Proof-Search in Intuitionistic Logic Based on Constraint Satisfaction and Related Complexity Problems

ANDREI VORONKOV, *Computer Science Department, University of Manchester, email voronkov@cs.man.ac.uk*

Abstract

We characterize provability in intuitionistic predicate logic in terms of derivation skeletons and constraints and study the problem of instantiations of a skeleton to valid derivations. We prove that for two different notions of a skeleton the problem is respectively polynomial and NP-complete. As an application of our technique, we demonstrate PSPACE-completeness of the prenex fragment of intuitionistic logic. We outline some applications of the proposed technique in automated reasoning.

Keywords: Intuitionistic logic, automated reasoning, complexity, derivation skeletons

1 Introduction

The characterization of provability for classical logic in terms of matrices was given by Kanger [9, 10] and Prawitz [20, 21] and is in a fact a reformulation of the fundamental Herbrand theorem. It is the basis of many theorem-proving methods, including the tableau method and the connection (or mating) method.

This characterization says that a formula $\exists\bar{x}\varphi(\bar{x})$, where $\varphi(\bar{x})$ is a quantifier-free formula, is provable in classical logic iff there is a *matrix* M of the form $\varphi(\bar{x}_1) \vee \ldots \vee \varphi(\bar{x}_n)$ constructed by this formula and a substitution σ such that every path in the matrix $M\sigma$ is inconsistent, i.e. contains a formula and its negation. For such a matrix M, the information on the number of quantifier duplications (i.e. the number n) is enough to find a proof using a non-deterministic polynomial algorithm (when n is fixed, the problem of the existence of such a substitution σ is NP-complete). There are many ways of constructing such a matrix and checking its paths. The way used in the tableau method can be considered as a construction of the skeleton of a derivation in the sequent calculus, and finding a substitution σ that makes this skeleton a derivation.

The situation with intuitionistic logic is more complicated. Even the propositional fragment of intuitionistic logic is PSPACE-complete due to non-invertible rules. Further, there is no prenex normal form of formulas so that the number of quantifier duplications does not give enough information about the structure of the derivation. In addition, there is no Skolem normal form and one has to deal with essentially universal quantifiers.

Our article has two main aims. The first aim is to develop a characterization of provability in intuitionistic predicate logic. For the connection method, such a characterization was developed by Wallen [31]. Our characterization is based on derivation skeletons and constraint satisfaction. We use constraints to design a machine-oriented version $\mathbf{LJ}_c$ of the intuitionistic sequent calculus. Our characterization of intuitionistic provability is more simple than that of Wallen [31] and can be used for constructing a family of sequent-based proof procedures, for example a tableau-based procedure, an inverse method-based procedure and their modifications on the basis of $\mathbf{LJ}_c$. The second aim of this article is to establish complexity results for the problem of instantiation of a given derivation skeleton to a derivation (the skeleton instantiation problem). We also outline some potential applications of our results and techniques.

This article has the following structure. Section 2 defines the sequent calculus **LJ** and skeletons of derivations in **LJ**. The proof-search in **LJ** can be considered as the search for a derivation skeleton and the instantiation of this skeleton to a valid derivation. In Section 3 we define the constraint calculus $\mathbf{LJ}_c$ in which the skeleton instantiation problem is encoded by constraints and prove a theorem characterizing the relation between derivations in **LJ** and $\mathbf{LJ}_c$. In Section 4 we prove the existence of a polynomial time algorithm for constraint satisfiability. As a consequence, we characterize the complexity of the skeleton instantiation problem. Using the technique of derivation skeletons and constraints developed in previous sections, in Section 5 we prove PSPACE-completeness of the prenex fragment of intuitionistic logic. In Section 6 we briefly consider related work and some further potential applications of our technique.

This article is an extended version on [29].

2 Preliminaries. The calculus LJ

Formulas are defined as usual, using atomic formulas, the constant Λ, the connectives $\wedge, \vee, \supset$ and the quantifiers $\forall, \exists$. Negation $\neg\varphi$ can be introduced as $\varphi \supset \Lambda$. A *sequent* is an expression of the form $\Gamma \to \varphi$ where Γ is a finite sequence of formulas and φ is a formula. A formula is *closed* iff it has no free variables. A sequent $\Gamma \to \varphi$ is *closed* iff all formulas in Γ, φ are closed.

The notation $\rightleftharpoons$ stands for "equal by definition". $s \approx t$ denotes that terms s and t are identical. $\not\approx$ is the negation of $\approx$.

By X we denote a countable infinite set of variables. The set of all terms in the signature Σ with variables in X is denoted by $T_\Sigma(X)$. By T_Σ we denote $T_\Sigma(\emptyset)$, i.e. the set of all ground terms in the signature Σ. *Constants* are regarded as function symbols of arity 0. Variables are denoted by u, v, w, x, y, z, maybe with indices. We assume that all formulas use variables in the set X.

A *substitution* is any mapping θ from X to $T_\Sigma(X)$ such that there is only a finite number of variables $x \in X$ such that $\theta(x) \not\approx x$. The *domain* $dom(\theta)$ of the substitution θ is the set $\{x \mid x\theta \not\approx x\}$. The *range* $ran(\theta)$ of the substitution θ is defined by $ran(\theta) \rightleftharpoons \{x\theta \mid x \in dom(\theta)\}$. Substitutions θ with the domain $x_1, \ldots, x_n$ are denoted by $\{x_1\theta/x_1, \ldots, x_n\theta/x_n\}$. Let E be a term, formula, sequence of formulas or sequent. For a substitution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$, $E\theta$ denotes the result of *applying θ to E*, i.e. renaming of those bound variables in E that coincide with a variable occurring in any t_i and the replacement of all free occurrences of x_i by t_i.

$$\frac{}{\Gamma, A, \Delta \to A}\ (Ax) \qquad \frac{}{\Gamma, \Lambda, \Delta \to \varphi}\ (\Lambda)$$

$$\frac{\Gamma, \varphi, \psi, \Delta \to \chi}{\Gamma, \varphi \wedge \psi, \Delta \to \chi}\ (\wedge \to) \qquad \frac{\Gamma \to \varphi \quad \Gamma \to \psi}{\Gamma \to \varphi \wedge \psi}\ (\to \wedge)$$

$$\frac{\Gamma, \varphi, \Delta \to \chi \quad \Gamma, \psi, \Delta \to \chi}{\Gamma, \varphi \vee \psi, \Delta \to \chi}\ (\vee \to) \qquad \frac{\Gamma \to \varphi}{\Gamma \to \varphi \vee \psi}\ (\to \vee_1) \qquad \frac{\Gamma \to \psi}{\Gamma \to \varphi \vee \psi}\ (\to \vee_2)$$

$$\frac{\Gamma, \psi, \Delta \to \chi \quad \Gamma, \varphi \supset \psi, \Delta \to \varphi}{\Gamma, \varphi \supset \psi, \Delta \to \chi}\ (\supset\to) \qquad \frac{\varphi, \Gamma \to \psi}{\Gamma \to \varphi \supset \psi}\ (\to\supset)$$

$$\frac{\Gamma, \varphi\{t/x\}, \forall x\varphi, \Delta \to \chi}{\Gamma, \forall x\varphi, \Delta \to \chi}\ (\forall \to) \qquad \frac{\Gamma \to \varphi\{y/x\}}{\Gamma \to \forall x\varphi}\ (\to \forall)$$

$$\frac{\Gamma, \varphi\{y/x\}, \Delta \to \chi}{\Gamma, \exists x\varphi, \Delta \to \chi}\ (\exists \to) \qquad \frac{\Gamma \to \varphi\{t/x\}}{\Gamma \to \exists x\varphi}\ (\to \exists)$$

In the rules $(\exists \to)$ and $(\to \forall)$ the variable y has no free occurrences in the conclusions of the rules. The variable y is called the *eigenvariable* of these rules.

FIG. 1. The calculus **LJ**

2.1 The calculus **LJ**

The calculus **LJ** for intuitionistic logic is shown in Figure 1. All rules labelled with $(C \to)$, where C is a connective or a quantifier are called *antecedent rules,* all rules labelled with $(\to C)$ are called *succedent rules.*

Derivations in **LJ** are trees defined as usual. When we say that a formula φ is *provable* in **LJ** or in any other sequent calculus, we mean that the sequent $\to \varphi$ is derivable in this calculus.

2.2 Derivation skeletons

The following definition is one of the main notions studied in this article:

DEFINITION 2.1 (Weak skeleton)
A *weak derivation skeleton* is a tree such that

1. Its nodes are labelled by the names of inference rules in **LJ**;
2. The number of parents of a node labelled by a name of an inference rule is equal to the number of the premises in such a rule in **LJ**.

In weak derivation skeletons, we display labels to the right of the bar.

EXAMPLE 2.2
The following picture displays a weak derivation skeleton:

$$\dfrac{\dfrac{\dfrac{}{}\,(Ax) \qquad \dfrac{\dfrac{\dfrac{}{}\,(Ax) \qquad \dfrac{\dfrac{\dfrac{}{}\,(Ax) \qquad \dfrac{\dfrac{}{}\,(Ax)}{}\,(\rightarrow\supset)}{}\,(\supset\rightarrow)}{}\,(\rightarrow\supset)}{}\,(\supset\rightarrow)}{}\,(\rightarrow\supset)}{}\,(\supset\rightarrow)}{}\,(\rightarrow\supset)$$

Weak derivation skeletons are abstractions of derivations that encode information about the structures of derivations. They abstract from the instantiation of variables in quantifier rules. Before formally defining the connection between derivations and skeletons, we introduce a stronger notion of skeletons:

Definition 2.3 (Skeleton)
A *skeleton* is defined similar to the weak skeleton with the following difference: all nodes labelled by antecedent rules or (Ax) are additionally labelled by a natural number.

The natural number in an antecedent rule uniquely identifies the main formula of the rule application. This number is the index of the main formula in the antecedent of the rule (or, equivalently, the number of formulas in Γ).

Definition 2.4 (Skeleton of a derivation)
Let Π be a derivation in **LJ**. *The weak skeleton* Π' *of* Π and *the skeleton* Π'' *of* Π are defined inductively as follows:

1. If Π has the form $\dfrac{}{\Gamma, A, \Delta \to A}\,(Ax)$ then Π' and Π'' are $\dfrac{}{}\,(Ax)$ and $\dfrac{}{}\,(Ax_n)$ respectively, where n is the number of formulas in Γ.
2. If Π has the form $\dfrac{}{\Gamma, \Lambda, \Delta \to \varphi}\,(\Lambda)$ then both Π' and Π'' are $\dfrac{}{}\,(\Lambda)$
3. If Π has the form $\dfrac{\begin{matrix}\Pi_1\\ \varphi, \Gamma \to \psi\end{matrix}}{\Gamma \to \varphi \supset \psi}\,(\rightarrow\supset)$ and Π_1' and Π_1'' are, respectively, the weak skeleton and the skeleton of Π_1, then Π' and Π'' are $\dfrac{\Pi_1'}{}\,(\rightarrow\supset)$ and $\dfrac{\Pi_1''}{}\,(\rightarrow\supset)$ respectively, and similar for all other succedent rules.
4. If Π has the form $\dfrac{\begin{matrix}\Pi_1\\ \Gamma, \psi, \Delta \to \chi\end{matrix} \quad \begin{matrix}\Pi_2\\ \Gamma, \varphi \supset \psi, \Delta \to \varphi\end{matrix}}{\Gamma, \varphi \supset \psi, \Delta \to \chi}\,(\supset\rightarrow)$ and Π_1', Π_2' and Π_1'', Π_2'' are, respectively, the weak skeletons and the skeletons of Π_1, Π_2, then Π' and Π'' are $\dfrac{\Pi_1' \quad \Pi_2'}{}\,(\supset\rightarrow)$ and $\dfrac{\Pi_1'' \quad \Pi_2''}{}\,(\supset\rightarrow_n)$ respectively, where n is the number of formulas in Γ, and similar for all other antecedent rules.

