Peter Landin Semantics Seminar
Edmund Robinson
Queen Mary University of London
Logical Relations and Mathematical Foundations
Ex 10.11

Assuming an implemented language in which the following phrase is a definition, give an experiment-by-context intended to settle whether it is a block or not (i.e., whether the first defined name in each case is hidden or supplied).

\[
\text{let } \text{stopper type } \text{a } \text{[77, 87]}
\]

\[
\text{let } \text{list type } \text{b } \text{[stopper type]}
\]

\[
\text{let } \text{of } \text{number@[lettype]}
\]

Ex 10.12

Assuming an implemented language in which the following phrase is a definition, give an experiment-by-context intended to settle whether it is a block or not (i.e., whether the first defined name in each case is hidden or supplied).

\[
\text{let } \text{a } \text{=} \text{[10 b a + 66]}
\]

Ex 10.13

Assuming an implemented language in which the following phrase is a definition, give an experiment-by-context intended to settle whether it is a block or not (i.e., whether the first defined name in each case is hidden or supplied).

\[
\text{let } \text{f(y) } \text{=} \text{y + 11}
\]

\[
\text{in } \text{(let } \text{g(y) } \text{=} \text{f(y) + 77)}
\]

Ex 10.14

Assuming an implemented language in which the following phrase is a definition, give an experiment-by-context intended to settle whether it is a block or not (i.e., whether the first defined name in each case is hidden or supplied).

\[
\text{let } \text{pair } \text{=} \text{number * number}
\]

\[
\text{in } \text{(let tree = nil (tree * pair * tree))}
\]
Performance

• Peter was very interested in describing what programs do.
Change in Semantics

- Move from proving programs “correct” in some absolute sense
- To providing tools to improve quality
  - and those tools have to fit in with the development chain
Formal models

• But you still have to produce a formal model
Mathematics

- The language we use when we want to do calculations about a system
- But the calculations are never about the actual system
- They are about models of the system
What happens if we use different models: do we get the same results?
Logical Relations
Two key messages

• Basic ideas are quite simple, and if you focus, then you can keep them like that.

• We can use them to justify (in fact derive) some standard notions of process equivalence.
Logical Relations

- Robert Milne: thesis - proving equivalence of implementations

- Mike Gordon: unpublished discussions

- Gordon Plotkin:
Logical Relation Inference and Multiview Information Interaction for Domain Adaptation Person Re-Identification
S Li, F Li, J Li, H Li, B Zhang, D Tao - IEEE Transactions on ..., 2023 - ieeexplore.ieee.org

A Novel Logical Neural Network Structure for Representing Logical Relations Clearly and Incrementally in a More Direct Mapping Manner
Z Han - papers.ssr.com

Logical Relations for Session-Typed Concurrency
"logical relation" POPL

Articles

About 837 results (0.14 sec)

Any time

Since 2023

Since 2022

Since 2019

Custom range...

Sort by relevance

Sort by date

Any type

Review articles

include patents

include citations

Create alert

[PDF] Automatic Differentiation for ML-Family Languages: Correctness via Logical Relations
F Lucatelli Nunes, M Vákár - 2023 - publications.mfo.de
INTRODUCTION AD and the PL community. Automatic differentiation (AD) is a popular technique for computing derivatives of functions implemented by a piece of code, particularly …

CLOMO: Counterfactual Logical Modification with Large Language Models
Y Huang, R Hong, H Zhang, W Shao, Z Yang... - arXiv preprint arXiv ..., 2023 - arxiv.org
... logical relation. The objective for these models is to adeptly modify the argument text until the specified logical relation is ... Argument: Statement1: It is widely assumed that people need to ...

