

A Graph Calculus for Predicate Logic*

Paulo A. S. Veloso

COPPE-UFRJ
Systems and Computer Engin. Program
UFRJ: Federal University of Rio de Janeiro
RJ, Brazil
pasveloso@gmail.com

Sheila R. M. Veloso

FEN-UERJ
Systems and Computer Engin. Dept., Fac. of Engineering
UERJ: State University of Rio de Janeiro
RJ, Brazil
sheila.murgel.bridge@gmail.com

We introduce a refutation graph calculus for classical first-order predicate logic, which is an extension of previous ones for binary relations. One reduces logical consequence to establishing that a constructed graph has empty extension, i. e. it represents \perp . Our calculus establishes that a graph has empty extension by converting it to a normal form, which is expanded to other graphs until we can recognize conflicting situations (equivalent to a formula and its negation).

1 Introduction

We present a refutation graph calculus for classical first-order predicate logic. This approach is based on reducing logical consequence to showing that a constructed graph has empty extension, representing the logical constant \perp . Our sound and complete calculus establishes when a graph has empty extension.

For instance, given formulas ψ , θ and ϕ , to establish that ϕ follows from $\{\psi, \theta\}$, we construct a graph G corresponding to $\{\psi, \theta\} \cup \{\neg\phi\}$ and show that G has empty extension. Now, our calculus establishes that a graph has empty extension by converting it to a normal form, which is expanded to other graphs until we can recognize conflicting situations (equivalent to a formula and its negation).

Formulas are often written down on a single line [3]. Graph calculi rely on two-dimensional representations providing better visualization [2].¹ In the realm of binary relations, a simple calculus (with linear derivations) [2, 3] was extended for handling complement: direct calculi [5, 7] and refutation calculi [13]. Our new calculus is a further extension, inheriting much of the earlier terminology (such as ‘graph’, ‘slice’ and ‘arc’), together with some ideas from Peirce’s diagrams for relations [11, 4]. The present calculus involves two new aspects: extension to arbitrary predicates (which affects the representation) and allowing formulas within the graphs.

The structure of this paper is as follows. Section 2 motivates the underlying ideas with some illustrative examples. Section 3 introduces our graph language: syntax, semantics and some constructions. In Section 4 we introduce our graph calculus: its rules and goal. Section 5 presents some concluding remarks, including comparison with related works.

2 Motivation

We begin by motivating our ideas with some illustrative examples.

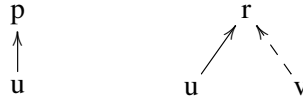
*Research partly sponsored by the Brazilian agencies CNPq and FAPERJ.

¹The structure of $(x + y) \cdot (z \div w)$ is more apparent in the notation $\begin{pmatrix} x \\ + \\ y \end{pmatrix} \cdot \begin{pmatrix} z \\ \div \\ w \end{pmatrix}$ (see also [1]).

We know that consequence can be reduced to unsatisfiability. We will indicate how one can represent formulas graphically and then establish consequence by graphical means.

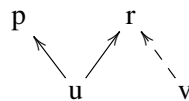
First, we indicate how we can represent (some) formulas graphically (see 3.1 for more details).

We represent an atomic formula by arrows to predicate symbols coming from its arguments. So, we represent the formulas $p(u)$ and $r(u, v)$, respectively, as follows:



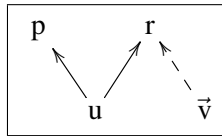
The former illustrates a 1-ary arc. The latter is an example of a 2-ary arc, which will be satisfied by the choices of values a and b for u and v , respectively, with the pair (a, b) in the 2-ary relation interpreting r .

We obtain a representation for a conjunction by joining those of its formulas. So, we represent the formula $p(u) \wedge r(u, v)$ as follows:



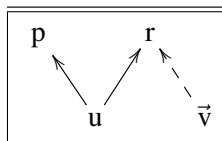
This set of 2 arcs is an example of a draft, which also represents the set $\{p(u), r(u, v)\}$. This draft D will be satisfied exactly by the assignments satisfying both $p(u)$ and $r(u, v)$.

To represent an existential quantification, we hide the node corresponding to the quantified variable, leaving only the rest visible. For instance, from formula $p(u) \wedge r(u, v)$, we obtain $\exists x (p(x) \wedge r(x, v))$. We can use the representation of the former to represent the latter: we place the above draft D within a box and mark v as visible, which we represent as follows:



This is an example of a 1-ary slice. The interpretation of this slice S is the 1-ary relation consisting of the values b such that, for some a , the assignment $u \mapsto a, v \mapsto b$ satisfies the underlying draft D .

Now, we can represent formula $\neg \exists x (p(x) \wedge r(x, v))$ by complementing this slice S . As \neg stands for complement, we represent $\neg \exists x (p(x) \wedge r(x, v))$ as follows:

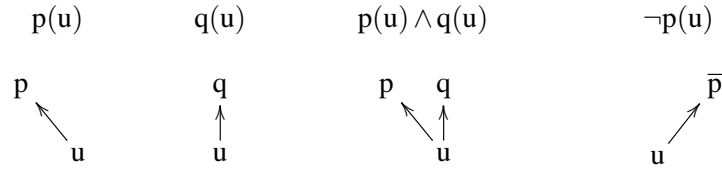


A draft consists of finite sets of names and of arcs (giving constraints on the names). A slice consists of a draft and a list of distinguished names, which we indicate by special marks, such as ' \rightarrow '.

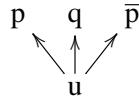
Next, we illustrate how one can establish consequence by graphical means. The idea is reducing unsatisfiability of a (finite) set of formulas to that of its corresponding draft.

We begin with an example that is basically propositional. Then, we examine other examples with equality \doteq and existential quantifiers (see 3.2 and 4.1 for more details).

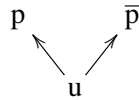
Example 2.1. Consider $p(u) \wedge q(u) \models p(u)$. As mentioned, we reduce it to $\{p(u) \wedge q(u), \neg p(u)\} \models \perp$. We can represent the formulas by (sets of) arcs as follows:



We can obtain a representation for the set $\{p(u) \wedge q(u), \neg p(u)\}$ by joining those of its formulas:



Within this draft for $\{p(u) \wedge q(u), \neg p(u)\}$, we find the conflicting situation (as $\bar{}$ stands for complement):



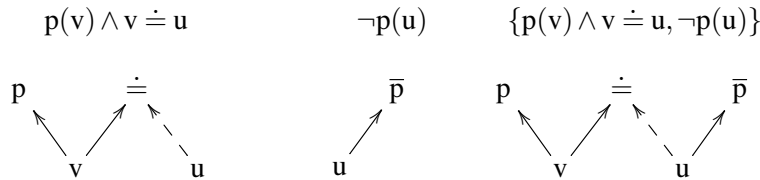
Thus, the representation of $\{p(u) \wedge q(u), \neg p(u)\}$ is unsatisfiable.

Example 2.2. We know that $p(u) \not\models p(v)$, i. e. $\{p(u), \neg p(v)\} \not\models \perp$. The corresponding draft is:

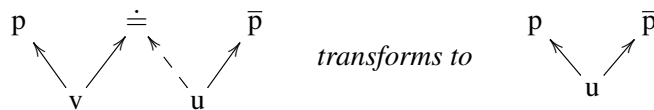


Here, we do not find conflicting arcs.² In fact, we can read from the representation a model $\mathfrak{M} = \langle M, p^{\mathfrak{M}} \rangle$, with $M := \{u, v\}$ and $p^{\mathfrak{M}} := \{u\}$, where one can satisfy $p(u)$ and $\neg p(v)$.

Example 2.3. We reduce $p(v) \wedge v \doteq u \models p(u)$ to the unsatisfiability of the set $\{p(v) \wedge v \doteq u, \neg p(u)\}$. We have the graphical representations as sets of arcs as follows:



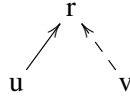
Now, we can simplify the representation of $\{p(v) \wedge v \doteq u, \neg p(u)\}$, by renaming v to u :



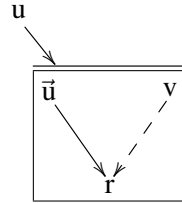
This final representation is not satisfiable (cf. Example 2.1).