For example, the weak derivation skeleton of Example 2.2 is the weak skeleton of the following derivation:

$$
\dfrac{\dfrac{}{R \to R}\,(Ax) \qquad \dfrac{\dfrac{\dfrac{}{P, (((P \supset R) \supset P) \supset P) \supset R \to P}\,(Ax) \qquad \dfrac{\dfrac{\dfrac{}{P, (P \supset R) \supset P, R \to R}\,(Ax) \qquad \dfrac{\dfrac{}{(P \supset R) \supset P, P, (P \supset R) \supset P, (((P \supset R) \supset P) \supset P) \supset R \to P}\,(Ax)}{P, (P \supset R) \supset P, (((P \supset R) \supset P) \supset P) \supset R \to ((P \supset R) \supset P) \supset P}\,(\to\supset)}{P, (P \supset R) \supset P, (((P \supset R) \supset P) \supset P) \supset R \to R}\,(\supset\to)}{(P \supset R) \supset P, (((P \supset R) \supset P) \supset P) \supset R \to P \supset R}\,(\to\supset)}{(P \supset R) \supset P, (((P \supset R) \supset P) \supset P) \supset R \to P}\,(\supset\to)}{(((P \supset R) \supset P) \supset P) \supset R \to ((P \supset R) \supset P) \supset P}\,(\to\supset)}{\dfrac{(((P \supset R) \supset P) \supset P) \supset R \to R}{\to ((((P \supset R) \supset P) \supset P) \supset R) \supset R}\,(\to\supset)}\,(\supset\to)
$$

The skeleton of this derivation is

$$
\dfrac{\dfrac{\;}{\;}\,(Ax_0) \qquad \dfrac{\dfrac{\dfrac{\;}{\;}\,(Ax_0) \qquad \dfrac{\dfrac{\dfrac{\;}{\;}\,(Ax_2) \qquad \dfrac{\dfrac{\;}{\;}\,(Ax_1)}{\;}\,(\to\supset)}{\;}\,(\supset\to_2)}{\;}\,(\to\supset)}{\;}\,(\supset\to_0)}{\;}\,(\to\supset)}{\dfrac{\;}{\;}\,(\to\supset)}\,(\supset\to_0)
$$

There may be many derivations of the same formula (even of a propositional one) with the same weak skeleton due to the choice of different subformulas:

Example 2.5
Consider the following two different derivations of the formula $((P \supset \Lambda) \wedge (Q \supset \Lambda) \wedge P \wedge Q) \supset \Lambda$ (here we assume that $\wedge$ is left-associative):

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Lambda, (Q \supset \Lambda), P, Q \to \Lambda}\,(\Lambda) \qquad \dfrac{}{(P \supset \Lambda), (Q \supset \Lambda), P, Q \to P}\,(Ax)}{(P \supset \Lambda), (Q \supset \Lambda), P, Q \to \Lambda}\,(\supset\to)}{(P \supset \Lambda), (Q \supset \Lambda), P \wedge Q \to \Lambda}\,(\wedge\to)}{(P \supset \Lambda), (Q \supset \Lambda) \wedge P \wedge Q \to \Lambda}\,(\wedge\to)}{(P \supset \Lambda) \wedge (Q \supset \Lambda) \wedge P \wedge Q \to \Lambda}\,(\wedge\to)}{\to ((P \supset \Lambda) \wedge (Q \supset \Lambda) \wedge P \wedge Q) \supset \Lambda}\,(\to\supset)
$$

and

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(P \supset \Lambda), \Lambda, P, Q \to \Lambda}\,(\Lambda) \qquad \dfrac{}{(P \supset \Lambda), (Q \supset \Lambda), P, Q \to Q}\,(Ax)}{(P \supset \Lambda), (Q \supset \Lambda), P, Q \to \Lambda}\,(\supset\to)}{(P \supset \Lambda), (Q \supset \Lambda), P \wedge Q \to \Lambda}\,(\wedge\to)}{(P \supset \Lambda), (Q \supset \Lambda) \wedge P \wedge Q \to \Lambda}\,(\wedge\to)}{(P \supset \Lambda) \wedge (Q \supset \Lambda) \wedge P \wedge Q \to \Lambda}\,(\wedge\to)}{\to ((P \supset \Lambda) \wedge (Q \supset \Lambda) \wedge P \wedge Q) \supset \Lambda}\,(\to\supset)
$$

They have the same weak skeleton shown in Example 2.2 but different skeletons:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{}\,(\wedge)\qquad\cfrac{}{}\,(Ax_2)}{}\,(\supset\to_0)}{}\,(\wedge\to_2)}{}\,(\wedge\to_1)}{}\,(\wedge\to_0)}{}\,(\to\supset)
\qquad\qquad
\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{}\,(\wedge)\qquad\cfrac{}{}\,(Ax_3)}{}\,(\supset\to_1)}{}\,(\wedge\to_2)}{}\,(\wedge\to_1)}{}\,(\wedge\to_0)}{}\,(\to\supset)$$

Skeletons uniquely identify derivations of the same sequent without quantifier rules. However, there may be many derivations of the same sequent with the same skeleton due to different instantiations of terms in the quantifier rules. For example, the following two different derivations of the formula $\forall xP(x) \supset \exists xP(x)$ have the same skeleton:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{P(x), \forall xP(x) \to P(x)}\,(Ax)}{\forall xP(x) \to P(x)}\,(\forall\to)}{\forall xP(x) \to \exists xP(x)}\,(\to\exists)}{\to \forall xP(x) \supset \exists xP(x)}\,(\to\supset)
\qquad\qquad
\cfrac{\cfrac{\cfrac{\cfrac{}{P(f(a)), \forall xP(x) \to P(f(a))}\,(Ax)}{\forall xP(x) \to P(f(a))}\,(\forall\to)}{\forall xP(x) \to \exists xP(x)}\,(\to\exists)}{\to \forall xP(x) \supset \exists xP(x)}\,(\to\supset)$$

There are several obvious procedures for constructing derivation skeletons. They can take into account invertibility and permutability of rules in **LJ** and use special strategies for non-invertible rules. We do not consider this heavily investigated subject. We shall consider the problem of instantiations of a (weak) skeleton to a derivation. We will show that this problem is decidable and characterize its complexity in Section 4. We conclude this section by formally defining the corresponding algorithmic problems:

DEFINITION 2.6
The *(weak) skeleton instantiation problem* is the following problem. Given a closed sequent S and a (weak) skeleton $\mathcal{S}$, whether there is a derivation in **LJ** of S with the (weak) skeleton $\mathcal{S}$.

Note that the condition on S to be closed is inessential since one can always replace free variables of a sequent by fresh constants.

3 The Constraint Calculus

In this section we define the constraint calculus $\mathbf{LJ}_c$. It can be used for defining sequent style proof procedures for automatic theorem proving in intuitionistic logic. The proof procedures based on $\mathbf{LJ}_c$ consist of two parts: the construction of a derivation in $\mathbf{LJ}_c$ and constraint satisfaction. The constraint to be checked for satisfiability is uniquely identified by the skeleton of a derivation.

In Section 3.1 we define constraints and their semantics. In Section 3.2 we introduce the constraint calculus $\mathbf{LJ}_c$ and prove its equivalence to **LJ**. Then we show that the skeleton instantiation problem can be reduced to the constraint satisfiability problem.

3.1 Constraints

DEFINITION 3.1 (Constraint)
Constraints are defined inductively as follows:

1. $\top$ and $\bot$ are constraints;
2. For any terms s and t, $s = t$ is a constraint;
3. For any terms $s, t_1, \ldots, t_n$, where $n \geq 0$, $s \not\preceq \{t_1, \ldots, t_n\}$ is a constraint;
4. If $\mathcal{C}_1$ and $\mathcal{C}_2$ are constraints, then $\mathcal{C}_1 \wedge \mathcal{C}_2$ is a constraint;
5. If $\mathcal{C}$ is a constraint and x is variable, then $\exists x \mathcal{C}$ is a constraint.

The constraints $\top$, $\bot$, $s = t$ and $s \not\preceq \{t_1, \ldots, t_n\}$ are called *atomic.*

Constraints can be considered as first order formulas in a signature whose predicate symbols are $\top$, $\bot$, $=$ and the symbols $\ldots \not\preceq \{\ldots\}$ of all arities ≥ 1. Constraints of the form $s = t$ are called *equality constraints.* We shall not distinguish the constraint $s = t$ from $t = s$.

Constraints describe the domain of substitutions. If a constraint $\mathcal{C}$ is true on a substitution σ we say that σ *satisfies* $\mathcal{C}$:

DEFINITION 3.2 (Constraint satisfiability)
The notion *"a substitution σ satisfies the constraint $\mathcal{C}$"* (denoted $\sigma \models \mathcal{C}$) is defined inductively as follows:

1. $\sigma \models \top$ for every substitution σ;
2. There is no substitution σ such that $\sigma \models \bot$;
3. $\sigma \models s = t$ iff $s\sigma \approx t\sigma$;
4. $\sigma \models s \not\preceq \{t_1, \ldots, t_n\}$ iff $s\sigma$ is a variable and $s\sigma$ does not occur in $t_1\sigma, \ldots, t_n\sigma$;
5. $\sigma \models \mathcal{C}_1 \wedge \mathcal{C}_2$ iff $\sigma \models \mathcal{C}_1$ and $\sigma \models \mathcal{C}_2$;
6. $\sigma \models \exists x \mathcal{C}$ iff there is a substitution τ such that $\tau \models \mathcal{C}$ and $y\tau \approx y\sigma$ for every variable y different from x.

Constraints $\mathcal{C}_1$ and $\mathcal{C}_2$ are *equivalent* iff for every substitution σ we have $\sigma \models \mathcal{C}_1$ iff $\sigma \models \mathcal{C}_2$. A constraint $\mathcal{C}$ is *satisfiable* iff there is a substitution σ satisfying $\mathcal{C}$.

3.2 Calculus $\mathbf{LJ}_c$

The calculus $\mathbf{LJ}_c$ derives *constrained sequents*:

DEFINITION 3.3 (Constrained sequent)
A *constrained sequent* is an expression $S \cdot \mathcal{C}$ where S is a sequent and $\mathcal{C}$ is a constraint. The *constraint calculus* $\mathbf{LJ}_c$ of constrained sequents is shown in Figure 2.

Consider two examples of derivations in $\mathbf{LJ}_c$:

EXAMPLE 3.4
The example of a derivation in $\mathbf{LJ}_c$ of the formula $\exists x \forall y P(x, y) \supset \forall y \exists x P(x, y)$ with a suitable constraint:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P(v,w), \forall y P(v,y) \to P(u,z) \cdot v = u \wedge w = z}\ (Ax)}{\forall y P(v,y) \to P(u,z) \cdot \exists w (v = u \wedge w = z)}\ (\forall \to)}{\forall y P(v,y) \to \exists x P(x,z) \cdot \exists u \exists w (v = u \wedge w = z)}\ (\to \exists)}{\exists x \forall y P(x,y) \to \exists x P(x,z) \cdot \exists v (v \not\preceq \{z\} \wedge \exists u \exists w (v = u \wedge w = z))}\ (\exists \to)}{\exists x \forall y P(x,y) \to \forall y \exists x P(x,y) \cdot \exists z (z \not\preceq \{\} \wedge \exists v (v \not\preceq \{z\} \wedge \exists u \exists w (v = u \wedge w = z)))}\ (\to \forall)}{\to \exists x \forall y P(x,y) \supset \forall y \exists x P(x,y) \cdot \exists z (z \not\preceq \{\} \wedge \exists v (v \not\preceq \{z\} \wedge \exists u \exists w (v = u \wedge w = z)))}\ (\to\supset)
$$

$$\frac{}{\Gamma, P(s_1,\ldots,s_n), \Delta \to P(t_1,\ldots,t_n)\cdot s_1 = t_1 \wedge \ldots \wedge s_n = t_n}\ (Ax) \qquad \frac{}{\Gamma, \Lambda, \Delta \to \varphi \cdot \top}\ (\Lambda)$$

$$\frac{\Gamma, \varphi, \psi, \Delta \to \chi \cdot \mathcal{C}}{\Gamma, \varphi \wedge \psi, \Delta \to \chi \cdot \mathcal{C}}\ (\wedge \to) \qquad \frac{\Gamma \to \varphi \cdot \mathcal{C}_1 \quad \Gamma \to \psi \cdot \mathcal{C}_2}{\Gamma \to \varphi \wedge \psi \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (\to \wedge)$$

$$\frac{\Gamma, \varphi, \Delta \to \chi \cdot \mathcal{C}_1 \quad \Gamma, \psi, \Delta \to \chi \cdot \mathcal{C}_2}{\Gamma, \varphi \vee \psi, \Delta \to \chi \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (\vee \to)$$

$$\frac{\Gamma \to \varphi \cdot \mathcal{C}}{\Gamma \to \varphi \vee \psi \cdot \mathcal{C}}\ (\to \vee_1) \qquad \frac{\Gamma \to \psi \cdot \mathcal{C}}{\Gamma \to \varphi \vee \psi \cdot \mathcal{C}}\ (\to \vee_2)$$

$$\frac{\Gamma, \psi, \Delta \to \chi \cdot \mathcal{C}_1 \quad \Gamma, \varphi \supset \psi, \Delta \to \varphi \cdot \mathcal{C}_2}{\Gamma, \varphi \supset \psi, \Delta \to \chi \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (\supset \to) \qquad \frac{\varphi, \Gamma \to \psi \cdot \mathcal{C}}{\Gamma \to \varphi \supset \psi \cdot \mathcal{C}}\ (\to \supset)$$

$$\frac{\Gamma, \varphi\{y/x\}, \forall x\varphi \to \chi, \Delta \cdot \mathcal{C}}{\Gamma, \forall x\varphi, \Delta \to \chi \cdot \exists y \mathcal{C}}\ (\forall \to) \qquad \frac{\Gamma \to \varphi\{y/x\} \cdot \mathcal{C}}{\Gamma \to \forall x\varphi \cdot \exists y(y \not\preceq \{v_1,\ldots,v_n\} \wedge \mathcal{C})}\ (\to \forall)$$

$$\frac{\Gamma, \varphi\{y/x\}, \Delta \to \chi \cdot \mathcal{C}}{\Gamma, \exists x\varphi, \Delta \to \chi \cdot \exists y(y \not\preceq \{v_1,\ldots,v_n\} \wedge \mathcal{C})}\ (\exists \to) \qquad \frac{\Gamma \to \varphi\{y/x\} \cdot \mathcal{C}}{\Gamma \to \exists x\varphi \cdot \exists y \mathcal{C}}\ (\to \exists)$$

For any of the rules $(\forall \to)$–$(\to \exists)$, the variable y has no free occurrences in the sequent of the conclusion of the rule. In the rule $(\to \forall)$, $v_1,\ldots,v_n$ are all free variables of $\Gamma, \forall x\varphi$. In the rule $(\exists \to)$, $v_1,\ldots,v_n$ are all free variables of $\Gamma, \exists x\varphi, \Delta, \chi$.