[PDF] Engineering logical relations for MLTT in Coq
A Adjej12, M Lennon-Bertrand, K Maillard... - ... Conference on Types for ... - meven.ac
... [1] formalize an inductive-recursive [5] definition of a logical relation for a representative ... Thus, we reformulate the logical relation using small induction-recursion, which can in turn ...
Basic types and operations
A Simple View

- Our structure comprises:
- Some basic entities (objects, A, B,...)
- And operations between them.
A Simple View

STRUCTURE

MODELS
A Simple View

STRUCTURE

A

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

DB

MODELS

A

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

DB

DB

DB

DB
 Putting things together: operations compose

A

B

C

A

B

C
Putting things together: operations compose
Putting things together: relations compose
Putting things together: relations compose
A Simple View

\[
a \cdot a = a \\
A \cdot A = B
\Rightarrow f(a \cdot b) \leq f(a \cdot b)
\]
A system of relations is “Logical” if operations respect relations.

\[
\text{a } \text{A } \text{a } \text{A } \text{b } \text{b } \Rightarrow f(a,b) \leq f(a,b)
\]
A system of relations between models is “logical” if the operations respect the relations.
Algebra

- Objects - giving basic sorts
- Operations - between objects
- Equations - between operations

Usual interpretation:
- Object = Set
- Operation = Function
Example: Groups

• One basic objects/sort: the group carrier

• Three operations:

  - Plus equations: associativity, identity, inverse
Models: Groups

• Actual groups:
Definition. A group $G$ is a system of elements which is closed under a single-valued binary operation which is associative, and relative to which $G$ contains an element satisfying the identity law, and with each element another element (called its inverse) satisfying the inverse law.

Corollary. A group has only one identity element, and only one inverse $a^{-1}$ for each element $a$. 

Birkhoff and Mac Lane: A Survey of Modern Algebra
Models

- Models are actual groups:

- A relation \( G \rightarrow G \) is logical iff

  - \( g \rightarrow g \Delta \Rightarrow g' \rightarrow g' \Rightarrow g \ast g' \rightarrow g \ast g' \)

  - \( 1 \rightarrow 1 \)

  - \( g \rightarrow g \Rightarrow g' \rightarrow g' \)
Models

• If \( f \) is a function \( G \to G \) and \( G \) and \( G \) are groups, then
  • \( \text{Graph}(f) \) is a logical relation for the group operations
  • \( \text{iff} \ f \) is a group homomorphism.

• This result holds for arbitrary algebraic theories.

• Logical relations generalise, and encapsulate a standard algebraic concept.
First-order types and a bit of category theory
We want to use more than just the basic objects

- First-order types:
  - Products and sums

- If we have a product of types in our structure, then we want to generate a relation between the corresponding products in our two different models.
Products of relations

A → A

B → B

A × A

A × B

B × B

a → a

b → b

and
Sums of relations

\[ A \to A \]

\[ B \to B \]

\[ A + B \]

\[ a \to a \]

\[ b \to b \]
n-ary operations

- A n-ary operation is equivalent to a unary operation on the product of the inputs.
amalgamating operations

- Having two operations is equivalent to a unary operation on the sum of the inputs.

- Example: groups
Algebra and Co-algebra

• Classically, both deal with one-sorted theories, ie one basic type

• algebra says that elements of that type can be combined into others by applying operations

• co-algebra says that elements of that type can be decomposed into the result of applying such operations to other elements of the type.
Multiplication and co-multiplication
Category Theory

• In the categorical account of algebra, terms are packaged up into a functor.

• $TA = \text{terms built from algebra operations and constants that are elements of } A$

• Algebra: $\xrightarrow{TA} A$

• Coalgebra: $\xrightarrow{A} TA$
Compositionality

• At this level everything is fine.

• Operations compose.

• We can use type-theoretic operations we expect (projection, tupling, injection, case).

• Logical relations compose.
Higher-order types: Functions
Exponentiation of relations

- Given pairs of types
- And relations
- Can we get a relation
A Simple View

- Structure
- Models

A → A
B → B

a → a
f a → f a
Exponentiation of relations

- Given pairs of types
- And relations
- Can we get a relation
- Ans: yes
How this works

- Application is OK:

\[ f \Delta a \Rightarrow fa \]

- As is lambda abstraction:

\[ \lambda a. fab \Rightarrow \lambda a. fab \]
How it breaks down: failure of composition

