Algorithmic Correspondence for Relevance Logics, Bunched Implication Logics, and Relation Algebras via an Implementation of the Algorithm PEARL

Willem Conradie[1], Valentin Goranko[2] and Peter Jipsen[3]

[1]University of the Witwatersrand, South Africa
[2]Stockholm University, Sweden
[3]Chapman University, USA

Relational and Algebraic Methods in Computer Science, RAMiCS 2021
CIRM, Marseille, France

November 2-5, 2021

Relevance logic: syntax, relevance frames

The language of propositional relevance logic $\mathcal{L}_R$ over a fixed set of propositional variables VAR is given by

$$A = p \mid \bot \mid \top \mid \mathbf{t} \mid \sim A \mid (A \wedge A) \mid (A \vee A) \mid (A \circ A) \mid (A \to A)$$

A relevance frame is a tuple $\mathcal{F} = \langle W, O, R, ^* \rangle$, where:

- W is a non-empty set of states (possible worlds);
- $O \subseteq W$ is the subset of normal states;
- $R \subseteq W^3$ is a relevant accessibility relation;
- $^* : W \to W$ is a function, called the Routley star, used to provide semantics for $\sim$.

Relevance logic: Routley-Meyer frames

The following binary relation $\preceq$ is defined in every relevance frame:

$$u \preceq v \text{ iff } \exists o(o \in O \wedge Rouv)$$

A Routley-Meyer frame (or RM-frame) is a relevance frame satisfying the following conditions for all $u, v, w, x, y, z \in W$:

1. $x \preceq x$
2. If $x \preceq y$ and $Ryuv$ then $Rxuv$.
3. If $x \preceq y$ and $Ruyv$ then $Ruxv$.
4. If $x \preceq y$ and $Ruvx$ then $Ruvy$.
5. If $x \preceq y$ then $y^* \preceq x^*$.
6. O is upward closed w.r.t. $\preceq$, i.e. if $o \in O$ and $o \preceq o'$ then $o' \in O$.

It follows that $\preceq$ is a preorder.

Adding a valuation $V : \mathrm{VAR} \to \mathcal{P}^{\uparrow}(W)$ produces a Routley-Meyer model

$$\mathcal{M} = \langle W, O, R, ^*, V\rangle$$

Relevance logic: Relational Semantics

Truth of a formula A in a RM-model $\mathcal{M} = \langle W, O, R,^*, V\rangle$ at a state $u \in W$, denoted $\mathcal{M}, u \Vdash A$, is defined as follows:

- $\mathcal{M}, u \Vdash p$ iff $u \in V(p)$;
- $\mathcal{M}, u \Vdash \mathbf{t}$ iff $u \in O$;
- $\mathcal{M}, u \Vdash {\sim}A$ iff $\mathcal{M}, u^* \nVdash A$;
- $\mathcal{M}, u \Vdash A \wedge B$ iff $\mathcal{M}, u \Vdash A$ and $\mathcal{M}, u \Vdash B$;
- $\mathcal{M}, u \Vdash A \vee B$ iff $\mathcal{M}, u \Vdash A$ or $\mathcal{M}, u \Vdash B$;
- $\mathcal{M}, u \Vdash A \rightarrow B$ iff $\forall v, w$ if $Ruvw$ and $\mathcal{M}, v \Vdash A$ then $\mathcal{M}, w \Vdash B$.
- $\mathcal{M}, u \Vdash A \circ B$ iff $\exists v, w$ $Rvwu$ and $\mathcal{M}, v \Vdash A$ and $\mathcal{M}, w \Vdash B$.

Relevance logic: Algebraic Semantics

A structure $\mathbb{A} = \langle A, \wedge, \vee, \circ, \to, \sim, \mathbf{t}, \top, \bot \rangle$ is called a relevant algebra [Urquhart, 1996] if it satisfies the following conditions:

1. $\langle A, \wedge, \vee, \top, \bot \rangle$ is a bounded distributive lattice,
2. $a \circ (b \vee c) = (a \circ b) \vee (a \circ c)$,
3. $(b \vee c) \circ a = (b \circ a) \vee (c \circ a)$,
4. $\sim(a \vee b) = \sim a \wedge \sim b$,
5. $\sim(a \wedge b) = \sim a \vee \sim b$,
6. $\sim\top = \bot$ and $\sim\bot = \top$,
7. $a \circ \bot = \bot \circ a = \bot$,
8. $\mathbf{t} \circ a = a$, and
9. $a \circ b \leq c$ iff $a \leq b \to c$.

