν_2		
x	y	z	x	y	z
alice	$*$	$-$	alice	bob	$-$
alice	$*$	$-$	alice	carol	$-$
carol	$*$	$-$	carol	dave	$-$

Semantics of atomic modules (or relations) are BRVs.

Facebook Example

Consider a database D with a binary relation:

Friends

alice	bob
alice	carol
carol	dave

All pairs (ν_1, ν_2) such that
$D, (\nu_1, \nu_2) \models \mathit{Friends}(x; y)$

ν_1			ν_2		
x	y	z	x	y	z
alice	$*$	$-$	alice	bob	$-$
alice	$*$	$-$	alice	carol	$-$
carol	$*$	$-$	carol	dave	$-$

Semantics of atomic modules (or relations) are BRVs.

Binary Relation on Valuations (BRV)

BRV is

- a set of pairs of valuations
- the dynamic semantics of a module

Semantics of Complex Modules operations on BRVs $\Rightarrow$ BRVs
E.g. $R_1(x; y) \cup R_2(u; v)$ union of BRVs
E.g. $R_1(x; y) \circ R_2(y; z)$ composition of BRVs

Forward-LIF (FLIF) & its Syntax

FLIF expressions E are:

$$E ::= \tau \mid E \circ E \mid E \cup E \mid E \cap E \mid E - E$$

Atomic expressions τ are (where x and y are variables, and c is a constant):

$$\tau ::= \underbrace{R(\bar{x};\bar{y})}_{\text{relation atom}} \mid \underbrace{(x = y) \mid (x = c)}_{\text{selection}} \mid \underbrace{(x := y) \mid (x := c)}_{\text{assignment}}$$

Evaluation Problem

Attempt #1

Given a FLIF expression E, the evaluation problem for expression E on instance D:

Input: A valuation ν_1

Output: All valuations ν_2 such that $D, (\nu_1, \nu_2) \models E$

Evaluation Problem

Attempt #1

Given a FLIF expression E, the evaluation problem for expression E on instance D:

Input: A valuation ν_1

Output: All valuations ν_2 such that $D, (\nu_1, \nu_2) \models E$

Not practical...

- variables are infinitely many
  $\Rightarrow$ give values only for input variables
- most of variables remain unchanged
  $\Rightarrow$ return only values for output variables

Semantic Inputs and Outputs of FLIF Expressions

Atomic modules (relations):

- input arguments are specified in the vocabulary
- remaining arguments are outputs

Example

Relation *Friend* of input arity 1, total arity 2
Expression $\mathit{Friend}(x; y)$ has input variable x, output variable y

Semantic Inputs and Outputs of FLIF Expressions

Atomic modules (relations):

- input arguments are specified in the vocabulary
- remaining arguments are outputs

Example

Relation *Friend* of input arity 1, total arity 2
Expression $\mathit{Friend}(x; y)$ has input variable x, output variable y

For complicated expressions, not so obvious

Semantic Inputs and Outputs of FLIF Expressions

Atomic modules (relations):

- input arguments are specified in the vocabulary
- remaining arguments are outputs

Example

Relation *Friend* of input arity 1, total arity 2
Expression $\mathit{Friend}(x; y)$ has input variable x, output variable y

For complicated expressions, not so obvious

Intuitively,

- **Outputs** $O(E)$: can change during the evaluation
- **Inputs** $I(E)$: are needed from the beginning to determine $O(E)$

Evaluation Problem (Revisited)

Ideal Formalization $Eval_E(D, \nu_{\text{in}})$

Given a FLIF expression E, the evaluation problem for expression E on instance D:

Input: A valuation ν_{in} on the input variables

Output: Projection on output variables of

$$\{\nu_{\text{out}} \mid \exists \nu'_{\text{in}} \supseteq \nu_{\text{in}} : D, (\nu'_{\text{in}}, \nu_{\text{out}}) \models E\}$$

Evaluation Problem (Revisited)

Ideal Formalization $Eval_E(D, \nu_{\mathrm{in}})$

Given a FLIF expression E, the evaluation problem for expression E on instance D:

Input: A valuation ν_{in} on the input variables

Output: Projection on output variables of

$$\{\nu_{\mathrm{out}} \mid \exists \nu'_{\mathrm{in}} \supseteq \nu_{\mathrm{in}} : D, (\nu'_{\mathrm{in}}, \nu_{\mathrm{out}}) \models E\}$$

Not feasible. . .