FIG. 2. Calculus $\mathbf{LJ}_c$

As we shall see later, the resulting constraint $\exists z(z \not\preceq \{\} \wedge \exists v(v \not\preceq \{z\} \wedge \exists u \exists w(v = u \wedge w = z)))$ is satisfiable.

EXAMPLE 3.5
The example of a derivation in $\mathbf{LJ}_c$ of the formula $\forall x \exists y P(x,y) \supset \exists y \forall x P(x,y)$ with a suitable constraint (this formula is not provable in $\mathbf{LJ}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P(v,w), \forall x\exists y P(x,y) \to P(u,z) \cdot v = u \wedge w = z}\ (Ax)}{\exists y P(v,y), \forall x\exists y P(x,y) \to P(u,z) \cdot \exists w(w \not\preceq \{v,u,z\} \wedge v = u \wedge w = z)}\ (\exists \to)}{\exists y P(v,y), \forall x\exists y P(x,y) \to \forall x P(x,z) \cdot \exists u(u \not\preceq \{v,z\} \wedge \exists w(w \not\preceq \{v,u,z\} \wedge v = u \wedge w = z))}\ (\to \forall)}{\forall x\exists y P(x,y) \to \forall x P(x,z) \cdot \exists v \exists u(u \not\preceq \{v,z\} \wedge \exists w(w \not\preceq \{v,u,z\} \wedge v = u \wedge w = z))}\ (\forall \to)}{\forall x\exists y P(x,y) \to \exists y \forall x P(x,y) \cdot \exists z \exists v \exists u(u \not\preceq \{v,z\} \wedge \exists w(w \not\preceq \{v,u,z\} \wedge v = u \wedge w = z))}\ (\to \exists)}{\to \forall x\exists y P(x,y) \supset \exists y \forall x P(x,y) \cdot \exists z \exists v \exists u(u \not\preceq \{v,z\} \wedge \exists w(w \not\preceq \{v,u,z\} \wedge v = u \wedge w = z))}\ (\to \supset)$$

As we shall see later, the resulting constraint $\exists z \exists v \exists u(u \not\preceq \{v,z\} \wedge \exists w(w \not\preceq \{v,u,z\} \wedge v = u \wedge w = z))$ is unsatisfiable.

The notions of a skeleton and a weak skeleton for $\mathbf{LJ}_c$-derivations are defined in the same way as for $\mathbf{LJ}$-derivations. The following statement connects the derivability in $\mathbf{LJ}$ with the derivability in $\mathbf{LJ}_c$:

THEOREM 3.6
Let $\Gamma \to \varphi$ be a sequent, θ be a substitution and $\mathcal{S}$ be a skeleton. Then the following conditions are equivalent:

1. There is a derivation of $\Gamma\theta \to \varphi\theta$ in **LJ** with the skeleton $\mathcal{S}$;
2. There is a constraint $\mathcal{C}$ and a derivation of $\Gamma \to \varphi \cdot \mathcal{C}$ in $\mathbf{LJ}_c$ with the skeleton $\mathcal{S}$ such that $\theta \models \mathcal{C}$.

PROOF. By induction on the derivation in **LJ** (respectively, $\mathbf{LJ}_c$). Consider several cases depending on the last inference rule applied. In all cases the condition on the skeletons to be the same is straightforward.

1. The last rule applied in the **LJ**-derivation is (Ax):

$$\frac{}{\Gamma\theta \to P(t_1,\ldots,t_n)\theta}\ (Ax)$$

Then $P(t_1,\ldots,t_n)\theta \in \Gamma\theta$. It means that there is a formula $P(s_1,\ldots,s_n) \in \Gamma$ such that $s_i\theta \approx t_i\theta$, for all $i \in \{1,\ldots,n\}$. Define $\mathcal{C}$ as $s_1 = t_1 \wedge \ldots \wedge s_n = t_n$. The following is the required $\mathbf{LJ}_c$-derivation:

$$\frac{}{\Gamma \to P(s_1,\ldots,s_n) \cdot \mathcal{C}}\ (Ax)$$

The property $\theta \models \mathcal{C}$ is obvious.

2. The last rule applied in the $\mathbf{LJ}_c$-derivation is (Ax):

$$\frac{}{\Gamma, P(s_1,\ldots,s_n), \Delta \to P(t_1,\ldots,t_n) \cdot s_1 = t_1 \wedge \ldots \wedge s_n = t_n}\ (Ax)$$

Suppose $\theta \models s_1 = t_1 \wedge \ldots \wedge s_n = t_n$. Then $s_i\theta \approx t_i\theta$ for all $i \in \{1,\ldots,m\}$. The required **LJ**-derivation is

$$\frac{}{\Gamma\theta, P(s_1,\ldots,s_n)\theta, \Delta\theta \to P(t_1,\ldots,t_n)\theta}\ (Ax)$$

3. The last rule applied in the **LJ**-derivation is $(\to \wedge)$. Then the **LJ**-derivation has the form

$$\frac{\begin{array}{c}\Pi_1\\ \Gamma\theta \to \varphi\theta\end{array} \quad \begin{array}{c}\Pi_2\\ \Gamma\theta \to \psi\theta\end{array}}{\Gamma\theta \to \varphi\theta \wedge \psi\theta}\ (\to \wedge)$$

By the induction hypothesis, there are $\mathbf{LJ}_c$-derivations Π'_1 and Π'_2 of the constrained sequents $\Gamma \to \varphi \cdot \mathcal{C}_1$ and $\Gamma \to \psi \cdot \mathcal{C}_2$, respectively, such that $\theta \models \mathcal{C}_1$ and $\theta \models \mathcal{C}_2$. By the definition of constraint satisfaction, $\theta \models \mathcal{C}_1 \wedge \mathcal{C}_2$. Define $\mathcal{C}$ as $\mathcal{C}_1 \wedge \mathcal{C}_2$. Then the required $\mathbf{LJ}_c$-derivation is

$$\frac{\begin{array}{c}\Pi'_1\\ \Gamma \to \varphi \cdot \mathcal{C}_1\end{array} \quad \begin{array}{c}\Pi'_2\\ \Gamma \to \psi \cdot \mathcal{C}_2\end{array}}{\Gamma \to \varphi \wedge \psi \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (\to \wedge)$$

4. The last rule applied in the $\mathbf{LJ}_c$-derivation is $(\to \wedge)$. Then the $\mathbf{LJ}_c$-derivation has the form

$$\frac{\begin{array}{c}\Pi_1\\ \Gamma \to \varphi \cdot \mathcal{C}_1\end{array} \quad \begin{array}{c}\Pi_2\\ \Gamma \to \psi \cdot \mathcal{C}_2\end{array}}{\Gamma \to \varphi \wedge \psi \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (\to \wedge)$$

Let $\theta \models \mathcal{C}_1 \wedge \mathcal{C}_2$. Then $\theta \models \mathcal{C}_1$ and $\theta \models \mathcal{C}_2$. By the induction hypothesis, there are **LJ**-derivations Π'_1 and Π'_2 of the sequents $\Gamma\theta \to \varphi\theta$ and $\Gamma\theta \to \psi\theta$, respectively. Then the required **LJ**-derivation is

$$\frac{\begin{array}{cc}\Pi_1' & \Pi_2' \\ \Gamma\theta \to \varphi\theta & \Gamma\theta \to \psi\theta\end{array}}{\Gamma\theta \to \varphi\theta \wedge \psi\theta}\ (\to \wedge)$$

5. The last rule applied in the **LJ**-derivation is $(\to \forall)$. Let y be the eigenvariable of this rule and $\Sigma \to \psi$ be the premise of the rule. Then it is easy to see that for any variable z not occurring in Σ or ψ, the sequent $\Sigma\{z/y\} \to \psi\{z/y\}$ is also derivable in **LJ** with the same skeleton. Hence, without loss of generality we can assume that the eigenvariable does not occur in Γ or φ. Then the **LJ**-derivation has the form

$$\frac{\begin{array}{c}\Pi \\ \Gamma\theta' \to \varphi\{y/x\}\theta'\end{array}}{\Gamma\theta \to (\forall x\varphi)\theta}\ (\to \forall)$$

where θ' is a substitution such that $v\theta \approx v\theta'$ for all variables v different from y and $y\theta' \approx y$. By the induction hypothesis, there is an $\mathbf{LJ}_c$-derivation Π' of a constrained sequent $\Gamma \to \varphi\{y/x\} \cdot \mathcal{C}$ such that $\theta' \models \mathcal{C}$. Let $v_1, \ldots, v_m$ be all free variables of $\Gamma, \forall x\varphi$. Since $y\theta' \approx y$ and y does not occur in Γ, φ, we have $\theta' \models y \not\preceq \{v_1, \ldots, v_m\} \wedge \mathcal{C}$. It follows that $\theta \models \exists y(y \not\preceq \{v_1, \ldots, v_m\} \wedge \mathcal{C})$. The required $\mathbf{LJ}_c$-derivation is

$$\frac{\begin{array}{c}\Pi' \\ \Gamma \to \varphi\{y/x\} \cdot \mathcal{C}\end{array}}{\Gamma \to \forall x\varphi \cdot \exists y(y \not\preceq \{v_1, \ldots, v_m\} \wedge \mathcal{C})}\ (\to \forall)$$

6. The last rule applied in the $\mathbf{LJ}_c$-derivation is $(\to \forall)$. Then the $\mathbf{LJ}_c$-derivation has the form

$$\frac{\begin{array}{c}\Pi \\ \Gamma \to \varphi\{y/x\} \cdot \mathcal{C}\end{array}}{\Gamma \to \forall x\varphi \cdot \exists y(y \not\preceq \{v_1, \ldots, v_n\} \wedge \mathcal{C})}\ (\to \forall)$$

Let $\theta \models \exists y(y \not\preceq \{v_1, \ldots, v_n\} \wedge \mathcal{C})$. By the definition of constraint satisfaction, there is a substitution θ' such that $v\theta' \approx v\theta$ for all variables different from y and $\theta' \models y \not\preceq \{v_1, \ldots, v_m\} \wedge \mathcal{C}$. It means that $\theta' \models \mathcal{C}$ and $y\theta'$ is a variable not occurring in $v_1\theta', \ldots, v_m\theta'$. By the induction hypothesis, there is derivation Π' in **LJ** of the sequent $\Gamma\theta' \to \varphi\{y/x\}\theta'$. Then the required **LJ**-derivation is

$$\frac{\begin{array}{c}\Pi' \\ \Gamma\theta' \to \varphi\{y/x\}\theta'\end{array}}{\Gamma\theta \to (\forall x\varphi)\theta}\ (\to \forall)$$

The other cases are similar.

∎

The following two theorems are straightforward consequences of Theorem 3.6:

THEOREM 3.7
Let S be a closed sequent and $\mathcal{S}$ be a skeleton. Then S has a derivation in **LJ** with the skeleton $\mathcal{S}$ iff there is a satisfiable constraint $\mathcal{C}$ such that $S \cdot \mathcal{C}$ has a derivation in $\mathbf{LJ}_c$ with the skeleton $\mathcal{S}$.

The same holds with "skeleton" replaced by "weak skeleton".

THEOREM 3.8
Let S be a closed sequent. Then S is derivable in **LJ** iff there is a satisfiable constraint $\mathcal{C}$ such that $S \cdot \mathcal{C}$ is derivable in $\mathbf{LJ}_c$.

Note that the constraint $\mathcal{C}$ such that $S \cdot \mathcal{C}$ is derivable in $\mathbf{LJ}_c$ with a skeleton $\mathcal{S}$ is uniquely defined by S and $\mathcal{S}$ in the following sense. Suppose that the constrained sequents $S \cdot \mathcal{C}_1$ and $S \cdot \mathcal{C}_2$ have $\mathbf{LJ}_c$-derivations with the same skeleton. Then $\mathcal{C}_1$ is obtained from $\mathcal{C}_2$ by renaming bound variables.

Thus, the proof-search in intuitionistic logic can be considered as consisting of two parts: constructing a skeleton and constraint satisfaction. The two parts should not necessarily be processed in this order, for example the constraint satisfaction algorithm can check constraints after each step of the skeleton construction.

4 Constraint satisfaction

In this section we prove that there is a polynomial time algorithm for the constraint satisfiability problem. The algorithm we use in the proof is a modification of the unification algorithm of Martelli and Montanari [15]. It transforms the input constraint $\mathcal{C}$ step by step into a constraint of a special form for which satisfiability or unsatisfiability can easily be checked. Each transformation step yields a constraint equivalent to $\mathcal{C}$.

