

Reducing Predicate Logic to Propositional Logic Part II

Hajnal Andr eka and Istv an N emeti

R enyi Institute

LaPoM, December 9, 2011

Today we want to prove the following theorem.

L_ω = “usual FOL with countable many relation symbols of arbitrary arities”.

$\mathcal{L}d_3$ = “3-variable FOL with one ternary relation symbol and without equality or substitutions, together with a weak proof-system \vdash^d ”.

Theorem 1

L_ω can be reduced to $\mathcal{L}d_3$, i.e.,
there is a computable meaning preserving translation mapping
 $\text{Tr} : L_\omega \rightarrow \mathcal{L}d_3$ such that

$$\text{Ax} \models \varphi \Leftrightarrow \text{Tr}(\text{Ax}) \models \text{Tr}(\varphi) \Leftrightarrow \text{Tr}(\text{Ax}) \vdash^d \text{Tr}(\varphi)$$

for all $\text{Ax} \subseteq L_\omega$ and $\varphi \in L_\omega$.

We cut the statement into two parts, and prove the two parts.
First part:

Theorem 2

We define $\text{tr} : L_\omega \rightarrow \mathcal{L}d_3$ such that
 $Ax \models \varphi \Leftrightarrow \text{tr}(Ax) \models \text{tr}(\varphi)$, for all $Ax \subseteq L_\omega$ and $\varphi \in L_\omega$.

Second part:

Theorem 3

We define $\text{tr} : \mathcal{L}d_3 \rightarrow \mathcal{L}d_3$ such that for all $Ax \subseteq \mathcal{L}d_3$, $\varphi \in \mathcal{L}d_3$
 $Ax \models \varphi \Leftrightarrow \text{tr}(Ax) \models \text{tr}(\varphi) \Leftrightarrow \text{tr}(Ax) \not\models \text{tr}(\varphi)$.

The first part is the easier one, the second part is the harder and novel one. We begin by proving Thm.2.

Proof of Thm.2.

First we concentrate on the $Ax = \emptyset$ case. This part will consist of 3 steps, the middle one of these is the most important one. We begin with this “middle” step.

$$L_\omega \longrightarrow L_\omega \longrightarrow L_3 \longrightarrow \mathcal{L}d_3$$

Middle step: How do we get rid of the infinity of variables?

With the pairing functions technique (due to Tarski).

What is the pairing functions technique?

A technique to code two pieces of data into one.

Proof of Thm.2.

First we concentrate on the $Ax = \emptyset$ case. This part will consist of 3 steps, the middle one of these is the most important one. We begin with this “middle” step.

$$L_\omega \longrightarrow L_\omega \longrightarrow L_3 \longrightarrow \mathcal{L}d_3$$

Middle step: How do we get rid of the infinity of variables?

With the **pairing functions** technique (due to Tarski).

What is the pairing functions technique?

A technique to **code two pieces of data into one**.

We have two functions, p, q for which the following holds:

$$\pi \stackrel{d}{=} \forall xy \exists z (p(z) = x \wedge q(z) = y).$$

First we translate L_ω into a language which contains unary function symbols p, q in addition to the relation symbols of L_ω . Then we will eliminate these function symbols. In this middle step we also assume that in L_ω **we have only at most binary relation symbols**.

Writing up the recursive definitions of these translating functions is very much like programming. We will only tell the idea of the translating, we will not write up the actual code of the program.

We will describe the main step of the algorithm, and then we show the algorithm on an example.

Main step:

We want to translate $\exists w\varphi$, where $\exists w, \exists x, z$ do not occur in φ .

We write $u_0 \stackrel{d}{=} p(u)$ and $u_1 \stackrel{d}{=} q(u)$.

We often write just comma in place of \wedge .

Idea: code x, w into a “new” x such that

$(\text{new}x)_0 = x, (\text{new}x)_1 = w$:

$$\text{tr}(\exists w\varphi) \stackrel{d}{=} \exists z(z_0 = x, \exists x(x = z, \text{tr}(\varphi(x/x_0, w/x_1))))).$$

We will “do nothing” with the other connectives.

Drawing on the blackboard.

Question.

Example: $\varphi \stackrel{d}{=} \forall xy \exists z (R_{xz}, R_{yz})$.

Algorithm: First we rename the variables so that neither of $\exists x, z$ occurs in φ , and all quantifiers occur only once.

$\forall uy \exists v R_{uv}, R_{yv}$. This is the same as

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$. Now we will process the quantifiers.