An $\mathcal{L}_R$-formula ϕ is valid on a relevant algebra $\mathbb{A}$ if the inequality $\mathbf{t} \leq \phi$ is valid on $\mathbb{A}$.

The relevance logic $\mathbf{RL} = \{\phi \mid \mathbf{t} \leq \phi$ is valid in all relevant algebras$\}$.

Some well-known correspondences for Relevance Logic [Routley, Plumwood, Meyer, Brady, 1982]

The basic relevance logic **B** is **RL** extended with $A \vee {\sim}A$ and ${\sim}{\sim}A$.

	Axiom	Correspondent on RM-frames
B1.	$A \wedge (A \to B) \to B$	$Raaa$
B6.	$A \to ((A \to B) \to B)$	$Rabc \Rightarrow Rbac$
B10.	$A \to (B \to B)$	$Rabc \Rightarrow b \leq c$
B11.	$B \to (A \to B)$	$Rabc \Rightarrow a \leq c$
B13.	$A \to (B \to A \wedge B)$	$Rabc \Rightarrow a \leq c \;\&\; b \leq c$
B19.	$A \vee B \to ((A \to B) \to B)$	$Rabc \Rightarrow (Rbac \;\&\; a \leq c)$
D3.	$(A \to {\sim}A) \to {\sim}A$	$Raa^{*}a$
D4.	$(A \to {\sim}B) \to (B \to {\sim}A)$	$Rabc \Rightarrow Rac^{*}b^{*}$

Extensions of **B** with these axioms are sound and complete with respect to corresponding classes of RM-frames.

Modal Correspondence (and Canonicity) Theory

- Sahlqvist-van Benthem class (mid 1970's)
- Many, many extensions and variations subsequently.
- Purely algebraic proof of canonicity of Sahlqvist-van Benthem formulas [Jònsson, 1994]
- Canonicity-via-correspondence approach [Sambin, Vacarro, 1989].
- Sahlqvist theory for distributive modal logic [Gehrke, Nagahashi, Venema, 2005]
- Extension to inductive formulas [Goranko, Vakarelov, 2006]
- Classical Algorithmic correspondence, SQEMA [Conradie, Goranko, Vakarelov, 2006]
- 'Unified Correspondence' [Conradie, Palmigiano 2012, 2019], [Conradie, Ghilardi, Palmigiano, 2014]

Relevant Correspondence (and Canonicity) Theory

"*Correspondence theory in the case of modal and intuitionistic logic has been extensively studied, but the analogous theory for the case of relevant logics is surprisingly neglected.*" – Urquhart, 1996

Precursors:

- A Sahlqvist theorem for relevant modal logics [Seki, 2003]
- A Sahlqvist theorem for substructural logic [Suzuki, 2013]
- On Sahlqvist formulas in relevant logic [Badia, 2018]

Current work:

- Within the Unified Correspondence framework:
  - Definition of Inductive and Sahlqvist formulas for Relevance Logic
  - A calculus of rewrite rules (specializing ALBA)
- PEARL algorithm = Calculus + Relevance Logic-specific translation + simplification
- Implementation of PEARL

The PEARL algorithm

PEARL = Propositional variable Elimination Algorithm for Relevance Logic

Extended language:

$$\begin{array}{rcl}\phi & = & p \mid \mathbf{i} \mid \mathbf{m} \mid \top \mid \bot \mid \mathbf{t} \mid {\sim}\phi \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \circ \phi) \mid (\phi \to \phi) \mid \\ & & {\sim}^{\flat}\phi \mid {\sim}^{\sharp}\phi \mid (\phi \prec \phi) \mid (\phi \Rightarrow \phi) \mid (\phi \hookrightarrow \phi)\end{array}$$

where $p \in \mathrm{VAR}$, $\mathbf{i} \in \mathrm{NOM}$ and $\mathbf{m} \in \mathrm{CNOM}$.

Applies a calculus of rewrite rules to eliminate propositional variables in favor of nominals and co-nominals.

