Towards an efficient relational deductive system for propositional non-classical logics

Andrea Formisano

Dipartimento di Informatica
Università di L'Aquila - ITALY

formisano@di.univaq.it

Marianna Nicolosi Asmundo

Dipartimento di Matematica ed Informatica
Università di Catania - ITALY

nicolosi@dmi.unict.it

Motivation

Relational formalization of many non-classical logics has been studied in a systematic and rigorous way (Orłowska et al.)

As a result
Relational renderings of the logics have been embedded in a common framework, at the

- Syntactical and semantical level: enriched algebras of binary relations
- Deduction level: relational Rasiowa-Sikorski proof system

The algebra of binary relations

- $\mathcal{L}^{\times}$ is an equational language devoid of variables (Tarski)
- it permits to state properties of binary relations over a domain $\mathcal{U}$

- Expressions are terms of the signature:

$$\mathsf{Z} \quad \mathsf{U} \quad \mathsf{I} \quad \mathsf{D} \qquad \text{(constants)}$$
$$\smile \quad ^{-} \qquad \text{(monadic)}$$
$$\cap \quad \triangle \quad ; \quad - \quad \cup \quad \dagger \qquad \text{(binary)}$$

- Equalities $Q = R$ where Q and R are expressions

- Semantics:
  - a domain $\mathcal{U}$ of discourse
  - usual interpretation for construcs
  - a subset of $\mathcal{U}^2$ for each expression

Which logic?

Any (propositional) logic expressible in relational terms, e.g.

- modal logic
- multi-modal logics
- logics of information (cf. Demri, Orłowska,...)
- lattice-based modal logics (cf. Orłowska, Vakarelov,...)
- temporal logic
- ...

Basic translation: propositional modal logic

A "classical" translation from modal propositional sentences φ to relational expressions $t(\varphi)$:

- $t(p_i) \quad =_{\text{Def}} \quad p'_i \,;\, \mathsf{U}$, where p'_i is a relational variable
- $t(\neg\psi) \quad =_{\text{Def}} \quad \overline{t(\psi)}$
- $t(\psi \wedge \chi) \quad =_{\text{Def}} \quad t(\psi) \cap t(\chi)$,
- $t(\Diamond\, \psi) \quad =_{\text{Def}} \quad \mathsf{R} \,;\, t(\psi)$

  where R is a constant designating the *accessibility relation*

... we have that: φ is a theorem iff $t(\varphi) = \mathsf{U}$

Goal

Mechanization

To produce an efficient implementation of the unified relational Rasiowa-Sikorski deductive system:

Relational translation of a logic (accessibility / axioms)

$$+$$

Relational translation of a formula

$$\Downarrow$$

A proof of the formula, if it is a theorem for the logic

How to reason?

Take advantage from the duality of Tableaux and Rasiowa-Sikorski (Golinska-Pilarek, Orłowska 06)

Transfer into relational Rasiowa Sikorski system, proof techniques and heuristics from the tableau methodology

Result:

A Prolog prover able to reason either automatically or interactively

What we have already done

- Optimized translator into the calculus of relations (FOO 2006)

- Efficient graph based representation of formulae and derivations by adapting BDD to relational dual tableaux

- refinements based on orderings and on the semantics of relational constants

we gain:

– maximization of structure sharing

– greater efficiency in the proof-search procedure (e.g., closure procedure)

..and what has still to be done

Further development of the prover

- comparisons with existing proof systems for non-classical logics
- introduction of free variables
- unification
- ...