Here we give a direct proof for arbitrary constraints. In fact, derivations in $\mathbf{LJ}_c$ do not give arbitrary constraints, and we could have used a different technique for these specialized constraints using the following idea. Using skolemization, we can get rid of constraints of the form $x \not\preceq \{v_1, \ldots, v_n\}$. Then the constraint satisfiability problem reduces to the unifiability problem for which linear time algorithms exist.

The interest in a direct proof for arbitrary constraints lies with Theorem 3.6 that characterizes all instantiations of derivations with a given skeleton. This characterization implies that we can replace a constraint constructed by an $\mathbf{LJ}_c$-derivation by an arbitrary equivalent constraint, and the new constraint will still characterize all instantiations of derivations with the given skeleton. A constraint $\mathcal{C}'$ obtained from a constraint $\mathcal{C}$ by skolemization is equally satisfiable to $\mathcal{C}$, but it is not equivalent. Thus, when we replace constraints by equivalent constraints or when we try to compose $\mathbf{LJ}_c$-derivations, it is more helpful to prove satisfiability of general constraints.

From here and up to Theorem 4.8 we introduce several purely technical definitions and lemmas. A reader not interested in technical details can skip this part of the article.

For simplicity, we shall deal with *prenex constraints,* but the technique used below is (with minor modifications) applicable to arbitrary constraints.

DEFINITION 4.1 (Prenex constraint, isolated constraint)
A constraint $\mathcal{C}$ is *prenex* iff it has the form $\exists \bar{x}(\mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_n)$, where $\mathcal{C}_i$ are atomic constraints.

Let $\mathcal{C}$ be a prenex constraint of this form. A variable x is called *isolated in* $\mathcal{C}$ iff some $\mathcal{C}_i$ has the form $x = t$ and x does not occur in t and in any $\mathcal{C}_j$ for $j \neq i$. A constraint $\mathcal{C}_i$ is called *isolated in* $\mathcal{C}$ iff an isolated variable occurs in it.

Any constraint can be transformed to an equivalent prenex one by means of the standard prenexing transformation $\mathcal{C}[\ldots \exists x \varphi \ldots] \Rightarrow \exists y \mathcal{C}[\ldots \varphi\{y/x\} \ldots]$. Obviously, there is a polynomial time prenexing algorithm.

We define the notion of a saturated constraint. A constraint $\mathcal{C}$ is saturated iff no transformation step defined in Lemma 4.5 below is applicable to $\mathcal{C}$.

DEFINITION 4.2 (Saturated prenex constraint)
Let $\mathcal{C}$ be a prenex constraint $\exists\bar{x}(\mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_n)$, where $\mathcal{C}_1, \ldots, \mathcal{C}_n$ are atomic. It is called *saturated* iff

1. For every $\mathcal{C}_i$ of the form $s \not\preceq \{t_1, \ldots, t_m\}$, the term s is a variable;
2. Every equality constraint $\mathcal{C}_i$ is either isolated in $\mathcal{C}$ or has the form $x = t$, where x is a variable and t is not a variable;
3. There is no variable x such that some $\mathcal{C}_i$ has the form $x \not\preceq \{t_1, \ldots, t_n\}$ and some $\mathcal{C}_j$ has the form $x = t$, where t is not a variable;
4. There is no variable x such that some $\mathcal{C}_i$ has the form $x = s$ and some $\mathcal{C}_j$ with $j \neq i$ has the form $x = t$.

We shall describe an algorithm transforming any constraint to an equivalent saturated prenex constraint and give necessary and sufficient conditions for a saturated prenex constraint to be satisfiable.

DEFINITION 4.3
Let s, t be two terms. The *common part* of s and t, denoted $cp(s,t)$, is a term and the *disagreement constraint* of s, t, denoted $DC(s,t)$, is a constraint defined inductively as follows:

1. If $s \approx t$, then $cp(s,t) \rightleftharpoons s$ and $DC(s,t) \rightleftharpoons \top$;
2. Otherwise, if s is a variable, then $cp(s,t) \rightleftharpoons s$ and $DC(s,t) \rightleftharpoons (s = t)$;
3. Otherwise, if t is a variable, then $cp(s,t) \rightleftharpoons t$ and $DC(s,t) \rightleftharpoons (t = s)$;
4. Otherwise, if s has the form $f(s_1, \ldots, s_n)$ and t has the form $g(t_1, \ldots, t_m)$ for different function symbols f, g, then $cp(s,t)$ and $DC(s,t)$ are undefined;
5. Otherwise, s has the form $f(s_1, \ldots, s_n)$ and t has the form $f(t_1, \ldots, t_n)$. If, for all $i \in \{1, \ldots, n\}$ the term $cp(s_i, t_i)$ is defined, then $cp(s,t) \rightleftharpoons f(cp(s_1,t_1), \ldots, cp(s_n,t_n))$ and $DC(s,t) \rightleftharpoons DC(s_1,t_1) \wedge \ldots \wedge DC(s_n,t_n)$. Otherwise, $cp(s,t)$ and $DC(s,t)$ are undefined.

For example, for the terms $f(x, g(y))$ and $f(x, c)$ the common part is undefined. For the terms $f(x, g(h(x,y)), z)$ and $f(x, g(z), g(y))$, the common part is $f(x, g(z), z)$ and the disagreement constraint is $\top \wedge z = h(x,y) \wedge z = g(y)$. In the following, we shall often remove $\top$ from conjunctions of constraints. Then we can say that the disagreement constraint for these terms is $z = h(x,y) \wedge z = g(y)$.

LEMMA 4.4
Let s, t be terms and x be a variable. If $cp(s,t)$ is undefined, then the constraint $s = t$ is unsatisfiable. If $cp(s,t)$ is defined, then the constraint $x = s \wedge x = t$ is equivalent to $x = cp(s,t) \wedge DC(s,t)$.

PROOF. Straightforward induction on the depth of s, t, using Definition 4.3. ∎

LEMMA 4.5
There is a polynomial time algorithm transforming any constraint $\mathcal{C}$ into an equivalent saturated prenex constraint $\mathcal{C}'$.

PROOF. Without loss of generality we can assume that $\mathcal{C}$ is quantifier-free. Then $\mathcal{C}$ has the form $\mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_k$ where all $\mathcal{C}_i$ are atomic. The algorithm consists of the steps transforming $\mathcal{C}$ into an equivalent constraint according to the following rules that can be applied in any order. The algorithm terminates when no step is applicable.

(Isolate) Some equality constraint $\mathcal{C}_i$ not isolated in $\mathcal{C}$ has the form $x = y$. If $x \approx y$, then replace $\mathcal{C}_i$ by $\top$ in $\mathcal{C}$. Otherwise, replace the variable x by the variable y in all constraints $\mathcal{C}_j$ for $j \neq i$.

(Merge) Some constraints $\mathcal{C}_i$ and $\mathcal{C}_j$, $i \neq j$ have the form $x = s$ and $x = t$, where s and t are not variables. If $cp(s,t)$ is undefined, set $\mathcal{C}$ to $\perp$. Otherwise, replace in $\mathcal{C}$ the constraint $\mathcal{C}_i$ by $x = cp(s,t)$ and the constraint $\mathcal{C}_j$ by $DC(s,t)$.

(Split) Some constraint $\mathcal{C}_i$ has the form $s = t$, where s and t are not variables. If $cp(s,t)$ is undefined, set $\mathcal{C}$ to $\perp$. Otherwise, replace $\mathcal{C}_i$ in $\mathcal{C}$ by $DC(s,t)$.

(Variable fail) Some constraint $\mathcal{C}_i$ has the form $t \not\preceq \{s_1, \ldots, s_n\}$ such that either t is not a variable or some constraint $\mathcal{C}_j$ has the form $t = s$, where s is not a variable. Set $\mathcal{C}$ to $\perp$.

First, we check that each of the above steps transforms the constraint $\mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_k$ into an equivalent constraint. For (Isolate) and (Variable fail) it is evident. For (Merge) and (Split) it follows from Lemma 4.4.

Then, we prove that this algorithm works in polynomial time and yields a saturated prenex constraint.

Note that the conditions for each step can be checked in polynomial time. Hence, it is enough to prove that the algorithm terminates in a polynomial number of steps. Let us call the *height* of $\mathcal{C}$ the natural number $n + m$, where n is the number of non-isolated variables in $\mathcal{C}$ and m is the number of occurrences of function symbols in $\mathcal{C}$. Initially, the height of $\mathcal{C}$ is bound by the size of $\mathcal{C}$. Every (Isolate) step decreases the number of non-isolated variables and does not change the number of occurrences of function symbols. Every (Merge) or (Split) step decreases the number of occurrences of function symbols in $\mathcal{C}$ and does not increase the number of non-isolated variables. Every (Variable fail) step obviously decreases the height of $\mathcal{C}$. Thus, every step of the algorithm decreases the height of $\mathcal{C}$. Since the height is non-negative, the algorithm terminates in a number of steps linear in the size of the input.

Now we prove that after the termination, the resulting constraint is saturated. We check all conditions of Definition 4.2.

1. *For every $\mathcal{C}_i$ of the form $s \not\preceq \{t_1, \ldots, t_m\}$, the term s is a variable.* Indeed, if s is not a variable, then (Variable fail) is applicable.
2. *Every equality constraint $\mathcal{C}_i$ is either isolated in $\mathcal{C}$ or has the form $x = t$, where x is a variable and t is not a variable.* Suppose $\mathcal{C}_i$ has the form $r = s$. If both r, s are not variables, then (Split) is applicable. If both r, s are variables then either (Isolate) is applicable or $\mathcal{C}_i$ is isolated. If one of r, s is a variable and one of these terms is not a variable, then the condition is trivially satisfied.
3. *There is no variable x such that some $\mathcal{C}_i$ has the form $x \not\preceq \{t_1, \ldots, t_n\}$ and some $\mathcal{C}_j$ has the form $x = t$, where t is not a variable.* This is true because otherwise (Variable fail) would be applicable.
4. *There is no variable x such that some $\mathcal{C}_i$ has the form $x = s$ and some $\mathcal{C}_j$ with $j \neq i$ has the form $x = t$.* This is true because otherwise (Merge) would be applicable.

The proof is completed. ∎

DEFINITION 4.6 (Relations $<^*_\mathcal{C}$, $\leq^*_\mathcal{C}$ and $\not\preceq_\mathcal{C}$)
Let $\mathcal{C}$ be a saturated prenex constraint of the form $\exists\bar{u}(\mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_k)$, where $\mathcal{C}_1, \ldots, \mathcal{C}_k$ are atomic. Introduce the binary relation $<_\mathcal{C}$ on the set of variables non-isolated in $\mathcal{C}$ in the following way: for two non-isolated variables x and y, we have $x <_\mathcal{C} y$ iff some $\mathcal{C}_i$ has the form $y = t$ and x occurs in t. The *relation* $<^*_\mathcal{C}$ is the transitive closure of $<_\mathcal{C}$. The *relation* $\leq^*_\mathcal{C}$ is the reflexive and transitive closure of $<_\mathcal{C}$. The *binary relation* $\not\preceq_\mathcal{C}$ on the set of non-isolated variables of $\mathcal{C}$ is defined in the following way: for two variables x, y non-isolated in $\mathcal{C}$ we have $x \not\preceq_\mathcal{C} y$ iff some $\mathcal{C}_i$ has the form $x \not\preceq \{t_1, \ldots, t_n\}$ and y occurs in $\{t_1, \ldots, t_n\}$.

LEMMA 4.7
Let $\mathcal{C}$ be a saturated prenex constraint of the form $\exists\bar{u}(\mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_k)$, where $\mathcal{C}_1, \ldots, \mathcal{C}_k$ are atomic. It is satisfiable iff

1. The relation $<^*_\mathcal{C}$ is irreflexive, i.e. there is no variable x such that $x <^*_\mathcal{C} x$;
2. $\leq^*_\mathcal{C} \cap \not\preceq_\mathcal{C} = \emptyset$, i.e. there is no pair of variables x, y such that $x \leq^*_\mathcal{C} y$ and $x \not\preceq_\mathcal{C} y$.

PROOF. Define a new transformation step on the saturated prenex constraints:

> (Replacement) If for some i the constraint $\mathcal{C}_i$ has the form $x = t$, the variable x does not occur in t and occurs in some $\mathcal{C}_j$ for $j \neq i$, replace x by t in all $\mathcal{C}_j$ such that $j \neq i$.

Note that the constraint obtained from a saturated prenex constraint by (Replacement) is also saturated. We prove several intermediate statements.

Let $\mathcal{C}'$ be a constraint obtained from $\mathcal{C}$ by (Replacement).

(I) *$\mathcal{C}$ and $\mathcal{C}'$ are equivalent.*
This statement is obvious.