Example 2.4. We reduce $r(u, v) \models \exists z r(u, z)$ to $\{r(u, v), \neg \exists z r(u, z)\} \models \perp$. As before, we can represent formula $r(u, v)$ by the single-arc draft:

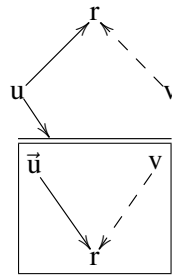
²Indeed, we have: $\begin{matrix} p \\ \uparrow \\ u \end{matrix}$ but not $\begin{matrix} \bar{p} \\ \uparrow \\ u \end{matrix}$ and $\begin{matrix} \bar{p} \\ \uparrow \\ v \end{matrix}$ but not $\begin{matrix} p \\ \uparrow \\ v \end{matrix}$.



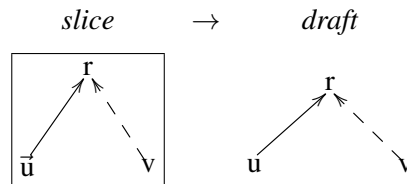
Also, we can represent $\neg\exists zr(u, z)$ by the following 1-ary arc:



Thus, we can represent $\{r(u, v), \neg\exists zr(u, z)\}$ by the draft:

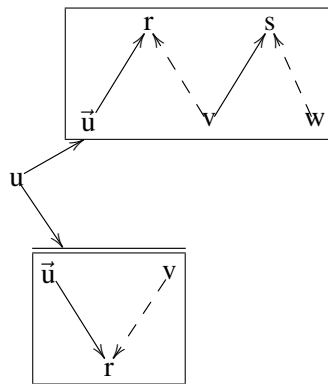


Now, with $\vec{u} \mapsto u, v \mapsto v$, we have a copy of the slice under complement within the draft, namely:

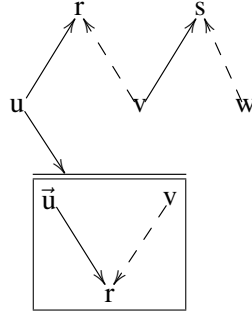


So, the representation of $\{r(u, v), \neg\exists zr(u, z)\}$ is not satisfiable.

Example 2.5. We reduce $\exists x\exists y[r(u, x) \wedge s(x, y)] \models \exists zr(u, z)$ to $\{\exists x\exists y[r(u, x) \wedge s(x, y)], \neg\exists zr(u, z)\} \models \perp$. Proceeding as before, we can be represent $\{\exists z\exists y[r(x, z) \wedge s(z, y)], \neg\exists zr(u, z)\}$ as:



We can transform this representation into the following one, which is, much as before, unsatisfiable.



3 Graph Language

We now introduce our concepts: expressions, slices and graphs will give relations, whereas arcs, sketches and drafts will correspond to constraints. We will examine syntax and semantics (in 3.1) and then some concepts and constructions (in 3.2).

We first introduce some notations. Given a function $f : A \rightarrow B$, we use $f(a)$ or a^f for its *value* at an element $a \in A$; which we extend to lists and sets. For a list $a = \langle a_1, \dots, a_k \rangle \in A^k$, we use $f(a)$ or a^f for the *list of values* $\langle a_1^f, \dots, a_k^f \rangle \in B^k$; for a set N , we use $f(N)$ or N^f for the *set of values* $\{a^f : a \in N\}$. Given a list $a = \langle a_1, \dots, a_k \rangle \in A^k$, we employ \underline{a} for its *set of components*. The *null list* is $\lambda := \langle \rangle$. We sometimes write a list $\langle a_1, \dots, a_k \rangle$ simply as $a_1 \dots a_k$.

We will use names (or parameters) for marking free places and variables for marking bound places, as usual in Proof Theory [12]. To quantify a formula φ we replace a name u by a new variable (not appearing in φ) obtaining $\exists x \varphi[u/x]$ and $\forall x \varphi[u/x]$. Also, given lists u , of n distinct names, and x , of n distinct variables not occurring in φ , we have the formulas $\exists^n x \varphi[u/x]$ and $\forall^n x \varphi[u/x]$.

We will consider first-order predicate languages (without function symbols, except the constant \perp), each one characterized by pairwise disjoint sets as follows:

- (Nm) an infinite linearly ordered *set of names* Nm ;
- (Nr) a denumerably infinite *set of variables* Vr ;
- (Pr) (possibly empty, but pairwise disjoint) sets Pr_n of n -ary *predicate symbols*, for $n \in \mathbb{N}$.

Given $m \in \mathbb{N}_+$, we use u_m for the m th name. Given $n \in \mathbb{N}$, we use $u^n := \langle u_1, \dots, u_n \rangle$ for the *list of the first n names* (with $u^0 = \lambda$). Also, given a set $v \subseteq Nm$ of names, we use \vec{v} for the list of the names in v in the ordering of Nm . For a formula φ , we use $NF[\varphi]$ for the *set of names occurring in φ* .

3.1 Syntax and semantics

We now introduce the syntax and semantics of our concepts. We first examine the syntax of our concepts.

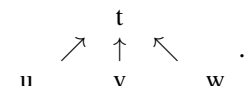
The objects of our graph language are defined by mutual recursion as follows.

- (E) An n -ary *expression* is an n -ary predicate symbol, a formula with n names, an n -ary slice or graph (see below), or \bar{E} , where E is an n -ary expression. For instance, \perp is a 0-ary expression, \doteq and $\ddot{=}$ are 2-ary expressions, whereas $p(u)$ and \bar{p} (for $p \in Pr_1$) are 1-ary expressions.
- (a) An m -ary *arc* a over set $N \subseteq Nm$ of names is a pair E/v (also noted $\frac{E}{v}$), where E is an m -ary expression and $v \in N^m$. Examples are \perp/λ , $\doteq/u v$, \bar{p}/u , $q(u)/v$ (for $p, q \in Pr_1$) and $s/u w$ (for $s \in Pr_2$).
- (Σ) A *sketch* $\Sigma = \langle N, A \rangle$ consists of sets $N \subseteq Nm$ of *names* and A of *arcs* over N .

- (D) A *draft* $D = \langle N, A \rangle$ is a sketch with finite sets N of names and A of arcs. An example of draft is $D' = \langle \{u, u', v, w, w'\}, \{\bar{p}/u, q(u)/v, \dot{=} / w w', s / u w\} \rangle$.
- (S) An n -ary *slice* $S = \langle \underline{S} : \hat{s} \rangle$ consists of its *underlying draft* $\underline{S} := \langle N, A \rangle$ and a *distinguished list* \hat{s} , with $\hat{s} \in N^n$. For instance, $S = \langle \{u, u', v, w, w'\}, \{\bar{p}/u, q(u)/v, \dot{=} / w w', s / u w\} : u v v \rangle$ is a 3-ary slice with underlying draft $\underline{S} = D'$ (as above) and distinguished list $\hat{s} = \langle u, v, v \rangle$.
- (G) An n -ary *graph* is a finite set of n -ary slices.

In particular, the empty graph $\{ \}$ has no slice. Example 4.5 (in 4.2) will show a 2-slice graph.

Note that expressions, arcs, slices and graphs are finite objects, whereas sketches are not necessarily so. Sketches will be useful for representing models and constructing co-limits. Also, some concepts and results do not depend on finiteness (see 3.2), which will be important in Section 4. We wish to represent these finite objects graphically by drawings (cf. the examples in Section 2). For this purpose, we employ two sorts of nodes: name nodes (labeled by names) and expression nodes (labeled by expressions). Some representations aiming at precision and readability are as follows.

We represent an m -ary arc E/v , with $v = \langle v_1, \dots, v_m \rangle$, by m arrows connecting each node labeled by v_i to the node labeled by E . For instance, we can draw a 3-ary arc $t/\langle u, v, w \rangle$ as 

To clarify (as in $\frac{t}{u v u}$), we may use distinct kinds of lines or label them by numbers.³ A more compact version uses $v_1 \xrightarrow{r} v_2$ for the 2-arc $r/v_1 v_2$, representing 1-ary, 3-ary and 4-ary arcs, respectively, as:

$$\begin{array}{c} p \\ | \\ v_1 \end{array}, v_1 \xrightarrow{t} v_3 \text{ and } v_1 \xrightarrow{q} v_4 .$$

$$\begin{array}{c} \vdots \\ v_2 \end{array} \xrightarrow{\text{dotted}} v_3 \quad \begin{array}{c} \vdots \\ v_2 \end{array} \xrightarrow{\text{wavy}} v_3$$

We can indicate the components of a distinguished list by marking their nodes, say with numbers, e. g. $\langle u, v, u \rangle$ by $u^{1,3}v^2$. Also, it may be convenient (for easier visualization) to enclose a slice S within a full box, \boxed{S} , and a graph G within a dashed box, \boxed{G} . For instance, Example 2.5 (in Section 2) shows a 0-ary slice $S = \langle \{u, v, w\}, A : \lambda \rangle$, with $A = \{r/uv, s/vw, \bar{T}/u\}$, where T is the 1-ary slice $\langle \{u, v\}, \{r/uv\} : u \rangle$.

