Theory and Practice of Logic Programming **1** (4): 409–445, July 2001.

Constraint Logic Programming with Hereditary Harrop formulas

JAVIER LEACH, SUSANA NIEVA and MARIO RODRÍGUEZ-ARTALEJO*

Departamento Sistemas Informáticos y Programación,
Av. Complutense s/n, Universidad Complutense de Madrid,
E-28040 Madrid, Spain
e-mail: {leach,nieva,mario}@sip.ucm.es

Abstract

Constraint Logic Programming (CLP) and Hereditary Harrop formulas (HH) are two well known ways to enhance the expressivity of Horn clauses. In this paper, we present a novel combination of these two approaches. We show how to enrich the syntax and proof theory of HH with the help of a given constraint system, in such a way that the key property of HH as a logic programming language (namely, the existence of uniform proofs) is preserved. We also present a procedure for goal solving, showing its soundness and completeness for computing answer constraints. As a consequence of this result, we obtain a new strong completeness theorem for CLP that avoids the need to build disjunctions of computed answers, as well as a more abstract formulation of a known completeness theorem for HH.

KEYWORDS: constraint systems, hereditary Harrop formulas, uniform proofs, goal solving

1 Introduction

Traditionally, the logic of Horn clauses has been considered as the basis for logic programming (Van Emden and Kowalski, 1976). In spite of its Turing completeness (Andréka and Németi, 1978), the lack of expressivity of Horn clauses for programming purposes is widely acknowledged. During the last decade, different extensions of Horn clauses have been proposed, with the aim of increasing expressivity without sacrificing the declarative character of pure logic programming. Among such extensions, two important approaches are Constraint Logic Programming (CLP) and Hereditary Harrop Formulas (HH).

The CLP scheme (Jaffar and Lassez, 1987) goes beyond the limitations of the Herbrand universe by providing the ability to program with Horn clauses over different computation domains, whose logical behaviour is given by constraint systems. CLP languages keep all the good semantic properties of pure logic programming, including soundness and completeness results (Jaffar *et al.*, 1996). Their implementation relies on the combination of *SLD* resolution with dedicated algorithms

* This is a substantially revised and extended version of an earlier paper (Leach, Nieva and Rodríguez-Artalejo, 1997). The authors have been partially supported by the Spanish National Project TIC 98-0445-C03-02 *TREND* and the Esprit BRA Working Group EP-22457 *CCLII*.

for constraint entailment, solving and simplification. Therefore, efficient and yet declarative programs can be written to solve complex combinatorial problems. See Jaffar and Maher (1994) for a survey of the foundations, implementation issues and applications of CLP languages.

On the other hand, the HH approach (Miller, Nadathur and Scedrov, 1987) overcomes the inability of Horn clauses to provide a logical basis for several constructions commonly found in modern programming languages, such as scoping, abstraction and modularity. This is achieved by extending Horn clauses to a richer fragment of intuitionistic logic that allows us to use disjunctions, implications and quantifiers in goals. In fact, HH is a typical example of an *abstract logic programming language*, in the sense of Miller, Nadathur, Pfenning and Scedrov (1991). Abstract logic programming languages are characterized by the fact that the declarative meaning of a program, given by provability in a deduction system, can be interpreted operationally as goal-oriented search for solutions. Technically, the existence of *uniform proofs* for all provable goal formulas permits the search interpretation of provability. The implementation of programming languages based on HH, such as λ-Prolog (Miller and Nadathur, 1986; Nadathur and Miller, 1988), requires the resolution of the problem of unifying terms occurring under the scope of arbitrary quantifier prefixes. Correct unification algorithms for such problems have been studied by Miller (1992) and Nadathur (1993). Moreover, Nadathur (1993) shows in detail the soundness and completeness of a goal solving procedure for the first-order HH language.

The aim of this paper is to present a framework for the combination of the CLP and HH approaches, that incorporates the benefits of expressivity and efficiency that HH and CLP bring to logic programming, respectively. We will enrich the syntax of first-order HH with constraints coming from a given constraint system. The resulting language is such that all constructions and results are valid for any constraint system $\mathscr{C}$, therefore we can speak of a *scheme* $HH(X)$ with *instances* $HH(\mathscr{C})$, as in CLP. We will define an amalgamated proof system that combines inference rules from intuitionistic sequent calculus with constraint entailment, in such a way that the key property of an abstract logic programming language is preserved. Moreover, we will also present a sound and complete procedure for goal solving. As in CLP, the result of solving a goal using a program will be an *answer constraint*.

The following simple program Δ, goal G and constraint R belong to the instance $HH(\mathscr{R})$ given by the constraint system $\mathscr{R}$ for real numbers. We will refer to this as the *disc example* in the sequel.

$$\Delta \equiv \{\forall x \forall y (x^2 + y^2 \leqslant 1 \Rightarrow disc\ (x, y))\}$$
$$G \equiv \forall y (y^2 \leqslant 1/2 \Rightarrow disc\ (x, y))$$
$$R \equiv x^2 \leqslant 1/2$$

In the example, the formula R turns out to be a correct and computable answer constraint in the resolution of G from Δ. Due to the soundness and completeness of the goal solving procedure, G can be deduced from Δ and R in the amalgamated proof system. In Figure 1 a uniform proof is presented of the sequent $\Delta; R \vdash G$, using the inferences rules of the calculus $\mathscr{UC}$ which will be presented in Section 4.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

$$\frac{x^2 \leqslant 1/2, y^2 \leqslant 1/2 \vdash_{\mathscr{R}} \exists u \exists v (x \approx u \wedge y \approx v \wedge u^2 + v^2 \leqslant 1)}{\Delta; x^2 \leqslant 1/2, y^2 \leqslant 1/2 \vdash \exists u \exists v (x \approx u \wedge y \approx v \wedge u^2 + v^2 \leqslant 1)} (C_R)$$

$$\frac{\Delta; x^2 \leqslant 1/2, y^2 \leqslant 1/2 \vdash \exists u \exists v (x \approx u \wedge y \approx v \wedge u^2 + v^2 \leqslant 1)}{\Delta; x^2 \leqslant 1/2, y^2 \leqslant 1/2 \vdash disc\ (x, y)} (Clause)$$

$$\frac{\Delta; x^2 \leqslant 1/2, y^2 \leqslant 1/2 \vdash disc\ (x, y)}{\Delta; x^2 \leqslant 1/2 \vdash y^2 \leqslant 1/2 \Rightarrow disc\ (x, y)} (\Rightarrow C_R)$$

$$\frac{\Delta; x^2 \leqslant 1/2 \vdash y^2 \leqslant 1/2 \Rightarrow disc\ (x, y)}{\Delta; x^2 \leqslant 1/2 \vdash \forall y (y^2 \leqslant 1/2 \Rightarrow disc\ (x, y))} (\forall_R)$$

Fig. 1. Uniform proof of the sequent $\Delta; R \vdash G$.

From a technical point of view, for the particular case of the Herbrand constraint system, our completeness result boils down to a more abstract formulation of the completeness theorem in Nadathur (1993). In the case of CLP programs using only Horn clauses with constraints, our goal solving procedure reduces to constrained resolution, and our completeness theorem yields a form of strong completeness for success that avoids the need to build disjunctions of computed answers, in contrast to Maher (1997, Theorem 2) (see also Jaffar, Maher, Marriott and Stuckey, 1996, Theorem 4.12). The reason for this discrepancy is that our amalgamated proof system uses more constructive inference mechanisms to deduce goals from program clauses, as we will see.

The rest of this paper is organized as follows. Section 2 shows some programming examples that illustrate the specific benefits of the combination of CLP and HH. In Section 3 we recall the notion of a constraint system and we define the syntax of HH with constraints. In Section 4 we present an intuitionistic proof system for HH with constraints, and show the existence of uniform proofs; then an equivalent proof system allowing only uniform proofs is defined. Based on this second calculus, a sound and complete procedure for goal solving is presented as a transformation system in Section 5. In Section 6 we summarize conclusions and possible lines for future research. To improve readability of the paper, some proofs have been omitted or compressed in the main text. Full proofs appear in the Appendix.

2 Examples

Although simple, the programs of this section exemplify the programming style in $HH(X)$ languages, combining the characteristic utilities of HH – such as to add temporarily facts to the program or to limit the scope of the names – with the advantages of using constraint solvers, instead of syntactical unification. The syntax used in the examples is basically that of HH languages, with the addition of constraints in clause bodies and goals. In particular, the notation $t \approx t'$ will be used for equality constraints. More formal explanations will follow in Section 3.

The programs below are based on a constraint system which is defined as a combination of $\mathscr{R}$ (real numbers) and $\mathscr{H}$ (Herbrand universe). This constraint system underlies the well known language $CLP(\mathscr{R})$ (Jaffar, Michaylov, Stuckey and

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Yap, 1992). The elements in the intended computation domain can be represented as trees whose internal nodes are labelled by constructors, and whose leaves are labelled either by constant constructors or by real numbers. In particular this includes the representation of lists, possibly with real numbers as members. We will use Prolog's syntax for the list constructors.

Example 2.1 (*Hypothetical queries in a database system*)
The following program keeps record of the marks of different students in two exercises they have to do to pass an exam.

$$
\begin{array}{l}
exercise1(bob, 4).\\
exercise1(fran, 3).\\
exercise2(fran, 6).\\
exercise1(pep, 5).\\
exercise2(pep, 6).\\
\\
pass(X) \Leftarrow exercise1(X, N1) \wedge exercise2(X, N2) \wedge (N1 + N2)/2 > 5.
\end{array}
$$

While the goal $G \equiv pass(bob)$ fails, $G' \equiv exercise2(bob, 6.5) \Rightarrow pass(bob)$ succeeds. To resolve this last goal, the fact $exercise2(bob, 6.5)$ is added to the program, but not permanently. If we again put the query $G \equiv pass(bob)$ it would fail again.