Uniform notation for standard relational constructs

Conjunctive formulae, α-formulae	$\alpha = xR{\cap}Sy$	$\alpha_1 = xRy$	$\alpha_2 = xSy$	$(\wedge)$
	$\alpha = x\overline{R{\cup}S}y$	$\alpha_1 = x\overline{R}y$	$\alpha_2 = x\overline{S}y$	$(\neg\vee)$
Disjunctive formulae, β-formulae	$\beta = xR{\cup}Sy$	$\beta_1 = xRy$	$\beta_2 = xSy$	$(\vee)$
	$\beta = x\overline{R{\cap}S}y$	$\beta_1 = x\overline{R}y$	$\beta_2 = x\overline{S}y$	$(\neg\wedge)$
	$\beta = x\overline{\overline{R}}y$	$\beta_1 = xRy$		$(\neg\neg)$
δ^α-formulae:	$\delta^\alpha = xR;Sy$	$\delta_0^{\alpha 1} = xRz$	$\delta_0^{\alpha 2} = zSy$	$(\exists\wedge)$
	$\delta^\alpha = x\overline{R \dagger S}y$	$\delta_0^{\alpha 1} = x\overline{R}z$	$\delta_0^{\alpha 2} = z\overline{S}y$	$(\neg\forall\vee)$
	where z is an existentially quantified variable $(\delta^\alpha \equiv (\exists z)(\delta_0^{\alpha 1}(z) \wedge \delta_0^{\alpha 2}(z)))$			
γ^β-formulae	$\gamma^\beta = x\overline{R;S}y$	$\gamma_0^{\beta 1} = x\overline{R}z$	$\gamma_0^{\beta 2} = z\overline{S}y$	$(\neg\exists\wedge)$
	$\gamma^\beta = xR \dagger Sy$	$\gamma_0^{\beta 1} = xRz$	$\gamma_0^{\beta 2} = zSy$	$(\forall\vee)$
	where z is a universally quantified variable $(\gamma^\beta \equiv (\forall z)(\gamma_0^{\beta 1}(z) \vee \gamma_0^{\beta 2}(z)))$			
κ-formulae	$\kappa = xR^{\smile}y$	$\kappa_1 = yRx$		
	$\kappa = x\overline{R^{\smile}}y$	$\kappa_1 = y\overline{R}x$		

RS decomposition rules for standard relational constructs

$$\frac{x\alpha y}{x\alpha_1 y | x\alpha_2 y} \qquad \begin{array}{c} x\beta y \\ \hline x\beta_1 y \\ [x\beta_2 y] \end{array}$$

$$\frac{x\delta^\alpha y}{x\delta_0^{\alpha_1} z, x\delta^\alpha y | z\delta_0^{\alpha_2} y, x\delta^\alpha y} \qquad \begin{array}{c} x\gamma^\beta y \\ \hline x\gamma_0^{\beta_1} w \\ w\gamma_0^{\beta_2} y \end{array}$$

$$\frac{x\kappa y}{x\kappa_1 y}$$

We confine equality handling to the closure phase by adding suitable axiomatic sequences

Example

$$(1.)\ \{x(Q\cup(P;R;S))\cup(\overline{P}\dagger\overline{R;S}))y\}$$

$$\mid\ \beta$$

$$(2.)\ \{xQy,\ xP;R;Sy,\ x\overline{P}\dagger\overline{R;S}y\}$$

$$\mid\ \gamma^{\beta}$$

$$(3.)\ \{xQy,\ xP;R;Sy,\ x\overline{P}e_1,\ e_1\overline{R}e_2,\ e_2\overline{S}y\}$$

$$\delta^{\alpha}$$

$$(4.)\ \{xQy,\ xPe_1,\ xP;R;Sy,\ x\overline{P}e_1,\ e_1\overline{R}e_2,\ e_2\overline{S}y\}$$

$$\delta^{\alpha}$$

$$(5.)\ \{xQy,\ e_1R;Sy,\ xP;R;Sy,\ x\overline{P}e_1,\ e_1\overline{R}e_2,\ e_2\overline{S}y\}$$

$$\delta^{\alpha}$$

$$(6.)\ \{xQy,\ e_1Re_2,\ e_1R;Sy,\ xP;R;Sy,\ x\overline{P}e_1,\ e_1\overline{R}e_2,\ e_2\overline{S}y\}$$

$$\delta^{\alpha}$$

$$(7.)\ \{xQy,\ e_2Sy,\ e_1R;Sy,\ xP;R;Sy,\ x\overline{P}e_1,\ e_1\overline{R}e_2,\ e_2\overline{S}y\}$$