Given a list w of names, the *arcless w slice* is the slice $\top_w := \langle \underline{w}, \emptyset : w \rangle$. The *arcless m -ary slice* is the slice $\top_m := \top_{u^m}$ (u^m is the list of the first m names) and the *m -node arcless draft* is $\bar{\top}_m = \langle u^m, \emptyset \rangle$. The *arc of formula* ϕ , with set v of names, is $a[\phi] := \phi/\bar{v}$. (The arc of a sentence τ is 0-ary: $a[\tau] = \tau/\lambda$, which we represent as the expression node τ .) The *sketch of the set of arcs* A is the sketch $\text{Sk}[A] := \langle N, A \rangle$, where N consists of the names occurring in the arcs of A : $N := \bigcup \{ \underline{w} \subseteq Nm : E/w \in A \}$. For instance, $\text{Sk}[\{s/uv, p(v)/w\}] = \langle \{u, v, w\}, \{s/uv, p(v)/w\} \rangle$.

We may wish to add an arc $a = E/v$ to a sketch, a slice or a graph. For a sketch $\Sigma = \langle N, A \rangle$, we set $\Sigma + a := \langle N \cup \underline{v}, A \cup \{a\} \rangle$; for a slice $S = \langle \underline{S} : \hat{s} \rangle$, we set $S + a := \langle \underline{S} + a : \hat{s} \rangle$; for a graph G , we set $G + a := \{ S + a : S \in G \}$. The *difference slice* of a finite set of arcs A with respect to an arc $a = E/v$ is the 0-ary slice $DS[A \angle a] := \langle \text{Sk}[A] + \bar{E}/v : \lambda \rangle$. For instance, Example 2.3 (in Section 2) represents the set $\{p(v) \wedge v \dot{=} u, \neg p(u)\}$ by the difference slice $DS[\{p/v, \dot{=} / vu\} \angle p/u]$. We will give some intuition for using 0-ary slices in 3.2.

We now examine the semantics of our concepts and related ideas.

A *model* \mathfrak{M} has as its universe a set $M \neq \emptyset$ and realizes each n -ary predicate symbol $p \in \text{Pr}_n$ as an n -ary relation $p^{\mathfrak{M}} \subseteq M^n$ (with $\dot{=}^{\mathfrak{M}} := \{ \langle a, a \rangle \in M^2 : a \in M \}$). An M -assignment for set $N \subseteq Nm$ of names is a function $g : N \rightarrow M$. A formula ϕ with set v of n names *defines* the n -ary relation $\phi^{\mathfrak{M}} \subseteq M^n$ consisting

³We often employ full, dashed, dotted and wavy lines, respectively, for the 1st, 2nd, 3rd and 4th arguments of expressions.

of the values of its ordered names for the assignments satisfying φ : $\varphi^{\mathfrak{M}} := \{\bar{v}^h \in M^n : \mathfrak{M} \models \varphi[\bar{h}]\}$. For instance, for 2-ary predicate symbol r , $r(u_1, u_2)^{\mathfrak{M}} = r^{\mathfrak{M}}$, $r(u_2, u_1)^{\mathfrak{M}} = \{\langle b, a \rangle \in M^2 : \langle a, b \rangle \in r^{\mathfrak{M}}\}$ and $r(u_1, u_1)^{\mathfrak{M}} = \{\langle a \rangle \in M^1 : \langle a, a \rangle \in r^{\mathfrak{M}}\}$. Also, $\perp^{\mathfrak{M}} := \emptyset$.

We now introduce the meanings of the concepts, again by mutual recursion.

- (E) We define the *relation* of an expression as follows. For a predicate symbol p we have its relation: $[p]_{\mathfrak{M}} := p^{\mathfrak{M}}$; for formula φ we have its defined relation: $[\varphi]_{\mathfrak{M}} := \varphi^{\mathfrak{M}}$; for a slice S or graph G , we use the extensions: $[S]_{\mathfrak{M}} := \llbracket S \rrbracket_{\mathfrak{M}}$ and $[G]_{\mathfrak{M}} := \llbracket G \rrbracket_{\mathfrak{M}}$ (see below); for \bar{E} , where E is an n -ary expression, we use the complement: $[\bar{E}]_{\mathfrak{M}} := M^n \setminus [E]_{\mathfrak{M}}$.
- (a) An M -assignment $g : N \rightarrow M$ satisfies an m -ary arc E/v over N in \mathfrak{M} (noted $g \Vdash_{\mathfrak{M}} E/v$) iff $v \in N^m$ and $v^g \in [E]_{\mathfrak{M}}$. For instance, $g \Vdash_{\mathfrak{M}} \dot{=} u/v$ iff $u^g = v^g$ and $g \Vdash_{\mathfrak{M}} p/w$ iff $w^g \in [p]_{\mathfrak{M}} = p^{\mathfrak{M}}$.
- (Σ) An assignment g satisfies a sketch $\Sigma = \langle N, A \rangle$ in \mathfrak{M} (noted $g : \Sigma \rightarrow \mathfrak{M}$) iff g satisfies every arc $a \in A$.
- (S) The *extension* of a slice is the relation consisting of values of its distinguished list for the assignments satisfying its underlying draft; for an n -ary slice $S = \langle \underline{S} : \hat{s} \rangle$, $\llbracket S \rrbracket_{\mathfrak{M}} := \{s^g \in M^n : g : \underline{S} \rightarrow \mathfrak{M}\}$.
- (G) The *extension* of a graph is the union of those of its slices: $\llbracket G \rrbracket_{\mathfrak{M}} := \bigcup_{S \in G} \llbracket S \rrbracket_{\mathfrak{M}}$.

Clearly, $g \Vdash_{\mathfrak{M}} \bar{E}/v$ iff $g \not\Vdash_{\mathfrak{M}} E/v$. Also, the arcless m -ary slice \top_m has extension $\llbracket \top_m \rrbracket_{\mathfrak{M}} = M^m$.

An expression E is *null* iff $[E]_{\mathfrak{M}} = \emptyset$ in every model \mathfrak{M} . For instance, the empty graph $\{ \}$ is null.

Given a sketch $\Sigma = \langle N, A \rangle$ and an arc $a = E/v$, we say that a is a *consequence* of Σ (noted $\Sigma \models a$) iff, for every model \mathfrak{M} and M -assignment $g : N \cup \underline{v} \rightarrow M$, g satisfies a whenever g satisfies Σ . Call expressions E and F *equivalent* (noted $E \equiv F$) iff, for every model \mathfrak{M} , $[E]_{\mathfrak{M}} = [F]_{\mathfrak{M}}$. A slice S and the singleton graph $\{S\}$ are equivalent (so they may be identified).

We can reduce consequence to the difference slice: an arc a is a consequence of a draft D iff the difference slice $DS[A \angle a]$ is null. So, we can also reduce logical consequence to a difference slice.⁴

Proposition 3.1. *Given a finite set Ψ of formulas and a formula θ : $\Psi \models \theta$ iff the difference slice $DS[\{a[\psi] : \psi \in \Psi\} \angle a[\theta]]$ is null.*

Proof. By the preceding remark, since $g \Vdash_{\mathfrak{M}} a[\varphi]$ iff $\mathfrak{M} \models \varphi[\bar{g}]$. □

Section 4 will present a calculus for establishing that an expression is null.

3.2 Concepts and constructions

We now examine some concepts and constructions.

We first introduce morphisms for comparing sketches.

Consider sketches $\Sigma' = \langle N', A' \rangle$ and $\Sigma'' = \langle N'', A'' \rangle$. A function $\eta : N'' \rightarrow N'$ is a *morphism* from Σ'' to Σ' (noted $\eta : \Sigma'' \dashrightarrow \Sigma'$) iff it preserves arcs: for every arc $E/v \in A''$, we have $E/v^{\eta} \in A'$. We use $\text{Mor}[\Sigma'', \Sigma']$ for the *set of morphisms* from Σ'' to Σ' .

Example 3.1. *Given $p \in \text{Pr}_1$ and $q, r, s, t, a, b \in \text{Pr}_2$, consider the drafts $D' = \langle N', A' \rangle$ and $D'' = \langle N'', A'' \rangle$, with sets of nodes $N' = \{u, v, v', w, w'\}$ and $N'' = \{u_1, u_2, u_3, v, v_1, v_2, w, w_1, w_2, w'\}$, and sets of arcs*

$$A' = \{q/vw, p/w', r/vw', s/vu, t/uw, a/uv', b/v'w\} \text{ and}$$

$$A'' = \{q/v_1w_1, q/v_2w_2, p/w', r/vw', r/v_1w', r/v_2w', s/v_2u_3, t/u_2w_1, a/u_1v', a/u_3v', b/v'w, b/v'w_1, b/v'w_2\}.$$

⁴Recall that $\Psi \models \theta$ iff, for every model \mathfrak{M} and assignment h , h satisfies θ whenever h satisfies every $\psi \in \Psi$.