Suppose now we want to know the requirements a student has to fulfil to pass, then we add to the program the clauses:

$$
\begin{array}{l}
need\text{-}to\text{-}pass(A, []) \Leftarrow pass(A).\\
need\text{-}to\text{-}pass(A, [ex1(X)|L]) \Leftarrow (exercise1(A, X) \Rightarrow need\text{-}to\text{-}pass(A, L)).\\
need\text{-}to\text{-}pass(A, [ex2(X)|L]) \Leftarrow (exercise2(A, X) \Rightarrow need\text{-}to\text{-}pass(A, L)).
\end{array}
$$

The goal $G \equiv need\text{-}to\text{-}pass(bob, L)$ will produce an answer equivalent in the constraint system to $\exists N(L \approx [ex2(N)] \wedge N > 6)$.

To get this answer, the intermediate goal $exercise2(A, X) \Rightarrow need\text{-}to\text{-}pass(A, L1)$ should be solved with the constraint $A \approx bob$. This would require:

1. To introduce the fact $exercise2(A, X)$ in the base. Note that the effect is different to adding a clause in Prolog with *assert*, since this implies the universal quantification of A and X.
2. Try to solve the goal $need\text{-}to\text{-}pass(A, [])$ with the first clause of this predicate, so to solving $pass(A)$, with the constraint $A \approx bob$ and $L1 \approx []$. This will add the constraints $X \approx N, (4 + N)/2 > 5$.

A similar example is shown in Hodas (1994). Here the benefit is in the use of constraints allowing us to write conditions about the real numbers that help to solve the goal more efficiently.

Example 2.2 (*Fibonacci numbers*)
Cohen (1990) uses the computation of Fibonacci numbers as a simple example to illustrate the advantages of constraint solving w.r.t. built-in arithmetic (as available in Prolog). The recursive definition of Fibonacci sequence immediately gives rise to the following $CLP(\mathscr{R})$ program:

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

$$fib(0,1).$$
$$fib(1,1).$$
$$fib(N, F1 + F2) \Leftarrow N \geqslant 2 \wedge fib(N-1, F1)) \wedge fib(N-2, F2).$$

Thanks to the abilities of the constraint solver, this program is reversible. In addition to goals such as $fib(10, X)$, with answer $X \approx 89$, we can also solve goals as $fib(N, 89)$ with answer $N \approx 10$. However, the program is based on an extremely inefficient double recursion. As a consequence, it runs in exponential time, and multiple recomputations of the same Fibonacci number occur.

In $HH(\mathscr{R})$ we can avoid this problem by using implications in goals to achieve the effect of tabulation. At the same time, the program remains reversible and close to the mathematical specification of the Fibonacci sequence.

$$fib(N, X) \Leftarrow (memfib(0,1) \Rightarrow (memfib(1,1) \Rightarrow getfib(N, X, 1))).$$
$$getfib(N, X, M) \Leftarrow 0 \leqslant N \wedge N \leqslant M \wedge memfib(N, X).$$
$$getfib(N, X, M) \Leftarrow N > M \wedge memfib(M-1, F1) \wedge memfib(M, F2) \wedge (memfib(M+1, F1+F2) \Rightarrow getfib(N, X, M+1)).$$

A predicate call of the form $getfib(N, X, M)$ assumes that the Fibonacci numbers fib_i, with $0 \leqslant i \leqslant M$, are memorized as atomic clauses for *memfib* in the local program. The call computes the Nth Fibonacci number in X; at the same time, the Fibonacci numbers fib_i, with $M < i \leqslant N$ are memorized during the computation.

Let us consider two simple goals for this program:

1. $G_1 \equiv fib(2, X)$. To solve G_1, $memfib(0,1)$ and $memfib(1,1)$ are added to the local program, and the goal $getfib(2, X, 1)$ is solved. Since $2 > 1$, the first clause for *getfib* fails. The second clause for *getfib* puts $memfib(2,2)$ into the local program and produces the new goal $getfib(2, X, 2)$, which is solved with answer $X \approx 2$ by means of the first clause.
2. $G_2 \equiv fib(N, 2)$. Analogously, G_2 is solved by solving $getfib(N, 2, 1)$ after adding $memfib(0,1)$ and $memfib(1,1)$ into the local program. The first clause for *getfib* fails. Therefore, the constraint $N > 1$ is assumed and the new goal $getfib(N, 2, 2)$ must be solved, after putting the atom $memfib(2,2)$ into the local program. Now, the first clause for *getfib* leads easily to the answer $N \approx 2$.

In general, all goals of the two forms:

1. $fib(n, X)$, n given,
2. $fib(N, f)$, f a given Fibonacci number,

can be solved by our goal solving procedure. Moreover, goals of the form (1) can be solved in $O(n)$ steps. Miller (1989) showed that implicational goals can be used to store previously computed Fibonacci numbers, thus leading to an HH program that runs in time $O(n)$. Later Hodas (1994) gave another memorized version of the computation of Fibonacci numbers, closer to the naive doubly recursive algorithm. Hodas' version combines implicational goals with a continuation-passing programming style which relies on higher-order predicate variables. The benefit of our version w.r.t. that of Miller (1989) and Hodas (1994) is the reversibility of the predicate *fib* that is enabled by constraint solving.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Example 2.3 (*Relating some simple parameters in a mortgage*)
The following program Δ is presented by Jaffar and Michaylov (1987) as an application of $CLP(\mathscr{R})$.[1]

$$\begin{array}{l} mortgage(P,T,I,M,B) \Leftarrow 0 \leqslant T \wedge T \leqslant 3 \wedge TotalInt \approx T * (P * I/1200) \wedge \\ \qquad\qquad B \approx P + TotalInt - (T * M). \\ mortgage(P,T,I,M,B) \Leftarrow T > 3 \wedge QuartInt \approx 3 * (P * I/1200) \wedge \\ \qquad\qquad mortgage(P + QuartInt - 3 * M, T - 3, I, M, B). \end{array}$$

where P stands for principal Payment, T for Time in months, I for Interest rate, M for Monthly payment, and B for outstanding Balance.

In $CLP(\mathscr{R})$ the goal $G \equiv mortgage(P,6,10,M,0)$, produces the answer $0 \approx 1.050625 * P - 6.075 * M$. From this answer we can deduce that $P/(T * M) \approx P/(6 * M) \approx 0.9637$ (the number 0.9637 is calculated as an approximation), where $P/(T * M)$ represents the quotient of loss for delayed payment.

We consider now a more complicated problem, namely to find *Imin*, *Imax* (with $0 \leqslant Imin \leqslant Imax$) such that any mortgage whose quotient of loss lies in the interval [0.9637 . . 0.97] can be balanced in six months with some interest rate I lying in the interval [*Imin* . . *Imax*]. This problem can be formulated in $HH(\mathscr{R})$ by the goal:

$$\begin{array}{c} G \equiv \forall M \forall P (0.9637 \leqslant P/(6 * M) \leqslant 0.97 \Rightarrow \\ \exists I (0 \leqslant Imin \leqslant I \leqslant Imax \wedge mortgage(P,6,I,M,0))). \end{array}$$

Using the goal transformation rules (1)–(8) of Section 5, we can show a resolution of G from Δ that computes the answer constraint:

$$Imax \approx 10 \wedge Imin \approx 8.219559 \text{ (approx.)}.$$

More details on the resolution of this goal will be given in Example 5.3 at the end of Section 5.

3 Hereditary Harrop formulas with constraints

As explained in the introduction, the framework presented in this paper requires the enrichement of the syntax of *Hereditary Harrop Formulas* (HH) (Miller, Nadathur and Scedrov, 1987; Miller *et al*., 1991) with constraints coming from a given *constraint system*. Following Saraswat (1992), we view a constraint system as a pair $\mathscr{C} = (\mathscr{L}_{\mathscr{C}}, \vdash_{\mathscr{C}})$, where $\mathscr{L}_{\mathscr{C}}$ is the set of formulas allowed as constraints and $\vdash_{\mathscr{C}} \subseteq \mathscr{P}(\mathscr{L}_{\mathscr{C}}) \times \mathscr{L}_{\mathscr{C}}$ is an *entailment relation*. We use C and Γ to represent a constraint and a finite set of constraints, respectively. Therefore, $\Gamma \vdash_{\mathscr{C}} C$ means that the constraint C is entailed by the set of constraints Γ. We write just $\vdash_{\mathscr{C}} C$ if Γ is empty. In (Saraswat, 1992), $\mathscr{L}_{\mathscr{C}}$ and $\vdash_{\mathscr{C}}$ are required to satisfy certain minimal assumptions, mainly related to the logical behaviour of $\wedge$ and $\exists$. Since we have to work with other logical symbols, our assumptions must be extended to account for their proper behaviour. Therefore, we assume:

[1] This example is considered anew in Jaffar *et al.* (1992).

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

(i) $\mathscr{L}_{\mathscr{C}}$ is a set of formulas including $\top$ (true), $\bot$ (false) and all the equations $t \approx t'$ between terms over some fixed signature, and closed under $\wedge, \Rightarrow, \exists, \forall$ and the application of substitutions of terms for variables.

(ii) $\vdash_{\mathscr{C}}$ is *compact*, i.e., $\Gamma \vdash_{\mathscr{C}} C$ holds iff $\Gamma_0 \vdash_{\mathscr{C}} C$ for some finite $\Gamma_0 \subseteq \Gamma$. $\vdash_{\mathscr{C}}$ is also *generic*, i.e. $\Gamma \vdash_{\mathscr{C}} C$ implies $\Gamma\sigma \vdash_{\mathscr{C}} C\sigma$ for every substitution σ.

(iii) All the inference rules related to $\wedge, \Rightarrow, \exists, \forall$ and $\approx$ valid in the intuitionistic fragment of first-order logic are also valid to infer entailments in the sense of $\vdash_{\mathscr{C}}$.