If successful, translates the result into first-order logic and simplifies.

PEARL calculus: Ackermann-rules

$$\frac{\alpha \leq p,\ \Delta(p) \implies \gamma(p) \leq \beta(p)}{\Delta(\alpha/p) \implies \gamma(\alpha/p) \leq \beta(\alpha/p)}\ (RAR)$$

$$\frac{p \leq \alpha,\ \Gamma(p) \implies \beta(p) \leq \gamma(p)}{\Gamma(\alpha/p) \implies \beta(\alpha/p) \leq \gamma(\alpha/p)}\ (LAR)$$

The Right Ackermann-rule (RAR) and Left Ackermann-rule (LAR) are subject to the following conditions:

- p does not occur in α,
- β is positive in p,
- γ is negative in p,
- Γ is negative in p,
- Δ is positive in p,

"positive in p" means all occurrences of p have an even number of $\sim$, $\_ \to \phi$, $\_ \Rightarrow \phi$ in their scope. "negative in p" means the opposite.

PEARL calculus: Monotone variable elimination rules

$$\frac{\Gamma(p) \implies \gamma(p) \leq \beta(p)}{\Gamma(\top/p) \implies \gamma(\bot/p) \leq \beta(\bot/p)} (\bot) \qquad \frac{\Delta(p) \implies \beta(p) \leq \gamma(p)}{\Delta(\bot/p) \implies \beta(\top/p) \leq \gamma(\top/p)} (\top)$$

provided that $\beta(p)$ and Γ are positive in p, while $\gamma(p)$ and $\Delta(p)$ are negative in p.

For example $A \leq ({\sim}A \to B)$ rewrites to $A \leq ({\sim}A \to \bot)$.

PEARL calculus: Approximation rules (sample)

$$\frac{\Gamma \implies \phi \leq \psi}{\mathbf{j} \leq \phi,\ \psi \leq \mathbf{m},\ \Gamma \implies \mathbf{j} \leq \mathbf{m}}$$

where **j** is a nominal and **m** is a co-nominal not occurring in the premise.

$$\frac{\chi \to \phi \leq \mathbf{m},\ \Gamma \implies \alpha \leq \beta}{\mathbf{j} \leq \chi,\ \mathbf{j} \to \phi \leq \mathbf{m},\ \Gamma \implies \alpha \leq \beta}\ (\to\text{Appr-Left})$$

$$\frac{\mathbf{i} \leq \chi \circ \phi,\ \Gamma \implies \alpha \leq \beta}{\mathbf{j} \leq \chi,\ \mathbf{i} \leq \mathbf{j} \circ \phi,\ \Gamma \implies \alpha \leq \beta}\ (\circ\text{Appr-Left})$$

$$\frac{\sim\phi \leq \mathbf{m},\ \Gamma \implies \alpha \leq \beta}{\phi \leq \mathbf{n},\ \sim\mathbf{n} \leq \mathbf{m},\ \Gamma \implies \alpha \leq \beta}\ (\sim\text{Appr-Left})$$

where **j** a nominal and **n** is a co-nominal not appearing in the premises.

PEARL calculus: Residuation rules

$$\frac{\phi \leq \chi \vee \psi,\ \Gamma \implies \alpha \leq \beta}{\phi \prec \chi \leq \psi,\ \Gamma \implies \alpha \leq \beta}\ (\vee\mathsf{Res})$$

$$\frac{\phi \wedge \chi \leq \psi,\ \Gamma \implies \alpha \leq \beta}{\phi \leq \chi \Rightarrow \psi,\ \Gamma \implies \alpha \leq \beta}\ (\wedge\mathsf{Res})$$

$$\frac{\phi \leq \chi \rightarrow \psi,\ \Gamma \implies \alpha \leq \beta}{\phi \circ \chi \leq \psi,\ \Gamma \implies \alpha \leq \beta}\ (\rightarrow\mathsf{Res})$$

$$\frac{\psi \leq \phi \hookrightarrow \chi,\ \Gamma \implies \alpha \leq \beta}{\phi \circ \psi \leq \chi,\ \Gamma \implies \alpha \leq \beta}\ (\hookrightarrow\mathsf{Res})$$