These drafts D' and D'' can be represented as in Figure 1. The mapping $v' \mapsto v'$; $w' \mapsto w'$; $v, v_1, v_2 \mapsto v$; $w, w_1, w_2 \mapsto w$ and $u_1, u_2, u_3 \mapsto u$ preserves arcs.⁵ So, we have a morphism $\eta : D'' \dashrightarrow D'$. We also have formulas $\delta(D')$ and $\delta(D'')$ such that $g : D' \rightarrow \mathfrak{M}$ iff $\mathfrak{M} \models \delta(D') \llbracket g \rrbracket$ and $g : D'' \rightarrow \mathfrak{M}$ iff $\mathfrak{M} \models \delta(D'') \llbracket g \rrbracket$.⁶

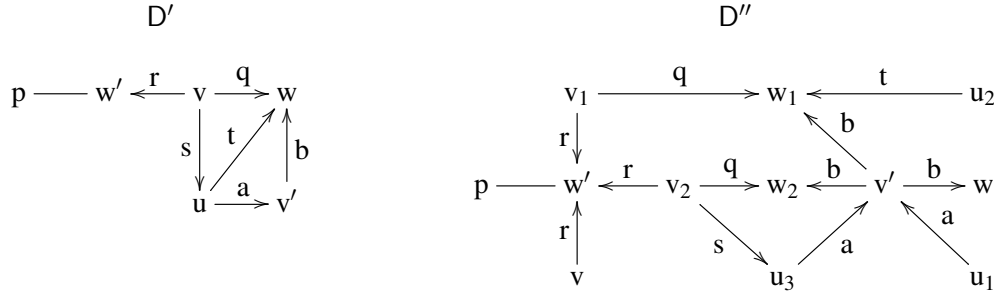


Figure 1: Drafts D' and D'' (Example 3.1)

A morphism transfers satisfying assignments by composition.

Lemma 3.1. *Given a morphism $\eta : \Sigma'' \dashrightarrow \Sigma'$, for every assignment $g : \mathbb{N}_{\Sigma'} \rightarrow M$ satisfying Σ' , the composite $g \cdot \eta : \mathbb{N}_{\Sigma''} \rightarrow M$ is an assignment satisfying Σ'' .*

Proof. For every arc $E/v \in A_{\Sigma''}$, we have $E/v^\eta \in A_{\Sigma'}$, thus $v^{\mathfrak{g} \cdot \eta} \in [E]_{\mathfrak{M}}$, whence $g \cdot \eta \models E/v$. \square

We now use morphisms to introduce zero sketches, slices and graphs.

A sketch $\Sigma = \langle N, A \rangle$ is *zero* iff there exist a slice $T = \langle \underline{T} : \hat{t} \rangle$ and a morphism $\eta : \underline{T} \dashrightarrow \Sigma$ such that \bar{T}/\hat{t}^η is an arc in A . A slice S is *zero* iff its underlying draft \underline{S} is a zero sketch. A graph is *zero* iff all its slices are zero slices. The sets of zero drafts, zero slices and zero graphs are all decidable, since, for drafts D' and D'' , the set $\text{Mor}[D'', D']$ is finite.

Example 3.2. *Consider the following draft D and 2-ary slice T :*

$$D = \langle \{u', v', w'\}, \{r/u'v', \bar{T}/u'w', s/v'w'\} \rangle \quad T = \langle \{u, v, w\}, \{r/uv, s/vw\} : uw \rangle$$

$$\begin{array}{ccc} u' & \xrightarrow{r} & v' \\ & \searrow & \downarrow s \\ & \bar{T} & w' \end{array} \quad \begin{array}{ccc} u^1 & \xrightarrow{r} & v \\ & & \downarrow s \\ & & w^2 \end{array}$$

The mapping $u \mapsto u'$, $v \mapsto v'$, $w \mapsto w'$ gives a morphism $\eta : \underline{T} \dashrightarrow D$, with $\hat{t}^\eta = \langle u^\eta, w^\eta \rangle = \langle u', w' \rangle$. Thus, draft D is zero. So, slices $\langle D : \lambda \rangle$, $\langle D : u' \rangle$, $\langle D : v'w' \rangle$ and $\langle D : u' \rangle$, $\langle D : u'v'w' \rangle$ are zero slices.⁷

Lemma 3.2. *No assignment can satisfy a zero sketch.*

Proof. By Lemma 3.1, $g : \Sigma \rightarrow \mathfrak{M}$ yields $g \cdot \eta : \underline{T} \rightarrow \mathfrak{M}$, thus $g \models_{\mathfrak{M}} \bar{T}/\hat{t}^\eta$ whence $g \not\models_{\mathfrak{M}} \bar{T}/\hat{t}^\eta$. \square

Corollary 3.1. *Zero slices and zero graphs are null.*

Proof. By Lemma 3.2: if $\llbracket S \rrbracket_{\mathfrak{M}} \neq \emptyset$, then some assignment satisfies \underline{S} . \square

⁵For instance, for arc p/w of D'' , we have arc p/w' of D' ; for arcs $q/v_1 w_1$ and $q/v_2 w_2$ of D'' , we have arc $q/v w$ of D' .

⁶Take $\delta(D')$ as $q(v, w) \wedge p(w') \wedge r(v, w') \wedge s(v, u) \wedge t(u, w) \wedge a(u, v') \wedge b(v', w)$ and $\delta(D'')$ as the conjunction of $q(v_1, w_1)$, $q(v_2, w_2)$, $p(w')$, $r(v, w')$, $r(v_1, w')$, $r(v_2, w')$, $s(v_2, u_3)$, $t(u_2, w_1)$, $a(u_1, v')$, $a(u_3, v')$, $b(v', w)$, $b(v', w_1)$ and $b(v', w_2)$.

⁷The extension of slice T can be described by the formula $\exists y (r(u, y) \wedge s(y, w))$.

We can now clarify the intuition behind using 0-ary difference slices (cf. 3.1). We know that a formula is satisfiable iff its existential closure is so. The latter will convert to a 0-ary (basic) graph, by Proposition 4.1 (in 4.1). Now, whether a slice is zero does not hinge on its distinguished list.

We now examine some categorical constructions: co-limits and pushouts [8].

The category of sketches and morphisms has co-limits. Given a diagram of sketches $\Sigma_i = \langle N_i, A_i \rangle$, its *co-limit* can be obtained as expected: obtain the co-limit N of the sets of names N_i and then transfer arcs, by the functions $v_i : N_i \rightarrow N$, i. e. $A := \bigcup_{i \in I} A_i^{v_i}$. In particular, the pushout of drafts gives a draft.

We wish to glue a slice T onto a draft or a slice via a designated list of names. This involves adding the arcs of T with its distinguished list identified to the designated list of names.

Gluing can be introduced as an amalgamated sum (of drafts). Consider an m -ary slice $T = \langle \underline{T} : \hat{t} \rangle$. Given a draft $D = \langle N, A \rangle$ and a list $w \in N^m$ of m names, the *glued draft* $D^w T$ is the pushout of the drafts $D + \underline{w} := \langle N \cup \underline{w}, A \rangle$ and \underline{T} over the m -ary arcless draft $\underline{T}_m = \langle \underline{u}^m, \emptyset \rangle$ and the natural morphisms μ' and μ'' ($\mu' : u_i \mapsto w_i$ and $\mu'' : u_i \mapsto \hat{t}_i$), as shown in Figure 2. Note that $v'(w) = v''(\hat{t})$.

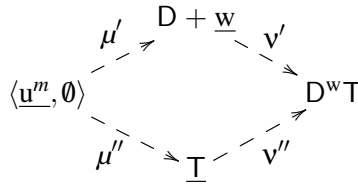
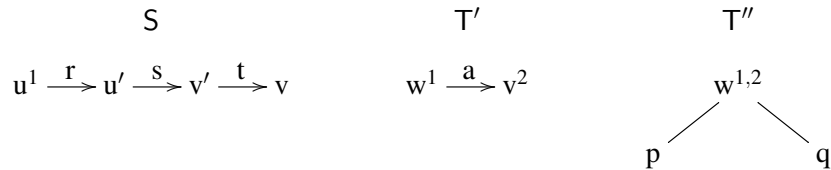


Figure 2: Pushout of drafts

Given an n -ary slice $S = \langle \underline{S} : \hat{s} \rangle$, we obtain the *glued slice* $S^w T$ by transferring the distinguished list of S to the glued draft $\underline{S}^w T : S^w T := \langle \underline{S}^w T : v'(\hat{s}) \rangle$.⁸ We glue a graph by gluing its slices, i. e. $S^w H$ is the graph $\{S^w T : T \in H\}$. We glue onto a graph by gluing onto its slices, i. e. $G^w H := \bigcup_{S \in G} S^w H$.