- deciding whether a variable is output (input) of some given expression is **undecidable** [KR2020]

  $\Rightarrow$ work with syntactic approximations

Syntactic Approximations of Inputs and Outputs

E	$I(E)$	$O(E)$
$R(\bar{x};\bar{y})$	$\bar{x}$	$\bar{y}$
$E_1 \circ E_2$	$I(E_1) \cup (I(E_2) - O(E_1))$	$O(E_1) \cup O(E_2)$
$E_1 \cup E_2$	$I(E_1) \cup I(E_2) \cup (O(E_1) \triangle O(E_2))$	$O(E_1) \cup O(E_2)$
$E_1 \cap E_2$	$I(E_1) \cup I(E_2) \cup (O(E_1) \triangle O(E_2))$	$O(E_1) \cap O(E_2)$
$E_1 - E_2$	$I(E_1) \cup I(E_2) \cup (O(E_1) \triangle O(E_2))$	$O(E_1)$
$(x = y)$	$\{x, y\}$	$\emptyset$
$(x := y)$	$\{y\}$	$\{x\}$
$(x = c)$	$\{x\}$	$\emptyset$
$(x := c)$	$\emptyset$	$\{x\}$

Can We Express FLIF in FOL?

What happens if a variable is an input and an output?

Relation a binary relation *Increment* $= \{(0,1),(1,2),(2,3),\ldots\}$.

Expression $\mathit{Increment}(x;x)$ increments the value of x

Discrepancy $\mathit{Increment}(x;x)$ is not satisfied in FOL

Can We Express FLIF in FOL?

What happens if a variable is an input and an output?

Relation a binary relation *Increment* $= \{(0,1),(1,2),(2,3),\ldots\}$.

Expression *Increment*$(x;x)$ increments the value of x

Discrepancy *Increment*$(x;x)$ is not satisfied in FOL

Solution work with $(\mathrm{FLIF}^{\mathrm{io}})$: the input-output disjoint fragment of FLIF.

How does $\mathrm{FLIF}^{\mathrm{io}}$ compare to FLIF?

not a simple equivalence ...

How does $\mathrm{FLIF}^{\mathrm{io}}$ compare to FLIF?

not a simple equivalence ...

Example

Consider the following two **semantically** equivalent expressions

- $E_1 = (R_1(x;y) \cup R_2(y;x)) - (R_1(x;y) \triangle R_2(y;x))$
- $E_2 = (R_1(x;y) \cap R_2(y;x))$

	E_1	E_2
$\mathrm{FLIF}^{\mathrm{io}}$	*no*	*yes*
I	$\{x, y\}$	$\{x, y\}$
O	$\{x, y\}$	$\{\}$

How does $\mathrm{FLIF}^{\mathrm{io}}$ compare to FLIF?

not a simple equivalence ...

Example

Consider the following two **semantically** equivalent expressions

- $E_1 = (R_1(x;y) \cup R_2(y;x)) - (R_1(x;y) \triangle R_2(y;x))$
- $E_2 = (R_1(x;y) \cap R_2(y;x))$

	E_1	E_2
$\mathrm{FLIF}^{\mathrm{io}}$	*no*	*yes*
I	$\{x, y\}$	$\{x, y\}$
O	$\{x, y\}$	$\{\}$

It is not good to

- require more inputs
- return less outputs

How does $\mathrm{FLIF}^{\mathrm{io}}$ compare to FLIF?

Attempt #1 (Rename Output Variables)

Let E be an FLIF expression, and ρ be a renaming of output variables of E. Then there exists an $\mathrm{FLIF}^{\mathrm{io}}$ expression E' such that:

1. inputs of E and E' are the same;
2. outputs of E' are the outputs of E renamed by ρ; and
3. for every interpretation D and every valuation ν_{in} on inputs, we have

$$\mathit{Eval}_E(D, \nu_{\mathrm{in}}) \circ \rho = \mathit{Eval}_{E'}(D, \nu_{\mathrm{in}})$$

How does $\mathrm{FLIF}^{\mathrm{io}}$ compare to FLIF?

Attempt #1 (Rename Output Variables)

Let E be an FLIF expression, and ρ be a renaming of output variables of E. Then there exists an $\mathrm{FLIF}^{\mathrm{io}}$ expression E' such that:

1. inputs of E and E' are the same;
2. outputs of E' are the outputs of E renamed by ρ; and
3. for every interpretation D and every valuation ν_{in} on inputs, we have

$$Eval_E(D, \nu_{\mathrm{in}}) \circ \rho = Eval_{E'}(D, \nu_{\mathrm{in}})$$

For trivial expressions, it works:

- For $E = R(x; x)$ and $\rho = \{x \mapsto y\}$
  Take $E' = R(x; y)$

How does $\mathrm{FLIF}^{\mathrm{io}}$ compare to FLIF?