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists y \neg \exists v R_{x_1 v}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z, \neg \exists v R_{x_{01} v}, R_{x_1 v}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z,$

$\neg \exists z z_0 = x, \exists x x = z, R_{x_{001} x_1}, R_{x_{01} x_1}$

Here is the result:

$\neg \exists z (z_0 = x, \exists x (x = z, \exists z (z_0 = x, \exists x (x = z,$
 $\neg \exists z (z_0 = x, \exists x (x = z, (R_{x_{001} x_1}, R_{x_{01} x_1}))))))$.

Example: $\varphi \stackrel{d}{=} \forall xy \exists z (R_{xz}, R_{yz})$.

Algorithm: First we rename the variables so that neither of $\exists x, z$ occurs in φ , and all quantifiers occur only once.

$\forall uy \exists v R_{uv}, R_{yv}$. This is the same as

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$. Now we will process the quantifiers.

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists y \neg \exists v R_{x_1 v}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z, \neg \exists v R_{x_{01} v}, R_{x_1 v}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z,$

$\neg \exists z z_0 = x, \exists x x = z, R_{x_{001} x_1}, R_{x_{01} x_1}$

Here is the result:

$\neg \exists z (z_0 = x, \exists x (x = z, \exists z (z_0 = x, \exists x (x = z, \neg \exists z (z_0 = x, \exists x (x = z, (R_{x_{001} x_1}, R_{x_{01} x_1}))))))$.

Example: $\varphi \stackrel{d}{=} \forall xy \exists z (R_{xz}, R_{yz})$.

Algorithm: First we rename the variables so that neither of $\exists x, z$ occurs in φ , and all quantifiers occur only once.

$\forall uy \exists v R_{uv}, R_{yv}$. This is the same as

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$. Now we will process the quantifiers.

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists y \neg \exists v R_{x_1 v}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z, \neg \exists v R_{x_{01} v}, R_{x_1 v}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z,$

$\neg \exists z z_0 = x, \exists x x = z, R_{x_{001} x_1}, R_{x_{01} x_1}$

Here is the result:

$\neg \exists z (z_0 = x, \exists x (x = z, \exists z (z_0 = x, \exists x (x = z, \neg \exists z (z_0 = x, \exists x (x = z, (R_{x_{001} x_1}, R_{x_{01} x_1}))))))$.

Example: $\varphi \stackrel{d}{=} \forall xy \exists z (R_{xz}, R_{yz})$.

Algorithm: First we rename the variables so that neither of $\exists x, z$ occurs in φ , and all quantifiers occur only once.

$\forall uy \exists v R_{uv}, R_{yv}$. This is the same as

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$. Now we will process the quantifiers.

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists y \neg \exists v R_{x_1 v}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z, \neg \exists v R_{x_01 v}, R_{x_1 v}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z,$

$\neg \exists z z_0 = x, \exists x x = z, R_{x_001 x_1}, R_{x_01 x_1}$

Here is the result:

$\neg \exists z (z_0 = x, \exists x (x = z, \exists z (z_0 = x, \exists x (x = z, \neg \exists z (z_0 = x, \exists x (x = z, (R_{x_001 x_1}, R_{x_01 x_1}))))))$.

Example: $\varphi \stackrel{d}{=} \forall xy \exists z (Rxz, Ryz)$.

Algorithm: First we rename the variables so that neither of $\exists x, z$ occurs in φ , and all quantifiers occur only once.

$\forall uy \exists v Ruv, Ryv$. This is the same as

$\neg \exists u \exists y \neg \exists v Ruv, Ryv$. Now we will process the quantifiers.

$\neg \exists u \exists y \neg \exists v Ruv, Ryv$

$\neg \exists z z_0 = x, \exists x x = z, \exists y \neg \exists v Rx_1 v, Ryv$

$\neg \exists z z_0 = x, \exists x x = z, \exists y \neg \exists v Rx_1 v, Ryv$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z, \neg \exists v Rx_{01} v, Rx_{1v}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z,$

$\neg \exists z z_0 = x, \exists x x = z, Rx_{001} x_1, Rx_{01} x_1$

Here is the result:

$\neg \exists z (z_0 = x, \exists x (x = z, \exists z (z_0 = x, \exists x (x = z, \neg \exists z (z_0 = x, \exists x (x = z, (Rx_{001} x_1, Rx_{01} x_1))))))$.

Example: $\varphi \stackrel{d}{=} \forall xy \exists z (Rxz, Ryz)$.

