On existentially quantified conjunctions of atomic formulae of $\mathcal{L}^+$ *

Domenico Cantone[†]
University of Catania, Italy

Alessandra Cavarra[‡]
University of Catania, Italy

Eugenio Omodeo[§]
University of L'Aquila, Italy

Abstract

Sufficient conditions for an $\exists^*$, $\forall\exists^*$, or $\forall\forall\exists^*$ prenex $\mathcal{L}^+$-sentence to be translatable into the variable-free formalism $\mathcal{L}^\times$ will be singled out in what follows. An efficient test based on such conditions will also be described. Through minor modifications of this testing algorithm, one can obtain the translation when the sufficient conditions are met.

1 Introduction

$\mathcal{L}^\times$ is a ground equational formalism that can compete with first-order predicate logic as a support for number theories as well as for full-blown theories of sets (cf. [TG87]). Superficially, it resembles the relational languages grown inside the database and knowledge base field more than the specification languages inspired by traditional logic. In analogy with first-order logic, fragments of which have been endowed with an executable semantics, $\mathcal{L}^\times$ has been proposed as a support for programming languages (cf. [BL92] and [FL94]). From the standpoint of automated deduction, $\mathcal{L}^\times$ has the appeal of being devoid of quantifiers—and even of variables: accordingly, performing deductions in $\mathcal{L}^\times$ is akin to the process of carrying out algebraic simplifications, without even the burden of having to instantiate variables.

In spite of the power it demonstrates when the aim is to formalize —and to reason inside— a strong theory (i.e. a theory where the operation of pairing and conjugated projections are definable), $\mathcal{L}^\times$ shows limitations when compared with plain predicate calculus. It is easily seen that any formula of $\mathcal{L}^\times$ can be translated into a first-order sentence involving at most three variables; however there are first-order sentences that have no logically equivalent sentence in three variables (the collection of all such sentences is not even recursive, cf. [TG87]), and moreover there are cases when a valid sentence in three variables cannot be proved unless by calling a fourth variable into play.

*Work partially supported by C.N.R. of Italy under Grant n. 96.01944.CT12, Research Project SETA.

[†]Dipartimento di Matematica, Università degli Studi di Catania, Viale A. Doria 6, I-95125 Catania, Italy, cantone@dipmat.unict.it

[‡]Dipartimento di Matematica, Università degli Studi di Catania, Viale A. Doria 6, I-95125 Catania, Italy, cavarra@boole.dipmat.unict.it

[§]Dipartimento di Matematica Pura e Applicata, Università degli Studi di L'Aquila, Via Vetoio, Località Coppito, I-67010 L'Aquila, Italy, omodeo@univaq.it

The inverse translation of first-order sentences in three variables into $\mathcal{L}^{\times}$, although not very easy, is possible too, and we own a program that does the job. More generally, our Prolog program accepts in input —and translates into $\mathcal{L}^{\times}$— sentences in three variables of the formalism $\mathcal{L}^{+}$ that extends both $\mathcal{L}^{\times}$ and a first-order language containing equality and an arbitrary number of binary predicate symbols, being devoid of constants and function symbols. One can hence express in $\mathcal{L}^{+}$, with relative ease, notions such as the ones of bisimulation, graph isomorphism, transitive $\in$-closure of a set, and have them translated automatically into compact $\mathcal{L}^{\times}$ equivalents. However, the limitation of having but three variables at one's disposal is unbearable after a while. E.g., every instance

$$\neg(\exists x_0, \ldots, x_{N+1})(x_{N+1} = x_0 \wedge \textstyle\bigwedge_{i=0}^{N} x_i \in x_{i+1})$$

of the acyclicity of membership can be stated with three variables (and indeed it can be expressed by the sentence $(\in \underbrace{\circ \in \ldots \circ \in}_{n \text{ times}}) \cap \iota{=}\emptyset$ of $\mathcal{L}^{\times}$), but why should one reformulate it before the translation? This paper addresses the problem of directly moving from $\mathcal{L}^{+}$ to $\mathcal{L}^{\times}$, when this is possible, without the preliminary obligation of moving quantifiers inwards (either manually or automatically) to comply with the three-variable restriction.