PEARL calculus: Adjunction rules

$$\frac{\phi \vee \chi \leq \psi,\ \Gamma \implies \alpha \leq \beta}{\phi \leq \psi,\ \chi \leq \psi,\ \Gamma \implies \alpha \leq \beta}\ (\vee\mathsf{Adj})$$

$$\frac{\psi \leq \phi \wedge \chi,\ \Gamma \implies \alpha \leq \beta}{\psi \leq \phi,\ \psi \leq \chi,\ \Gamma \implies \alpha \leq \beta}\ (\wedge\mathsf{Adj})$$

$$\frac{\sim \phi \leq \psi,\ \Gamma \implies \alpha \leq \beta}{\sim^{\flat} \psi \leq \phi,\ \Gamma \implies \alpha \leq \beta}\ (\sim\mathsf{Left\text{-}Adj})$$

$$\frac{\phi \leq \sim \psi,\ \Gamma \implies \alpha \leq \beta}{\psi \leq \sim^{\sharp} \phi,\ \Gamma \implies \alpha \leq \beta}\ (\sim\mathsf{Right\text{-}Adj})$$

PEARL for bunched implication algebras and RAs

A **bunched implication algebra** $\mathbf{A} = \langle A, \wedge, \vee, \top, \bot, \circ, \mathbf{t}, \rightarrow, \Rightarrow\rangle$ such that $\langle A, \wedge, \vee, \bot, \top\rangle$ is a bounded (distributive) lattice, $a \circ b \leq c$ iff $a \leq b \rightarrow c$, $a \wedge b \leq c$ iff $a \leq b \Rightarrow c$, and $\langle A, \circ, \mathbf{t}\rangle$ is a commutative monoid.

The frames of bunched implication algebras are relevant frames without *.

A **relation algebra** is a relevant algebra expanded with a Boolean negation $\neg$ such that $\langle A, \circ, \mathbf{t}\rangle$ is a monoid, $a \rightarrow b = {\sim}({\sim}b \circ a)$, and $\neg{\sim}(a \circ b) = \neg{\sim}b \circ \neg{\sim}a$.

The term $\neg{\sim}a$ is the **converse** of a (usually written $a^{\smile}$).

The relevant frames of relation algebras are also called **atom structures**, and the Boolean negation ensures that the partial order is an antichain.

Implementation of PEARL

The input is a LaTeX string using the standard syntax of relevance logic expressions.

Intuitionistic implication $\Rightarrow$, coimplication $\prec$, the right residual $\hookrightarrow$ of $\circ$, and the adjoints $\sim^\sharp$ and $\sim^\flat$ can also appear in an input formula.

The expression is parsed with a simple top-down Pratt parser [1973] using standard rules of precedence.

For well-formed formulas, an abstract syntax tree (AST) based on Python dictionaries and lists of arguments is created for each formula.

Parsing example

E.g., the formula $A = $ "$p \to q \wedge \mathbf{t}$" is translated to the AST representation

```
A={"id":"\to","a":[
    {"id":"p","a":[]},
    {"id":"\\land","a":[
        {"id":"q","a":[]},
        {"id":"\mathbf t","a":[]}
    ]}
 ]}
```

The implication symbol $\to$ is referenced by `A.id`

and the two arguments are `A.a[0]` and `A.a[1]`.

Preprocessing phase

Five short recursive Python functions are used to transform the AST representation step-by-step according to the specific groups of PEARL transformation rules.

The function `preprocess(st)` takes a LaTeX string `st` as input and parses it, producing an AST A.

If the formula A is not well-formed, an error-string is returned.

If it has a top-level $\to$ symbol, it is replaced with a $\leq$ to turn the formula into an inequality, and otherwise the equivalent inequality $\mathbf{t} \leq A$ is constructed.

Subsequently the splitting rules and monotonicity rules are applied and the resulting list of inequalities is returned.

Preprocessing example

For example, with `r"p\to q\land\mathbf t"` as input, the formula is parsed, rewritten as $p \leq q \wedge \mathbf{t}$

The splitting rules produce the list $[p \leq q, p \leq \mathbf{t}]$ and monotonicity returns $[\top \leq \bot, \top \leq \mathbf{t}]$.

The function `approximate(As)` takes this list as input, and applies the first approximation rule to each formula, followed by all possible left and right approximations interleaved with further applications of the splitting rule.