- Other constructors preserve order on relations, \(\rightarrow\) does not.
- Compositionality of relations fails
How it breaks down: failure of composition
How it breaks down: failure of composition

\[ [R_A \rightarrow R_B] [S_A \rightarrow S_B] \]

\[ + [R_A S_A \rightarrow R_B S_B] \]

But we do get \( \leq \)
So far all very concrete: Sets are the go to mathematical structure for building things
But logic is the go to tool for reasoning about them.
Get rid of the sets: a logic-based approach
Look at the proofs

- Proofs all use logic and basic type theory, not really set theory
- We need a predicate logic, with sorts and predicates over sorts.
- Start with a unary version.
Logic as a type theory

- Types: sort (=context) + predicate defined in that context.
- Terms: have two components -
  - substitution
  - entailment
Logic as a type theory

- Functions between contexts generalise terms (they are substitutions).
  \[ x : A \quad y := e(x) \rightarrow y : B \]

- Predicates have a context (their free variables).
  \[ x : A \vdash P(x) \]
Logic as a type theory

- Predicates have a context (their free variables).

\[ x : A \vdash P(x) \]

- Functions between predicates are a substitution and an entailment.

\[ x : A \vdash P(x) \overset{\text{sub}}{\longrightarrow} y : B \vdash Q(y) \]
\[ y := e(x) \]
\[ x : A \vdash P(x) \rightarrow Q(e(x)) \]
Logic as a type theory

- Model of type theory
- Homomorphism of structure
- Model of type theory
Can also do this for binary predicates (relations)
Core idea

```
Rel

Type

Type
```
Core idea

Rel → Pred

Type × Type → Type
Sets again

- This story works semantically for sets and relations.
- Ditch the idea that a relation is just the set of elements.
- A relation also has to know what sets it is a relation between.
Why should we care?

• Ans: not everything is a set, not every construction uses set-theoretic constructions.

• Example: Kripke logical relations, step-indexed logical relations

• Idea: work in a world where everything is, say, Kripke. Kripke gives good interpretation of logic. Binary predicates give logical relations.

• Key to understanding lots of complicated papers is that they are just talking about this simple picture in the context of a complicated world.
Using structure to derive congruences
Example: State transition systems
Different forms of bisimulation can be derived from different ways of modelling systems.
Labelled non-deterministic state transition system

• A set of States

• a labelled transition relation
Labelled non-deterministic state transition system: bisimulation (Park-Milner)

- Two systems
- Relation between them is a bisimulation if
Formalising

\[ \alpha : A \times S \rightarrow P(S) \]
\[ \alpha(a, s) = \{ s' | s \xrightarrow{a} s' \} \]

Given \[ \alpha : A \times S \rightarrow P(S) \]
\[ \alpha : A \times S \rightarrow P(S) \]
for what relation \( S \xrightarrow{a} S \) is \( \alpha \alpha \alpha \)?
Formalising

Given \( \alpha : A \times S \rightarrow P(s) \)

for what relation \( S \overset{\alpha}{\rightarrow} S \) is \( \alpha \circ \alpha \)?

- Does not depend on exact way model is structured, e.g. \( A \overset{\alpha}{\rightarrow} [s \mapsto P(s)] \)
- \( \alpha \) depend on how we extend \( P(s) \) to relations.
Power-set as a type constructor: possibility 1

- Interpret the powerset of S as functions S -> Bool: \( P(S) = S \rightarrow \text{Bool} \)
- really strong, relational version of the contravariant power-set functor.
Power-set as a type constructor: possibility 2

- $\mathbb{P}(S)$ covariant power set functor,

- is the “free complete sup-semi-lattice on $S$”

  - algebraic theory

  - have $V_x$ for any set $X$.

  - equations between the $V_x$

  - (Proper class of operations and proper class of equations, but up to equality only a set of operations for each set).
Extension to Rel

- What is the free complete sup-semilattice in Rel?

- Given R a relation between A and B, we need \( P(R) \) defined to be a relation between \( P(A) \) and \( P(B) \)

- \( U \ P(R) \ V \) iff
  - there is an \( S \) subset of \( R \) such that \( \pi_0 S = U \) and \( \pi_1 S = V \)
  - iff for all \( u \) in \( U \) there is a \( v \) in \( V \) such that \( uRv \), and for all \( v \) in \( V \) there is a \( u \) in \( U \) such that \( uRv \).
Extension to Rel

• What is the free complete sup-semilattice in Rel?