2 The languages $\mathcal{L}^{\times}$ and $\mathcal{L}^{+}$

The language $\mathcal{L}^{\times}$. Syntax and semantics

Predicate expressions in the language $\mathcal{L}^{\times}$ can be built from denumerably many binary predicate letters ($p_1, p_2, \ldots$), an *identity* map symbol (ι, occasionally written $=$), two binary operators ($\cap$ and $\circ$), two unary operators ($\overline{\ }$ and $^{-1}$), and two constants ($\emptyset$ and $\mathbb{1}$). One can relate predicate expressions through the equality symbol ($=$) to produce atomic formulae. These, in turn, can be combined into compounds by means of propositional connectives. An example of a well-formed $\mathcal{L}^{\times}$-formula is: $(\overline{p_1 \circ p_5^{-1}} \circ \mathbb{1}) \cap p_3{=}(\overline{p_3 \circ p_1})^{-1} \vee p_1 \cap p_5 \neq \emptyset$.

An *interpretation* of the language $\mathcal{L}^{\times}$ is given by a pair $\Im = [\mathcal{U}, (\mathcal{R}_1, \mathcal{R}_2, \ldots)]$, where $\Im$ is a non-empty domain and each $\mathcal{R}_i$ is a binary relation over $\mathcal{U}$, i.e., $\mathcal{R}_i \subseteq \mathcal{U} \times \mathcal{U}$.

For any interpretation $\Im = [\mathcal{U}, (\mathcal{R}_1, \mathcal{R}_2, \ldots)]$, we convene that:

- $\iota^{\Im}$ is the identity relation on $\mathcal{U}$;
- $p_i^{\Im}$ is the binary relation $\mathcal{R}_i$;
- given two predicate expressions A and B,

$$\begin{array}{rcl}
(A \cap B)^{\Im} & =_{\text{Def}} & \{[x,y] \mid [x,y] \text{ in } A^{\Im} \cap B^{\Im}\}, \\
\overline{A}^{\Im} & =_{\text{Def}} & \{[x,y] \mid x, y \text{ in } \mathcal{U} \text{ and } [x,y] \text{ not in } A^{\Im}\}, \\
(A \circ B)^{\Im} & =_{\text{Def}} & \{[x,y] \mid \text{for some } z \text{ in } \mathcal{U},\ [x,z] \text{ in } A^{\Im} \text{ and } [z,y] \text{ in } B^{\Im}\}, \\
(A^{-1})^{\Im} & =_{\text{Def}} & \{[x,y] \mid [y,x] \text{ in } A^{\Im}\}.
\end{array}$$

An equality $A{=}B$ is true in $\Im$ if and only if $A^{\Im}$ and $B^{\Im}$ denote the same binary relation over $\mathcal{U}$. Finally, connectives are interpreted in the usual way.

The language $\mathcal{L}^{+}$

$\mathcal{L}^{+}$ is a version of first-order language, with the above p_is as predicate symbols and with atomic formulae of the following two types:

- xAy, where x and y are individual variables and A is an $\mathcal{L}^{\times}$-predicate expression;

- $A{=}B$, where A and B are $\mathcal{L}^{\times}$-predicate expressions.

An example of an $\mathcal{L}^{+}$-formula is $(\forall w, z)(\, w\overline{(\ell \circ p^{-1}) \cap \iota z} \leftrightarrow w(r \circ s^{-1} \circ \mathbb{1}) \cap \iota z\,) \wedge p \circ p^{-1} {=} \iota$.

The semantics of $\mathcal{L}^{+}$ is a usual first-order logic semantics, where predicate expressions and predicate equalities are interpreted as above.

Sufficient conditions for an $\exists^*$, $\forall\exists^*$, or $\forall\forall\exists^*$ prenex $\mathcal{L}^{+}$-sentence to be translatable into the variable-free formalism $\mathcal{L}^{\times}$ will be singled out in what follows. An efficient test based on such conditions will also be described. Through minor modifications of this testing algorithm, one can obtain the translation when the sufficient conditions are met.