The notation $C\sigma$ used above means application to a constraint C of a substitution $\sigma = [t_1/x_1, \ldots, t_n/x_n]$, using proper renaming of the variables bound in C to avoid capturing free variables from the terms t_i, $1 \leqslant i \leqslant n$. $\Gamma\sigma$ represents the application of σ to every constraint of the set Γ. In the sequel, the notation $F\sigma$ will also be used for other formulas F, not necessarily constraints.

Note that the three conditions (i)–(iii) are meant as minimal requirements. In particular, the availability of the equality symbol $\approx$ is granted in any constraint system, and it will always stand for a congruence. However, other specific axioms for equality may be different in different constraint systems.

Observe also that item (iii) above does not mean that $\vdash_{\mathscr{C}}$ is restricted to represent deducibility in some intuitionistic theory. On the contrary, our assumptions allow us to consider constraint systems $\mathscr{C}$ such that $\mathscr{L}_{\mathscr{C}}$ is a full first-order language with classical negation, and $\Gamma \vdash_{\mathscr{C}} C$ holds iff $Ax_{\mathscr{C}} \cup \Gamma \vdash C$, where $Ax_{\mathscr{C}}$ is a suitable set of first-order axioms and $\vdash$ is the entailment relation of classical first-order logic with equality. In particular, three important constraint systems of this form are: $\mathscr{H}$, where $Ax_{\mathscr{H}}$ is Clark's axiomatization of the Herbrand universe (Clark, 1978); $\mathscr{CFT}$, where $Ax_{\mathscr{CFT}}$ is Smolka and Treinen's axiomatization of the domain of *feature trees* (Smolka and Treinen, 1994); and $\mathscr{R}$, where $Ax_{\mathscr{R}}$ is Tarski's axiomatization of the real numbers (Tarski, 1951). In these three cases, the constraint system is known to be *effective*, in the sense that the validity of entailments $\Gamma \vdash_{\mathscr{C}} C$, with finite Γ, can be decided by an effective procedure.

The previous systems include the use of disjunctions. In CLP there is a well known completeness theorem due to Maher (1987), which relies on the possibility of building finite disjunctions of computed answer constraints. As we will see in Section 5, disjunctions are not needed to prove completeness of goal solving in our setting. This is the reason why we do not enforce $\mathscr{L}_{\mathscr{C}}$ to be closed under $\vee$ in the general case.

In the sequel, we assume an arbitrarily fixed effective constraint system $\mathscr{C}$. By convention, the notation $\Gamma \vdash_{\mathscr{C}} \Gamma'$ will mean that $\Gamma \vdash_{\mathscr{C}} C$ holds for all $C \in \Gamma'$, and $C \vdash\!\dashv_{\mathscr{C}} C'$ will abbreviate that $C \vdash_{\mathscr{C}} C'$ and $C' \vdash_{\mathscr{C}} C$ hold. In addition to this, we will say that a constraint C with free variables $x_1, \ldots, x_n$ is $\mathscr{C}$-satisfiable iff $\vdash_{\mathscr{C}} \exists x_1 \ldots \exists x_n C$.

To define the syntax of the first-order formulas of $HH(\mathscr{C})$, we shall assume a set $PS = \bigcup_{n \in \mathbb{N}} PS^n$ of ranked predicate symbols (which is disjoint from the symbols occurring in $\mathscr{L}_{\mathscr{C}}$), which are used to build atomic formulas A of the form $P(t_1, \ldots, t_n)$, with $P \in PS^n$.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Definition 3.1
The set of *definite clauses*, with elements noted D, and the set of *goals*, with elements noted G, are defined by the following syntactic rules:

$$D := A \mid D_1 \wedge D_2 \mid G \Rightarrow A \mid \forall x D$$
$$G := A \mid C \mid G_1 \wedge G_2 \mid G_1 \vee G_2 \mid D \Rightarrow G \mid C \Rightarrow G \mid \exists x G \mid \forall x G$$

This syntax is the natural extension of first-order HH as presented in Nadathur (193). The novelty is that constraints can occur in goals of the forms C and $C \Rightarrow G$, and therefore also in definite clauses of the form $G \Rightarrow A$. Some variants could be considered, such as dropping $D_1 \wedge D_2$ or replacing $G \Rightarrow A$ by $G \Rightarrow D$, but these changes would render a logically equivalent system. In the rest of the paper, by a *program* we understand any finite set Δ of definite clauses. This includes both CLP programs and first-order HH programs as particular cases.

As usual in the HH framework (e.g. see Nadathur, 1993), we will work with a technical device (so-called *elaboration*) for decomposing the clauses of a given program into a simple form. This is useful for a natural formulation of goal solving procedures.

Definition 3.2
We define the *elaboration of a program* Δ as the set $elab(\Delta) = \bigcup_{D \in \Delta} elab(D)$, where $elab(D)$ is defined by case analysis in the following way:
– $elab(A) = \{\top \Rightarrow A\}$.
– $elab(D_1 \wedge D_2) = elab(D_1) \cup elab(D_2)$.
– $elab(G \Rightarrow A) = \{G \Rightarrow A\}$.
– $elab(\forall x D) = \{\forall x D' \mid D' \in elab(D)\}$.

Note that all clauses in $elab(\Delta)$ have the form $\forall x_1 \ldots \forall x_n (G \Rightarrow A), n \geqslant 0$. We still need another technicality. A *variant* of such a clause is any clause of the form $\forall y_1 \ldots \forall y_n (G\sigma \Rightarrow A\sigma)$ where $y_1, \ldots, y_n$ are new variables not occurring free in the original clause, and $\sigma = [y_1/x_1, \ldots, y_n/x_n]$.

4 Proof systems

In this section we present an amalgamated proof system $\mathscr{IC}$ that combines the usual inference rules from intuitionistic logic with the entailment relation $\vdash_{\mathscr{C}}$ of a constraint system $\mathscr{C}$. We will derive sequents of the form $\Delta; \Gamma \vdash\!\!\!- G$ where Δ is a program, Γ represents a finite set of constraints and G is an arbitrary goal. We also show that $\mathscr{IC}$ enjoys completeness of uniform proofs, and we present a second proof system $\mathscr{UC}$ which is equivalent to $\mathscr{IC}$ in deductive power, but is tailored to build uniform proofs only.

4.1 The calculus $\mathscr{IC}$

$\mathscr{IC}$ stands for an **I**ntuitionistic sequent calculus for $HH(\mathscr{C})$ that allows to deduce a goal from defined clauses in the presence of **C**onstraints.

The intuitionistic calculus with constraints $\vdash_{\mathscr{IC}}$ is defined as follows. $\Delta; \Gamma \vdash_{\mathscr{IC}} G$

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

if and only if the sequent $\Delta; \Gamma \vdash G$ has a proof using the rules of the proof system $\mathscr{IC}$ that we introduce in the following. A proof of a sequent is a tree whose nodes are sequents, the root is the sequent to be proved and the leaves match axioms of the calculus. The rules regulate relationship between child nodes and parent nodes. In the representation of the rules, we have added to the premises the side conditions relating to the existence of proofs in the constraint system; these entailment relations are not considered as nodes of the proofs seen as trees. This notation simplifies the reading of both inference rules and proof trees.

- *Axioms to deal with atomic goals or constraints*:

$$\frac{\Gamma \vdash_{\mathscr{C}} C}{\Delta; \Gamma \vdash C}\ (C_R) \qquad \frac{\Gamma \vdash_{\mathscr{C}} A \approx A'}{\Delta, A; \Gamma \vdash A'}\ (Atom)$$

  In (*Atom*), A, A' are assumed to begin with the same predicate symbol. $A \approx A'$ abbreviates $t_1 \approx t'_1 \wedge \ldots \wedge t_n \approx t'_n$, where $A \equiv P(t_1, \ldots, t_n)$, $A' \equiv P(t'_1, \ldots, t'_n)$.
- *Rules introducing the connectives and quantifiers of the Hereditary Harrop formulas*:

$$\frac{\Delta; \Gamma \vdash G_i}{\Delta; \Gamma \vdash G_1 \vee G_2}\ (\vee_R)\ (i = 1, 2)$$

$$\frac{\Delta, D_1, D_2; \Gamma \vdash G}{\Delta, D_1 \wedge D_2; \Gamma \vdash G}\ (\wedge_L) \qquad \frac{\Delta; \Gamma \vdash G_1 \quad \Delta; \Gamma \vdash G_2}{\Delta; \Gamma \vdash G_1 \wedge G_2}\ (\wedge_R)$$

$$\frac{\Delta; \Gamma \vdash G_1 \quad \Delta, A; \Gamma \vdash G}{\Delta, G_1 \Rightarrow A; \Gamma \vdash G}\ (\Rightarrow_L)$$

$$\frac{\Delta, D; \Gamma \vdash G}{\Delta; \Gamma \vdash D \Rightarrow G}\ (\Rightarrow_R) \qquad \frac{\Delta; \Gamma, C \vdash G}{\Delta; \Gamma \vdash C \Rightarrow G}\ (\Rightarrow C_R)$$

$$\frac{\Delta; \Gamma, C \vdash G[y/x] \quad \Gamma \vdash_{\mathscr{C}} \exists y C}{\Delta; \Gamma \vdash \exists x G}\ (\exists_R)$$

y does not appear free in the sequent of the conclusion.

$$\frac{\Delta, D[y/x]; \Gamma, C \vdash G \quad \Gamma \vdash_{\mathscr{C}} \exists y C}{\Delta, \forall x D; \Gamma \vdash G}\ (\forall_L) \qquad \frac{\Delta; \Gamma \vdash G[y/x]}{\Delta; \Gamma \vdash \forall x G}\ (\forall_R)$$

in both, y does not appear free in the sequent of the conclusion.