Some considerations

- Trees are the most natural structure to represent proofs in analytic systems
- They reflect the idea of recursively decompose the formula to be proved till elementary tautologies are found
- The frequent presence of redundant parts in the trees makes the proof search highly inefficient
- More efficient data structures are needed

Graph based data structure

Graph based representation of proofs and formulae has already been proposed in the context of tableaux systems: Binary Decision Diagrams, Shannon Graphs

We developed a similar representation for

- Relational expressions
- Checking validity instead of unsatisfiability

We deal with Boolean constructs, Peircean constructs and non-standard constructs

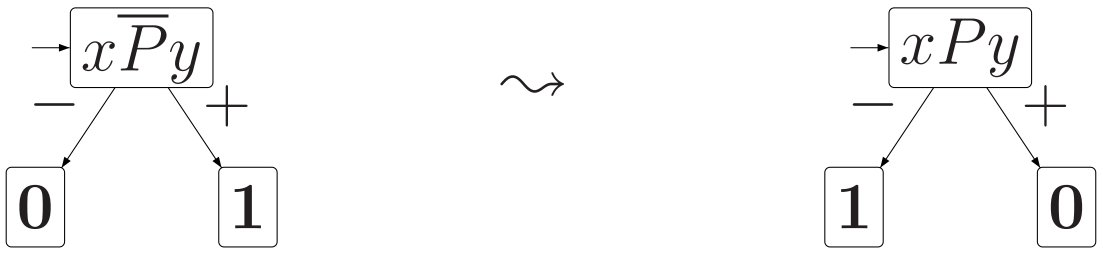
Relational decision graph (RDG)

- Nodes are labelled with relational literals: xRy
- Only two leaves labelled with $\mathbf{0}$ and $\mathbf{1}$
- Outgoing edges are labelled with $-$ and $+$
- Are constructed in a bottom-up manner
- Exploit nesting of (sub-)graphs