3 Two translation problems

The present paper addresses the following two problems:

(1) Given a sentence α of $\mathcal{L}^{+}$ of the form

$$\alpha \leftrightarrow_{\text{Def}} \exists\, x_0\, \exists\, x_1 \cdots \exists\, x_N\, \exists\, x_{N+1}\, \left(\bigwedge_{h=1}^{H} x_{i_h}\, R_h\, x_{j_h} \right) ,$$

where each R_h is a predicate expression of $\mathcal{L}^{\times}$, establish whether α *can be translated* into $\mathcal{L}^{\times}$, in the sense that for a suitable predicate Q of $\mathcal{L}^{\times}$ the following holds:

$$\models^{+}\ \alpha \leftrightarrow Q \neq \emptyset\,.$$

(2) Given a formula $\varphi[x_0, x_{N+1}]$ of $\mathcal{L}^{+}$ of the form

$$\varphi[x_0, x_{N+1}] \leftrightarrow_{\text{Def}} \exists\, x_1 \cdots \exists\, x_N\, \left(\bigwedge_{h=1}^{H} x_{i_h}\, R_h\, x_{j_h} \right) ,$$

where, as above, R_h is a predicate expression of $\mathcal{L}^{\times}$, establish whether φ is *definable* in $\mathcal{L}^{\times}$, in the sense that for a suitable predicate Q of $\mathcal{L}^{\times}$the following holds:

$$\models^{+}\ \forall\, x_0\, \forall\, x_{N+1}\ \left(\varphi[x_0, x_{N+1}] \leftrightarrow x_0 Q x_{N+1}\right)\,.$$

Notice that problem (2) is more general than problem (1), since

- any predicate expression Q which solves problem (2), solves the corresponding problem (1) too;
- one can translate any instance of problem (1) into an instance of problem (2) by adding the two conjuncts $x_{-1}\mathbb{1}x_0$ and $x_{N+1}\mathbb{1}x_{N+2}$.

Moreover, w.l.o.g., in the formulation of problems (1) and (2) we can always require $i_h \leq j_h$ to hold for all $h \in \{1, \ldots, H\}$.

Even though apparently narrow, problem (2) deserves good treatment; indeed, an algorithm to solve it is an essential ingredient of clever techniques for translating $\mathcal{L}^{+}$-sentences into $\mathcal{L}^{\times}$.

4 Graph rendering of the problems

Let $\varphi[x_0, x_{N+1}]$ be a formula of $\mathcal{L}^+$ having the said form

$$\varphi[x_0, x_{N+1}] \leftrightarrow_{\text{Def}} \exists\, x_1 \cdots \exists\, x_N \left(\bigwedge_{h=1}^{H} x_{i_h}\, R_h\, x_{j_h} \right) ,$$

where R_h is a predicate of $\mathcal{L}^\times$ and $i_h \leq j_h$ for all $h \in \{1, \ldots, H\}$. We begin by associating with $\varphi[x_0, x_{N+1}]$ the labelled graph $G'_\varphi = (V', E')$, where $V' = \{\nu_0, \nu_1, \ldots, \nu_N, \nu_{N+1}\}$ and where a labelled directed arc $\nu_{i_h} \stackrel{R_h}{\longrightarrow} \nu_{j_h}$ is made to correspond in E' to each one of the atoms $x_{i_h} R_h x_{j_h}$ of φ.

Notice that G'_φ is a *dag* (i.e., acyclic), save for possible self-loops. However, each self-loop σ can be eliminated from G'_φ as follows. Introduce in V' a new node ν_σ and replace σ by the new arc $\nu \stackrel{R \cap \iota}{\longrightarrow} \nu_\sigma$, where σ is $\nu \stackrel{R}{\longrightarrow} \nu$; then replace every arc $\nu \stackrel{R'}{\longrightarrow} \nu'$ with $\nu' \neq \nu_\sigma$ in G'_φ by the new arc $\nu_\sigma \stackrel{R'}{\longrightarrow} \nu'$.