Note that the rule of contraction seems to be absent from this system, but in fact it is implicitly present because Δ and Γ are viewed as sets (rather than sequences) in any sequent $\Delta; \Gamma \vdash G$. In many respects, the inference rules of $\mathscr{UC}$ are similar to those used for HH in the literature (e.g. see Miller *et al.* (1991) and Nadathur (1993)). However, the presence of constraints induces some modifications. Of particular importance are the modifications introduced to $(\exists_R)$ and $(\forall_L)$. A simple reformulation of the traditional version of $(\exists_R)$, using a constraint $y \approx t$ instead of a substitution $[t/x]$, representing an instance of x, could be:

$$\frac{\Delta; \Gamma, y \approx t \vdash G[y/x]}{\Delta; \Gamma \vdash \exists x G}$$

if y does not occur in t, and it does not appear free in the conclusion.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

In our constraint-oriented formulation of $(\exists_R)$ we allow any satisfiable constraint C (not necessary of the form $y \approx t$) instead of the substitution, to guess an instance of x. The next example shows that this extra generality is necessary.

Example 4.1
This example is based on $HH(\mathscr{R})$. Consider
$\Delta \equiv \{\forall x(x^2 \approx 2 \Rightarrow r(x))\}$,
$G \equiv \exists x\ r(x)$.
The sequent $\Delta; \vdash G$ is expected to be derivable. However, the traditional formulation of $(\exists_R)$ does not work, because no term t in the language $\mathscr{L}_{\mathscr{R}}$ denotes a square root of 2. With our $(\exists_R)$, choosing the $\mathscr{R}$-satisfiable constraint $C \equiv x^2 \approx 2$, the problem is reduced to the easy derivation of the sequent $\Delta; x^2 \approx 2 \vdash r(x)$.

Our definition of $(\forall_L)$ is dual to $(\exists_R)$ and follows the same idea, since $(\forall_L)$ also relies on guessing an instance for x. On the other hand, rule $(\forall_R)$ has a universal character. Therefore, the traditional formulation by means of a new variable has been kept in this case.

For technical reasons, we need to measure the *size* of proofs. We formalize this notion as the number of sequents in it, that coincides with the number of nodes of the proof seen as a tree.

In the sequel we will use some technical properties of $\mathscr{IC}$-provability. Let us state them in the following lemmas, the proofs of which can be found in the Appendix.

The first lemma guarantees that substitution of a term for a variable in a sequent, preserves $\mathscr{IC}$-provability.

Lemma 4.1
For any Δ, Γ, G, x and t, if $\Delta; \Gamma \vdash_{\mathscr{IC}} G$, then there is a proof of the same size of $\Delta[t/x]; \Gamma[t/x] \vdash G[t/x]$.

The next lemma shows that a sequent continues to be provable if we strengthen the set of constraints.

Lemma 4.2
For any Δ, Γ, G, if Γ' is a set of constraints such that $\Gamma' \vdash_{\mathscr{C}} \Gamma$, and $\Delta; \Gamma \vdash_{\mathscr{IC}} G$, then $\Delta; \Gamma' \vdash G$ has a proof of the same size.

Corollary 4.3
For any Δ, Γ, G, x and u, if $\Delta; \Gamma \vdash_{\mathscr{IC}} G$, then $\Delta[u/x]; \Gamma, x \approx u \vdash G[u/x]$ has a proof of the same size.

Proof
By Lemma 4.1, $\Delta[u/x]; \Gamma[u/x] \vdash G[u/x]$ has a proof of the same size as $\Delta; \Gamma \vdash G$. Hence, applying Lemma 4.2, $\Delta[u/x]; \Gamma, x \approx u \vdash G[u/x]$ has a proof of the same size, because $\Gamma, x \approx u \vdash_{\mathscr{C}} \Gamma[u/x]$. □

The next lemma ensures that free variables that appear only in the set of constraints of a sequent can be considered as existentially quantified in the proof of the sequent.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Lemma 4.4
For any Δ, Γ, C, G, if $\Delta; \Gamma, C \vdash_{\mathscr{IC}} G$ and x is a variable that does not appear free in Δ, Γ, G, then $\Delta; \Gamma, \exists x C \vdash\!\!- G$ has a proof of the same size.

4.2 Uniform proofs

We are aiming at an *abstract logic programming language* in the sense of Miller *et al.* (1991). This means that *uniform proofs* must exist for all provable sequents. In our setting, the idea of uniform proof consists of breaking down a goal into its components until obtaining an atomic formula or a constraint, before using the rules for introduction of connectives on the left or resorting to constraint entailment.

More formally, the notion of uniform proof is as follows.

Definition 4.1
An $\mathscr{IC}$-proof is called *uniform proof* when each internal node in the proof tree is a sequent whose right-hand side G is neither a constraint nor an atomic formula. Moreover, the inference rule relating this node to its children must be one of the right-introduction rules $(\vee_R)$, $(\wedge_R)$, $(\Rightarrow_R)$, $(\Rightarrow C_R)$, $(\exists_R)$, $(\forall_R)$, according to the outermost logical symbol of G.

To prove that uniform proofs exist for all $\mathscr{IC}$-provable sequents, we follow the same approach that in Miller *et al.* (1991), showing that any given $\mathscr{IC}$-proof can be transformed into a uniform proof. This is achieved by the next lemma.

Lemma 4.5 (*Proof transformation*)
If G is a goal, Δ a program and Γ a set of constraint formulas, such that $\Delta; \Gamma \vdash\!\!- G$ has a proof of size l, then:

1. For $G \equiv A$, there are n constraint formulas $C_1, \ldots, C_n$ $(n \geqslant 0)$ and a formula $\forall x_1 \ldots \forall x_n\ (G' \Rightarrow A')$ that is a variant of some formula in $elab(\Delta)$ such that $x_1, \ldots, x_n$ are new distinct variables not appearing free in Δ, Γ, A, where x_i does not appear free in $C_1, \ldots, C_{i-1}$, for $1 < i \leqslant n$, and A' begins with the same predicate symbol as A. In addition it holds:
   (a) $\Gamma \vdash_{\mathscr{C}} \exists x_1 C_1; \quad \Gamma, C_1 \vdash_{\mathscr{C}} \exists x_2 C_2; \quad \ldots; \quad \Gamma, C_1, \ldots, C_{n-1} \vdash_{\mathscr{C}} \exists x_n C_n.$
   (b) $\Gamma, C_1, \ldots, C_n \vdash_{\mathscr{C}} A' \approx A.$
   (c) $\Delta; \Gamma, C_1, \ldots, C_n \vdash\!\!- G'$ has a proof of size less than l, or $G' \equiv \top$.
2. If $G \equiv C$, then $\Gamma \vdash_{\mathscr{C}} C$.
3. If $G \equiv G_1 \wedge G_2$, then $\Delta; \Gamma \vdash\!\!- G_1$ and $\Delta; \Gamma \vdash\!\!- G_2$ have proofs of size less than l.
4. If $G \equiv G_1 \vee G_2$, then $\Delta; \Gamma \vdash\!\!- G_i$ has a proof of size less than l for $i = 1$ or 2.
5. If $G \equiv D \Rightarrow G_1$, then $\Delta, D; \Gamma \vdash\!\!- G_1$ has a proof of size less than l.
6. If $G \equiv C \Rightarrow G_1$, then $\Delta; \Gamma, C \vdash\!\!- G_1$ has a proof of size less than l.
7. For $G \equiv \exists x G_1$, if y is a variable not appearing free in Δ, Γ, G, then there is a constraint formula C such that:
   (a) $\Gamma \vdash_{\mathscr{C}} \exists y C$.
   (b) $\Delta; \Gamma, C \vdash\!\!- G_1[y/x]$ has a proof of size less than l.
8. If $G \equiv \forall x G_1$, then $\Delta; \Gamma \vdash\!\!- G_1[y/x]$ has a proof of size less than l, where y is a variable that does not appear free in Δ, Γ, G.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Proof

We reason by induction on the size l of the proof of $\Delta;\Gamma \vdash\!\!- G$, analysing cases according to the last inference rule applied in the proof of the sequent $\Delta;\Gamma \vdash\!\!- G$. A detailed proof can be found in the Appendix. As novelties w.r.t. Miller *et al.* (1991), we must deal with constraints and with the new formulation of rules $(\exists_R)$, $(\forall_L)$. Here we only sketch the case where $(\forall_L)$ is the last inference rule applied and $G \equiv \exists w G_1$. Let us show graphically the proof transformation, in which we will essentially switch the applications of $(\forall_L)$ and $(\exists_R)$. By the induction hypothesis, the initial proof has the form:

$$\dfrac{\dfrac{\begin{array}{c}\Delta', D[u/x];\Gamma, C' \wedge C, u \approx y \vdash\!\!- G_1[z/w] \\ \uparrow \text{ Cor. 4.3, Lem. 4.2} \\ \Delta', D[y/x];\Gamma, C', C \vdash\!\!- G_1[z/w]\end{array} \qquad \Gamma, C' \vdash_{\mathscr{C}} \exists z C}{\Delta', D[y/x];\Gamma, C' \vdash\!\!- \exists w G_1} (\exists_R) \qquad \Gamma \vdash_{\mathscr{C}} \exists y C'}{\Delta', \forall x D;\Gamma \vdash\!\!- \exists w G_1} (\forall_L)$$

where:

– y is not free in Δ', $\forall x D$, Γ, $\exists w G_1$.
– z is not free in Δ', $D[y/x]$, Γ, C', $\exists w G_1$.
– u is a new variable.