Example 3.3. Consider the three slices: 1-ary $S = \langle \{u, u', v', v\}, \{r/uu', s/u'v', t/v'v\} : u \rangle$ as well as 2-ary $T' = \langle \{v, w\}, \{a/wv\} : wv \rangle$ and $T'' = \langle \{w\}, \{p/w, q/w\} : ww \rangle$.⁹ They are represented as follows:



We obtain 1-ary glued slices as follows:

$$S^{(u',v')} T' = \langle \{u, u', v', v\}, \left\{ \begin{array}{l} r/uu', s/u'v', t/v'v, \\ a/u'v' \end{array} \right\} : u \rangle \quad S^{(u',v')} T'' = \left\langle \left\{ \begin{array}{l} u, v, \\ w \end{array} \right\}, \left\{ \begin{array}{l} r/uw, s/ww, t/wv, \\ p/w, q/w \end{array} \right\} : u \right\rangle$$



⁸A glued draft and slice are unique up to isomorphism. They can be made unique by a suitable choice of names. As isomorphic objects have the same behavior, we often consider a sketch or a slice up to isomorphism.

⁹The extension of slice T'' can be described by the formula $p(w) \wedge q(w) \wedge w \doteq w'$.

Addition of a slice-arc is equivalent to gluing the slice. For instance, with the slices of Example 3.3: $S + T'/u'v' \equiv S^{(u',v')}T'$ and $S + T''/u'v' \equiv S^{(u',v')}T''$.

Proposition 3.2. *Given a slice S and an arc T/w : $S + T/w \equiv S^wT$.*

Proof. By Lemma 3.1 and the pushout property . □

It is not difficult to translate our graph language to the underlying first-order predicate language. It suffices to express the semantics of the graph language (in 3.1) by formulas.

4 Graph Calculus

We now introduce our graph calculus, with conversion and expansion rules. We employ R^* for the *reflexive-transitive closure* of a binary relation R on a set, as usual.

Our conversion and expansion rules will transform an expression to an equivalent one. Thus, one can apply such a rule in any context. For instance, we will have a rule converting \perp to the empty graph $\{\}$; so, we can apply it to convert $\overline{\perp}$ to $\{\}$ and $S + \perp/\lambda$ to $S + \{\}/\lambda$, for any slice S . Also, we can identify a singleton graph with its slice (cf. 3.1): if $S \triangleright F$ then $\{S\} \triangleright F$ and if $E \triangleright T$ then $E \triangleright \{T\}$.

4.1 Conversion

We now introduce the basic objects and the conversion rules.

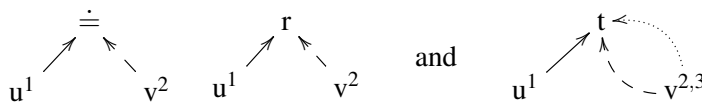
The *basic* objects are defined (by mutual recursion) as follows. The basic expressions are the predicate symbols, other than \doteq , and \overline{T} , where T is a basic slice (see below). An arc E/v is basic iff E is a basic expression. A sketch is basic iff all its arcs are basic. A slice is basic iff its underlying draft is a basic sketch. A graph is basic iff its slices are all basic. For instance, the drafts D' and D'' , of Example 3.1, and D , of Example 3.2, (in 3.2) are basic, whereas those in Examples 2.1, 2.2 and 2.3 are not basic.

The conversion rules will transform an expression to an equivalent basic graph.

The formula rules will come from some equivalences between formulas and expressions. We now illustrate some of these equivalences. For a 1-ary predicate p , formula $p(v)$ is equivalent to the 1-ary slice $\langle \{v\}, \{p/v\} : v \rangle$, thus $\neg p(v)$ is equivalent to the 1-ary expression $\overline{p(v)}$. Now, consider formulas $r(u, v)$ and $s(v, w)$. For the conjunction $r(u, v) \wedge s(v, w)$, we have a 3-ary slice S equivalent to it, namely the slice $S = \langle N, A : u v w \rangle$, with sets $N = \{u, v, w\}$ and $A = \{r(u, v)/u v, s(v, w)/v w\}$. For the disjunction $r(u, v) \vee s(v, w)$ we have a 3-ary graph G such that $r(u, v) \vee s(v, w) \equiv G$, namely the graph G with 2 slices: $\langle \{u, v, w\}, \{r(u, v)/u v\} : u v w \rangle$ and $\langle \{u, v, w\}, \{s(v, w)/v w\} : u v w \rangle$. Also, as the conditional formula $r(u, v) \rightarrow s(v, w)$ is logically equivalent to $\neg r(u, v) \vee s(v, w)$, it is equivalent to the 3-ary graph $\{\langle \{u, v, w\}, \{r(u, v)/u v\} : u v w \rangle, \langle \{u, v, w\}, \{s(v, w)/v w\} : u v w \rangle\}$. The existential formula $\exists y t(u, y, w)$ is equivalent to the 2-ary slice $\langle \{u, v, w\}, \{t(u, v, w)/u v w\} : u w \rangle$. Also, as the universal formula $\forall y t(u, y, w)$ is logically equivalent to $\neg \exists y \neg t(u, y, w)$, it is equivalent to the 2-ary expression $\overline{\langle \{u, v, w\}, \{t(u, v, w)/u v w\} : u w \rangle}$.

The *formula rules* are the following 8 conversion rules eliminating formulas.

(α) For an atomic formula $p(w)$: $p(w) \triangleright \langle \underline{w}, \{p/w\} : w \rangle$. So, we replace $u \doteq v$, $r(u, v)$ and $t(u, v, v)$ by

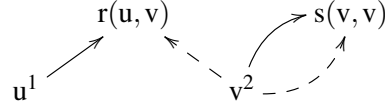


(\perp) $\perp \triangleright \{ \}$, i. e. we replace 0-ary formula \perp by the empty graph.

(\neg) $\neg\phi \triangleright \overline{\phi}$. So, we replace $\neg(r(u, v) \rightarrow s(v, w))$ by the 3-ary expression $\overline{r(u, v) \rightarrow s(v, w)}$.

• Given formulas ψ and θ , with $u := NF[\psi]$ and $v := NF[\theta]$, set $w := u \cup v$.

(\wedge) $\psi \wedge \theta \triangleright \langle w, \{\psi/\vec{u}, \theta/\vec{v}\} : \vec{w} \rangle$. Thus, we can replace formula $r(u, v) \wedge s(v, v)$ by the 2-ary slice $\langle \{u, v\}, \{r(u, v)/u v, s(v, v)/v\} : u v \rangle$, which we can represent as:

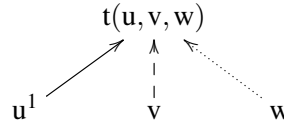


(\vee) $\psi \vee \theta \triangleright \{ \langle w, \{\psi/\vec{u}\} : \vec{w} \rangle, \langle w, \{\theta/\vec{v}\} : \vec{w} \rangle \}$. So, we can replace formula $r(u, v) \vee s(v, v)$ by the 2-ary graph $\left\{ \begin{array}{l} \langle \{u, v\}, \{r(u, v)/u v\} : u v \rangle, \\ \langle \{u, v\}, s(v, v)/v\} : u v \rangle \end{array} \right\}$.¹⁰

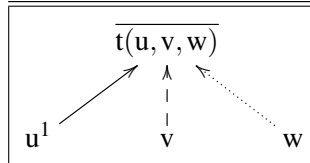
(\rightarrow) $\psi \rightarrow \theta \triangleright \langle w, \{\overline{\psi}/\vec{u}, \theta/\vec{v}\} : \vec{w} \rangle$. So, we can replace formula $p(u) \rightarrow r(v, w)$ by the 3-ary graph $\{ \langle \{u, v, w\}, \{p(u)/u\} : u v w \rangle, \langle \{u, v, w\}, r(v, w)/v w \rangle : u v w \}$.

• Given a formula ϕ and a set v of names, set $u := w \setminus v$, where $w := NF[\phi]$.