A graph G_φ will result at the end, acyclic by what we have managed to do, and to be called the *graph rendering of* φ. We call the nodes ν_0 and ν_{N+1} *preferred* source and sink of G_φ, respectively.

5 Solution method for the second problem

3-saturated and 3-embeddable directed acyclic graphs

DEFINITION 5.1 *A* 3-saturated *graph is a triple* $[G, s, t]$*, where* G *is a* dag*,* s *is a* source *(i.e, a node with no incoming edge) of* G*,* t *is a* sink *(i.e., a node with no outgoing edge) of* G*, and either*

- G *consists of the single arc* $s \longrightarrow t$*, or*
- G *can be obtained from two 3-saturated graphs* $[G', s', t']$ *and* $[G'', s'', t'']$*, having disjoint sets of nodes, in either of the following two ways:*
  - *by fusing* t' *with* s'' *and by putting* $s =_{\text{Def}} s'$ *and* $t =_{\text{Def}} t''$*;*
  - *by fusing* s' *with* s'' *and* t' *with* t''*, and by putting* $s =_{\text{Def}} \{s', s''\}$ *and* $t =_{\text{Def}} \{t', t''\}$.[1]

A dag G *is said to be* 3-embeddable *w.r.t. a source* s *and to a sink* t *if it can be extended into a 3-saturated graph* $[G', s, t]$ *based on the same nodes.* □

An algorithm to detect 3-embeddability

ALGORITHM 3-EMB
Input: A dag G with designated source s and sink t.
Output: Answer to the 3-embeddability problem for G.

- While there is a node v in G, with $v \neq s, t$, which is not isolated and such that it has at most one incoming edge and at most one outgoing edge, remove from G all edges incident to v; moreover, if v has exactly one incoming edge, $u \longrightarrow v$, and exactly one outgoing edge, $v \longrightarrow w$, then add to G the edge $u \longrightarrow w$, if not already present.
- If G contains at most the single edge $s \longrightarrow t$ then return "G is 3-embeddable" otherwise return "G is not 3-embeddable".

[1] Here $\{s', s''\}$ and $\{t', t''\}$ stand for the nodes obtained by fusing s' with s'', and t' with t'', respectively.

REMARK 5.2 It is not hard to see that algorithm 3-EMB admits an $\mathcal{O}(n^3)$ implementation, where n is the number of nodes in the input graph. □

Translating existential formulae from $\mathcal{L}^+$ to $\mathcal{L}^\times$

As usual, let $\varphi[x_0, x_{N+1}]$ be an instance of problem (2), and let G_φ be its graph rendering, with preferred source s and sink t. We have the following result.

LEMMA 5.3 *If G_φ is 3-embeddable, then φ can be translated in the $\mathcal{L}^\times$-formalism.* ■

To perform the translation of a given formula φ in the $\mathcal{L}^\times$-formalism, algorithm 3-EMB can be adapted in such a way that while it recognizes the 3-embeddability of the graph rendering G_φ of φ, it also constructs a 3-saturated graph G'_φ which extends G_φ (called a *3-completion* of G_φ) by adding edges labelled $\mathbb{1}$.

The resulting saturated graph G'_φ can then be processed by the following procedure to produce the seeked for translation.

ALGORITHM $\mathcal{L}^\times$-TRANSLATOR
Input: A tranlatable instance $\varphi[x_0, x_{N+1}]$ of problem (2).
Output: A translation of $(\forall x_0, x_{N+1})\varphi$ in the $\mathcal{L}^\times$-formalism.

- Let G_φ be the graph rendering of φ, with preferred source and sink s and t, and let G'_φ be the 3-completion of G_φ.
- While there is a node v in G'_φ, with $v \neq s, t$, having exactly one incoming edge, $u \xrightarrow{A} v$, and exactly one outgoing edge, $v \xrightarrow{B} w$, remove $u \xrightarrow{A} v$, $v \xrightarrow{B} w$, and v from G'_φ; if an edge $u \xrightarrow{C} w$ is present in G'_φ then substitute its label C with $C \cap (A \circ B)$, otherwise add to G'_φ a new edge $u \xrightarrow{A \circ B} w$, with label $A \circ B$.
- If G'_φ has been reduced to the single edge $s \xrightarrow{R} t$ then return "$x_0 R x_{N+1}$ is a $\mathcal{L}^\times$-translation of $\varphi[x_0, x_{N+1}]$".