Algorithm: First we rename the variables so that neither of $\exists x, z$ occurs in φ , and all quantifiers occur only once.

$\forall uy \exists v Ruv, Ryv$. This is the same as

$\neg \exists u \exists y \neg \exists v Ruv, Ryv$. Now we will process the quantifiers.

$\neg \exists u \exists y \neg \exists v Ruv, Ryv$

$\neg \exists z z_0 = x, \exists x x = z, \exists y \neg \exists v Rx_1v, Ryv$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z, \neg \exists v Rx_{01}v, Rx_{1v}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z, \neg \exists v Rx_{01}v, Rx_{1v}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z,$

$\neg \exists z z_0 = x, \exists x x = z, Rx_{001}x_1, Rx_{01}x_1$

Here is the result:

$\neg \exists z (z_0 = x, \exists x (x = z, \exists z (z_0 = x, \exists x (x = z, \neg \exists z (z_0 = x, \exists x (x = z, (Rx_{001}x_1, Rx_{01}x_1))))))$.

Example: $\varphi \stackrel{d}{=} \forall xy \exists z (R_{xz}, R_{yz})$.

Algorithm: First we rename the variables so that neither of $\exists x, z$ occurs in φ , and all quantifiers occur only once.

$\forall uy \exists v R_{uv}, R_{yv}$. This is the same as

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$. Now we will process the quantifiers.

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists y \neg \exists v R_{x_1 v}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z, \neg \exists v R_{x_{01} v}, R_{x_1 v}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z,$

$\neg \exists z z_0 = x, \exists x x = z, R_{x_{001} x_1}, R_{x_{01} x_1}$

Here is the result:

$\neg \exists z (z_0 = x, \exists x (x = z, \exists z (z_0 = x, \exists x (x = z, \neg \exists z (z_0 = x, \exists x (x = z, (R_{x_{001} x_1}, R_{x_{01} x_1}))))))$.

Example: $\varphi \stackrel{d}{=} \forall xy \exists z (R_{xz}, R_{yz})$.

Algorithm: First we rename the variables so that neither of $\exists x, z$ occurs in φ , and all quantifiers occur only once.

$\forall uy \exists v R_{uv}, R_{yv}$. This is the same as

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$. Now we will process the quantifiers.

$\neg \exists u \exists y \neg \exists v R_{uv}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists y \neg \exists v R_{x_1 v}, R_{yv}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z, \neg \exists v R_{x_{01} v}, R_{x_1 v}$

$\neg \exists z z_0 = x, \exists x x = z, \exists z z_0 = x, \exists x x = z,$

$\neg \exists z z_0 = x, \exists x x = z, R_{x_{001} x_1}, R_{x_{01} x_1}$

Here is the result:

$\neg \exists z (z_0 = x, \exists x (x = z, \exists z (z_0 = x, \exists x (x = z, \neg \exists z (z_0 = x, \exists x (x = z, (R_{x_{001} x_1}, R_{x_{01} x_1}))))))$.

Let tr denote the just outlined algorithm.

Claim 1

$\pi \models \varphi \leftrightarrow \text{tr}(\varphi)$ *holds for all sentences φ of L_ω .*

How do we **get rid of the function symbols**?

We consider p, q not as function symbols but as binary relations:

$p(x, y) \stackrel{d}{=} p(x) = y$, $q(x, y) \stackrel{d}{=} q(x) = y$. In the next step we use that **we have only at most binary relation symbols** in the language.

Example:

Rx_0x_1

$\exists yz(Ryz, x_0 = y, x_1 = z)$

$\exists yz(Ryz, x_0 = y, x_1 = z)$

$\exists yz(Ryz, \exists z(x_0 = z, z_1 = y), x_1 = z)$

$\exists yz(Ryz, \exists z(x_0 = z, z_1 = y), x_1 = z)$

$\exists yz(Ryz, \exists z(\exists y(x_0 = y, y_1 = z), z_1 = y), x_1 = z)$.

We now define $\text{tr}_2(\varphi) \stackrel{d}{=} \pi \rightarrow \text{tr}(\varphi)$. Then we have

$$\models \pi \rightarrow \varphi \quad \Leftrightarrow \quad \models \text{tr}_2(\varphi).$$

Where do we get the pairing functions from? We do this in the first step:

$$L_\omega \longrightarrow L_\omega^{bin} \longrightarrow L_3 \longrightarrow \mathcal{L}d_3$$

Idea: In L_ω the relation symbols are R_1, R_2, \dots . We introduce a new binary relation symbol E (for “element”) and a new unary relation symbol M (for “old universe”). Then from E we define p, q , we state that these are pairing functions, and by using the pairing functions we code the relations M, R_1, R_2, \dots together into a new unary relation R . At the end we will have the following relation symbols: E, R .