We can transform this into the following proof:

$$\dfrac{\dfrac{\Delta', D[u/x];\Gamma, C' \wedge C, u \approx y \vdash\!\!- G_1[z/w] \qquad \Gamma, C' \wedge C \vdash_{\mathscr{C}} \exists u(u \approx y)}{\Delta', \forall x D;\Gamma, C' \wedge C \vdash\!\!- G_1[z/w]} (\forall_L)}{\begin{array}{l}\downarrow \text{ Lem. 4.4} \\ \Delta', \forall x D;\Gamma, \exists y(C' \wedge C) \vdash\!\!- G_1[z/w] \qquad \Gamma \vdash_{\mathscr{C}} \exists z \exists y(C' \wedge C)\end{array}}{\Delta', \forall x D;\Gamma \vdash\!\!- \exists w G_1} (\exists_R)$$

where:

– z is not free in Δ', $\forall x D$, Γ, $\exists w G_1$.
– u is not free in Δ', $\forall x D$, Γ, $C' \wedge C$, $G_1[z/w]$. □

The next main theorem follows now as a straightforward consequence of the Proof Transformation Lemma 4.5.

Theorem 4.6 (*Uniform proofs*)

Every $\mathscr{IC}$-provable sequent has a uniform proof.

Proof

Given an $\mathscr{IC}$-provable sequent with a proof of size l, the existence of a uniform proof is established reasoning by induction on l, using Lemma 4.5. □

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

4.3 The calculus $\mathcal{UC}$

Now we know that uniform proofs are complete for $\mathcal{IC}$, and their goal-oriented format renders them close to the goal solving procedure we are looking for. However, as an intermediate step we will present a second proof system $\mathcal{UC}$ for $HH(\mathcal{C})$, which will enjoy three properties:

(a) $\mathcal{UC}$ and $\mathcal{IC}$ have the same provable sequents.
(b) $\mathcal{UC}$ builds only **U**niform proofs, and it is parameterized by a given **C**onstraint system.
(c) $\vdash_{\mathcal{UC}}$ replaces the left-introduction rules by a backchaining mechanism.

$\mathcal{UC}$-derivations are *very close* to our intended computations. Therefore, the $\mathcal{UC}$ system will be very useful for designing a sound and complete goal solving procedure in the next section.

Provability in $\mathcal{UC}$ is defined as follows. $\Delta;\Gamma \vdash_{\mathcal{UC}} G$ if and only if the sequent $\Delta;\Gamma \vdash\!\!\!- G$ has a proof using the following rules:

- *Axiom to deal with constraints*:

$$\frac{\Gamma \vdash_{\mathcal{C}} C}{\Delta;\Gamma \vdash\!\!\!- C}\ (C_R)$$

- *Backchaining rule for atomic goals*:

$$\frac{\Delta;\Gamma \vdash\!\!\!- \exists x_1 \ldots \exists x_n((A \approx A') \wedge G)}{\Delta;\Gamma \vdash\!\!\!- A'}\ (Clause)$$

  where A, A' begin with the same predicate symbol and $\forall x_1 \ldots \forall x_n(G \Rightarrow A)$ is a variant of a formula of $elab(\Delta)$, where $x_1, \ldots, x_n$ do not appear free in the sequent of the conclusion.
- *Rules introducing the connectives and quantifiers of the goals*:

$$(\vee_R),\ (\wedge_R),\ (\Rightarrow_R),\ (\Rightarrow C_R),\ (\exists_R),\ (\forall_R).$$

  Defined as in the system $\mathcal{IC}$.

The structure of the rule ($Clause$), which encapsulates a backchaining mechanism, corresponds to the method by which atomic goals, A', will be solved by the goal solving procedure, which to be presented in Section 5 below. As is usual in logic programming, an 'instance' of a clause with a head A and a body G is searched, in such a way that $A \approx A'$ and G can be proved. By the definition of $\mathcal{UC}$, the existential quantification on the right-hand side of the premise sequent forces a search for this 'instance' (managed by means of constraints in our system). Note that a similar behaviour would result from the application of $(\forall_L)$, if we were to make use of $\mathcal{IC}$.

The next auxiliary lemma is needed to show that $\mathcal{UC}$ and $\mathcal{IC}$ have the same deductive power. It can be viewed as a particular kind of cut elimination for $\mathcal{IC}$, where the cut formula is taken from the elaboration of the program in the left-hand side of the sequent. We cannot apply directly any classical cut elimination result, because constraint entailment is embedded into our proof system.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Lemma 4.7 (*Elaboration*)
For any Δ, Γ, A and $F \in elab(\Delta)$: if $\Delta, F; \Gamma \vdash_{\mathscr{IC}} A$, then $\Delta; \Gamma \vdash_{\mathscr{IC}} A$.

Proof
It appears in the Appendix. □

Now we can prove the promised equivalence between $\mathscr{UC}$ and $\mathscr{IC}$.

Theorem 4.8
The proof systems $\mathscr{IC}$ and $\mathscr{UC}$ are equivalent. That means, for any program Δ, for any set of constraints Γ, and for any goal G it holds:

$$\Delta; \Gamma \vdash_{\mathscr{IC}} G \text{ if and only if } \Delta; \Gamma \vdash_{\mathscr{UC}} G.$$

Proof
We prove both implications by induction on the size of proofs.

⇒ Assuming $\Delta; \Gamma \vdash_{\mathscr{IC}} G$, we prove $\Delta; \Gamma \vdash_{\mathscr{UC}} G$ by case analysis on the structure of G.
If $G \equiv A$, by the Proof Transformation Lemma (4.5) there are n ($n \geqslant 0$) constraints $C_1, \ldots, C_n$, a variant $\forall x_1 \ldots \forall x_n (G' \Rightarrow A')$ of some formula of $elab(\Delta)$, with $x_1, \ldots, x_n$ new distinct variables, x_i not appearing free in $C_1, \ldots, C_{i-1}$, for $1 < i \leqslant n$, and A, A' beginning with the same predicate symbol, such that:

(a) $\Gamma \vdash_{\mathscr{C}} \exists x_1 C_1$; $\Gamma, C_1 \vdash_{\mathscr{C}} \exists x_2 C_2$; …; $\Gamma, C_1, \ldots, C_{n-1} \vdash_{\mathscr{C}} \exists x_n C_n$.
(b) $\Gamma, C_1, \ldots, C_n \vdash_{\mathscr{C}} A' \approx A$.
(c) $\Delta; \Gamma, C_1, \ldots, C_n \vdash_{\mathscr{IC}} G'$, with a shorter proof, or $G' \equiv \top$.

By (b) and (C_R), $\Delta; \Gamma, C_1, \ldots, C_n \vdash_{\mathscr{UC}} A' \approx A$. By (c) and the induction hypothesis, $\Delta; \Gamma, C_1, \ldots, C_n \vdash_{\mathscr{UC}} G'$. Note that if $G' \equiv \top$, the proof of this sequent is a direct consequence of the rule (C_R). So applying $(\wedge_R)$, $\Delta, \Gamma, C_1, \ldots, C_n \vdash_{\mathscr{UC}} (A' \approx A) \wedge G'$. Now, in accordance with (a) and the conditions on $x_1, \ldots, x_n$, it is possible to apply $(\exists_R)$ n times obtaining $\Delta; \Gamma \vdash_{\mathscr{UC}} \exists x_1 \ldots \exists x_n ((A' \approx A) \wedge G')$.

Therefore, using (*Clause*), $\Delta; \Gamma \vdash_{\mathscr{UC}} A$.

The cases for non-atomic formulas are immediate due to the Proof Transformation Lemma (4.5), the definition of $\mathscr{UC}$ and the induction hypothesis.

⇐ Let us also prove only the atomic case, the others are proved using the induction hypothesis and the definition of the calculi $\mathscr{UC}$, $\mathscr{IC}$.
Assume $\Delta; \Gamma \vdash_{\mathscr{UC}} A$, then by the definition of $\mathscr{UC}$ the rule (*Clause*) has been applied and $\Delta; \Gamma \vdash_{\mathscr{UC}} \exists x_1 \ldots \exists x_n ((A' \approx A) \wedge G')$, with a shorter proof, where $\forall x_1 \ldots \forall x_n (G' \Rightarrow A')$ is a variant of a formula of $elab(\Delta)$ with $x_1, \ldots, x_n$ new variables and A, A' beginning with the same predicate symbol. Because of the form of $\mathscr{UC}$'s inference rules, the only way to derive this sequent is by n successive applications of $(\exists_R)$. Since $x_1, \ldots, x_n$ are new[2], we can assume:

(a) $\Gamma \vdash_{\mathscr{C}} \exists x_1 C_1$; $\Gamma, C_1 \vdash_{\mathscr{C}} \exists x_2 C_2$; …; $\Gamma, C_1, \ldots, C_{n-1} \vdash_{\mathscr{C}} \exists x_n C_n$.

[2] Without loss of generality we can consider that x_i does not appear free in $C_1, \ldots, C_{i-1}$, for $1 < i \leqslant n$.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

(b) $\Delta; \Gamma, C_1, \ldots, C_n \vdash_{\mathcal{UC}} (A' \approx A) \wedge G'$, with a shorter proof.

Then by (b) and according to the definition of $\mathcal{UC}$, $\Delta; \Gamma, C_1, \ldots, C_n \vdash_{\mathcal{UC}} A' \approx A$ and $\Delta; \Gamma, C_1, \ldots, C_n \vdash_{\mathcal{UC}} G'$ with shorter proofs. Therefore, by the induction hypothesis,

$$\Delta; \Gamma, C_1, \ldots, C_n \vdash_{\mathcal{IC}} A' \approx A \ (\dagger) \text{ and}$$
$$\Delta; \Gamma, C_1, \ldots, C_n \vdash_{\mathcal{IC}} G' \ (\ddagger).$$

($\dagger$) implies $\Gamma, C_1, \ldots, C_n \vdash_{\mathcal{C}} A' \approx A$, by the Proof Transformation Lemma (4.5). Then, by (*Atom*),

$$\Delta, A'; \Gamma, C_1, \ldots, C_n \vdash_{\mathcal{IC}} A \ (\diamond),$$

so applying ($\Rightarrow_L$) to ($\ddagger$) and ($\diamond$),

$$\Delta, G' \Rightarrow A'; \Gamma, C_1, \ldots, C_n \vdash_{\mathcal{IC}} A.$$

Now by n applications of ($\forall_L$), using (a) and the conditions on $x_1 \ldots, x_n$, we obtain

$$\Delta, \forall x_1 \ldots \forall x_n (G' \Rightarrow A'); \Gamma \vdash_{\mathcal{IC}} A.$$

Therefore by the Elaboration Lemma (4.7) $\Delta; \Gamma \vdash_{\mathcal{IC}} A$. □

The properties stated in Lemma 4.2 and Lemma 4.4 hold also for $\mathcal{UC}$-derivability. This is ensured by the next two lemmas that are proved in the Appendix.