• Given R a relation between A and B, we need P(R) defined to be a relation between P(A) and P(B)

• U P(R) V iff

  • there is an S subset of R such that pi_o S = U and pi_1 S = V

  • iff forall u in U there is a v in V such that uRv, and for all v in V there is a u in U such that uRv.
Extension to Rel

- $U \mathcal{P}(R) V$
  
  - iff there is an $S$ subset of $R$ such that $\pi_0 S = U$ and $\pi_1 S = V$
  
  - iff for all $u$ in $U$ there is a $v$ in $V$ such that $uRv$, and for all $v$ in $V$ there is a $u$ in $U$ such that $uRv$. 
Strong bisimulation

Given \( S R S \) and \( \alpha : A \times S \rightarrow P(S) \)

then \( \alpha \sim \alpha \) iff when \( s \sim s \) then

\[ \alpha(a, s) \sim \alpha(a, s) \]

i.e. when \( s' \in \alpha(a, s) \) (\( s = a, s' \))

then there is \( s' \sim s' \in \alpha(a, s) \) (\( s = a, s' \))

etc.

i.e. iff \( R \) is a strong bisimulation.
Other forms of bisimulation
Other forms of bisimulation

- weak bisimulation
- branching bisimulation
- semi-branching bisimulation
- probabilistic bisimulation
Basic strategy

• There are other ways of modelling state transition systems.

• For weak bisimulation we are interested in systems that have silent internal computations.

• For branching bisimulation we have silent internal computations, but also synchronisation points.

• For probabilistic bisimulation we need models of stochastic processes.
State transition systems as monoid HM

- Our model only deals with single transitions.

- We could ask it to account for sequences of transitions.

- A monoid homomorphism

\[
\begin{align*}
A \times S & \rightarrow P(S) \\
\overset{\circ}{A} & \rightarrow [S \rightarrow P(S)] \\
A^* & \rightarrow [S \rightarrow P(S)]
\end{align*}
\]
Weak bisimulation (Milner)

• Processes have silent tau actions, representing internal computation.

**Definition 20.** (Milner (1989)) Let $S$ be a labelled transition system for $A = L + \{\tau\}$, and $v \in L^*$, then

$$s \xrightarrow{v} s' \text{ iff there is a } w \in A^* = (L + \{\tau\})^* \text{ such that } v = \hat{w} \text{ and } s \xrightarrow{w} s'.$$

We can type $\Rightarrow$ as $\Rightarrow : [L^* \rightarrow [S \rightarrow \mathcal{P} S]]$, and we refer to it as the system derived from $\rightarrow$.
Weak bisimulation (Milner)

- Processes have silent tau actions, representing internal computation.

**Definition 20.** (Milner (1989)) Let $S$ be a labelled transition system for $A = L + \{\tau\}$, and $v \in L^*$, then

$$s \xrightarrow{v} s' \text{ iff there is a } w \in A^* = (L + \{\tau\})^* \text{ such that } v = \hat{w} \text{ and } s \xrightarrow{w} s'.$$

We can type $\Rightarrow$ as $\Rightarrow : [L^* \to [S \to \mathcal{P} S]]$, and we refer to it as the system derived from $\rightarrow$.

**Definition 21.** If $S$ and $T$ are two labelled transition systems for $A = L + \{\tau\}$, then a relation $R \subseteq S \times T$ is a weak bisimulation iff for all $a \in A = L + \{\tau\}$, whenever $sRt$

- for all $s \xrightarrow{a} s'$, there is $t'$ such that $t \xrightarrow{a} t'$ and $s'Rt'$
- and for all $t \xrightarrow{a} t'$, there is $s'$ such that $s \xrightarrow{a} s'$ and $s'Rt'$.
Weak Bisimulation

\[
\begin{align*}
S & \xrightarrow{a} S' \\
\text{matched by} & \\
S & \xrightarrow{T} T' \ldots \xrightarrow{a} T' \ldots \xrightarrow{T} S' \\
\text{or if } a = T & \text{ additionally} \\
S & \xrightarrow{a=T} S' \\
S & \xrightarrow{a} S'
\end{align*}
\]
Weak bisimulation (1):
saturation