6 Conclusions and envisaged extensions

Two problems that naturally generalize problem (2) of Sec.3 are:

(3) Given a formula $\varphi[x_0, x_{N+1}]$ of $\mathcal{L}^+$ of the form $\varphi[x_0, x_{N+1}] \leftrightarrow_{\text{Def}} \exists\, x_1 \cdots \exists\, x_N \left(\bigwedge_{h=1}^{H} x_{i_h}\, R_h\, x_{j_h} \right)$, where, as before, each R_h is a predicate expression of $\mathcal{L}^\times$, and assuming that a subset $F_1, \ldots, F_K$ of the set of R_hs has been declared to consist of single-valued maps, establish whether φ is *definable* in $\mathcal{L}^\times$, in the sense that for a suitable predicate Q of $\mathcal{L}^\times$ the following holds:

$$\bigwedge_{k=1}^{K} (\, F_k^{-1} \circ F_k \subseteq \iota \,) \models^+ \forall\, x_0\, \forall\, x_{N+1}\ (\varphi[x_0, x_{N+1}] \leftrightarrow x_0 Q x_{N+1})\ .$$

(4) Given a formula $\varphi[x_{N+1}, \ldots, x_{N+M}] \leftrightarrow_{\text{Def}} \exists\, x_1 \cdots \exists\, x_N \left(\bigwedge_{h=1}^{H} x_{i_h}\, R_h\, x_{j_h} \right)$, with $M > 2$, involving predicate constructs and quantifiers, establish whether φ is *definable* in purely relational terms.

It seems worth of investigation, concerning in particular problem (3), how additional information can suggest local transformations to the graph rendering of a given φ, that may enforce 3-embeddability. An illustration of the alluded kind of transformations will be seen in the last example in the appendix. Problem (4) requires, at least when $M > 3$, a generalization of our techniques to higher dimensions.

Well performing techniques for giving algebraic form to logical sentences are a presupposition for any fair comparison between, and fruitful combination of, reasoning methods directly rooted on predicate calculus and methods originating from algebra. Although less familiar to the majority of computer scientists than other formalisms successfully designed for database systems and for taxonomic reasoning, $\mathcal{L}^{\times}$ is the archetype of a rich variety of relational languages. This makes us confident that the translation techniques we are designing can be exported from the still somewhat esoteric context we have been focusing on, and find real applications.

To what extent one could take advantage of existing equational theorem-proving techniques to support a sophisticated translation from logic to algebra is at present unclear to the authors. At any rate, we count very much on the aid of such techniques for the proof-generation phase that should follow the translation phase.

References

[BL92] P. Broome and J. Lipton. Constructive relational programming: A declarative approach to program correctness and high level optimization. In *ARO Report 92-1, Transactions of the 9th Army Conference on Applied Mathematics and Computing*, 1992.

[FL94] P. Freyd and J. Lipton. 1994 report: ONR Contract Number 4331-001-srp-01, 1994, http://www.cs.wesleyan.edu/~lipton/ONR/www-eoyl94.html.

[TG87] A. Tarski and S. Givant. *A formalization of Set Theory without variables*, volume 41 of *Colloquium Publications*. American Mathematical Society, 1987.

Appendix: Examples drawn from elementary arithmetic

Conjugated projections ℓ and r

The following table contains a characterization in the formalisms $\mathcal{L}^+$ and $\mathcal{L}^\times$ of two conjugated projections, ℓ ed r, which are supposed to invert a pairing function over the natural numbers.