Lemma 4.9
For any Δ, Γ, G, if Γ' is a set of constraints such that $\Gamma' \vdash_{\mathcal{C}} \Gamma$, and $\Delta; \Gamma \vdash_{\mathcal{UC}} G$, then $\Delta; \Gamma' \vdash\!\!- G$ has a $\mathcal{UC}$-proof of the same size.

Lemma 4.10
For any Δ, Γ, C, G, if $\Delta; \Gamma, C \vdash_{\mathcal{UC}} G$ and x is a variable that does not appear free in Δ, Γ, G, then $\Delta; \Gamma, \exists x C \vdash\!\!- G$ has a $\mathcal{UC}$-proof of the same size.

From now on we will work only with the calculus $\mathcal{UC}$.

5 A goal solving procedure

We now turn to the view of $HH(\mathcal{C})$ as a logic programming language. Solving a goal G using a program Δ means finding a $\mathcal{C}$-satisfiable constraint R such that

$$\Delta; R \vdash_{\mathcal{UC}} G.$$

Any constraint R with this property is called a *correct answer constraint*. For instance, $R \equiv x^2 \leqslant 1/2$ is a correct answer constraint for the *disc* example, as shown in the introduction.

We will present a goal solving procedure as a transition system. Goal solving will proceed by transforming an initial state through a sequence of intermediate states, ending in a final state. Each state will conserve the goals that remain to be solved and a partially calculated answer constraint. The final state will not have any goal to be solved. In the following we will formalize these ideas and show soundness and completeness of the proposed procedure.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Definition 5.1
A *state* w.r.t. a finite set of variables V, written $\mathscr{S}$, has the form $\Pi[S\Box\mathscr{G}]$, where $\mathscr{G}$ is a multiset of triples $\langle\Delta, C, G\rangle$ (Δ *local program*, C *local constraint* formula and G *local goal*). Π is a quantifier prefix $Q_1x_1 \ldots Q_kx_k$ where $x_1, \ldots, x_k$ are distinct variables not belonging to V, and every Q_i, $1 \leqslant i \leqslant k$, is the quantifier $\forall$ or $\exists$. S is a *global constraint* formula.

This complex notion of state is needed because the goal solving transformations, presented below, introduce local clauses and local constraints. Of course, local clauses also arise in HH (see Nadathur (1993)). Initial states are quite simple as can be seen in Definition 5.3.

We say that a *state* $\Pi[S\Box\mathscr{G}]$ is *satisfiable* iff the associated constraint formula ΠS, also called *partially calculated answer constraint*, is $\mathscr{C}$-satisfiable.

If Π', Π are quantifier prefixes such that Π' coincides with the first k elements of Π, $0 \leqslant k \leqslant n$, where n is the number of elements of Π, then $\Pi - \Pi'$ represents the result of eliminating Π' of Π. For instance $\forall x\forall y\exists z\forall u\exists v - \forall x\forall y\exists z \equiv \forall u\exists v$.

To represent a multiset $\mathscr{G}$, we will simply write its elements separated by commas, assuming that repetitions are relevant but ordering is not. In particular, the notation $\mathscr{G}, \langle\Delta, C, G\rangle$ stands for any multiset which includes at least one occurrence of the triple $\langle\Delta, C, G\rangle$.

Definition 5.2 (*Rules for transformation of states*)
The transformations permitting to pass from a state $\mathscr{S}$ w.r.t. a set of variables V, to another state $\mathscr{S}'$ w.r.t. V, written as $\mathscr{S} \Vdash \mathscr{S}'$, are the following:

(1) *Conjunction*
$\Pi[S\Box\mathscr{G}, \langle\Delta, C, G_1 \wedge G_2\rangle] \Vdash \Pi[S\Box\mathscr{G}, \langle\Delta, C, G_1\rangle, \langle\Delta, C, G_2\rangle]$.

(2) *Disjunction*
$\Pi[S\Box\mathscr{G}, \langle\Delta, C, G_1 \vee G_2\rangle] \Vdash \Pi[S\Box\mathscr{G}, \langle\Delta, C, G_i\rangle]$, for $i = 1$ or 2 (don't know choice).

(3) *Implication with local clause*
$\Pi[S\Box\mathscr{G}, \langle\Delta, C, D \Rightarrow G\rangle] \Vdash \Pi[S\Box\mathscr{G}, \langle\Delta \cup \{D\}, C, G\rangle]$.

(4) *Implication with local constraint*
$\Pi[S\Box\mathscr{G}, \langle\Delta, C, C' \Rightarrow G\rangle] \Vdash \Pi[S\Box\mathscr{G}, \langle\Delta, C \wedge C', G\rangle]$.

(5) *Existential quantification*
$\Pi[S\Box\mathscr{G}, \langle\Delta, C, \exists xG\rangle] \Vdash \Pi\exists w[S\Box\mathscr{G}, \langle\Delta, C, G[w/x]\rangle]$, where w does not appear in Π nor in V.

(6) *Universal quantification*
$\Pi[S\Box\mathscr{G}, \langle\Delta, C, \forall xG\rangle] \Vdash \Pi\forall w[S\Box\mathscr{G}, \langle\Delta, C, G[w/x]\rangle]$, where w does not appear in Π nor in V.

(7) *Constraint*
$\Pi[S\Box\mathscr{G}, \langle\Delta, C, C'\rangle] \Vdash \Pi[S \wedge (C \Rightarrow C')\Box\mathscr{G}]$.
If $\Pi(S \wedge (C \Rightarrow C'))$ is $\mathscr{C}$-satisfiable.

(8) *Clause of the program*
$\Pi[S\Box\mathscr{G}, \langle\Delta, C, A\rangle] \Vdash \Pi[S\Box\mathscr{G}, \langle\Delta, C, \exists x_1 \ldots \exists x_n((A' \approx A) \wedge G)\rangle]$.
Provided that $\forall x_1 \ldots \forall x_n(G \Rightarrow A')$ is a variant of some clause in $elab(\Delta)$ (don't know choice), $x_1, \ldots, x_n$ do not appear in Π nor in V, and A', A begin with the same predicate symbol.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Note that every transformation can be applied to an arbitrary triple $\langle \Delta, C, G \rangle$ within the state, since $\mathscr{G}$ is viewed as a multiset. Moreover, all choices involved in carrying out a sequence of state transformations are *don't care*, except those explicitly labelled as *don't know* in transformations (2) and (8) above. One can commit to don't care choices without compromising completeness. In other words: at the implementation level, backtracking is needed only for don't know choices.

The following definition formalizes the setting needed for goal solving.

Definition 5.3
The *initial state* for a program Δ and a goal G is a state w.r.t. the set of free variables of Δ and G consisting in $\mathscr{S}_0 \equiv [\top \square \langle \Delta, \top, G \rangle]$.

A *resolution of a goal G from a program* Δ is a finite sequence of states w.r.t. the free variables of Δ and G, $\mathscr{S}_0, \ldots, \mathscr{S}_n$, such that:

- $\mathscr{S}_0$ is the initial state for Δ and G.
- $\mathscr{S}_{i-1} \Vdash\!\!\!- \mathscr{S}_i$, $1 \leqslant i \leqslant n$, by means of any of the transformation rules.
- The *final state* $\mathscr{S}_n$ has the form $\Pi_n[S_n \square \emptyset]$.

The constraint $\Pi_n S_n$ is called the *answer constraint* of this resolution.

Example 5.1
Using Δ, G and R as given in the *disc example* (see the Introduction) it is possible to build a resolution of G from Δ with answer constraint R as follows:

$$
\begin{aligned}
&[\top \square \langle \Delta, \top, \forall y(y^2 \leqslant 1/2 \Rightarrow \mathit{disc}\ (x, y)) \rangle] \Vdash\!\!\!-_{(6)} \\
&\forall y[\top \square \langle \Delta, \top, y^2 \leqslant 1/2 \Rightarrow \mathit{disc}\ (x, y) \rangle] \Vdash\!\!\!-_{(4)} \\
&\forall y[\top \square \langle \Delta, y^2 \leqslant 1/2, \mathit{disc}\ (x, y) \rangle] \Vdash\!\!\!-_{(8)} \\
&\forall y[\top \square \langle \Delta, y^2 \leqslant 1/2, \exists u \exists v (x \approx u \wedge y \approx v \wedge u^2 + v^2 \leqslant 1/2) \rangle] \Vdash\!\!\!-_{(7)} \\
&\forall y[y^2 \leqslant 1/2 \Rightarrow \exists u \exists v (x \approx u \wedge y \approx v \wedge u^2 + v^2 \leqslant 1) \square \emptyset]
\end{aligned}
$$

since $\forall y(y^2 \leqslant 1/2 \Rightarrow \exists u \exists v(x \approx u \wedge y \approx v \wedge u^2 + v^2 \leqslant 1))$ is $\mathscr{R}$-satisfiable.
So the answer constraint is

$$
\begin{aligned}
&\forall y(y^2 \leqslant 1/2 \Rightarrow \exists u \exists v (x \approx u \wedge y \approx v \wedge u^2 + v^2 \leqslant 1)) \vdash\!\!\!\dashv_{\mathscr{R}} \\
&\forall y(y^2 \leqslant 1/2 \Rightarrow x^2 + y^2 \leqslant 1) \vdash\!\!\!\dashv_{\mathscr{R}} x^2 \leqslant 1/2.
\end{aligned}
$$

□

For CLP programs, the goal transformations (2), (3), (4) and (6) can never be applied. Therefore, the state remains of the form $\Pi[S \square \mathscr{G}]$, where Π includes only existential quantifiers and $\mathscr{G}$ is a multiset of triples $\langle \Delta, C, G \rangle$ such that Δ is the global program. For states of this kind, the goal transformations (1), (5), (7) and (8) specify constrained *SLD* resolution, as used in CLP (e.g. see Jaffar and Maher (1994) and Jaffar *et al.* (1996)). On the other hand, traditional HH programs can be emulated in our framework by using the Herbrand constraint system $\mathscr{H}$ and avoiding constraints in programs and initial goals. Then transformation (4) becomes useless, and the remaining goal transformations can be viewed as a more abstract formulation of the goal solving procedure from Nadathur (193). Transformation (8) introduces equational constraints in intermediate goals, and in transformation (7) the local constraint C is simply $\top$. Therefore, $\Pi(S \wedge (C \Rightarrow C'))$ is equivalent to $\Pi(S \wedge C')$, where $S \wedge C'$ can be assumed to be a conjunction of equations. Checking

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

$\mathcal{H}$-satisfiability of $\Pi(S \wedge C')$ corresponds to solving a unification problem under a mixed prefix in Nadathur (1993).