Let $\mathfrak{M} \stackrel{d}{=} \langle M, R_1, \dots \rangle$ be an arbitrary model for L_ω . We construct another model $\mathfrak{M}^+ \stackrel{d}{=} \langle U, E, M, R_1, \dots \rangle$ from this, the following way.

U is the closure of M via taking finite subsets successively, and E is the so obtained elementhood relation.

In more detail: We may assume that all elements of M are singletons. Now,

$$U_0 \stackrel{d}{=} M, E_0 \stackrel{d}{=} \emptyset,$$

$$U_{n+1} \stackrel{d}{=} U_n \cup \{X \subseteq U_n : X \text{ is finite}\},$$

$$E_{n+1} \stackrel{d}{=} E_n \cup \{\langle a, X \rangle : X \in U_{n+1}, a \in X\},$$

$$U \stackrel{d}{=} \bigcup \{U_n : n \in \omega\}, E \stackrel{d}{=} \bigcup \{E_n : n \in \omega\}.$$

Recall that in set theory we can express pairs as follows:

$$\langle a, b \rangle \stackrel{d}{=} \{\{a\}, \{a, b\}\}.$$

Fortunately, from E we can express the so obtained pairing functions with using only three variables:

$$u\varepsilon v \stackrel{d}{=} E(u, v), \quad \text{for } u, v \in \{x, y, z\},$$

$$x \dot{=} \{y\} \stackrel{d}{=} \forall z(z\varepsilon x \leftrightarrow z = y),$$

$$\{x\}\varepsilon y \stackrel{d}{=} \exists z(z \dot{=} \{x\}, z\varepsilon y),$$

$$x \dot{=} \{\{y\}\} \stackrel{d}{=} \exists z(z \dot{=} \{y\}, x \dot{=} \{z\}),$$

$$x\varepsilon \cup y \stackrel{d}{=} \exists z(x\varepsilon z, z\varepsilon y),$$

$$\text{op}(x) \stackrel{d}{=} \exists y \forall z(\{z\}\varepsilon x \leftrightarrow y = z) \wedge$$

$$\forall yz[(y\varepsilon \cup x, \neg\{y\}\varepsilon x, z\varepsilon \cup x, \neg\{z\}\varepsilon x) \rightarrow y \dot{=} z],$$

$$\forall y \exists z(y\varepsilon x \rightarrow z\varepsilon y),$$

$$p \stackrel{d}{=} \text{op}(x) \wedge \{y\}\varepsilon x,$$

$$q \stackrel{d}{=} \text{op}(x) \wedge [x \dot{=} \{\{y\}\} \vee (y\varepsilon \cup x, \neg\{y\}\varepsilon x)].$$

Now, p, q are formulas with free variables x, y written up from $E, =$. Also, it can be checked that in \mathfrak{M}^+

$$p(\langle a, b \rangle, y) \rightarrow y = a, \quad q(\langle a, b \rangle, y) \rightarrow y = b.$$

Thus in the models \mathfrak{M}^+ even the **strong pairing** formula is true for these p, q :

$$\mathfrak{M}^+ \models \pi^+$$

where

$$\begin{aligned} \pi^+ &\stackrel{d}{=} \pi \wedge \\ &\forall xy [\exists z (p(x, z), p(y, z)), \exists z (q(x, z), q(y, z)) \rightarrow x = y] \wedge \\ &\forall x (\exists y p(x, y) \leftrightarrow \exists y q(x, y)) \wedge \exists xy (x \neq y). \end{aligned}$$

Finally, we compress the relations M, R_1, \dots into a single unary relation R with the help of the pairing functions p, q : Recall that p, q are partial functions, they are not defined on each element. In the following, e.g., $u_{00} = u_{00}$ denotes that “ u_{00} exists”, i.e. that p is defined on u and p is defined on $p(u)$.

$$\mathcal{R}_M \stackrel{d}{=} \{u \in U : u_{00} = u_{00} \rightarrow M(u_{00})\},$$

$$\mathcal{R}_1 \stackrel{d}{=} \{u \in U : u_{10} = u_{10} \rightarrow R_1(u_{100}, u_{1010}, u_{1011})\},$$

etc, and then

$$R \stackrel{d}{=} \mathcal{R}_M \cap \bigcap \{\mathcal{R}_i : i \in \omega\},$$

$$\mathfrak{M}^{++} \stackrel{d}{=} \langle U, E, M, R \rangle.$$

We can express the old relations from R as follows:

$$Mx \stackrel{d}{=} \exists u (Ru, x = u_{00}),$$

$$R_1xyz \stackrel{d}{=} \exists u (Ru, x = u_{100}, y = u_{1010}, z = u_{1011}),$$

etc.