Attempt #1 (Rename Output Variables)

Let E be an FLIF expression, and ρ be a renaming of output variables of E. Then there exists an $\mathrm{FLIF}^{\mathrm{io}}$ expression E' such that:

1. inputs of E and E' are the same;
2. outputs of E' are the outputs of E renamed by ρ; and
3. for every interpretation D and every valuation ν_{in} on inputs, we have

$$Eval_E(D, \nu_{\mathrm{in}}) \circ \rho = Eval_{E'}(D, \nu_{\mathrm{in}})$$

For trivial expressions, it works:

- For $E = R(x; x)$ and $\rho = \{x \mapsto y\}$
  Take $E' = R(x; y)$

For complex expressions, not really...

Problems in Attempt#1

1. Requiring outputs of E' to be outputs of E renamed by ρ
   - expression $S(x;) - R(x;x)$ has no outputs

Problems in Attempt#1

1. Requiring outputs of E' to be outputs of E renamed by ρ
   - expression $S(x;) - R(x;x)$ has no outputs

2. Variable clashes
   - expression $R(x;x) \circ S(y;z)$ has x and z as outputs, using $\rho : \{x \mapsto y\}$
     - × $R(x;y) \circ S(y;z)$ wrong semantics

Final Attempt [ICDT2021]

Theorem Statement

Let E be an FLIF expression, F be a set of forbidden variables, and ρ be a bijection from the output variables of E to a set of variables disjoint from the variables in E. Then there exists an $\mathrm{FLIF}^{\mathrm{io}}$ expression E' such that:

1. inputs of E and E' are the same;
2. outputs of E' include the outputs of E renamed by ρ and the extra output variables are disjoint from F; and
3. for every interpretation D and every valuation ν_{in} on inputs, we have

$$Eval_E(D, \nu_{\mathrm{in}}) = Eval_{E'}(D, \nu_{\mathrm{in}}) \circ \rho$$

$$\mathrm{FLIF}^{\mathrm{io}} \equiv \mathrm{FLIF}$$

Examples

1. expression $S(x;) - R(x;x)$ has no outputs
   - $(S(x;) \circ (y := x)) - (R(x;y) \circ (y := x))$

2. expression $R(x;x) \circ S(x;x)$ has x as output, using $\rho : \{x \mapsto y\}$
   - $R(x;z) \circ S(z;y)$

3. expression $R(x,u;u,y) \cap S(y;y)$ has y as output, using $\rho : \{y \mapsto z\}$
   - $R(x,u;u_1,z_1) \circ (u_1 = u) \circ S(y;z) \circ (z_1 = z)$

4. expression $T(;) \cup (S(;x) \circ R(x;x))$ has x as output, using $\rho : \{x \mapsto y\}$
   - $T(;) \circ (z := x) \circ (y := x) \cup (S(;z) \circ R(z;y))$

Observations and Future Work

Observation on proving $\mathrm{FLIF}^{\mathrm{io}} \equiv \mathrm{FLIF}$:

- the need for extra output variables
  (their values is not important, though)

Observations and Future Work

Observation on proving $\mathrm{FLIF}^{\mathrm{io}} \equiv \mathrm{FLIF}$:

- the need for extra output variables
  (their values is not important, though)

Open problems:

- Will FLIF benefit from adding a new category of variables, namely intermediate variables?
- Does IO-disjoint LIF have the same expressive power of LIF?

References

E. Ternovska
An Algebra of Modular Systems: Static and Dynamic Perspectives.
Proceedings 12th International Symposium on Frontiers of Combining Systems (FroCos 2019).

H. Aamer, B. Bogaerts, D. Surinx, E. Ternovska, and J. Van den Bussche
Executable First-Order Queries in the Logic of Information Flows.
Proceedings 23rd International Conference on Database Theory (ICDT 2020).

H. Aamer, B. Bogaerts, D. Surinx, E. Ternovska, and J. Van den Bussche
Inputs, Outputs, and Composition in the Logic of Information Flows.
Proceedings 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020).

H. Aamer and J. Van den Bussche
Input-Output Disjointness for Forward Expressions in the Logic of Information Flows
Proceedings 24th International Conference on Database Theory (ICDT 2021).