The result is a list of quasi-equations that always have conclusion $\mathbf{i} \leq \mathbf{m}$ and premises that are irreducible with respect to the approximation and splitting rules.

Elimination phase

The function `eliminate(As)` then attempts to apply the Ackermann-rules to each quasi-equation by selecting each variable, first with positive polarity and, if that does not succeed, then with negative polarity.

Backtracking is used to ensure that all variables are tried in all possible orders.

If for some quasi-equations none of the variable orders allow all variables to be eliminated, then the function reports this result.

On the other hand, if for each quasi-equations some variable order succeeds to eliminate all formula variables then the resulting list of pure quasi-equations (i.e., containing no formula variables, but only nominals or co-nominals) is returned.

Simplification phase

Since these pure quasi-equations contain redundant premises, the function `simplify(As)` is used to eliminate them, and to also apply the left and right simplification rules.

Finally the variant of the standard translation is applied to the pure quasi-equations and produces a first-order formula on the Routley-Meyer frames.

Translation rules (Part 1)

- $\mathrm{Tr}(\mathbf{i} \leq \mathbf{j}) = x_{\mathbf{j}} \preceq x_{\mathbf{i}}$
- $\mathrm{Tr}(\mathbf{i} \leq \mathbf{m}) = x_{\mathbf{i}} \not\preceq y_{\mathbf{m}}$
- $\mathrm{Tr}(\mathbf{i} \leq \mathbf{t}) = O x_{\mathbf{i}}$
- $\mathrm{Tr}(\mathbf{i} \leq \bot) = \mathrm{False}$
- $\mathrm{Tr}(\mathbf{i} \leq \top) = \mathrm{True}$
- $\mathrm{Tr}(\mathbf{i} \leq {\sim}\mathbf{m}) = x_{\mathbf{i}}^{*} \preceq y_{\mathbf{m}}$
- $\mathrm{Tr}(\mathbf{i} \leq {\sim}\mathbf{j}) = x_{\mathbf{j}} \not\preceq x_{\mathbf{i}}^{*}$
- $\mathrm{Tr}(\mathbf{i} \leq {\sim}A) = \forall x_{\mathbf{j}}(\mathrm{Tr}(j \leq A) \to x_{\mathbf{j}} \not\preceq x_{\mathbf{i}}^{*})$
- $\mathrm{Tr}(\mathbf{i} \leq \mathbf{j} \circ \mathbf{k}) = R x_{\mathbf{j}} x_{\mathbf{k}} x_{\mathbf{i}}$
- $\mathrm{Tr}(\mathbf{i} \leq \mathbf{j} \circ B) = \exists x_{\mathbf{k}}(\mathrm{Tr}(\mathbf{k} \leq B) \wedge R x_{\mathbf{j}} x_{\mathbf{k}} x_{\mathbf{i}})$
- $\mathrm{Tr}(\mathbf{i} \leq A \circ B) = \exists x_{\mathbf{j}}(\mathrm{Tr}(\mathbf{j} \leq A) \wedge \mathrm{Tr}(\mathbf{i} \leq \mathbf{j} \circ B))$
- $\mathrm{Tr}(\mathbf{i} \leq A \to B) = \mathrm{Tr}(\mathbf{i} \circ A \leq B)$
- $\mathrm{Tr}(\mathbf{i} \leq A \hookrightarrow B) = \mathrm{Tr}(A \circ \mathbf{i} \leq B)$
- $\mathrm{Tr}(\mathbf{i} \leq A \Rightarrow B) = \mathrm{Tr}(\mathbf{i} \wedge A \leq B)$
- $\mathrm{Tr}(\mathbf{i} \leq A \wedge B) = \mathrm{Tr}(\mathbf{i} \leq A) \wedge \mathrm{Tr}(\mathbf{i} \leq B)$
- $\mathrm{Tr}(\mathbf{i} \leq A \vee B) = \mathrm{Tr}(\mathbf{i} \leq A) \vee \mathrm{Tr}(\mathbf{i} \leq B)$

Translation rules (Part 2)