(\exists^*) For formula $\exists^* x \phi[v/x], \exists^* x \phi[v/x] \triangleright \langle w, \{\phi/\vec{w}\} : \vec{u} \rangle$. Thus, we can replace $\exists y \exists z t(u, y, z)$ by the single-arc 1-ary slice $\langle \{u, v, w\}, \{t(u, v, w)/u v w\} : u \rangle$, which we can represent as:



(\forall^*) For formula $\forall^* x \phi[v/x], \forall^* x \phi[v/x] \triangleright \overline{\langle w, \{\phi/\vec{w}\} : \vec{u} \rangle}$. So, can we replace $\forall y \forall z t(u, y, z)$ by the 1-ary expression $\langle \{u, v, w\}, \overline{\{t(u, v, w)/u v w\}} : u \rangle$, which we can represent as:



Example 4.1. Consider a formula ϕ with list of names $\langle u, v, w \rangle$, noted $\phi(u, v, w)$.

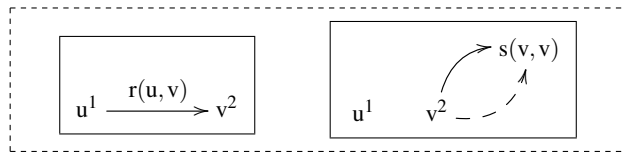
For the formula $\exists y \forall z \phi(u, y, z)$, we have the conversions:

$$\exists y \forall z \phi(u, y, z) \stackrel{(\exists^*)}{\triangleright} \langle \{u, v\}, \{ \frac{\forall z \phi(u, v, z)}{u v} \} : u \rangle \stackrel{(\forall^*)}{\triangleright} \langle \{u, v\}, \{ \overline{\langle \{u, v, w\}, \{\phi/u v w\} : u v \rangle} \} : u \rangle$$

For the formula $\forall y \exists z \phi(u, y, z)$, we have the conversions:

$$\forall y \exists z \phi(u, y, z) \stackrel{(\forall^*)}{\triangleright} \langle \{u, v\}, \{ \overline{\langle \exists z \phi(u, y, z) \rangle} \} : u \rangle \stackrel{(\exists^*)}{\triangleright} \langle \{u, v\}, \{ \overline{\langle \{u, v, w\}, \{\phi/u v w\} : u v \rangle} \} : u \rangle$$

¹⁰This graph can be represented as follows:



By applying the 8 formula rules in any context, one can transform an expression to an equivalent expression without connectives or quantifiers.

The *equality rule* is the following conversion rule, eliminating expression \doteq .

(\doteq) $\doteq \triangleright \langle \{u\}, \emptyset : \langle u, u \rangle \rangle$, where $u \in \text{Nm}$. So, we can replace slice $\langle \{u, v, w\}, \{r/uv, \doteq/vw, s/uw\} : vw \rangle$ by the slice $\langle \{v, w\}, \{r/uv, \frac{\langle \{u\}, \emptyset : \langle u, u \rangle \rangle}{vw}, s/uw\} : vw \rangle$.

By using these 9 rules, one can eliminate logical symbols and predicates, but arcs whose expressions are slices or graphs, perhaps complemented, may appear. For instance, this happens with $v \doteq w$ and $\exists y(r(u, y) \wedge s(y, w))$. The following rules will address these cases.

The *complementation rules* are the following 2 conversion rules, moving $\bar{}$ inside.

($\bar{}$) For an n -ary graph H : $\bar{H} \triangleright \langle \underline{w}, \{\bar{T}/w : T \in H\} : w \rangle$, where w is a list of n distinct names. So, we can replace the complemented 1-ary graph $\{\bar{S}, \bar{T}\}$ by the slice $\langle \{v\}, \{\bar{S}/v, \bar{T}/v\} : v \rangle$.

($\bar{\bar{}}$) $\bar{\bar{E}} \triangleright E$, i. e. eliminate double complementation.

By applying these 2 complementation rules in any context, one can eliminate arcs whose expressions are complemented graphs.

The *structural rules* are the following 3 conversion rules.

($\overset{U}{\rightarrow}$) $S + H/v \triangleright \{S + T/v : T \in H\}$, i. e. replace addition of graph arc by alternative addition of its slice arcs. So, we replace slice $S + \{T', T''\}/u$ by the graph $\{S + T'/u, S + T''/u\}$.

($\overset{T}{\rightarrow}$) $S + T/v \triangleright S^v T$, i. e. replace addition of slice arc by glued slice.

(\uparrow) For an n -ary expression E : $E \triangleright \langle \underline{w}, \{E/w\} : w \rangle$, where w is a list of n distinct names. So, for $r \in \text{Pr}_2$, we can replace 2-ary expression r by the 2-ary slice $\langle \{u_1, u_2\}, \{r/u_1 u_2\} : u_1 u_2 \rangle$.

By means of rules ($\overset{U}{\rightarrow}$) and ($\overset{T}{\rightarrow}$), one can eliminate arcs whose expressions are graphs or slices.

Rule (\uparrow) converts expressions to slices and serves to eliminate \bar{p} : $\bar{p} \triangleright \langle \underline{u^n}, \{\bar{p}/u^n\} : u^n \rangle$, for $p \in \text{Pr}_n$.

Example 4.2. Consider the formula $r(v, w)$. We proceed much as in Example 4.1.

Formula $\exists y \forall z r(y, z)$ converts to the 0-ary slice $S = \langle \{v\}, \{ \frac{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{r(v, w)/vw\} : v \rangle}}{vw} \rangle}{v} \} : \lambda \rangle$.

This slice S is not basic, but it can be converted to a basic slice by (α) as follows:

$$\langle \{v\}, \{ \frac{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{r(v, w)/vw\} : v \rangle}}{vw} \rangle}{v} \} : \lambda \rangle \stackrel{(\alpha)}{\triangleright} \langle \{v\}, \{ \frac{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{r(v, w)/vw\} : v \rangle}}{vw} \rangle}}{vw} \rangle}}{vw} \rangle}{v} \} : \lambda \rangle$$

Formula $\forall y \exists z r(y, z)$ converts to the 0-ary expression $E = \langle \{v\}, \{ \frac{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{r(v, w)/vw\} : v \rangle}}{vw} \rangle}{v} \} : \lambda \rangle$.

This expression E is not basic, but it can be converted to a basic expression F by (α) as follows:

$$\langle \{v\}, \{ \frac{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{r(v, w)/vw\} : v \rangle}}{vw} \rangle}{v} \} : \lambda \rangle \stackrel{(\alpha)}{\triangleright} \langle \{v\}, \{ \frac{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{ \frac{\overline{\langle \{v, w\}, \{r(v, w)/vw\} : v \rangle}}{vw} \rangle}}{vw} \rangle}}{vw} \rangle}}{v} \} : \lambda \rangle$$

Expression F can be converted to a basic 0-ary slice by (\uparrow).

Rule $(\overline{\top})$ gives some useful derived rules about arc addition, which we can use to shorten conversions (such shortenings were used in the examples of Section 2). We can replace addition of: a graph arc by gluing the graph $((\overline{H}) : S + H/v \triangleright^* S^v H)$, a complemented-graph arc by addition of parallel complemented-slice arcs $(S + \overline{H}/v \triangleright^* S + \{\overline{T}/v : T \in H\})$ and an equality arc by node renaming $(S + \dot{=} / u v \triangleright^* S[u/v], S + \dot{=} / u v \triangleright^* S[v/u])$.

We can also replace n conjunctions and disjunctions by slices and graphs, respectively.¹¹

Example 4.3. Consider the formula $s(v', w') \wedge \exists x[r(v', x) \wedge \neg \exists y(r(x, y) \wedge s(y, w'))]$. This expression E can be converted to the 2-ary slice $\langle D : v' w' \rangle$, where D is the draft of Example 3.2 (in 3.2).

We can convert expressions in a modular way.

Lemma 4.1. If $S \triangleright^* G$ and $E \triangleright^* H$, then $S + E/v \triangleright^* G^v H$.

Proof. By (\overline{H}) rule: $S + E/v \triangleright^* G + H/v = \{P + H/v : P \in G\} \xrightarrow{(\overline{H})} \{P^v H : P \in G\} = G^v H$. \square

Thus, one can obtain a basic form for $S + E/v$ from basic forms S^b and E^b , for S and E .

Proposition 4.1. Every n -ary expression E can be effectively converted to a basic n -ary graph E^b .