Drawing on the blackboard.

We can write up the translation function: Let $\varphi \in L_w$. Then φ^r denotes the **relativization** of φ to the unary relation symbol M : “relativization” means “restriction”.

$$R_1(xyz)^r \stackrel{d}{=} R_1(xyz) \wedge Mx \wedge My \wedge Mz,$$

$$(\neg\varphi)^r \stackrel{d}{=} \neg(\varphi^r), \dots, (\exists x\varphi)^r \stackrel{d}{=} \exists x(Mx \wedge \varphi^r), \text{ etc.}$$

Also, φ^{rR} denotes the formula we get from φ^r by replacing the relation symbols M, R_1, \dots with their definitions from R .

Now, $\mathfrak{M} \models \varphi \Leftrightarrow \mathfrak{M}^+ \models \varphi^r \Leftrightarrow \mathfrak{M}^{++} \models \varphi^{rR}$, and so

$$\models \varphi \Leftrightarrow \models \pi \rightarrow \varphi^r \Leftrightarrow \models \pi \rightarrow \varphi^{rR}.$$

We define $\text{tr}_1(\varphi) \stackrel{d}{=} \pi \rightarrow \varphi^{rR}$. Then

$$\models \varphi \Leftrightarrow \models \text{tr}_1(\varphi).$$

We can write up the translation function: Let $\varphi \in L_\omega$. Then φ^r denotes the **relativization** of φ to the unary relation symbol M : “relativization” means “restriction”.

$$R_1(xyz)^r \stackrel{d}{=} R_1(xyz) \wedge Mx \wedge My \wedge Mz,$$

$$(\neg\varphi)^r \stackrel{d}{=} \neg(\varphi^r), \dots, (\exists x\varphi)^r \stackrel{d}{=} \exists x(Mx \wedge \varphi^r), \text{ etc.}$$

Also, φ^{rR} denotes the formula we get from φ^r by replacing the relation symbols M, R_1, \dots with their definitions from R .

Now, $\mathfrak{M} \models \varphi \Leftrightarrow \mathfrak{M}^+ \models \varphi^r \Leftrightarrow \mathfrak{M}^{++} \models \varphi^{rR}$, and so

$$\models \varphi \Leftrightarrow \models \pi \rightarrow \varphi^r \Leftrightarrow \models \pi \rightarrow \varphi^{rR}.$$

We define $\text{tr}_1(\varphi) \stackrel{d}{=} \pi \rightarrow \varphi^{rR}$. Then

$$\models \varphi \Leftrightarrow \models \text{tr}_1(\varphi).$$

We can write up the translation function: Let $\varphi \in L_\omega$. Then φ^r denotes the **relativization** of φ to the unary relation symbol M : “relativization” means “restriction”.

$$R_1(xyz)^r \stackrel{d}{=} R_1(xyz) \wedge Mx \wedge My \wedge Mz,$$

$$(\neg\varphi)^r \stackrel{d}{=} \neg(\varphi^r), \dots, (\exists x\varphi)^r \stackrel{d}{=} \exists x(Mx \wedge \varphi^r), \text{ etc.}$$

Also, φ^{rR} denotes the formula we get from φ^r by replacing the relation symbols M, R_1, \dots with their definitions from R .