Definition 26. Let $F : (L + \{\tau\}) \rightarrow [S \rightarrow \mathcal{P} S]$ be a transition system with internal action. We say that $F$ is saturated if

1. $id \leq F(\tau)$ and $F(\tau).F(\tau) \leq F(\tau)$ and
2. for all $a \in L$, $F(\tau).F(a).F(\tau) \leq F(a)$

(1) $s \not\rightarrow s$ $s \not\rightarrow s'$ $s \not\rightarrow t$, $s''$

(2) $s \not\rightarrow s'$ $s \not\rightarrow a$, $s''$ $s''$
Weak bisimulation (1):
saturation

Definition 26. Let $F : (L + \{\tau\}) \longrightarrow [S \rightarrow \mathcal{P} S]$ be a transition system with internal action. We say that $F$ is saturated if

1. $\text{id} \leq F(\tau)$ and $F(\tau) . F(\tau) \leq F(\tau)$ and
2. for all $a \in L$, $F(\tau) . F(a) . F(\tau) \leq F(a)$

Proposition 27. Suppose $F : (L + \{\tau\}) \longrightarrow [S \rightarrow \mathcal{P} S]$ and $G : (L + \{\tau\}) \longrightarrow [T \rightarrow \mathcal{P} T]$ are saturated transition systems with internal actions, then $R \subseteq S \times T$ is a weak bisimulation between the systems if and only if it is a strong bisimulation between them.
Weak bisimulation (1):
saturation

ORIGINAL MODEL $\alpha: (A + \{T\}) \times S \rightarrow P(S)$

SATURATE IT $\bar{\alpha}: (A + \{T\}) \times S \rightarrow P(S)$

- SAME STATE SPACE

- CHECK $\bar{\alpha}$ FOR STRONG BISIMULATION
Weak bisimulation (1):
saturation

Original model: visible action + T action

generates new model:
- sequences of visible action

\[ \alpha : A^* \rightarrow [s \rightarrow P(s)] \]

But \( \alpha(\varepsilon) \neq \lambda s. s \)?

So not a monoid \( HM \)

It doesn't respect concatenation (semi-group).
Weak Bisimulation

- Model is a semi-group HM
- constructed from an original
- A relation between two such models is logical iff it is a weak bisimulation between the original models.
Weak bisimulation (2):
lax HM

Definition 31. A lax transition system on an alphabet \( L \) (not including an internal action \( \tau \)) is a function \( F : L^* \rightarrow [S \rightarrow \mathcal{P} S] \) such that:

1. \( \text{id} \leq F(\varepsilon) \) (reflexivity)
2. \( F(vw) = F(v).F(w) \) (composition)

Definition 32. Let \( F : (L + \{\tau\}) \rightarrow [S \rightarrow \mathcal{P} S] \) be a transition system with internal action, then its laxification \( \hat{F} : L^* \rightarrow [S \rightarrow \mathcal{P} S] \) is the lax transition system defined by:

1. \( \hat{F}(\varepsilon) = F(\tau)^* \)
2. \( \hat{F}(a) = F(\tau)^*.F(a).F(\tau)^* \), for any \( a \in L \).
3. \( \hat{F}(vw) = \hat{F}(v).\hat{F}(w) \).

Lemma 35. Suppose \( F : (L + \{\tau\}) \rightarrow [S \rightarrow \mathcal{P} S] \) and \( G : (L + \{\tau\}) \rightarrow [T \rightarrow \mathcal{P} T] \) are transition systems with internal actions, and \( R \subseteq S \times T \). Then the following are equivalent:

1. \( R \) is a weak bisimulation between \( F \) and \( G \)
2. \( (\hat{F}, \hat{G}) \in [\text{Id}_{L^*} \rightarrow [R \rightarrow \mathcal{P} R]] \)
3. \( R \) is the state space of a lax transition system in \( \text{Rel} \) whose first projection is \( \hat{F} \) and whose second is \( \hat{G} \).
Weak bisimulation (2):
lax HM