Proof. By induction on the structure of expressions. \square

Example 4.4. Given the predicate symbols of Example 3.1 (in 3.2), consider the formula ψ :

$$q(v, w) \wedge \exists z[p(z) \wedge r(v, z) \wedge \exists x \exists y \exists y'(s(v, x) \wedge t(x, w) \wedge a(x, y) \wedge b(y, w))].$$

Consider also the formula $\theta := \exists^3 x_1 x_2 x_3 \exists y' \exists^2 y_1 y_2 \exists z' \exists^2 z_1, z_2 \chi$, where χ is as follows:

$$p(z') \wedge s(v_2, x_3) \wedge t(x_2, z_1) \wedge \left(\begin{array}{c} q(y_1, z_1) \\ \wedge \\ q(y_2, z_2) \end{array} \right) \wedge \left(\begin{array}{c} a(x_1, v') \\ \wedge \\ a(x_2, v') \end{array} \right) \wedge \left(\begin{array}{c} r(v, z') \\ \wedge \\ r(y_1, z') \\ \wedge \\ r(y_2, z') \end{array} \right) \wedge \left(\begin{array}{c} b(y', w) \\ \wedge \\ b(y', z_1) \\ \wedge \\ b(y', z_2) \end{array} \right).$$

Now, form the difference slice $DS[\{a[\psi]\} \angle a[\theta]] = \langle \{v, w\}, \{\psi/v w, \overline{\theta}/v w : \lambda\} \rangle$. Expressions ψ and θ can be respectively converted to the 2-ary slices $\underline{S} = \langle D' : \langle v, w \rangle \rangle$ and $\overline{T} = \langle D'' : \langle v, w \rangle \rangle$, where D' and D'' are the drafts of Example 3.1. Thus, we have $DS[\{a[\psi]\} \angle a[\theta]] \triangleright^* \langle \{v, w\}, \{S/v w, \overline{T}/v w : \lambda\} \rangle$. Now, we can see that $\langle \{v, w\}, \{S/v w, \overline{T}/v w : \lambda\} \rangle \xrightarrow{(\overline{\top})} \langle \underline{S} + \overline{T}/v w : \lambda \rangle$.¹² Hence $DS[\{a[\psi]\} \angle a[\theta]] \triangleright^* \langle \underline{S} + \overline{T}/v w : \lambda \rangle$.

4.2 Derivations

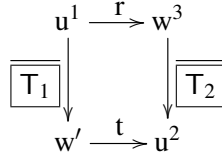
We now introduce the remaining rule and finish the presentation of our calculus.

First, let us review Examples 4.3 and 4.4 (in 4.1). Formula E of Example 4.3 converts to the 2-ary slice $\langle D : v' w' \rangle$, which was seen to be zero in Example 3.2 (in 3.2). Thus, formula E is unsatisfiable. Now, consider formulas ψ and θ of Example 4.4, where we have seen that $DS[\{a[\psi]\} \angle a[\theta]] \triangleright^* \langle \underline{S} + \overline{T}/v w : \lambda \rangle$. Now, Example 3.1 (in 3.2) shows a morphism $\eta : \underline{\mathbb{T}} \dashrightarrow \underline{\mathbb{S}}$, with $\hat{t}^\eta = \langle v^\eta, w^\eta \rangle = \langle v, w \rangle = \hat{s}$. Thus, draft $\underline{S} + \overline{T}/v w$ is zero, whence, slice $DS[\{a[\psi]\} \angle a[\theta]]$ is null. Therefore, we can conclude that $\psi \models \theta$.

¹¹For instance, with 3 formulas, we have $r(u, v) \wedge s(v, w) \wedge p(w) \triangleright^* \langle \{u, v, w\}, \{r(u, v)/u v, s(v, w)/v w, p(w)/w\} : u v w \rangle$ and $r(u, v) \vee s(v, w) \vee p(w) \triangleright^* \langle \{u, v, w\}, \{r(u, v)/u v\} : u v w \rangle, \langle \{u, v, w\}, s(v, w)/v w \rangle : u v w \rangle, \langle \{u, v, w\}, \{p(w)/w\} : u v w \rangle$.

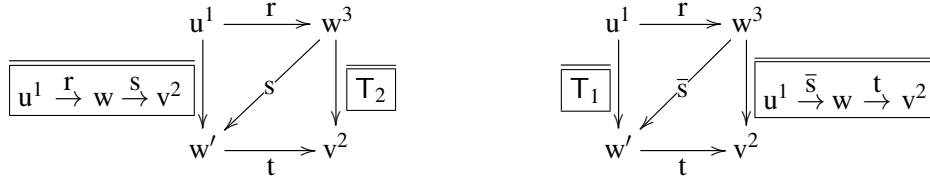
¹²Indeed: $\langle \{v, w\}, \{S/v w, \overline{T}/v w : \lambda\} \rangle = \langle \{v, w\}, \{\overline{T}/v w : \lambda\} \rangle + S/v w \xrightarrow{(\overline{\top})} \langle \{v, w\}, \{\overline{T}/v w : \lambda\} \rangle^{(v, w)} S = \langle \underline{S} + \overline{T}/v w : \lambda \rangle$.

Example 4.5. To introduce expansion and its usefulness, consider the 3-ary slice S :



where $T_1 := u^1 \xrightarrow{r} w \xrightarrow{s} v^2$, $T_2 := u^1 \xrightarrow{\bar{s}} w \xrightarrow{t} v^2$.

Slice S is not zero; but in any model \mathfrak{M} , the pair $(g(w), g(w'))$ is either in $[s]_{\mathfrak{M}}$ or in $[\bar{s}]_{\mathfrak{M}}$. So, S is equivalent to the 2-ary graph $G = \{S_+, S_-\}$, with slices S_+ and S_- , respectively as follows:



Slices S_+ and S_- are both zero, so graph G is zero. Thus, S is a null slice.

The expansion rule will replace a slice by a graph with 2 alternative slices.

(\triangleleft) For an m -ary slice T and $v \in N_S^m$: $S \triangleleft \{S^v T, S + \bar{T}/v\}$.

Note that both $S^v T$ and $S + \bar{T}/v$ are basic whenever S and T are basic.

Lemma 4.2. For a slice S , an m -ary slice T and $v \in Nm^m$: $S \equiv \{S^v T, S + \bar{T}/v\}$.

Proof. By Proposition 3.2 (in 3.2), $S + T/v \equiv S^v T$, and clearly $S \equiv \{S + T/v, S + \bar{T}/v\}$. \square

A *derivation* consists of applications of the conversion rules and the expansion rule: $\vdash := (\triangleright \cup \triangleleft)^*$. A derivation is *normal* iff applications of conversion rules precede applications of the expansion rule: $E \triangleright^* G \triangleleft^* H$. In practice, as we wish to derive a zero graph, we may erase slices already found to be zero.

Let φ be the formula $r(u, w) \wedge t(w', v) \wedge \neg \exists x [r(u, x) \wedge s(x, v)] \wedge \neg \exists y [\bar{s}(u, y) \wedge t(y, v)]$. Expression $E := \exists x \varphi[w'/x]$ converts to the slice S of Example 4.5, where it expands to the graph G . We thus have the normal derivation $E \triangleright^* S \triangleleft G$, with G a zero graph. Hence, formula φ is unsatisfiable.

4.3 Soundness and completeness

We now examine soundness and completeness of our calculus.

Soundness is clear (as $E \equiv F$, whenever $E \vdash F$): if $E \vdash H$ and H is zero, then E is null. We will show that a converse holds for basic graphs (if E^b is null then E^b expands to a zero graph), and we will have completeness of normal derivations: if E is null, then $E \triangleright^* E^b \triangleleft^* H$, for some zero graph H .

Henceforth, all sketches, drafts, slices and graphs will be basic. We define the following families of slices: the family Z_0 of *zero slices* (cf. 3.2); the family Z_* of *expansively zero slices*: the slices S such that, for some graph $G \subseteq Z_0$, $S \triangleleft^* G$; the family Z_∞ of *not expansively zero slices*: the slices outside Z_* .

The following simple properties of these families will be useful.

Lemma 4.3. For every graph G : $G \subseteq Z_*$ iff, for some graph $H \subseteq Z_0$, $G \triangleleft^* H$.

Proof. (\Rightarrow) If, for each $S \in G$, $S \triangleleft^* H_S$ and $H_S \subseteq Z_0$, then, with $H := \bigcup_{S \in G} H_S$, $G \triangleleft^* H$ and $H \subseteq Z_0$.