Rendering of a Boolean relational formula in RDG

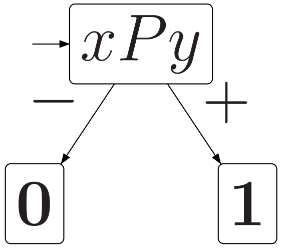
α- and β-(de)composition rules

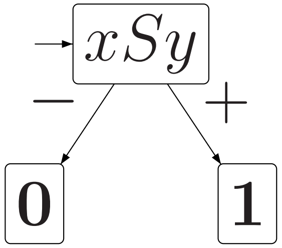
Complementation

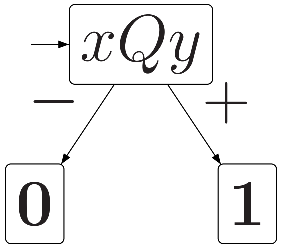


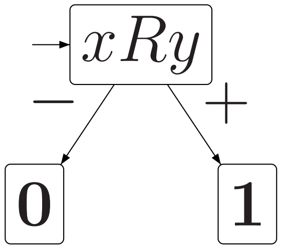
Example of construction

Relational formula $xP\cup((Q\cap R)\cup S)y$



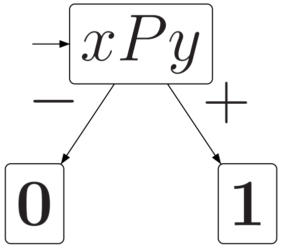


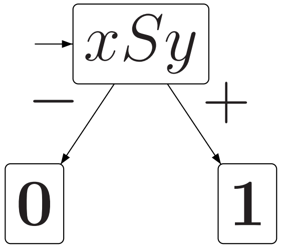




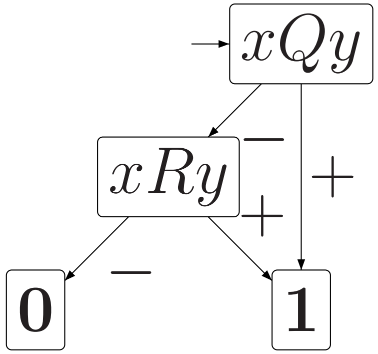
Example of construction

Relational formula $xP\cup((Q\cap R)\cup S)y$



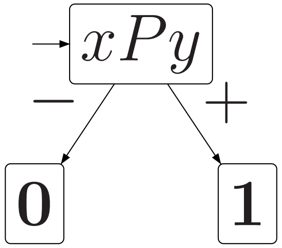


α-(de)composition

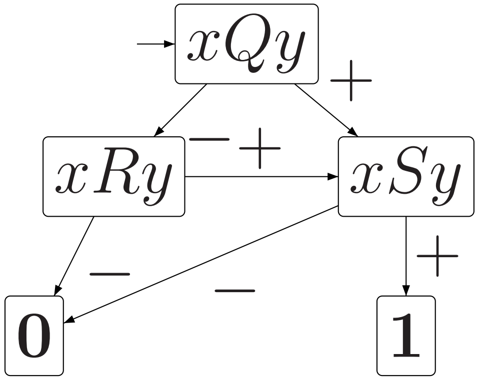


Example of construction

Relational formula $xP \cup ((Q \cap R) \cup S)y$



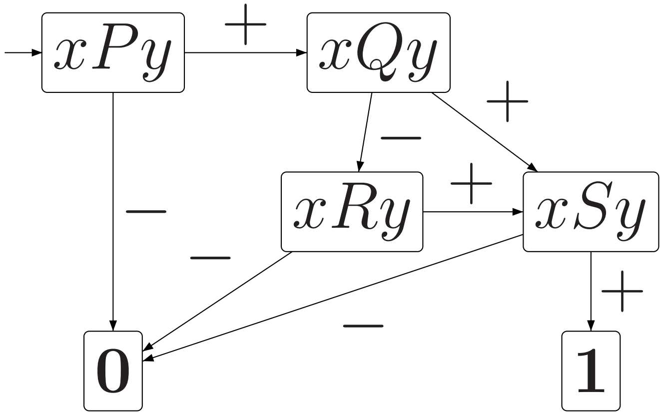
α-(de)composition
β-(de)composition



Example of construction

Relational formula $xP \cup ((Q \cap R) \cup S)y$

α-(de)composition
β-(de)composition
β-(de)composition



Dealing with Peircean constructs

Basic idea:

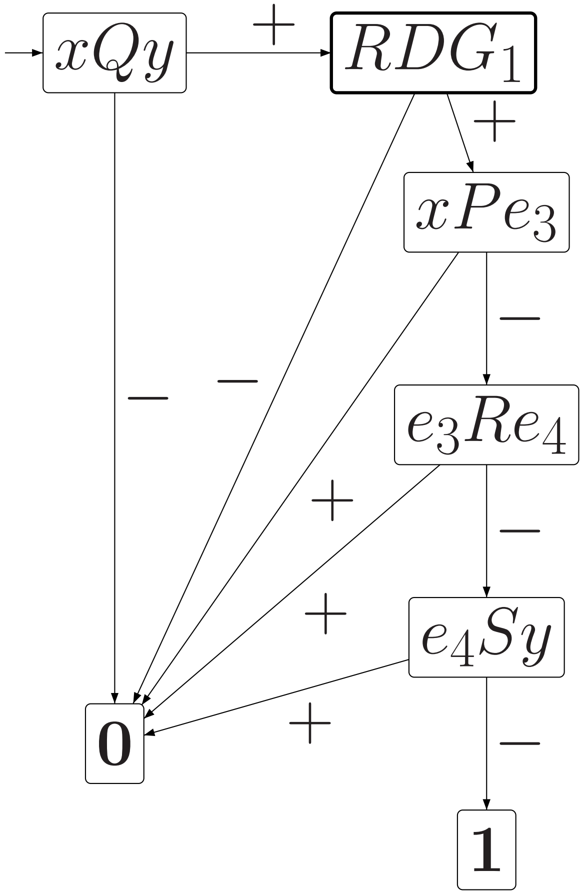
To allow an RDG to label a node in another RDG