Admittedly, the labelled unification algorithm presented in Nadathur (1993) is closer to an actual implementation, while our description of goal solving is more abstract. Note, however, that the goal solving transformations are open to efficient implementation techniques. In particular, when (7) adds a new constraint to the global constraint S, the satisfiability of the new partially calculated answer constraint should be checked incrementally, without repeating all the work previously done for ΠS. Of course, delaying the constraint satisfiability checks until the end is neither necessary nor convenient.

5.1 Soundness

Soundness of the goal solving procedure means that if R is the answer constraint of a resolution of a goal G from a program Δ, then the sequent $\Delta; R \vdash G$ has a $\mathcal{UC}$-proof.

The soundness theorem is based on two auxiliary results. The first one ensures that states remain satisfiable along any resolution.

Lemma 5.1
Let $\mathcal{S}_0, \ldots, \mathcal{S}_n$ be a resolution of a goal G from a program Δ, and V the set of free variables of Δ and G. Then, for any i, $0 \leqslant i \leqslant n$, if $\mathcal{S}_i \equiv \Pi_i[S_i \Box \mathcal{G}_i]$, then the following properties are satisfied:

1. The free variables of the formulas of $\mathcal{G}_i$, and S_i are in Π_i or in V.
2. $\mathcal{S}_i$ is satisfiable.

Proof
The first property is a consequence of the procedure used to build the prefix of a state. The initial state satisfies it by definition, and when passing from state $\mathcal{S}_{i-1}$ to state $\mathcal{S}_i$, $1 \leqslant i \leqslant n$, if we include new free variables, these will be quantified universally or existentially by Π_i.

For the second property, note that $S_0 \equiv \top$ by definition. Moreover, for each transformation step $\mathcal{S}_{i-1} \Vdash \mathcal{S}_i$, one of the three following cases applies:

- $S_i \not\equiv S_{i-1}$. Then the transition must correspond to the transformation (7) which requires $\mathcal{C}$-satisfiability of $\Pi_i(S_i)$.
- $S_i \equiv S_{i-1}$ and $\Pi_i \equiv \Pi_{i-1}$. This case is trivial.
- $S_i \equiv S_{i-1}$ and $\Pi_i \equiv \Pi_{i-1} Q w$, where Q is $\forall$ or $\exists$ and w is a new variable not free in S_{i-1}, and not occurring in Π_{i-1}. Under these conditions,

$$\Pi_i S_i \equiv \Pi_{i-1} Q w S_{i-1} \vdash\!\!\dashv_{\mathcal{C}} \Pi_{i-1} S_{i-1},$$

and $\mathcal{C}$-satisfiability propagates from $\Pi_{i-1} S_{i-1}$ to $\Pi_i S_i$. □

The second auxiliary lemma means that correct answer constraints are preserved by any resolution step.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Lemma 5.2
Assume $\mathscr{S} \equiv \Pi[S \square \mathscr{G}]$ and $\mathscr{S}' \equiv \Pi\Pi'[S' \square \mathscr{G}']$ are two states w.r.t. a set of variables V, such that $\mathscr{S} \| \!\!- \mathscr{S}'$. If R' is a constraint with its free variables in $\Pi\Pi'$ or in V, and such that $R' \vdash_{\mathscr{C}} S'$ and for any $\langle \Delta', C', G' \rangle \in \mathscr{G}'$, $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$, then $\Pi' R' \vdash_{\mathscr{C}} S$ and for any $\langle \Delta, C, G \rangle \in \mathscr{G}$, $\Delta; \Pi' R', C \vdash_{\mathscr{UC}} G$.

Proof
We analyse the different cases, according to the transformation applied. We show here the first case, the other cases appear in the Appendix.

(i) *Conjunction.* Π' is empty and $S \equiv S'$, so $\Pi' R' \vdash_{\mathscr{C}} S$ obviously. On the other hand, let $\langle \Delta, C, G \rangle \in \mathscr{G}$: If $\langle \Delta, C, G \rangle \in \mathscr{G}'$, then $\Delta; \Pi' R', C \vdash_{\mathscr{UC}} G$ by hypothesis, since $\Pi' R' \equiv R'$. If $\langle \Delta, C, G \rangle \notin \mathscr{G}'$, then $G \equiv G_1 \wedge G_2$ and $\langle \Delta, C, G_1 \rangle$, $\langle \Delta, C, G_2 \rangle \in \mathscr{G}'$. Therefore $\Delta; \Pi' R', C \vdash_{\mathscr{UC}} G_1$ and $\Delta; \Pi' R', C \vdash_{\mathscr{UC}} G_2$, by hypothesis, since $\Pi' R' \equiv R'$,
and consequently $\Delta; \Pi' R', C \vdash_{\mathscr{UC}} G$, by applying $(\wedge_R)$. □

Theorem 5.3 (*Soundness*)
Let Δ be any program. If G is a goal such that there is a resolution $\mathscr{S}_0, \ldots, \mathscr{S}_n$ of G from Δ with answer constraint $R \equiv \Pi_n S_n$, then R is $\mathscr{C}$-satisfiable and $\Delta; R \vdash_{\mathscr{UC}} G$.

Proof
The proof is direct from the previous lemmas. $\mathscr{C}$-satisfiability of R is a consequence of item 2 of Lemma 5.1. Besides using Lemma 5.2 we can prove, that for $0 \leqslant i \leqslant n$, $\Delta; (\Pi_n - \Pi_i) S_n, C \vdash_{\mathscr{UC}} G$, for any $\langle \Delta, C, G \rangle \in \mathscr{G}_i$, and $(\Pi_n - \Pi_i) S_n \vdash_{\mathscr{C}} S_i$. The case $i = 0$ of this result assures the theorem. Let us prove it by induction on the construction of $\mathscr{S}_0, \ldots, \mathscr{S}_n$, but beginning from the last state. The base case is obvious because $\mathscr{G}_n = \emptyset$ and $(\Pi_n - \Pi_n) S_n \vdash_{\mathscr{C}} S_n$ holds trivially. For the induction step, we suppose the result for $\mathscr{S}_{i+1}, \ldots, \mathscr{S}_n$, and we prove it for $\mathscr{S}_i$. Taking $(\Pi_n - \Pi_{i+1}) S_n$ as the constraint R' of Lemma 5.2, the induction hypothesis for $i + 1$ indicates that the conditions of Lemma 5.2 are satisfied for $\mathscr{S}' \equiv \mathscr{S}_{i+1}$, then this lemma affirms that the result is true for $\mathscr{S}_i$ as we wanted to prove. □

5.2 Completeness

Completeness of the goal solving procedure states that given a program Δ, and a goal G such that $\Delta; R_0 \vdash_{\mathscr{UC}} G$ for a $\mathscr{C}$-satisfiable constraint R_0, there is a resolution of G from Δ with answer constraint R that is entailed by R_0 in the constraint system $\mathscr{C}$, i.e. $R_0 \vdash_{\mathscr{C}} R$. Of course, this entailment means that the computed answer R is at least as general as the given correct answer R_0.

In order to prove this result, we introduce a well-founded ordering which measures the complexity of proving that a given constraint is a correct answer for a given state. The ordering is based on multisets.

Definition 5.4
Let Δ be a program, G a goal, and C, R, constraints such that $\Delta; R, C \vdash_{\mathscr{UC}} G$, then we define $\tau_R(\Delta, C, G)$ as the size of the shortest $\mathscr{UC}$-proof of the sequent $\Delta; R, C \vdash\!\!- G$.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Let $\mathscr{G}$ be a multiset of triples $\langle \Delta, C, G\rangle$. We define $\mathscr{M}_{\mathscr{G}R}$ as the multiset of sizes $\tau_R(\Delta, C, G)$, where the multiplicity of $\tau_R(\Delta, C, G)$ in $\mathscr{M}_{\mathscr{G}R}$ coincides with the multiplicity of $\langle \Delta, C, G\rangle$ in $\mathscr{G}$.

We use the notation $<<$ for the well-founded multiset ordering (Dershowitz and Manna, 1979) induced by the ordering $<$ over the natural numbers.

Next, we show that as long as a state can be transformed, the transformation can be chosen to yield a smaller state with respect to $<<$, while essentially keeping a given answer constraint R.