Now, $\mathfrak{M} \models \varphi \Leftrightarrow \mathfrak{M}^+ \models \varphi^r \Leftrightarrow \mathfrak{M}^{++} \models \varphi^{rR}$, and so

$$\models \varphi \Leftrightarrow \models \pi \rightarrow \varphi^r \Leftrightarrow \models \pi \rightarrow \varphi^{rR}.$$

We define $\text{tr}_1(\varphi) \stackrel{d}{=} \pi \rightarrow \varphi^{rR}$. Then

$$\models \varphi \Leftrightarrow \models \text{tr}_1(\varphi).$$

We can write up the translation function: Let $\varphi \in L_\omega$. Then φ^r denotes the **relativization** of φ to the unary relation symbol M : “relativization” means “restriction”.

$$R_1(xyz)^r \stackrel{d}{=} R_1(xyz) \wedge Mx \wedge My \wedge Mz,$$

$$(\neg\varphi)^r \stackrel{d}{=} \neg(\varphi^r), \dots, (\exists x\varphi)^r \stackrel{d}{=} \exists x(Mx \wedge \varphi^r), \text{ etc.}$$

Also, φ^{rR} denotes the formula we get from φ^r by replacing the relation symbols M, R_1, \dots with their definitions from R .

Now, $\mathfrak{M} \models \varphi \Leftrightarrow \mathfrak{M}^+ \models \varphi^r \Leftrightarrow \mathfrak{M}^{++} \models \varphi^{rR}$, and so

$$\models \varphi \Leftrightarrow \models \pi \rightarrow \varphi^r \Leftrightarrow \models \pi \rightarrow \varphi^{rR}.$$

We define $\text{tr}_1(\varphi) \stackrel{d}{=} \pi \rightarrow \varphi^{rR}$. Then

$$\models \varphi \Leftrightarrow \models \text{tr}_1(\varphi).$$

This finishes the first step of the first part. Remains the third step of the first part.

Now we turn to the third step, which is **getting rid of identity and substitutions**.

$$L_\omega \longrightarrow L_\omega^{bin} \longrightarrow L_3 \longrightarrow \mathcal{L}d_3$$

In L_3 we have equality and two relation symbols: $=, E, R$. With the equality at hand, we can do away with the substitutions by Tarski's trick for eliminating substitutions:

translate $E(xz)$ to $\exists y(y = z \wedge E(x, y))$, etc.

Thus we may assume that the only atomic formulas of L_3 are $x = y, y = z, E(x, y), R(x)$.

In $\mathcal{L}d_3$ the only atomic formula is $P(x, y, z)$, we will express the atomic formulas $x = y, y = z, E(x, y), R(x)$ of L_3 with this. This step is similar to the first step.

Let $\mathfrak{M} \stackrel{d}{=} \langle U, E, R \rangle$ be a model of L_3 . Define

$\mathfrak{M}^+ \stackrel{d}{=} \langle U, P \rangle$ where

$$P \stackrel{d}{=} \{ \langle u, u, u \rangle : u \in U \} \cup \{ \langle a, b, c \rangle : E(a, b) \vee R(c) \}.$$

We define $x = y$, $y = z$, $E(x, y)$, $R(x)$ from $P(x, y, z)$ as follows:
 We write P_{xyz} for $P(x, y, z)$ etc.

$$Exy \stackrel{d}{=} \forall z P_{xyz},$$

$$Rz \stackrel{d}{=} \forall x \forall y P_{xyz},$$

$$x \dot{=} y \stackrel{d}{=} \forall z (P_{xyz} \wedge \neg Exy \wedge \neg Rz),$$

$$y \dot{=} z \stackrel{d}{=} \forall x (P_{xyz} \wedge \neg Exy \wedge \neg Rz).$$

We define the formula Eq as follows:

$$Eq \stackrel{d}{=} \{u \dot{=} v \leftrightarrow \exists w (u \dot{=} w, w \dot{=} v), \forall u \exists w (u \dot{=} w),$$

$$Euv \wedge u \dot{=} w \rightarrow Ewv, \quad Euv \wedge v \dot{=} w \rightarrow Euw,$$

$$Ru \wedge u \dot{=} w \rightarrow Rw, \quad \exists xy \neg x \dot{=} y : \{u, v, w\} = \{x, y, z\}\}.$$

Let $\varphi \in L_3$. Then $\varphi(E, R, = / P)$ denotes the formula we get from φ by replacing the atomic formulas $x = y, y = z, E(x, y), R(x)$ of L_3 with the above defined ones. Then we define:

$$\text{tr}_3(\varphi) \stackrel{d}{=} \text{Eq} \rightarrow \varphi(E, R, = / P).$$

This will work for non-one-element models. We can complicate this a bit so that it works for all models, i.e.,

$$\models \varphi \Leftrightarrow \models \text{tr}_3(\varphi).$$

Now we define $\text{Tr}(\varphi) \stackrel{d}{=} \text{tr}_3\text{tr}_2\text{tr}_1(\varphi)$.

Now, introduce Ax.

Proof of Thm.2 is finished.