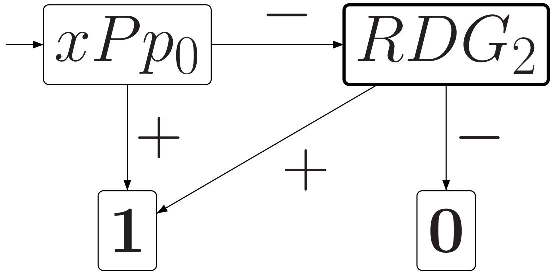
This affects the proof-search procedure:

- An *inner* procedure performing propositional reasoning
- An *outer* procedure generating δ-instantiations of the given formula

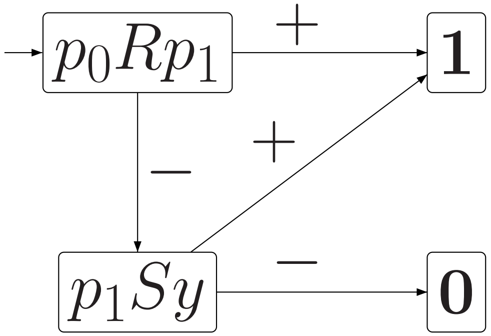
Example of use of nested RDGs

Consider the formula $x(Q\cup(P;R;S))\cup(\overline{P}\dagger\overline{R;S}))y$





RDG_1



RDG_2

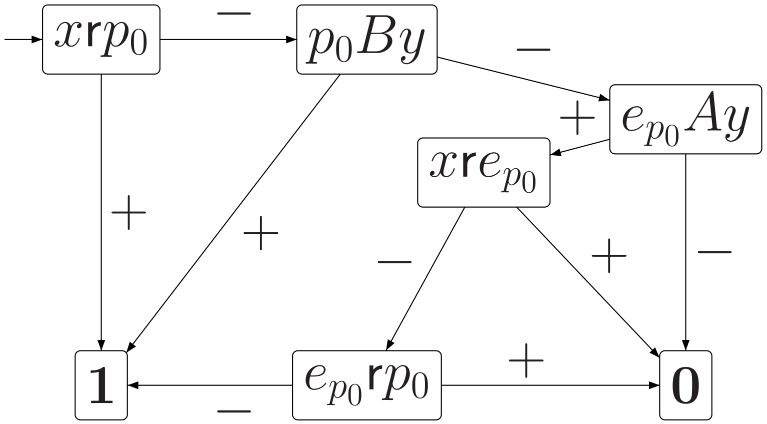
Non-standard relational constructs

We generalize the use of nestings: the first-order formula characterizing the non-standard construct is converted into and RDG

Example:

Until operator of temporal logic (relational construct $\mathtt{U}$)

$$x\,A\mathtt{U}B\,y \equiv \exists t(x\mathsf{r}t \wedge tBy \wedge (\forall u)(uAy \vee x\bar{\mathsf{r}}u \vee u\bar{\mathsf{r}}t)).$$