- $\mathrm{Tr}(\mathbf{n} \leq \mathbf{m}) = y_{\mathbf{m}} \preceq y_{\mathbf{n}}$
- $\mathrm{Tr}(\mathbf{t} \leq \mathbf{m}) = \neg O y_{\mathbf{m}}$
- $\mathrm{Tr}(\bot \leq \mathbf{m}) = \mathrm{True}$
- $\mathrm{Tr}(\top \leq \mathbf{m}) = \mathrm{False}$
- $\mathrm{Tr}({\sim}\mathbf{n} \leq \mathbf{m}) = y^*_{\mathbf{m}} \not\preceq y_{\mathbf{n}}$
- $\mathrm{Tr}({\sim}\mathbf{j} \leq \mathbf{m}) = x_{\mathbf{j}} \preceq y^*_{\mathbf{m}}$
- $\mathrm{Tr}({\sim}A \leq \mathbf{m}) = \exists x_{\mathbf{j}}(\mathrm{Tr}(j \leq A) \wedge x_{\mathbf{j}} \preceq y^*_{\mathbf{m}})$
- $\mathrm{Tr}(\mathbf{i} \circ \mathbf{j} \leq \mathbf{m}) = \neg R x_{\mathbf{i}} x_{\mathbf{j}} y_{\mathbf{m}}$
- $\mathrm{Tr}(\mathbf{i} \circ B \leq \mathbf{m}) = \forall x_{\mathbf{j}}(\mathrm{Tr}(\mathbf{j} \leq B) \to \neg R x_{\mathbf{i}} x_{\mathbf{j}} y_{\mathbf{m}})$
- $\mathrm{Tr}(A \circ B \leq \mathbf{m}) = \forall x_{\mathbf{i}}(\mathrm{Tr}(\mathbf{i} \leq A) \wedge \mathrm{Tr}(\mathbf{i} \circ B \leq \mathbf{m})$
- $\mathrm{Tr}(A \Rightarrow B \leq \mathbf{m}) = \forall x_{\mathbf{i}}(\mathrm{Tr}(\mathbf{i} \leq A \Rightarrow B) \to \mathrm{Tr}(\mathbf{i} \leq \mathbf{m}))$
- $\mathrm{Tr}(A \prec B \leq \mathbf{m}) = \mathrm{Tr}(A \leq B \vee \mathbf{m})$
- $\mathrm{Tr}(A \wedge B \leq \mathbf{m}) = \mathrm{Tr}(A \leq \mathbf{m}) \vee \mathrm{Tr}(B \leq \mathbf{m})$
- $\mathrm{Tr}(A \vee B \leq \mathbf{m}) = \mathrm{Tr}(A \leq \mathbf{m}) \wedge \mathrm{Tr}(B \leq \mathbf{m})$
- $\mathrm{Tr}(A \leq B) = \forall x_{\mathbf{j}}(\mathrm{Tr}(\mathbf{j} \leq A) \to \mathrm{Tr}(\mathbf{j} \leq B))$

The translation Tr is not restricted to pure quasi-inequalities and can be applied to arbitrary pure formulas.

The Python code can be used in any Jupyter notebook, with the output displayed in standard mathematical notation.

No special installation is needed to use the program in a personal Jupyter notebook or in a public cloud-based notebook such as Colab.google.com, and the output can be pasted into standard LaTeX documents.

Moreover the program can be easily extended to handle the syntax of other suitable logics and lattice-ordered algebras.

Try it out at in a Google Colab notebook

The Python code is available at `https://github.com/jipsen/PEARL`

It can also be copied and used directly in a browser at
`https://colab.research.google.com/drive/1p0PTkmyq7vTWgYDxCTFHVRwjaLeT45uX?usp=sharing`.