ℓ is a function	$(\forall\, x, y, z)(\ell(x, y) \wedge \ell(x, z) \rightarrow y = z)$	$(\ell \circ \overline{\iota}) \cap \ell = \emptyset$
r is a function		$(r \circ \overline{\iota}) \cap r = \emptyset$
ℓ is total	$(\forall\, x, u)\ \exists\, y\ (\ell(x, y) \wedge y \mathbb{1} u)$	$\ell \circ \mathbb{1} = \mathbb{1}$
r is total		$r \circ \mathbb{1} = \mathbb{1}$
ℓ and r invert	$(\forall\, x, y) \exists\, z\ (\ell(z, x) \wedge r(z, y))$	$\ell^{-1} \circ r = \mathbb{1}$
a pairing function	$\neg(\exists\, z, w)\ (\exists\, x, y)(\ell(z, x) \wedge \ell(w, x) \wedge r(z, y) \wedge r(w, y))$	$(\ell \circ \ell^{-1}) \cap (r \circ r^{-1}) = \emptyset$

Successor function s

s is an injective		$(s \circ \overline{\iota}) \cap s = \emptyset$
total function		$s \circ \mathbb{1} = \mathbb{1}$
	$(\forall\, x, y, z)(s^{-1}(x, y) \wedge s^{-1}(x, z) \rightarrow y = z)$	$(s^{-1} \circ \overline{\iota}) \cap s^{-1} = \emptyset$

All numbers but one are successor of some number

Here is a chain of transformations from $\mathcal{L}^+$ to $\mathcal{L}^\times$.

- $\exists\, z\, \forall\, y(z \neq y \leftrightarrow \exists\, x\, s(x, y))$
- $(\exists\, z, u)\ \neg \exists\, y\ \neg(z \neq y \leftrightarrow \exists\, x\ (s(x, y) \wedge x \mathbb{1} u))$
- $(\exists\, z, u)\ \neg \exists\, y\ ((z \neq y \wedge y\overline{s^{-1} \circ \mathbb{1}} u) \vee (z = y \wedge y s^{-1} \circ \mathbb{1} u))$
- $(\exists\, z, u)\ \neg(\exists\, y\ (z \neq y \wedge y\overline{s^{-1} \circ \mathbb{1}} u) \vee \exists\, y\ (z = y \wedge y s^{-1} \circ \mathbb{1} u))$
- $(\exists\, z, u)\ (\neg \exists\, y\ (z \neq y \wedge y\overline{s^{-1} \circ \mathbb{1}} u) \wedge \neg \exists\, y\ (z = y \wedge y s^{-1} \circ \mathbb{1} u))$
- $(\exists\, z, u)\ (z\overline{\overline{\iota} \circ \overline{s^{-1} \circ \mathbb{1}}} u \wedge z\overline{\iota \circ s^{-1} \circ \mathbb{1}} u))$
- $\overline{\overline{\iota} \circ \overline{s^{-1} \circ \mathbb{1}}} \cap \overline{\iota \circ s^{-1} \circ \mathbb{1}} \neq \emptyset$

s is acyclic

For all N,

- $\neg(\exists\, x_0, x_1, \ldots, x_N, x_{N+1})\ (s(x_0, x_1) \wedge s(x_1, s_2) \wedge \cdots \wedge s(x_N, x_{N+1}) \wedge x_0 = x_{N+1})$
- $(\underbrace{s \circ s \circ \cdots \circ s}_{N \text{ times}}) \cap \iota = \emptyset$

Properties of the sum function p

$v + 0 = v$

- $\forall\, w(\exists\, v\ (\ell(w, v) \wedge p(w, v)) \leftrightarrow \neg(\exists\, x, u)(r(w, u) \wedge s(x, u)))$;
- $(\forall\, w, z)(\neg \exists\, v\ (\ell(w, v) \wedge p(z, v) \wedge w = z) \leftrightarrow (\exists\, x, u)(r(w, u) \wedge s(x, u) \wedge w = z))$;
- $\overline{(\ell \circ p^{-1}) \cap \iota} = (r \circ s^{-1} \circ \mathbb{1}) \cap \iota$