Refinements: orderings

Simplifications of the Boolean part of the RDG

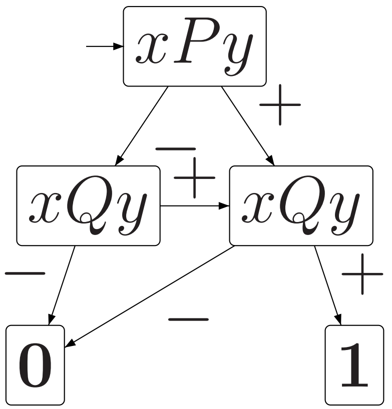
$$
\begin{array}{ll}
RDG(r(n), n^-, n^+) \rightsquigarrow n^- & \text{if } n^- = n^+ \\
RDG(r(n_1), RDG(r(n_2), n_2^-, n_2^+), n_1^+) \rightsquigarrow RDG(r(n_1), n_2^-, n_1^+) & \text{if } r(n_1) = r(n_2) \\
RDG(r(n_1), n_1^-, RDG(r(n_2), n_2^-, n_2^+)) \rightsquigarrow RDG(r(n_1), n_1^-, n_2^+) & \text{if } r(n_1) = r(n_2) \\
RDG(r(n_1), RDG(r(n_2), n_2^-, n_2^+), n_1^+) \rightsquigarrow & \\
\qquad RDG(r(n_2), RDG(r(n_1), n_2^-, n_1^+), RDG(r(n_1), n_2^+, n_1^+)) & \text{if } r(n_1) \succ r(n_2) \\
RDG(r(n_1), n_1^-, RDG(r(n_2), n_2^-, n_2^+)) \rightsquigarrow & \\
\qquad RDG(r(n_2), RDG(r(n_1), n_1^-, n_2^-), RDG(r(n_1), n_1^-, n_2^+)) & \text{if } r(n_1) \succ r(n_2)
\end{array}
$$

Handling of the basic relational constants

$$
\begin{array}{llll}
RDG(r(n), n^-, n^+) \rightsquigarrow n^- & \text{if } r(n){=}xUy & RDG(r(n), n^-, n^+) \rightsquigarrow n^+ & \text{if } r(n){=}xZy \\
RDG(r(n), n^-, n^+) \rightsquigarrow n^- & \text{if } r(n){=}xIx & RDG(r(n), n^-, n^+) \rightsquigarrow n^+ & \text{if } r(n){=}xDx
\end{array}
$$

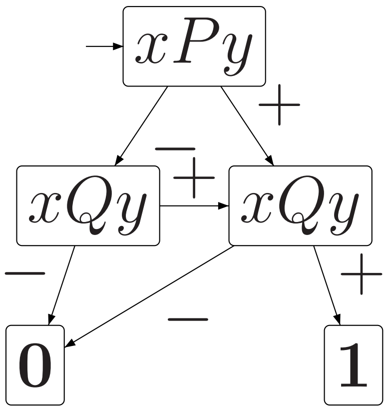
Example

Formula $x(P \cap Q) \cup Qy$



Example

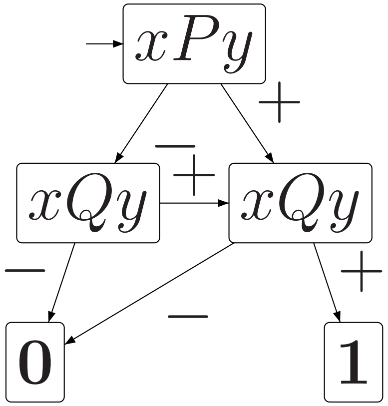
Formula $x(P \cap Q) \cup Qy$

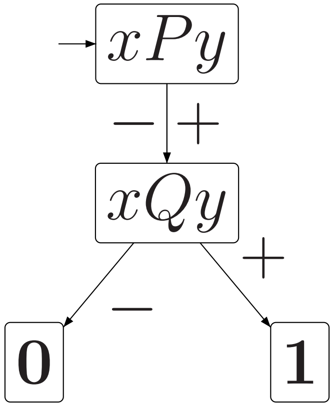


We have $RDG(r(n_1), n_1^-, RDG(r(n_2), n_2^-, n_2^+))$ and $r(n_1) = r(n_2)$. Thus the RDG is transformed into $RDG(r(n_1), n_1^-, n_2^+)$