(\Leftarrow) if $G \triangleleft^* H$, with $H \subseteq Z_0$, then for each $S \in G$, $S \triangleleft^* H_S$, with $H_S \subseteq H \subseteq Z_0$, whence $S \in Z_*$. \square

Lemma 4.4. *For every graph $G: G \subseteq Z_*$ iff, for some graph $H \subseteq Z_*$, $G \triangleleft^* H$.*

Proof. By Lemma 4.3, since $Z_0 \subseteq Z_*$. (\Rightarrow) If $G \subseteq Z_*$, then $G \triangleleft^* H$, with $H \subseteq Z_0 \subseteq Z_*$. (\Leftarrow) If $G \triangleleft^* H$, with $H \subseteq Z_*$, then $H \triangleleft^* H'$, with $H' \subseteq Z_0$, whence $G \triangleleft^* H'$, with $H' \subseteq Z_*$. \square

Corollary 4.1. *For $S \in Z_\infty$, m -slice T and $v \in N_S^m$: one of $S^v T$ and $S + \bar{T}/v$ is not expansively zero.*

Proof. By Lemma 4.4: if $\{S^v T, S + \bar{T}/v\} \subseteq Z_*$, then $S \in Z_*$. \square

We will show that a slice $S \in Z_\infty$ has a model \mathfrak{M} with $[[S]]_{\mathfrak{M}} \neq \emptyset$

Given a slice $S \in Z_\infty$, we can obtain a set of slices $S_n = \langle N_n, A_n : \hat{s}_n \rangle$ with $S_n \in Z_\infty$, for $n \in \mathbb{N}$, whose underlying drafts are connected by morphisms μ_n from \underline{S}_n to \underline{S}_{n+1} , which we extend naturally to morphisms $\mu_j^i : \underline{S}_i \dashrightarrow \underline{S}_j$, for $i \leq j$. Consider the co-limit of this draft diagram: sketch $\Sigma = \langle N, A \rangle$ with morphisms $\nu_n : \underline{S}_n \dashrightarrow \Sigma$ (cf. 3.2). We use this co-limit sketch Σ to define a *natural model* \mathfrak{M} with $M := N$, and $p^{\mathfrak{M}} := \{v \in M^n : p/v \in A\}$, for $p \in \text{Pr}_n$.

By construction, the co-limit sketch Σ is saturated in the following sense: given any m -ary slice $T = \langle \underline{T} : \hat{t} \rangle$ and $w \in N^m$, we have $\text{arc } \bar{T}/w \in A$ or there is a morphism $\eta : \underline{T} \dashrightarrow \Sigma$ with $\hat{t}^\eta = w$.

We can establish that satisfying assignments are morphisms.

Lemma 4.5. *Given a draft D and $g : N_D \rightarrow M$, $g : D \rightarrow \mathfrak{M}$ iff $g : D \dashrightarrow \Sigma$.*

Proof. By structural induction (on the total number of complemented slice arcs occurring in D). \square

Finally, since $\nu_0 : \underline{S} \dashrightarrow \Sigma$, we have $\nu_0(\hat{s}_0) \in [[S]]_{\mathfrak{M}} \neq \emptyset$.

Therefore, if $G \not\subseteq Z_*$, then G is not null.

Theorem 4.1. *Consider an n -ary expression E .*

(\vdash) *If $E \vdash H$ and H is zero, then E is null.*

(\triangleright^* ; \triangleleft^*) *If E is null, then $E \triangleright^* E^b \triangleleft^* H$, for some zero n -ary graph H .*

5 Conclusion

We now present some concluding remarks, including comparison with related works.

We have presented a refutation graph calculus for classical first-order predicate logic. This sound and complete calculus reduces logical consequence to establishing that a constructed graph is null, i. e. has empty extension in every model. Our calculus uses formulas directly and can represent them by arcs.

We have a simple strategy for establishing that a graph G is null: first convert G to basic form, then apply repeatedly the expansion rule, erasing slices found to be zero, which is decidable (cf. 3.2), trying to obtain the empty graph. Conversion to basic form, though tedious, can be automated (cf. 4.1); some ingenuity may be required in selecting which slice of a graph to expand and how to do it (cf. 4.2), but the embedded slices can provide a finer control. In fact, a (human-guided) system may be envisaged.

The idea of using graphical representations for logic appears in several works.

Girard's proof nets have been applied to classical logic [10], where sequent proofs are translated to proof nets. In our case, however, the (macroscopic) structure of normal derivations is rather simple: first conversions, then expansion (cf. 4.2).

Graph rewriting motivates a graphical representation of first-order predicate logic. For the binary fragment (with \doteq), a representation of formulas by graph predicates has been obtained by Rensink [9]: a correspondence between sets of graph predicates with depth up to n and a hierarchy $\exists(-\exists)^n$. There are

close similarities between some concepts (our sketches are his graphs), but his graph predicates involve morphisms (even though they may be reminiscent of our arcs).

Our approach does resemble Peirce's ideas [11] as formulated by Dau [4]. In our representation, we use names only for referring to them in the meta-language: if we erase these names, we obtain a representation quite close to the Peirce's ones (cf. Example 4.2 in 4.1). Besides our refutation approach with normal derivations, there are some differences: we allow formulas directly in the graphs (and need conversion rules), rather than pre-processing diagrams for them; Peirce considers the fragment \neg , \wedge and \exists , whereas we use graphs to cope with \vee , which seems to lead to less cumbersome representations; we handle $\dot{=}$, first as a 2-ary predicate and then as a special one, whereas Peirce represents it directly by identity lines, leading to more compact diagrams. So, there appear to be advantages and disadvantages on both sides.

Some further work on our calculus would be: add function symbols (for this purpose, some ideas used for structured nodes [6] seem promising); provide a detailed comparison between it and [9] (such a comparison between Peirce's and Rensink's approaches is reported difficult [9], p. 333); develop a "middle-ground" between our approach and Dau's [4], with the best features from each one.

References

- [1] T. Barkowsky (2010): *Diagrams in the mind: visual or spatial?*. In A. K. Goel, M. Jamnik & N. H. Narayanan, editors: *LNAI, Series 6170*, p. 1, Springer-Verlag, Berlin, doi:10.1007/978-3-540-92687-0.
- [2] S. Curtis & G. Lowe (1995): *A graphical calculus*. In B. Moller, editor: *Mathematics of Program Construction LNCS Series 947*, Springer-Verlag, Berlin, pp. 214–231, doi:10.1007/3-540-60117-1-12.
- [3] S. Curtis & G. Lowe (1996): *Proofs with graphs*. In R. Backhouse, editor: *Science of Computer Programming*, Elsevier, volume (26), pp. 197–216, doi:10.1016/0167-6423(95)00025-9.
- [4] F. Dau (2006): *Mathematical logic with diagrams, based on the existential graphs of Peirce*, Habil. thesis, TU Dresden, 2006, www.du-dau.net/publications.shtml.
- [5] R. Freitas, P. A. S. Veloso, S. R. M. Veloso & P. Viana (2008): *On a graph calculus for algebras of relations*. In W. Hodges & R. de Queiroz, editors: *LNAI, Series 5110*, Springer-Verlag, Heiderberg, pp. 298–312, doi:10.1007/978-3-540-69937-8.
- [6] R. Freitas, P. A. S. Veloso, S. R. M. Veloso & P. Viana (2009): *Positive fork graph calculus*. In S. Artemov, editor: *LNCS, Series 5407*, Springer-Verlag, New York, pp. 152–163, doi:10.1007/978-3-540-92687-0.
- [7] R. Freitas, P. A. S. Veloso, S. R. M. Veloso & P. Viana (2010): *A calculus for graphs with complement*. In A. K. Goel, M. Jamnik & N. H. Narayanan, editors: *LNAI, Series 6170*, pp. 84–98, Springer-Verlag, Berlin, doi:10.1007/978-3-540-92687-0.
- [8] S. MacLane (1998): *Categories for the Working Mathematician*, second edition, Springer-Verlag, Berlin.
- [9] A. Rensink (2004): *Representing first-order logic using graphs*, In H. Ehrig et al. editors: *LNCS, Series 3256*, pp. 319–335, Springer-Verlag, Heiderberg, doi:10.1007/978-3-540-30203-2.
- [10] E. Robinson (2003): *Proof nets for classical logic*, *J. Logic and Computat.* volume(13) number(5) , pp. 776–797, doi:10.1093/logcom/13.5.777.
- [11] J. F. Sowa (2011): *Existential graphs*, doi:10.1515/semi.2011.060.
- [12] G. Takeuti (1975): *Proof Theory*, North-Holland, Amsterdam.
- [13] P. A. S. Veloso and S. R. M. Veloso (2012): *On Graph refutation for relational inclusions*, In S. R. della Rocca and E. Pimentel, editors: *EPTCS volume (81)*, pp. 47–66. doi:10.4204/EPTCS.81.4.