Original model: visible action + T action

generates new model:
- sequences of visible action

\[ \alpha : A^* \rightarrow [s \rightarrow P(s)] \]

But \( \alpha(\varepsilon) \neq \lambda s. s \)

So not a monoid HM

Just respects concatenation (semi-group).
Branching bisimulation

**Definition 36.** A relation $R \subseteq S \times T$ is called a branching bisimulation if and only if whenever $sRt$:

- $s \xrightarrow{a} s'$ implies $((\exists t_1, t_2 \in T. t \xrightarrow{\tau} t_1 \xrightarrow{a} t_2 \land sRt_1 \land s'Rt_2) \lor (a = \tau \land s'Rt))$,
- $t \xrightarrow{a} t'$ implies $((\exists s_1, s_2 \in S. s \xrightarrow{\tau} s_1 \xrightarrow{a} s_2 \land s_1Rt \land s_2Rt') \lor (a = \tau \land sRt'))$. 

\[\begin{align*}
  &s \xrightarrow{a} s' \\
  &\Rightarrow \\
  &s \xrightarrow{a} t \\
  &\text{OR} \\
  &s \xrightarrow{a} t
\end{align*}\]
Branching bisimulation

**Definition 36.** A relation \( R \subseteq S \times T \) is called a branching bisimulation if and only if whenever \( sRt \):

- \( s \xrightarrow{a} s' \) implies \( ((\exists t_1, t_2 \in T. t \xrightarrow{\tau^*} t_1 \xrightarrow{a} t_2 \land sRt_1 \land s'Rt_2) \lor (a = \tau \land s'Rt)) \),
- \( t \xrightarrow{a} t' \) implies \( ((\exists s_1, s_2 \in S. s \xrightarrow{\tau^*} s_1 \xrightarrow{a} s_2 \land s_1Rt \land s_2Rt') \lor (a = \tau \land sRt')) \).

\[
F^b : (L + \{\tau\}) \longrightarrow [S \rightarrow \mathcal{P}(S \times S)]
\]

\[
F^b as = \{(s_1, s_2) \in S \times S | (s \xrightarrow{\tau^*} s_1 \xrightarrow{a} s_2) \lor (a = \tau \text{ and } s = s_1 = s_2)\}.
\]
Branching bisimulation

Definition 36. A relation $R \subseteq S \times T$ is called a branching bisimulation if and only if whenever $sRt$:

- $s \xrightarrow{a} s'$ implies $(\exists t_1, t_2 \in T. t \xrightarrow{\tau} t_1 \xrightarrow{a} t_2 \land sRt_1 \land s'Rt_2)$ or $(a = \tau \land s'Rt)$,
- $t \xrightarrow{a} t'$ implies $(\exists s_1, s_2 \in S. s \xrightarrow{\tau} s_1 \xrightarrow{a} s_2 \land s_1Rt \land s_2Rt')$ or $(a = \tau \land sRt')$.

$$F^b : (L + \{\tau\}) \longrightarrow [S \rightarrow \mathcal{P}(S \times S)]$$

$$F^b as = \{(s_1, s_2) \in S \times S \mid (s \xrightarrow{\tau} s_1 \xrightarrow{a} s_2) \text{ or } (a = \tau \text{ and } s = s_1 = s_2)\}.$$ 

Theorem 39. Let $R \subseteq S \times T$. Then $R$ is a branching bisimulation if and only if $(F^b, G^b) \in \text{[Id}_{L+\{\tau\}} \rightarrow [R \rightarrow \mathcal{P}(R \times R)]]$. 
Probabilistic Bisimulation

• Need to model stochastic processes not just state transition.

• Idea (Lawvere, Giry) process is given by a form of “Markov kernel”: an operator that relates a probability space on the domain to a measure space on the codomain and gives the probability of a transition function taking a value in a given measurable set.

• Notion of bisimulation arising from logical relations is strong probabilistic bisimulation.

• Have to work harder to get close to Pi-bisimulation.
References