Example

Formula $x(P \cap Q) \cup Qy$



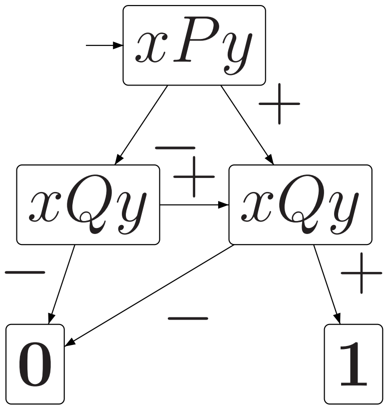


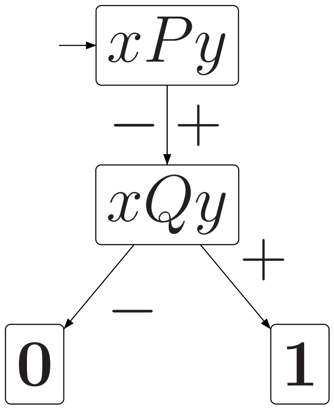
We have $RDG(r(n_1), n_1^-, RDG(r(n_2), n_2^-, n_2^+))$ and $r(n_1) = r(n_2)$. Thus the RDG is transformed into $RDG(r(n_1), n_1^-, n_2^+)$

Since $n^- = n^+$, apply $RDG(r(n), n^-, n^+) \rightsquigarrow n^-$

Example

Formula $x(P \cap Q) \cup Qy$



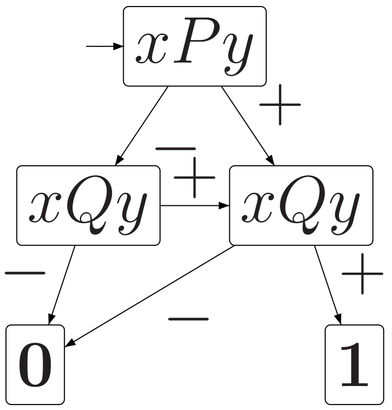


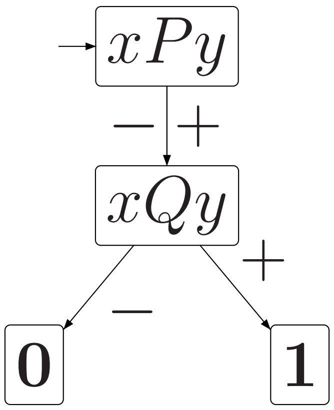
We have $RDG(r(n_1), n_1^-, RDG(r(n_2), n_2^-, n_2^+))$ and $r(n_1) = r(n_2)$. Thus the RDG is transformed into $RDG(r(n_1), n_1^-, n_2^+)$

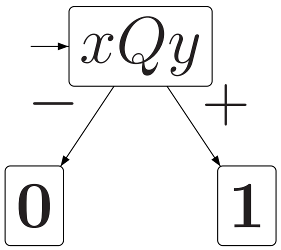
Since $n^- = n^+$, apply $RDG(r(n), n^-, n^+) \leadsto n^-$

Example

Formula $x(P{\cap}Q){\cup}Qy$







Further refinements

- R is a symmetric binary relation

$$RDG(xRy, n^-, n^+) \rightsquigarrow RDG(yRx, n^-, n^+) \quad \text{if } x \succ y$$

- R is a reflexive binary relation

$$RDG(xRy, n^-, n^+) \rightsquigarrow n^- \quad \text{if } x = y$$

- equality relator handling
- ...

The proof-search procedure

1. Bottom-up construction of the RDG
    - Boolean rules
    - Peircean rules
2. Normalization phase.

   If the RDG is made of a single $\mathbf{0}$-node then the initial formula is valid.