(II) *The relation $<^*_\mathcal{C}$ is irreflexive iff the relation $<^*_{\mathcal{C}'}$ is irreflexive.*
Let $y_1, \ldots, y_n$ be all variables of t. We have $x \notin \{y_1, \ldots, y_n\}$. Let

$$\begin{array}{c} z_1 = s_1 \\ \cdots \\ z_m = s_m \end{array} \tag{4.1}$$

be all non-isolated equality constraints among C_j such that x occurs in a non-variable term in C_j. Since $\mathcal{C}$ is saturated we have $x \notin \{z_1, \ldots, z_m\}$. Then all pairs belonging to $<_\mathcal{C}$ in which x occurs are

$$\begin{array}{ccc} y_1 <_\mathcal{C} x & \ldots & y_n <_\mathcal{C} x \\ x <_\mathcal{C} z_1 & \ldots & x <_\mathcal{C} z_m \end{array} \tag{4.2}$$

Note that in $\mathcal{C}'$ constraints (4.1) are replaced by

$$\begin{array}{c} z_1 = s_1\{t/x\} \\ \cdots \\ z_m = s_m\{t/x\} \end{array}$$

Hence, $<_{\mathcal{C}'}$ is obtained from $<_\mathcal{C}$ by removing all pairs in (4.2) and adding

$$\begin{array}{lll} y_1 <_{\mathcal{C}'} z_1 & \dots & y_1 <_{\mathcal{C}'} z_m \\ y_n <_{\mathcal{C}'} z_1 & \dots & y_n <_{\mathcal{C}'} z_m \end{array}$$

Using the fact that $x \notin \{y_1, \dots, y_n, z_1, \dots, z_m\}$, it is easy to see that $<^*_{\mathcal{C}}$ is irreflexive if and only if $<^*_{\mathcal{C}'}$ is irreflexive.

The same considerations prove that in general

(III) *$<^*_{\mathcal{C}'}$ is obtained from $<^*_{\mathcal{C}}$ by removing all pairs of the form $x <^*_{\mathcal{C}} u$ or $u <^*_{\mathcal{C}} x$. In particular, $<^*_{\mathcal{C}'} \subseteq <^*_{\mathcal{C}}$.*

Let us now prove

(IV) *$\leq^*_{\mathcal{C}} \cap \not\preceq_{\mathcal{C}} \neq \emptyset$ iff $\leq^*_{\mathcal{C}'} \cap \not\preceq_{\mathcal{C}'} \neq \emptyset$.*

Let $(u, v) \in \leq^*_{\mathcal{C}} \cap \not\preceq_{\mathcal{C}}$. Then $u \leq^*_{\mathcal{C}} v$ and $u \not\preceq_{\mathcal{C}} v$. Since $\mathcal{C}$ is saturated and $u \not\preceq_{\mathcal{C}} v$, we have $u \not\approx x$. Consider two cases.

(a) $v \not\approx x$. Then, evidently, $u \not\preceq_{\mathcal{C}'} v$. By $u \leq^*_{\mathcal{C}} v$ and (III) we get $u \leq^*_{\mathcal{C}'} v$. Hence, $(u, v) \in \leq^*_{\mathcal{C}'} \cap \not\preceq_{\mathcal{C}'}$.

(b) $v \approx x$. Then $u \not\preceq_{\mathcal{C}} x$ and $u \leq^*_{\mathcal{C}} x$. By $u \not\preceq_{\mathcal{C}} x$ we get $u \not\preceq_{\mathcal{C}'} y_1, \dots, u \not\preceq_{\mathcal{C}'} y_n$. Since $u \not\approx x$ and $u \leq^*_{\mathcal{C}} x$, for some $l \in \{1, \dots, n\}$ we have $u \leq^*_{\mathcal{C}} y_l$. By (III), we have $u \leq^*_{\mathcal{C}'} y_l$. Hence, $(u, y_l) \in \leq^*_{\mathcal{C}'} \cap \not\preceq_{\mathcal{C}'}$.

Suppose now $(u, v) \in \leq^*_{\mathcal{C}'} \cap \not\preceq_{\mathcal{C}'}$. Consider two cases

(a) $u \not\preceq_{\mathcal{C}} v$. Then using $\leq^*_{\mathcal{C}'} \subseteq \leq^*_{\mathcal{C}}$ we obtain $(u, v) \in \leq^*_{\mathcal{C}} \cap \not\preceq_{\mathcal{C}}$.

(b) It is not true that $u \not\preceq_{\mathcal{C}} v$. This means that for some j and l the constraint $\mathcal{C}_j$ has the form $u \not\preceq \{\bar{r}\}$, x occurs in $\bar{r}$ and $v \approx y_l$. Since $u \leq^*_{\mathcal{C}'} v$, we have $u \leq^*_{\mathcal{C}'} y_l$ and hence by (III) $u \leq^*_{\mathcal{C}} y_l$. Since $y_l \leq^*_{\mathcal{C}} x$, we obtain $u \leq^*_{\mathcal{C}} x$. Hence, $(u, x) \in \leq^*_{\mathcal{C}} \cap \not\preceq_{\mathcal{C}}$.

Thus, (IV) is proved.

An obvious consequence of (I), (II) and (IV) is

(V) *Let $\mathcal{C}'$ be a constraint obtained from $\mathcal{C}$ by several applications of (Replacement). Then*

(a) $\mathcal{C}'$ is equivalent to $\mathcal{C}$;

(b) $<^*_{\mathcal{C}}$ is irreflexive and $\leq^*_{\mathcal{C}} \cap \not\preceq_{\mathcal{C}} = \emptyset$ iff $<^*_{\mathcal{C}'}$ is irreflexive and $\leq^*_{\mathcal{C}'} \cap \not\preceq_{\mathcal{C}'} = \emptyset$.

Let us now apply (Replacement) to $\mathcal{C}$ repeatedly while it is applicable. We can apply it only a finite number of times since every application isolates some non-isolated variable. Denote the resulting constraint by $\mathcal{C}'$. Then $\mathcal{C}'$ has the form $\exists \bar{u}(\mathcal{C}'_1 \wedge \ldots \wedge \mathcal{C}'_k)$. Consider the following three possible cases:

1. Some $\mathcal{C}'_i$ has the form $x = t$ such that x occurs in t. Then evidently, $x <^*_{\mathcal{C}'} x$ and $\mathcal{C}'$ is unsatisfiable.
2. Some $\mathcal{C}'_i$ has the form $x \not\preceq \{\bar{r}\}$ such that x occurs in $\bar{r}$. Then evidently, $(x, x) \in \leq^*_{\mathcal{C}'} \cap \not\preceq_{\mathcal{C}'}$ and $\mathcal{C}'$ is unsatisfiable.
3. Otherwise, all equality constraints among $\mathcal{C}'_i$ have the form $x = t$, where the variable x is isolated in $\mathcal{C}'$ and all non-equality constraints have the form $y \not\preceq \{\bar{s}\}$ such that y does not occur in $\bar{s}$.

   Let all equality constraints in $\mathcal{C}'$ be

$$\begin{array}{l} x_1 = t_1 \\ \dots \\ x_n = t_n \end{array}$$

Since all x_i are isolated in $\mathcal{C}'$, the relation $<^*_{\mathcal{C}'}$ is empty and $\leq^*_{\mathcal{C}'} \cap \not\leq_{\mathcal{C}'} = \emptyset$. We note that the substitution $\{t_1/x_1, \ldots, t_n/x_n\}$ satisfies $\mathcal{C}'$.

In all three cases we have that $\mathcal{C}'$ is satisfiable if and only if $<^*_{\mathcal{C}'}$ is irreflexive and $\leq^*_{\mathcal{C}'} \cap \not\leq_{\mathcal{C}'} = \emptyset$. Applying (V), we get that $\mathcal{C}$ is satisfiable if and only if $<^*_{\mathcal{C}}$ is irreflexive and $\leq^*_{\mathcal{C}} \cap \not\leq_{\mathcal{C}} = \emptyset$. ∎

The proof of this lemma suggests that we can check satisfiability of saturated prenex constraints using (Replacement), but it can cause the exponential growth of the size of constraints.

THEOREM 4.8
There is a polynomial time algorithm checking constraint satisfiability. There is a polynomial time algorithm for the skeleton instantiation problem.

PROOF. By Lemma 4.5, there is a polynomial time algorithm translating any constraint into an equivalent saturated prenex constraint. By Lemma 4.7, satisfiability of a saturated prenex constraint can be checked using the conditions of that lemma. It remains to note that the conditions of Lemma 4.7 can be checked using a polynomial time algorithm. This is evident, since the relation $<^*_{\mathcal{C}}$ can be constructed using any polynomial time algorithm of the transitive closure of $<_{\mathcal{C}}$.

Using Theorem 3.7, we can reduce in polynomial time the skeleton instantiation problem to the constraint satisfiability problem. Thus, the skeleton instantiation problem can be solved in polynomial time. ∎

Consider the resulting constraint of the $\mathbf{LJ}_c$-derivation of Example 3.4: $\exists z(z \not\leq \{\} \wedge \exists v(v \not\leq \{z\} \wedge \exists u \exists w(v = u \wedge w = z)))$. Its prenex form is $\exists z \exists v \exists u \exists w(z \not\leq \{\} \wedge v \not\leq \{z\} \wedge v = u \wedge w = z)$. The variables u and w are isolated in this constraint. Thus, the constraint is saturated. (Replacement) is not applicable to this constraint and it is obviously satisfiable.

Consider now the resulting constraint of the $\mathbf{LJ}_c$-derivation of Example 3.5: $\exists z \exists v \exists u(u \not\leq \{v, z\} \wedge \exists w(w \not\leq \{v, u, z\} \wedge v = u \wedge w = z))$. Its prenex form is $\exists z \exists v \exists u \exists w(u \not\leq \{v, z\} \wedge w \not\leq \{v, u, z\} \wedge v = u \wedge w = z)$. We can apply (Isolate) to this constraint isolating the variable w:

$$\begin{array}{l} \exists z \exists v \exists u \exists w(u \not\leq \{v, z\} \wedge w \not\leq \{v, u, z\} \wedge v = u \wedge w = z) \Rightarrow \\ \exists z \exists v \exists u \exists w(u \not\leq \{v, z\} \wedge z \not\leq \{v, u, z\} \wedge v = u \wedge w = z) \end{array}$$

The resulting constraint is unsatisfiable since it contains $z \not\leq \{v, u, z\}$.

THEOREM 4.9
The weak skeleton instantiation problem is NP-complete.

PROOF. Given a closed sequent S and a weak skeleton $\mathcal{S}$, one can guess the corresponding skeleton of a derivation in $\mathbf{LJ}$ using a non-deterministic polynomial time algorithm. Applying Theorem 4.8 we obtain that the weak skeleton instantiation problem is in NP.

To prove NP-hardness, we shall use a polynomial-time reduction of the subsumption problem which is known to be NP-complete [7]. The subsumption problem can be formulated as follows: given terms $t_1, \ldots, t_n$ and ground terms $s_1, \ldots, s_m$, is there a substitution σ such that $\{t_1\sigma, \ldots, t_n\sigma\} \subseteq \{s_1, \ldots, s_m\}$. (If such a substitution σ exists we say that $\{t_1, \ldots, t_n\}$ *subsumes* $\{s_1, \ldots, s_m\}$.)

Let $t_1, \ldots, t_n$ be terms and $s_1, \ldots, s_m$ be ground terms. Let $v_1, \ldots, v_k$ be all variables of $t_1, \ldots, t_n$ and P be a unary predicate symbol. Consider the closed sequent $P(s_1), \ldots, P(s_m) \to \exists v_1 \ldots \exists v_k(P(t_1) \wedge \ldots \wedge P(t_n))$ (we could as well consider the formula $P(s_1) \wedge \ldots \wedge P(s_m) \supset \exists v_1 \ldots \exists v_k(P(t_1) \wedge \ldots \wedge P(t_n)))$. Consider the weak skeleton

$$\left.\begin{array}{c} \dfrac{\dfrac{}{\qquad}(Ax) \quad \dfrac{}{\qquad}(Ax)}{\qquad\qquad\qquad}(\to\wedge) \\ \vdots \\ \dfrac{\dfrac{}{\qquad}(Ax) \qquad \dfrac{}{\qquad}(\to\wedge)}{\qquad\qquad\qquad\qquad}(\to\wedge) \end{array}\right\} n-1 \text{ applications of } (\to\wedge)$$

$$\left.\begin{array}{c} \dfrac{}{\qquad}(\to\exists) \\ \vdots \\ \dfrac{}{\qquad}(\to\exists) \end{array}\right\} k \text{ applications of } (\to\exists)$$

Denote $P(s_1), \ldots, P(s_m)$ by Γ. Any derivation with this weak skeleton has the form (here we assume that $\wedge$ is right-associative):

$$\dfrac{\dfrac{\vdots}{\Gamma \to \exists v_2 \ldots \exists v_k(P(t_1) \wedge \ldots \wedge P(t_n))\sigma}}{\Gamma \to \exists v_1 \ldots \exists v_k(P(t_1) \wedge \ldots \wedge P(t_n))}(\to\exists)$$

where the part above the dots is

$$\dfrac{\dfrac{\dfrac{}{\Gamma \to P(t_1)\sigma}(Ax) \qquad \dfrac{\vdots \quad \dfrac{\dfrac{}{\Gamma \to P(t_{n-1})\sigma}(Ax) \quad \dfrac{}{\Gamma \to P(t_n)\sigma}(Ax)}{\Gamma \to P(t_{n-1})\sigma \wedge P(t_n)\sigma}(\to\wedge)}{\Gamma \to P(t_2)\sigma \wedge \ldots \wedge P(t_n)\sigma}}{\Gamma \to P(t_1)\sigma \wedge \ldots \wedge P(t_n)\sigma}(\to\wedge)}{\Gamma \to \exists v_k(P(t_1) \wedge \ldots \wedge P(t_n))\sigma}(\to\exists)$$

where σ is a substitution. Note that the top sequents of such derivations are axioms of **LJ** iff $\{t_1\sigma, \ldots, t_n\sigma\} \subseteq \{s_1, \ldots, s_m\}$. Thus, a derivation with this weak skeleton exists iff $\{t_1, \ldots, t_n\}$ subsumes $\{s_1, \ldots, s_m\}$. ∎

5 The prenex fragment of intuitionistic logic is PSPACE-complete

As an application of our technique, we prove that the prenex fragment of intuitionistic logic is PSPACE-complete.