Two examples of output from the PEARL implementation

- Input: `pearl(`$(A \to B) \wedge (B \to C) \to (A \to C)$`, "latex")`
- Translate to (list of) initial inequalit(ies):
  $[(A \to B) \wedge (B \to C) \leq A \to C]$
- Approximation phase:
  $\mathbf{i} \leq A \to B,\ \mathbf{i} \leq B \to C,\ \mathbf{j}_1 \to \mathbf{n}_1 \leq \mathbf{m},\ C \leq \mathbf{n}_1,\ \mathbf{j}_1 \leq A \implies \mathbf{i} \leq \mathbf{m}$
- Order of variables during the elimination phase: $['+A','+C','+B']$
- Elimination phase: $\mathbf{j}_1 \to \mathbf{n}_1 \leq \mathbf{m},\ \mathbf{i} \circ (\mathbf{i} \circ \mathbf{j}_1) \leq \mathbf{n}_1 \quad \implies \quad \mathbf{i} \leq \mathbf{m}$
- Apply simplification rules: $\mathbf{i} \circ (\mathbf{i} \circ \mathbf{j}_1) \leq \mathbf{n}_1 \quad \implies \quad \mathbf{i} \leq \mathbf{j}_1 \to \mathbf{n}_1$
- Apply Tr rules: $\forall x_2(Rx_0x_1x_2 \implies \neg(Rx_0x_2y_1)) \implies \neg(Rx_0x_1y_1)$
- Contrapose and simplify: $Rx_0x_1y_1 \implies \exists x_2(Rx_0x_1x_2 \wedge Rx_0x_2y_1)$

Second example

- Input command: `pearl(`$A \to (\sim A \to B)$`, "latex")`
- Initial inequality after monotone variable elimination: $[A \leq \sim A \to \bot]$
- Approximation phase:
  $\mathbf{i} \leq A,\ \mathbf{j}_1 \to \mathbf{n}_1 \leq \mathbf{m},\ \bot \leq \mathbf{n}_1,\ \mathbf{j}_1 \leq \sim\mathbf{n}_2,\ A \leq \mathbf{n}_2 \quad \Longrightarrow \quad \mathbf{i} \leq \mathbf{m}$
- Elimination phase:
  $\mathbf{j}_1 \to \mathbf{n}_1 \leq \mathbf{m},\ \bot \leq \mathbf{n}_1,\ \mathbf{j}_1 \leq \sim\mathbf{n}_2,\ \mathbf{i} \leq \mathbf{n}_2 \quad \Longrightarrow \quad \mathbf{i} \leq \mathbf{m}$
- Apply simplification rules: $\mathbf{j}_1 \leq \sim\mathbf{n}_2 \quad \Longrightarrow \quad \mathbf{n}_2 \leq \mathbf{j}_1 \to \mathbf{n}_1$
- Apply Tr rules: $x_1^* \preceq y_2 \implies \forall x_2(Rx_2x_1y_1 \implies x_2 \preceq y_2)$

Some References

- V. R. Pratt, P. C. Fischer and J. D. Ullman: Top Down Operator Precedence, ACM Symposium on Principles of Programming Languages, Boston, MA, 41-51, ACM Press, 1973.
- Jónsson, B., 1994. On the canonicity of Sahlqvist identities. Studia Logica, 53(4), 473-491.
- Sambin, G. and Vaccaro, V., 1989. A new proof of Sahlqvist's theorem on modal definability and completeness. The Journal of Symbolic Logic, 54(3), 992-999.
- Goranko, V. and Vakarelov, D., 2006. Elementary canonical formulae: extending Sahlqvist's theorem. Annals of Pure and Applied Logic, 141(1-2), 180-217.
- Conradie, W., Goranko, V. and Vakarelov, D., 2006. Algorithmic correspondence and completeness in modal logic. I. The core algorithm SQEMA. Logical Methods in Computer Science, 2(1), 1-26.
- Conradie, W., Ghilardi, S. and Palmigiano, A., 2014. Unified correspondence. In Johan van Benthem on logic and information dynamics, Springer, 933-975.
- Conradie, W. and Palmigiano, A., 2012. Algorithmic correspondence and canonicity for distributive modal logic. Annals of Pure and Applied Logic, 163(3), 338-376.
- Conradie, W. and Palmigiano, A., 2019. Algorithmic correspondence and canonicity for non-distributive logics. Annals of Pure and Applied Logic, 170(9), 923-974.

Some References II

Badia, G., 2018. On Sahlqvist formulas in relevant logic. Journal of philosophical logic, 47(4), 673-691.

Seki, T., 2003. A Sahlqvist theorem for relevant modal logics. Studia Logica, 73(3), 383-411.

Suzuki, T., 2013. A Sahlqvist theorem for substructural logic. Rev. Symb. Log., 6(2), 383-411.

Urquhart, A., 1996. Duality for algebras of relevant logics. Studia Logica, 56(1), 263-276.

THANKS!