$\boxed{x + s(y) = s(x + y)}$

$$\begin{array}{rrl} (\forall\, w,t)(& (\exists\, v,y)(& r(w,v) \wedge s(y,v) \wedge p(w,t)) \leftrightarrow \\ - & (\exists\, v,y, & x,u,z)(r(w,v) \wedge s(y,v) \wedge \ell(w,x) \wedge \\ & & \ell(u,x) \wedge r(u,y) \wedge p(u,z) \wedge s(z,t))\ )\ ; \end{array}$$

- $(r \circ s^{-1} \circ \mathbb{1}) \cap p = ((r \circ s^{-1} \circ r^{-1}) \cap (\ell \circ \ell^{-1})) \circ p \circ s.$

Properties of the product function m

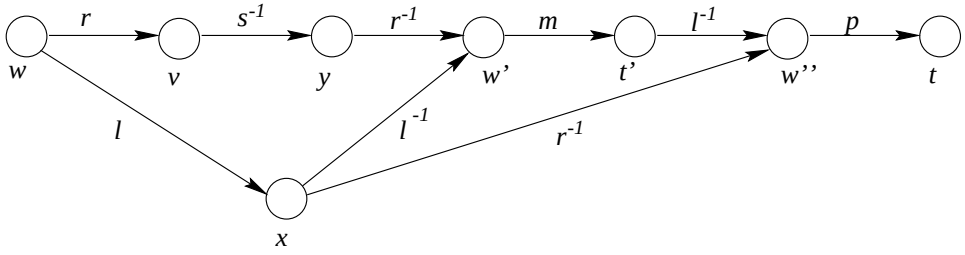
$\boxed{v \cdot 0 = 0}$

- $(\forall\, w,z)(\neg\exists\, v\, s(v,z) \rightarrow (m(w,z) \leftrightarrow \ell(w,z) \vee r(w,z)))$;
- $\overline{\mathbb{1} \circ s} \subseteq \overline{m \triangle (\ell \cup r)}$
- $m \triangle (\ell \cup r) \subseteq \mathbb{1} \circ s$

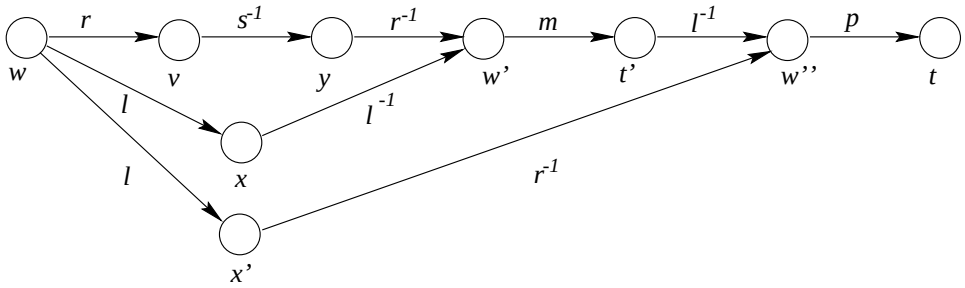
$\boxed{x \cdot s(y) = x \cdot y + x}$

$$\begin{array}{rrl} (\forall\, w,t)(& (\exists\, v,y)(& r(w,v) \wedge s(y,v) \wedge m(w,t)) \leftrightarrow \\ & (\exists\, v,y, & x,w',t',w'')(r(w,v) \wedge s(y,v) \wedge \ell(w,x) \wedge \\ & & r(w',y) \wedge \ell(w',x) \wedge m(w',t') \wedge \\ & & \ell(w'',t') \wedge r(w'',x) \wedge p(w'',t))\ )\ ; \end{array}$$

A graph rendering of the right-hand side of the above equivalence is the following:



It can easily be checked that the above graph is not 3-embeddable. Since ℓ has been characterized as a function, node x can be split into two nodes, by introducing a new node x', and the graph becomes



The above graph is 3-embeddable. Thus we obtain the following translation:

$$(r \circ s^{-1} \circ \mathbb{1}) \cap m = ((((r \circ s^{-1} \circ r^{-1}) \cap (\ell \circ \ell^{-1})) \circ m \circ \ell^{-1}) \cap (\ell \circ r^{-1})) \circ p\,.$$

■