The calculus $\mathbf{LJ}_c$ is not the most efficient computationally. In order to make it a useful procedure for theorem proving in intuitionistic logic, we have to modify $\mathbf{LJ}_c$. For example, if we know that above some application R of an inference rule there are no applications of quantifier rules, we can modify the rule $(\supset\to)$ so that the derivation above R in $\mathbf{LJ}_c$ will be loop-free. This technique yields a decision procedure for intuitionistic propositional logic (see Vorob'ev [26, 27], Lincoln, Scedrov, and Shankar [13], Dyckhoff [5], Hudelmaier [8]). We show how this technique together with our

$$\frac{}{\Gamma, A, \Delta \to A}\ (Ax) \qquad \frac{}{\Gamma, \Lambda, \Delta \to \varphi}\ (\Lambda)$$

$$\frac{\Gamma \to \varphi \quad \Gamma \to \psi}{\Gamma \to \varphi \wedge \psi}\ (GI1\wedge) \qquad \frac{\Gamma, \varphi \supset \chi, \Delta \to \varphi \quad \Gamma, \psi \supset \chi, \Delta \to \psi}{\Gamma, \varphi \wedge \psi \supset \chi, \Delta \to \varphi \wedge \psi}\ (GI2\wedge)$$

$$\frac{\Gamma \to \varphi}{\Gamma \to \varphi \vee \psi}\ (GI1\vee) \qquad \frac{\Gamma \to \psi}{\Gamma \to \varphi \vee \psi}\ (GI1\vee)$$

$$\frac{\Gamma, \varphi \supset P, \psi \supset P, P \supset \chi, \Delta \to \varphi}{\Gamma, (\varphi \vee \psi) \supset \chi, \Delta \to \varphi \vee \psi}\ (GI2\vee) \qquad \frac{\Gamma, \varphi \supset P, \psi \supset P, P \supset \chi, \Delta \to \psi}{\Gamma, (\varphi \vee \psi) \supset \chi, \Delta \to \varphi \vee \psi}\ (GI2\vee)$$

$$\frac{\varphi, \Gamma \to \psi}{\Gamma \to \varphi \supset \psi}\ (GI1 \supset) \qquad \frac{\Gamma, \varphi, \psi \supset \chi, \Delta \to \psi}{\Gamma, (\varphi \supset \psi) \supset \chi, \Delta \to \varphi \supset \psi}\ (GI2 \supset)$$

$$\frac{\Gamma, \varphi, \psi, \Delta \to \chi}{\Gamma, \varphi \wedge \psi, \Delta \to \chi}\ (GE\wedge) \qquad \frac{\Gamma, \varphi, \Delta \to \chi \quad \Gamma, \psi, \Delta \to \chi}{\Gamma, \varphi \vee \psi, \Delta \to \chi}\ (GE\vee)$$

$$\frac{\Gamma, A, \Delta, \psi, \Xi \to \chi}{\Gamma, A, \Delta, A \supset \psi, \Xi \to \chi}\ (GE \supset P) \qquad \frac{\Gamma, \psi, \Delta, A, \Xi \to \chi}{\Gamma, A \supset \psi, \Delta, A, \Xi \to \chi}\ (GE \supset P)$$

$$\frac{\Gamma, \varphi \supset (\psi \supset \chi), \Delta \to \rho}{\Gamma, (\varphi \wedge \psi) \supset \chi, \Delta \to \rho}\ (GE \supset \wedge)$$

$$\frac{\Gamma, \varphi \supset P, \psi \supset P, P \supset \chi, \Delta \to \rho}{\Gamma, (\varphi \vee \psi) \supset \chi, \Delta \to \rho}\ (GE \supset \vee) \qquad \frac{\Gamma, \varphi, \psi \supset \chi, \Delta \to \psi \quad \Gamma, \chi, \Delta \to \rho}{\Gamma, (\varphi \supset \psi) \supset \chi, \Delta \to \rho}\ (GE \supset\supset)$$

The rules satisfy the following restrictions:

1. Conclusions of $(GI1)$-rules must not be of the form $\Gamma, \varphi' \supset \psi', \Delta \to \varphi'$;
2. In every rule where P occurs, P is a propositional variable not occurring in the conclusion of the rule;
3. In the rule $(GE \supset \wedge)$ we have $\rho \neq \varphi \wedge \psi$, in the rule $(GE \supset \vee)$ we have $\rho \neq \varphi \vee \psi$, in the rule $(GE \supset\supset)$ we have $\rho \neq \varphi \supset \psi$.

FIG. 3. Calculus **LG**

constraint-based technique proves that the prenex fragment of intuitionistic logic is PSPACE-complete.

First, we recall some techniques of Hudelmaier [8]. The sequent calculus **LG** is shown in Figure 3. Hudelmaier [8] proves that this calculus is sound and complete for propositional intuitionistic logic. In addition, the backward search with backtracking of all possible proofs in **LG** of a given propositional formula φ gives us a decision procedure for intuitionistic logic working in space polynomial in the size of φ. We shall use these properties of **LG** and the constraint satisfaction technique of Section 4.

We introduce a technical definition.

DEFINITION 5.1
For any quantifier-free formulas φ, ψ, the constraint $\mathcal{E}_{\varphi=\psi}$ is defined inductively as follows:

1. If φ has the form $\varphi_1 \wedge \varphi_2$ and ψ has the form $\psi_1 \wedge \psi_2$, then $\mathcal{E}_{\varphi=\psi} \rightleftharpoons \mathcal{E}_{\varphi_1=\psi_1} \wedge \mathcal{E}_{\varphi_2=\psi_2}$;

2. If φ has the form $\varphi_1 \vee \varphi_2$ and ψ has the form $\psi_1 \vee \psi_2$, then $\mathcal{E}_{\varphi=\psi} \rightleftharpoons \mathcal{E}_{\varphi_1=\psi_1} \wedge \mathcal{E}_{\varphi_2=\psi_2}$;
3. If φ has the form $\varphi_1 \supset \varphi_2$ and ψ has the form $\psi_1 \supset \psi_2$, then $\mathcal{E}_{\varphi=\psi} \rightleftharpoons \mathcal{E}_{\varphi_1=\psi_1} \wedge \mathcal{E}_{\varphi_2=\psi_2}$;
4. If φ has the form $P(s_1, \dots, s_n)$ and ψ has the form $P(t_1, \dots, t_n)$, then $\mathcal{E}_{\varphi=\psi} \rightleftharpoons s_1 = t_1 \wedge \dots \wedge s_n = t_n$;
5. In all other cases $\mathcal{E}_{\varphi=\psi} \rightleftharpoons \bot$.

LEMMA 5.2
Let φ and ψ be quantifier-free formulas and θ be a substitution. Then $\varphi\theta = \psi\theta$ iff $\theta \models \mathcal{E}_{\varphi=\psi}$.

PROOF. Straightforward. ∎

DEFINITION 5.3 (Prenex sequent, $\mathbf{LG}_c$)
A formula is called *prenex* iff it has the form $Q_1 x_1 \dots Q_n x_n \varphi$, where every Q_i is a quantifier and φ is a quantifier-free formula. A sequent $\Gamma \to \varphi$ is called *prenex* iff either all formulas in Γ, φ are quantifier-free of $\Gamma = \emptyset$ and φ is a prenex formula. The *prenex fragment of intuitionistic logic* is the set of prenex formulas provable in **LJ**.

The *calculus* $\mathbf{LG}_c$ of constrained prenex sequents is shown in Figure 4.

LEMMA 5.4
Let S be a quantifier-free sequent and θ be a substitution. Then the following conditions are equivalent:

1. There is a derivation of $S\theta$ in **LJ**;
2. There is a constraint $\mathcal{C}$ and a derivation of $S \cdot \mathcal{C}$ in $\mathbf{LG}_c$ such that $\theta \models \mathcal{C}$.

PROOF. ($\Rightarrow$) Suppose $S\theta$ is derivable in **LJ**. Since **LG** is equivalent to **LJ** for propositional sequents, these calculi are also equivalent for quantifier-free sequents. Hence, $S\theta$ is derivable in **LG**. We use induction on the length of derivations in **LG**. The case of axioms is considered in Theorem 3.6. Consider non-axiom rules of **LG**.

1. The last rule of the **LG**-derivation is $(GI1\wedge)$:

$$\frac{\Gamma\theta \to \varphi\theta \quad \Gamma\theta \to \psi\theta}{\Gamma\theta \to \varphi\theta \wedge \psi\theta}\ (GI1\wedge)$$

   By the induction hypothesis, there are constraints $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\theta \models \mathcal{C}_1$ and $\theta \models \mathcal{C}_2$ and such that the constrained sequents $\Gamma \to \varphi \cdot \mathcal{C}_1$ and $\Gamma \to \psi \cdot \mathcal{C}_2$ are derivable in $\mathbf{LG}_c$. By the restriction on $(GI1)$-rules, no formula in $\Gamma\theta$ has the form $\varphi\theta \wedge \psi\theta \supset \chi$. Hence, no formula in Γ has the form $\varphi \wedge \psi \supset \chi'$. Hence, we can apply the $\mathbf{LG}_c$-rule

$$\frac{\Gamma \to \varphi \cdot \mathcal{C}_1 \quad \Gamma \to \psi \cdot \mathcal{C}_2}{\Gamma \to \varphi \wedge \psi \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (GI1\wedge)$$

   The condition $\theta \models \mathcal{C}_1 \wedge \mathcal{C}_2$ is obvious.
   The other $(GI1)$-rules are similar.
2. The last rule of the **LG**-derivation is $(GI2\vee)$:

$$\frac{\Gamma\theta, \varphi\theta \supset P, \psi\theta \supset P, P \supset \chi\theta, \Delta\theta \to \varphi\theta}{\Gamma\theta, (\varphi\theta \vee \psi\theta) \supset \chi\theta, \Delta\theta \to \varphi'\theta \vee \psi'\theta}\ (GI2\vee)$$

$$\frac{}{\Gamma, A, \Delta \to A' \cdot \mathcal{E}_{A=A'}}\ (Ax) \qquad \frac{}{\Gamma, \Lambda, \Delta \to \varphi \cdot \top}\ (\Lambda)$$

$$\frac{\Gamma \to \varphi \cdot \mathcal{C}_1 \quad \Gamma \to \psi \cdot \mathcal{C}_2}{\Gamma \to \varphi \wedge \psi \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (GI1\wedge)$$

$$\frac{\Gamma, \varphi \supset \chi, \Delta \to \varphi \cdot \mathcal{C}_1 \quad \Gamma, \psi \supset \chi, \Delta \to \psi \cdot \mathcal{C}_2}{\Gamma, \varphi \wedge \psi \supset \chi, \Delta \to \varphi' \wedge \psi' \cdot \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \mathcal{E}_{\varphi=\varphi'} \wedge \mathcal{E}_{\psi=\psi'}}\ (GI2\wedge)$$

$$\frac{\Gamma \to \varphi \cdot \mathcal{C}}{\Gamma \to \varphi \vee \psi \cdot \mathcal{C}}\ (GI1\vee) \qquad \frac{\Gamma \to \psi \cdot \mathcal{C}}{\Gamma \to \varphi \vee \psi \cdot \mathcal{C}}\ (GI1\vee)$$

$$\frac{\Gamma, \varphi \supset P, \psi \supset P, P \supset \chi, \Delta \to \varphi \cdot \mathcal{C}}{\Gamma, (\varphi \vee \psi) \supset \chi, \Delta \to \varphi' \vee \psi' \cdot \mathcal{C} \wedge \mathcal{E}_{\varphi=\varphi'} \wedge \mathcal{E}_{\psi=\psi'}}\ (GI2\vee)$$

$$\frac{\Gamma, \varphi \supset P, \psi \supset P, P \supset \chi, \Delta \to \psi \cdot \mathcal{C}}{\Gamma, (\varphi \vee \psi) \supset \chi, \Delta \to \varphi' \vee \psi' \cdot \mathcal{C} \wedge \mathcal{E}_{\varphi=\varphi'} \wedge \mathcal{E}_{\psi=\psi'}}\ (GI2\vee)$$

$$\frac{\varphi, \Gamma \to \psi \cdot \mathcal{C}}{\Gamma \to \varphi \supset \psi \cdot \mathcal{C}}\ (GI1 \supset) \qquad \frac{\Gamma, \varphi, \psi \supset \chi, \Delta \to \psi \cdot \mathcal{C}}{\Gamma, (\varphi \supset \psi) \supset \chi, \Delta \to \varphi' \supset \psi' \cdot \mathcal{C} \wedge \mathcal{E}_{\varphi=\varphi'} \wedge \mathcal{E}_{\psi=\psi'}}\ (GI2 \supset)$$