   Otherwise
3. Check that all the remaining $\mathbf{1}$-paths are closed
    - apply the closure rule to each $\mathbf{1}$-path
    - if a $\mathbf{1}$-path cannot be closed expand a $\boldsymbol{\delta}$-formula with a free variable in the RDG

Heuristics

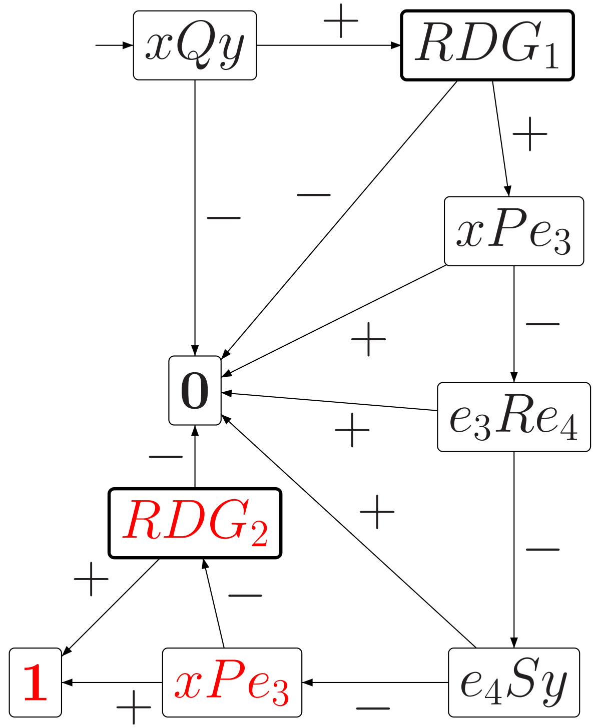
- Selection of δ-formula
- Selection of variable

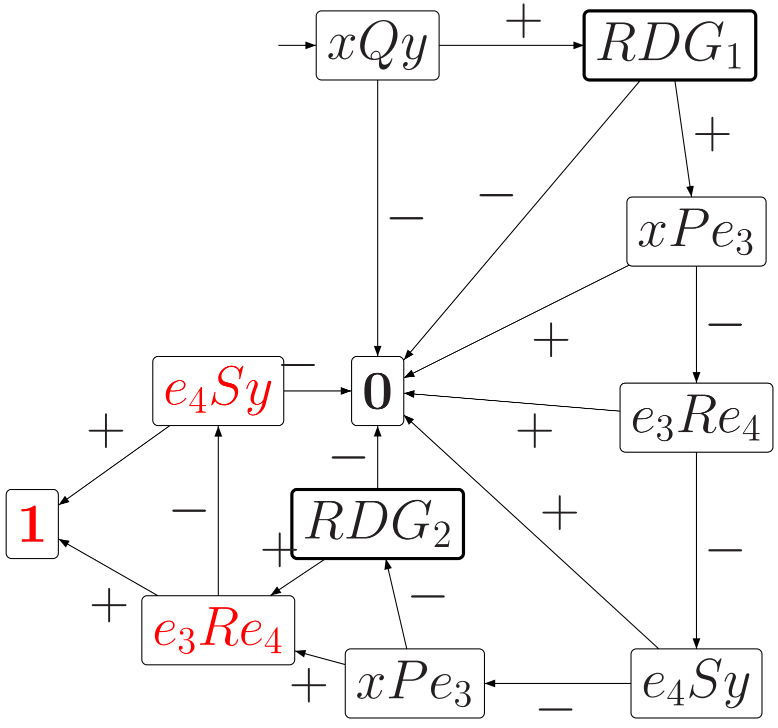
Currently: we initially order δ-formulae and variables choosing the first element on the list. To ensure fairness the list is rotated at each step

To avoid infinite computations we adopt a bounded deep-first iterative deepening

Example: the extension mechanism

Consider the formula $x(Q\cup(P;R;S))\cup(\overline{P}\dagger\overline{R;S}))y$





Conclusions

We have presented the first results toward the construction of an efficient relational prover for propositional non-classical logics based on the RS system

1. Efficient graph based representation of formulae and proofs together with refinements

2. A bounded deep-first iterative deepening strategy of proof search

Future work

1. Introduce efficient strategies for selection of δ-nodes and variables in the extension steps

2. Add features of theory reasoning to the closure phase

3. Add free variables and an unification

4. Compare with other proof systems and working theorem provers for non-classical logics

5. Compositional reasoning, formal verification..