$$\frac{\Gamma, \varphi, \psi, \Delta \to \chi \cdot \mathcal{C}}{\Gamma, \varphi \wedge \psi, \Delta \to \chi \cdot \mathcal{C}}\ (GE\wedge) \qquad \frac{\Gamma, \varphi, \Delta \to \chi \cdot \mathcal{C}_1 \quad \Gamma, \psi, \Delta \to \chi \cdot \mathcal{C}_2}{\Gamma, \varphi \vee \psi, \Delta \to \chi \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (GE\vee)$$

$$\frac{\Gamma, A, \Delta, \psi, \Xi \to \chi \cdot \mathcal{C}}{\Gamma, A, \Delta, A' \supset \psi, \Xi \to \chi \cdot \mathcal{C} \wedge \mathcal{E}_{A=A'}}\ (GE \supset P) \qquad \frac{\Gamma, \psi, \Delta, A, \Xi \to \chi \cdot \mathcal{C}}{\Gamma, A' \supset \psi, \Delta, A, \Xi \to \chi \cdot \mathcal{C} \wedge \mathcal{E}_{A=A'}}\ (GE \supset P)$$

$$\frac{\Gamma, \varphi \supset (\psi \supset \chi), \Delta \to \rho \cdot \mathcal{C}}{\Gamma, (\varphi \wedge \psi) \supset \chi, \Delta \to \rho \cdot \mathcal{C}}\ (GE \supset \wedge)$$

$$\frac{\Gamma, \varphi \supset P, \psi \supset P, P \supset \chi, \Delta \to \rho \cdot \mathcal{C}}{\Gamma, (\varphi \vee \psi) \supset \chi, \Delta \to \rho \cdot \mathcal{C}}\ (GE \supset \vee)$$

$$\frac{\Gamma, \varphi, \psi \supset \chi, \Delta \to \psi \cdot \mathcal{C}_1 \quad \Gamma, \chi, \Delta \to \rho \cdot \mathcal{C}_2}{\Gamma, (\varphi \supset \psi) \supset \chi, \Delta \to \rho \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (GE \supset\supset)$$

$$\frac{\to \varphi\{y/x\} \cdot \mathcal{C}}{\to \forall x \varphi \cdot \exists y (y \not\preceq \{v_1, \ldots, v_n\} \wedge \mathcal{C})}\ (\to \forall) \qquad \frac{\to \varphi\{y/x\} \cdot \mathcal{C}}{\to \exists x \varphi \cdot \exists y \mathcal{C}}\ (\to \exists)$$

The rules satisfy both the restrictions of **LG** and the restrictions of $\mathbf{LJ}_c$.

FIG. 4. Calculus $\mathbf{LG}_c$

where $\varphi\theta = \varphi'\theta$ and $\psi\theta = \psi'\theta$. By the induction hypothesis, there is a constraint $\mathcal{C}$ such that $\theta \models \mathcal{C}$ and such that the constrained sequent $\Gamma, \varphi \supset P, \psi \supset P, P \supset \chi, \Delta \to \varphi \cdot \mathcal{C}$ is derivable in $\mathbf{LG}_c$. Consider the $\mathbf{LG}_c$-rule

$$\frac{\Gamma, \varphi \supset P, \psi \supset P, P \supset \chi, \Delta \to \varphi \cdot \mathcal{C}}{\Gamma, (\varphi \vee \psi) \supset \chi \to \varphi' \vee \psi', \Delta \cdot \mathcal{C} \wedge \mathcal{E}_{\varphi=\varphi'} \wedge \mathcal{E}_{\psi=\psi'}}\ (GI2\vee)$$

It is obvious that P is a propositional variable not occurring in the conclusion of the rule. Since $\theta \models \mathcal{C}$, $\varphi\theta = \varphi'\theta$ and $\psi\theta = \psi'\theta$, we have $\theta \models \mathcal{C} \wedge \mathcal{E}_{\varphi=\varphi'} \wedge \mathcal{E}_{\psi=\psi'}$. The other $(GI2)$-rules are similar.

3. The last rule of the **LG**-derivation is $(GE \supset P)$:

$$\frac{\Gamma\theta, A\theta, \Delta\theta, \psi\theta, \Xi\theta \to \chi\theta}{\Gamma\theta, A\theta, \Delta\theta, A'\theta \supset \psi\theta, \Xi\theta \to \chi\theta}\ (GE \supset P)$$

where $A\theta = A'\theta$. By the induction hypothesis, there is a constraint $\mathcal{C}$ such that $\theta \models \mathcal{C}$ and such that the constrained sequent $\Gamma, A, \Delta, \psi, \Xi \to \chi \cdot \mathcal{C}$ is derivable in $\mathbf{LG}_c$. Consider the $\mathbf{LG}_c$-rule

$$\frac{\Gamma, A, \Delta, \psi, \Xi \to \chi \cdot \mathcal{C}}{\Gamma, A, \Delta, A' \supset \psi, \Xi \to \chi \cdot \mathcal{C} \wedge \mathcal{E}_{A=A'}}\ (GE \supset P)$$

Since $\theta \models \mathcal{C}$ and $A\theta = A'\theta$, we have $\theta \models \mathcal{C} \wedge \mathcal{E}_{A=A'}$.

4. The last rule of the $\mathbf{LG}$-derivation is $(GE \supset\supset)$:

$$\frac{\Gamma\theta, \varphi\theta, \psi\theta \supset \chi\theta, \Delta\theta \to \psi\theta \quad \Gamma\theta, \chi\theta, \Delta\theta \to \rho\theta}{\Gamma\theta, (\varphi\theta \supset \psi\theta) \supset \chi\theta, \Delta\theta \to \rho\theta}\ (GE \supset\supset)$$

By the induction hypothesis, there are constraints $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\theta \models \mathcal{C}_1$ and $\theta \models \mathcal{C}_2$ and such that the constrained sequents $\Gamma, \varphi, \psi \supset \chi, \Delta \to \psi \cdot \mathcal{C}_1$ and $\Gamma, \chi, \Delta \to \rho \cdot \mathcal{C}_2$ are derivable in $\mathbf{LG}_c$. By the restriction on $(GE \supset\supset)$-rules, the formula $\rho\theta$ does not have the form $\varphi\theta \supset \psi\theta$. Hence, the formula ρ does not have the form $\varphi \supset \psi$. Hence, we can apply the $\mathbf{LG}_c$-rule

$$\frac{\Gamma, \varphi, \psi \supset \chi, \Delta \to \psi \cdot \mathcal{C}_1 \quad \Gamma, \chi, \Delta \to \rho \cdot \mathcal{C}_2}{\Gamma, (\varphi \supset \psi) \supset \chi, \Delta \to \rho \cdot \mathcal{C}_1 \wedge \mathcal{C}_2}\ (GE \supset\supset)$$

The condition $\theta \models \mathcal{C}_1 \wedge \mathcal{C}_2$ is obvious.

(⇐) Similar to the proof of Theorem 3.6. Consider some specific rules of $\mathbf{LG}_c$:

1. The last rule of the $\mathbf{LG}_c$-derivation is $(GI2\wedge)$:

$$\frac{\Gamma, \varphi \supset \chi, \Delta \to \varphi \cdot \mathcal{C}_1 \quad \Gamma, \psi \supset \chi, \Delta \to \psi \cdot \mathcal{C}_2}{\Gamma, \varphi \wedge \psi \supset \chi, \Delta \to \varphi' \wedge \psi' \cdot \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \mathcal{E}_{\varphi=\varphi'} \wedge \mathcal{E}_{\psi=\psi'}}\ (GI2\wedge)$$

Suppose $\theta \models \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \mathcal{E}_{\varphi=\varphi'} \wedge \mathcal{E}_{\psi=\psi'}$. Then $\theta \models \mathcal{C}_1$, $\theta \models \mathcal{C}_2$, $\varphi\theta = \varphi'\theta$ and $\psi\theta = \psi'\theta$. By the induction hypothesis the sequents $\Gamma\theta, \varphi\theta \supset \chi\theta, \Delta\theta \to \varphi\theta$ and $\Gamma\theta, \psi\theta \supset \chi\theta, \Delta\theta \to \psi\theta$ are derivable in $\mathbf{LJ}$. Then the sequent $\Gamma\theta, \varphi\theta \wedge \psi\theta \supset \chi\theta, \Delta\theta \to \varphi\theta \wedge \psi\theta$ is derivable in $\mathbf{LJ}$. Using $\varphi\theta = \varphi'\theta$ and $\psi\theta = \psi'\theta$ we get that $\Gamma\theta, \varphi\theta \wedge \psi\theta \supset \chi\theta, \Delta\theta \to \varphi'\theta \wedge \psi'\theta$ is derivable in $\mathbf{LJ}$.

2. The last rule of the $\mathbf{LG}_c$-derivation is $(GE \supset \vee)$:

$$\frac{\Gamma, \varphi \supset P, \psi \supset P, P \supset \chi, \Delta \to \rho \cdot \mathcal{C}}{\Gamma, (\varphi \vee \psi) \supset \chi, \Delta \to \rho \cdot \mathcal{C}}\ (GE \supset \vee)$$

Suppose $\theta \models \mathcal{C}$. By the induction hypothesis, the sequent $\Gamma\theta, \varphi\theta \supset P, \psi\theta \supset P, P \supset \chi\theta, \Delta\theta \to \rho\theta$ is derivable in $\mathbf{LJ}$. Since P does not occur in $\Gamma, \Delta, \varphi, \psi, \chi, \rho$ any sequent obtained from this sequent by replacement of P by any formula, is derivable in $\mathbf{LJ}$. Replacing P by $\varphi\theta \vee \psi\theta$, we obtain that the sequent $\Gamma\theta, \varphi\theta \supset \varphi\theta \vee \psi\theta, \psi\theta \supset \varphi\theta \vee \psi\theta, \varphi\theta \vee \psi\theta \supset \chi\theta, \Delta\theta \to \rho\theta$ is derivable in $\mathbf{LJ}$. Hence, $\Gamma\theta, \varphi\theta \vee \psi\theta \supset \chi\theta, \Delta\theta \to \rho\theta$ is derivable in $\mathbf{LJ}$.

The other cases are similar.

∎

In fact, the condition on skeletons formulated in Theorem 3.6 for $\mathbf{LJ}$ and $\mathbf{LJ}_c$ is not true for $\mathbf{LG}$ and $\mathbf{LG}_c$: the calculus $\mathbf{LG}_c$ gives "more" skeletons than $\mathbf{LG}$. We can make $\mathbf{LG}_c$ equivalent to $\mathbf{LG}$ w.r.t. skeletons by introducing non-equality constraints

$s \neq t$ in $(GI1)$- and $(GE \supset)$- rules of **LG**, but the current formulation of $\mathbf{LG}_c$ is more simple and enough for our purposes.

Lemma 5.4 generalizes to prenex formulas:

LEMMA 5.5
Let φ be a prenex formula and θ be a substitution. Then the following conditions are equivalent:

1. There is a derivation of $\rightarrow \varphi\theta$ in **LJ**;
2. There is a constraint $\mathcal{C}$ and a derivation of $\rightarrow \varphi \cdot \mathcal{C}$ in $\mathbf{LG}_c$ such that $\theta \models \mathcal{C}$.

PROOF. By induction on the number q of quantifiers in φ.

($q = 0$) In this case φ is quantifier-free and the statement follows from Lemma 5.4.

($q > 0$) In this case the proof is similar to that of Theorem 3.6. We shall consider only one case, when φ has the form $\forall x\psi$ and only the direction $(\Leftarrow)$. In this case the last rule applied in the $\mathbf{LG}_c$-derivation is $(\rightarrow \forall)$. Then the $\mathbf{LG}_c$-derivation has the form

$$\dfrac{\begin{array}{c}\Pi\\ \rightarrow \psi\{y/x\} \cdot \mathcal{C}\end{array}}{\rightarrow \forall x\psi \cdot \exists y(y \not\preceq \{v_1, \ldots, v_n\} \wedge \mathcal{C})}\ (\rightarrow \forall)$$

Let $\theta \models \exists y(y \not\preceq \{v_1, \ldots, v_n\} \wedge \mathcal{C})$. By the definition of constraint satisfaction, there is a substitution θ' such that $v\theta' \approx v\theta$ for all variables different from y and $\theta' \models y \not\preceq \{v_1, \ldots, v_m\} \wedge \mathcal{C}$. It means that $\theta' \models \mathcal{C}$ and $y\theta'$ is a variable not occurring in $v_1\theta', \ldots, v_m\theta'$. By the induction hypothesis, there is derivation Π' in **LJ** of the sequent $\rightarrow \psi\{y/x\}\theta'$. Then the required **LJ**-derivation is

$$\dfrac{\begin{array}{c}\Pi'\\ \rightarrow \psi\{y/x\}\theta'\end{array}}{\rightarrow (\forall x\psi)\theta}\ (\rightarrow \forall)$$

∎

Lemma 5.5 immediately yields

THEOREM 5.6
Let φ be a closed prenex formula. Then the sequent $\rightarrow \varphi$ is derivable in **LJ** iff there is a satisfiable constraint $\mathcal{C}$ such that $\rightarrow \varphi \cdot \mathcal{C}$ is derivable in $\mathbf{LG}_c$.

DEFINITION 5.7 (*level*(S))
For any formula φ, denote by *level*(φ) the number $3d+2c+i+q$, where d, c, i and q are the numbers of occurrences in φ of $\vee, \wedge, \supset$ and quantifiers, respectively. For a sequence set of formulas $\Gamma = (\varphi_1, \ldots, \varphi_n)$, denote by *level*($\Gamma$) the number $\sum_{i=1}^{n} \mathit{level}(\varphi_i)$. For a sequent $\Gamma \rightarrow \varphi$, its *level,* denoted by *level*($\Gamma \rightarrow \varphi$), is defined by

$$\mathit{level}(\Gamma \rightarrow \varphi) \rightleftharpoons \begin{cases} \mathit{level}(\Gamma) & \text{if some formula in } \Gamma \text{ has the form } \varphi \supset \psi \\ \mathit{level}(\Gamma) + \mathit{level}(\varphi) & \text{otherwise} \end{cases}$$

The *level* of a constrained sequent $S \cdot \mathcal{C}$, denoted by *level*($S \cdot \mathcal{C}$), is defined as *level*(S).

LEMMA 5.8
In any application of an inference rule of $\mathbf{LG}_c$, the level of any of its premises is strictly less then the level of its conclusion.

PROOF. By straightforward analysis of inference rules of $\mathbf{LG}_c$. ∎

THEOREM 5.9
The prenex fragment of intuitionistic logic is PSPACE-complete.

PROOF. Since intuitionistic propositional logic is already PSPACE-hard (Statman [24]), we conclude that the prenex fragment of intuitionistic logic is PSPACE-hard. Thus, we have to show that this fragment is in PSPACE.

Let $\psi = (Q_1x_1 \ldots Q_nx_n\varphi)$ be any prenex formula, where φ is quantifier-free. Without loss of generality we assume that the variables $x_1, \ldots, x_n$ are pairwise different. Consider possible $\mathbf{LG}_c$-derivations of constrained sequents of the form $\rightarrow \psi \cdot \mathcal{C}$. Any such derivation has the form

$$\cfrac{\cfrac{\begin{array}{c}\Pi\\ \rightarrow \varphi \cdot \mathcal{C}'\end{array}}{\rightarrow Q_nx_n\varphi \cdot \mathcal{C}_n}}{\begin{array}{c}\vdots\\ \cfrac{\rightarrow Q_2x_2 \ldots Q_nx_n\varphi \cdot \mathcal{C}_2}{\rightarrow Q_1x_1 \ldots Q_nx_n\varphi \cdot \mathcal{C}}\end{array}}$$

up to renaming variables. Since φ is quantifier-free, the constraint $\mathcal{C}'$ is a conjunction $\mathcal{C}'_1 \wedge \ldots \wedge \mathcal{C}'_k$ such that every $\mathcal{C}'_i$ is either $\top$ or $\bot$ or has the form $s = t$ such that the terms s and t are arguments of an atomic formula occurring in φ. The number of such pairs of terms s, t is polynomial in the size of φ. Hence, after applying to $\mathcal{C}'$ equivalence-preserving transformations $s = t \wedge \mathcal{C}'' \wedge s = t \Rightarrow s = t \wedge \mathcal{C}''$, $\top \wedge \mathcal{C}'' \Rightarrow \mathcal{C}''$ and $\bot \wedge \mathcal{C}'' \Rightarrow \bot$ we obtain a constraint equivalent to $\mathcal{C}'$ whose size is polynomial in the size of φ.

Using Lemma 5.8, we note that all such derivations of $\rightarrow \psi \cdot \mathcal{C}$ can be obtained by a non-deterministic algorithm working in space polynomial in the size of ψ. Using the above consideration, we can note that $\mathcal{C}$ can be kept in the size polynomial in the size of ψ. Applying Theorem 5.6 on the equivalence of $\mathbf{LJ}$ and $\mathbf{LG}_c$ and Theorem 4.8 on the existence of a polynomial time procedure for constraint satisfiability, we obtain that the derivability problem for $\rightarrow \psi$ is in PSPACE. ∎

This result is interesting because the prenex fragment of classical logic (even without function symbols) is undecidable. In fact, in classical logic every formula is effectively equivalent to a prenex formula.

6 Conclusion and related work

The idea to use constraints instead of substitutions is used in many areas of automated deduction. It is presented in the unification algorithm of Martelli and Montanari [15], it is used in the semantics of logic programs, for example in Gabbrielli, Dore and Levi [6], and Maher [14]. It is implemented in the theorem proving system based on the sequent calculus for classical logic (Anufriev and Aselderov [1], Degtyarev, Zhezherun and Lyaletski [4], see also the survey by Degtyarev and Voronkov [2]). The use of constraints in automated deduction is investigated in many papers (see the survey by Kirchner [12]). In this article, we consider the use of constraints in intuitionistic logic. In particular, we obtain some complexity results for the problem of the instantiation of derivation skeletons and for the prenex fragment of intuitionistic logic.

It is generally well-known how to design sequent-based proof-procedures for intuitionistic logic. For example, a matrix characterization has been developed by Wallen [31], a tableau-based proof procedure has been described by Shankar [23] and systems based on the inverse method have been given in Voronkov [28] and Tammet [25]. Related logical papers are Vorob'ev [26, 27], Kanger [9], Maslov [16], Dyckhoff [5], Hudelmaier [8] and Miglioli, Moscato and Ornaghi [17]. The novelty of our article is an abstract characterization of provability and related complexity results.

This characterization and related complexity results can be used in several ways not pursued in the above mentioned papers. Let us first consider the complexity results. Both in Shankar [23] and Voronkov [28] the substitutions satisfying constraints are constructed in an "eager" way. Shankar [23] uses "local skolemization" to instantiate essentially universally quantified variables. Voronkov [28] uses ordinary variables which is possible due to the *locality* of the inverse method. In both cases, it is known that the resulting substitution may have size exponential in the size of the set of constraints. Thus, even for "almost propositional" fragments of intuitionistic logic, like the prenex fragment, both methods yield a procedure using exponential space. (In fact, the method of Shankar [23] does not even give a decision procedure, unless it is suitable modified.) Our technique allows us to find a polynomial space decision procedure for this fragment.

In the matrix characterization of Wallen [31], provability is characterized in terms of a multiplicity (the number of copies of subformulas to be used), existence of a substitution and certain ordering restrictions. The ordering restrictions are supposed to encode the order of the application of inference rules in **LJ**. For a given multiplicity, it is not clear what is the complexity of the problem of the existence of a substitution satisfying the ordering restrictions. For example, Otten and Kreitz [19] use an algorithm based on a string unification algorithm. The complexity of full string unification is not known and currently known upper bounds yield several exponents. (It is not clear whether Otten and Kreitz [19]) use full string unification, or only some special case.) In addition, the Wallen's characterization does not give a decision algorithm even for propositional intuitionistic logic.

Another interesting point in the characterization of provability using constraints is that such a characterization can be generalized to include built-in relations, for example arithmetical relations, in intuitionistic theorem provers. In [30] the technique of this article is developed for intuitionistic logic with equality. In [3] we prove some complexity results related to intuitionistic logic with equality. In the case of arbitrary built-in predicates, it is hard to expect either the tableau method or the inverse method to eagerly find a substitution closing a derivation. The constraint-handling mechanism for built-in relations can be integrated into our constraint satisfaction algorithms.

In our article, the use of constraints instead of explicit substitutions gives a polynomial time procedure instead of an exponential time procedure. When it comes to built-in relations, the use of constraints can result in more essential speedup as noted in Nieuwenhuis and Rubio [18]. Their paper has shown how to use an NP-complete constraint satisfaction problem instead of constructing a double exponential number of minimal AC-unifiers.

Our technique can be used for other non-classical logics, for example some modal logics. Our characterization of provability (Theorem 3.6) and related complexity results (Theorems 4.8 and 4.9) are also true for classical logic.

Acknowledgments

I thank Anatoli Degtyarev who made many valuable remarks and who pointed out an error in the first proof of Theorem 3.6. I also thank Evgeny Dantsin for his comments about complexity.

References

[1] F.V. Anufriev and Z.M. Aselderov. The algorithm of evidence (in Russian). *Kibernetika*, (5):29–60, 1972.

[2] A. Degtyarev and A. Voronkov. Automated theorem proving I. *Cybernetics*, 22(3):290–297, 1986.

[3] A. Degtyarev and A. Voronkov. Decidability problems for the prenex fragment of intuitionistic logic. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 503–512, New Brunswick, NJ, July 1996. IEEE Computer Society Press.

[4] A.I. Degtyarev, A.P. Zhezherun, and A.V. Lyaletski. Deductive tools of a mathematical text processing system (in Russian). *Kibernetika*, (5):105–108, 1978.

[5] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57(3):795–807, 1992.

[6] M. Gabbrielli, G.M. Dore, and G. Levi. Observable semantics for constraint logic programs. *Journal of Logic and Computation*, 5(2):133–171, 1995.

[7] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1979.

[8] J. Hudelmaier. An $O(n \log n)$-space decision procedure for intuitionistic propositional logic. *Journal of Logic and Computation*, 3(1):63–75, 1993.

[9] S. Kanger. *Provability in Logic*, volume 1 of *Studies in Philosophy*. Almqvist and Wicksell, Stockholm, 1957.

[10] S. Kanger. A simplified proof method for elementary logic. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, pages 87–94. North Holland, 1963. Reprinted as [11].

[11] S. Kanger. A simplified proof method for elementary logic. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 364–371. Springer Verlag, 1983. Reprinted from [10].

[12] H. Kirchner. On the use of constraints in automated deduction. In A. Podelski, editor, *Constraint Programming: Basics and Tools*, volume 910 of *Lecture Notes in Computer Science*, pages 128–146. Springer Verlag, 1995.

[13] P. Lincoln, A. Scedrov, and N. Shankar. Linearizing intuitionistic implication. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 51–62, 1991.

[14] M.J. Maher. A CLP view of logic programming. In *Proc. Conf. on Algebraic and Logic Programming*, volume 632 of *Lecture Notes in Computer Science*, pages 364–383, October 1992.

[15] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.

[16] S.Yu. Maslov. Invertible sequential variant of constructive predicate calculus (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 4:96–111, 1967. English Translation in: Seminars in Mathematics: Steklov Math. Inst. 4, Consultants Bureau, NY-London, 1969, p.36–42.

[17] P. Miglioli, U. Moscato, and M. Ornaghi. An improved refutation system for intuitionistic predicate logic. *Journal of Automated Reasoning*, 13:361–373, 1994.

[18] R. Nieuwenhuis and A. Rubio. AC-superposition with constraints: no AC-unifiers needed. In A. Bundy, editor, *Automated Deduction—CADE-12. 12th International Conference on Automated Deduction*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 545–559, Nancy, France, June/July 1994.

[19] J. Otten and C. Kreitz. A connection based proof method for intuitionistic logic. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 918 in Lecture Notes in Artificial Intelligence, pages 122–137, Schloß Rheinfels, St. Goar, Germany, May 1995.

[20] D. Prawitz. An improved proof procedure. *Theoria*, 26(2):102–128, 1960. Reprinted as [22].

[21] D. Prawitz. A proof procedure with matrix reduction. In M. Laudet, L. Lacombe, L. Nolin, and M. Schutzenberger, editors, *Symp. on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 207–213. Springer Verlag, 1970.

[22] D. Prawitz. An improved proof procedure. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 162–199. Springer Verlag, 1983. Reprinted from [20].

[23] N. Shankar. Proof search in the intuitionistic sequent calculus. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 522–536, Saratoga Springs, NY, USA, June 1992. Springer Verlag.

[24] R. Statman. Intuitionistic logic is polynomial-space complete. *Theoretical Computer Science*, 9(1):67–72, 1979.

[25] T. Tammet. A resolution theorem prover for intuitionistic logic. In M.A. McRobbie and J.K. Slaney, editors, *Automated Deduction—CADE-13*, volume 1104 of *Lecture Notes in Computer Science*, pages 2–16, New Brunswick, NJ, USA, 1996.

[26] N.N. Vorob'ev. The derivability problem in the constructive propositional calculus with strong negation. *Dokl. Akad. Nauk SSSR (in Russian)*, 58:689–692, 1952.

[27] N.N. Vorob'ev. A new derivability algorithm in the constructive propositional calculus. *Trudy Math. Inst. Steklov*, 52, 1958.

[28] A. Voronkov. Theorem proving in non-standard logics based on the inverse method. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 648–662, Saratoga Springs, NY, USA, June 1992. Springer Verlag.

[29] A. Voronkov. Proof search in intuitionistic logic based on constraint satisfaction. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods. 5th International Workshop, TABLEAUX '96*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 312–329, Terrasini, Palermo Italy, May 1996.

[30] A. Voronkov. Proof-search in intuitionistic logic with equality, or back to simultaneous rigid E-unification. *Journal of Automated Reasoning*, 21:205–231, 1998.

[31] L.A. Wallen. *Automated Proof-Search in Nonclassical Logics.* The MIT Press, 1990.

Received 10 March, 1996.