Lemma 5.4
Let $\mathscr{S} \equiv \Pi[S \square \mathscr{G}]$ be a non-final state w.r.t. a set of variables V, and let R be a constraint such that ΠR is $\mathscr{C}$-satisfiable and $R \vdash_{\mathscr{C}} S$. If $\Delta; R, C \vdash_{\mathscr{UC}} G$ for all $\langle \Delta, C, G\rangle \in \mathscr{G}$, then we can find a rule transforming $\mathscr{S}$ in a state $\mathscr{S}' \equiv \Pi'[S' \square \mathscr{G}']$ ($\mathscr{S} \Vdash \mathscr{S}'$) and a constraint R' such that:

1. $\Pi R \vdash_{\mathscr{C}} \Pi' R'$ and $R' \vdash_{\mathscr{C}} S'$.
2. $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$ for all $\langle \Delta', C', G'\rangle \in \mathscr{G}'$. Moreover $\mathscr{M}_{\mathscr{G}'R'} << \mathscr{M}_{\mathscr{G}R}$.

Proof
By induction on the structure of G, where $\langle \Delta, C, G\rangle \in \mathscr{G}$, analyzing cases. We show here an illustrative case, the proof for the other cases appears in the Appendix.

If G has the form $\exists x G_1$, applying the transformation v) we obtain $\mathscr{S}'$. Let w be the variable used in the substitution involved in this transformation. w does not appear in Π, V, and we can choose it also not free in R. By hypothesis $\Delta; R, C \vdash\!\!- \exists x G_1$ has a proof of size l, then by the definition of $\mathscr{UC}$, there is a constraint formula C_1 such that $\Delta; R, C, C_1 \vdash\!\!- G_1[w/x]$ has a proof of size less than l and $R, C \vdash_{\mathscr{C}} \exists w C_1$. Let $R' \equiv R \wedge (C \Rightarrow C_1)$.

1. $R \vdash_{\mathscr{C}} \exists w(R \wedge (C \Rightarrow C_1))$, since w is not free in R, C, and $R, C \vdash_{\mathscr{C}} \exists w C_1$, therefore $\Pi R \vdash_{\mathscr{C}} \Pi \exists w(R \wedge (C \Rightarrow C_1)) \equiv \Pi' R'$. Moreover, $S' \equiv S$, $R' \vdash_{\mathscr{C}} R$ and $R \vdash_{\mathscr{C}} S$ implies $R' \vdash_{\mathscr{C}} S'$.
2. Let $\langle \Delta', C', G'\rangle \in \mathscr{G}'$. If $\langle \Delta', C', G'\rangle \in \mathscr{G}$, then $\Delta'; R, C' \vdash_{\mathscr{UC}} G'$ by hypothesis, and therefore, using $R' \vdash_{\mathscr{C}} R$ and Lemma 4.9, $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$ and $\tau_{R'}(\Delta', C', G') \leqslant \tau_R(\Delta', C', G')$.

If $\langle \Delta', C', G'\rangle \notin \mathscr{G}$, then $G' \equiv G_1[w/x]$, $\Delta' \equiv \Delta$ and $C' \equiv C$. $\Delta; R', C \vdash\!\!- G_1[w/x]$ will also have a proof of size less than l, since $\Delta; R, C, C_1 \vdash\!\!- G_1[w/x]$ has such a proof, due to $R', C \vdash_{\mathscr{C}} R, C, C_1$ and Lemma 4.9. So $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$ for all $\langle \Delta', C', G'\rangle \in \mathscr{G}'$, $\tau_{R'}(\Delta', C', G') < \tau_R(\Delta, C, G)$, and $\mathscr{M}_{\mathscr{G}'R'} << \mathscr{M}_{\mathscr{G}R}$. □

Theorem 5.5 (*Completeness*)
Let Δ be a program, G a goal and R_0 a $\mathscr{C}$-satisfiable constraint such that $\Delta; R_0 \vdash_{\mathscr{UC}} G$. Then there is a resolution of G from Δ with answer constraint R such that $R_0 \vdash_{\mathscr{C}} R$.

Proof
Using Lemma 5.4, we can build a sequence $\mathscr{S}_0 \Vdash \mathscr{S}_1 \Vdash \ldots \Vdash \mathscr{S}_n$ of state transformations, ($\mathscr{S}_i \equiv \Pi_i[S_i \square \mathscr{G}_i]$, $0 \leqslant i \leqslant n$), that is a a resolution of G from Δ, and a sequence of constraints $R_0, \ldots, R_n$ satisfying that for all i, $1 \leqslant i \leqslant n$:

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

- $R_0 \vdash_{\mathscr{C}} \Pi_i R_i$,
- $R_i \vdash_{\mathscr{C}} S_i$,
- $\Delta'; R_i, C' \vdash_{\mathscr{UC}} G'$, for all $\langle \Delta', C', G' \rangle \in \mathscr{G}_i$.

We use an inductive construction that is guaranteed to terminate thanks to the well-founded ordering $<<$. Let $\mathscr{S}_0 \equiv [\top \square \langle \Delta, \top, G \rangle]$ be the initial state for Δ and G, which we know is not final, if we take R_0 as the constraint given by the theorem's hypothesis, we obtain $R_0 \vdash_{\mathscr{C}} \Pi_0 R_0$ and $R_0 \vdash_{\mathscr{C}} S_0$, since Π_0 is empty and $S_0 \equiv \top$. Moreover, by hypothesis, $\Delta; R_0 \vdash_{\mathscr{UC}} G$ is satisfied, and then also $\Delta; R_0, \top \vdash_{\mathscr{UC}} G$ because of $R_0, \top \vdash_{\mathscr{C}} R_0$ and Lemma 4.9.

Assume the result true for $\mathscr{S}_0, \ldots, \mathscr{S}_i$, if the state $\mathscr{S}_i$ is not final, then $\mathscr{S}_i$ and R_i fulfill the hypothesis of Lemma 5.4, thus there will be a state $\mathscr{S}_{i+1}$ ($\mathscr{S}_i \| \!\!- \mathscr{S}_{i+1}$) and a constraint R_{i+1} such that $R_{i+1} \vdash_{\mathscr{C}} S_{i+1}$ and $\Pi_i R_i \vdash_{\mathscr{C}} \Pi_{i+1} R_{i+1}$ (†) Furthermore, for all $\langle \Delta', C', G' \rangle \in \mathscr{G}_{i+1}$, $\Delta'; R_{i+1}, C' \vdash_{\mathscr{UC}} G'$ and $\mathscr{M}_{\mathscr{G}_{i+1} R_{i+1}} << \mathscr{M}_{\mathscr{G}_i R_i}$. Therefore, by the induction hypothesis, $R_0 \vdash_{\mathscr{C}} \Pi_i R_i$, and with (†) we obtain $R_0 \vdash_{\mathscr{C}} \Pi_{i+1} R_{i+1}$. By successive iteration, as $<<$ is well-founded, we must eventually get a final state $\mathscr{S}_n$ that will in fact satisfy $R_0 \vdash_{\mathscr{C}} \Pi_n R_n$ and $R_n \vdash_{\mathscr{C}} S_n$ and so $R_0 \vdash_{\mathscr{C}} \Pi_n S_n$, where $\Pi_n S_n \equiv R$ is the answer constraint of $\mathscr{S}_0, \ldots, \mathscr{S}_n$. In this way we conclude $R_0 \vdash_{\mathscr{C}} R$. □

For $HH(\mathscr{H})$ programs such that constraints appear neither in the left-hand side of implications nor in initial goals, Theorem 5.5 implies an alternative formulation of the completeness theorem given in Nadathur (1993) for a goal solving procedure for first-order HH. In our opinion, using constraints and constraint satisfiability instead of substitutions and unification under a mixed prefix, that requires low level representation details, we gain a more abstract presentation. For CLP programs, Theorem 5.5 becomes a stronger form of completeness, in comparison to the strong completeness theorem for success given in Maher (1987, Theorem 2) (see also Jaffar *et al.*, 1996, Theorem 4.12). There, assuming $\Delta; R \models_{\mathscr{C}} G$, the conclusion is that $R \vdash_{\mathscr{C}} \bigvee_{i=1}^{m} R_i$ where $R_1, \ldots, R_m$ are answer constraints computed in m different resolutions of G from Δ. Example 5.2 was used in Maher (1987) to illustrate the need of considering disjunctions of computed answers. In fact, there is no single computed answer R_0 such that $R \vdash_{\mathscr{H}} R_0$. However, this fact doesn't contradict Theorem 5.5, because $\Delta; R \| \!\!- G$ is not $\mathscr{UC}$-derivable, as we will see immediately.

Example 5.2
This example is borrowed from Maher (1987). It belongs to the instance $HH(\mathscr{H})$ given by the Herbrand constraint system. Consider

$\Delta \equiv \{D_1, D_2\}$, with $D_1 \equiv p(a, b)$, $D_2 \equiv \forall x (x \not\approx a \Rightarrow p(x, b))$,
$G \equiv p(x, y)$,
$R \equiv y \approx b$.

Up to trivial syntactic variants, this is a $CLP(\mathscr{H})$-program. According to the model theoretic semantics of $CLP(\mathscr{H})$, we get $\Delta; R \models_{\mathscr{H}} G$, because either $x \approx a$ or $x \not\approx a$ will hold in each $\mathscr{H}$-model of $\Delta \cup \{R\}$. In contrast to this, in $\mathscr{UC}$ we only can derive $\Delta; R \wedge x \approx a \| \!\!- G$ (using D_1) and $\Delta; R \wedge x \not\approx a \| \!\!- G$ (using D_2). And it is easy to check that both answers $R \wedge x \approx a$ and $R \wedge x \not\approx a$ can be computed by the goal

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

solving transformations. But we do not obtain $\Delta; R \vdash_{\mathscr{UC}} G$. Since $R \nVdash_{\mathscr{H}} x \approx a$, $R \nVdash_{\mathscr{H}} x \not\approx a$, neither D_1 nor D_2 can be used to build a $\mathscr{UC}$-proof.

The example shows a difference between the model-theoretic semantics used in CLP (Maher, 1987) and our proof-theoretical semantics, based on provability in the calculus $\mathscr{UC}$. The latter deals with the logical symbols in goals and clauses according to the inference rules of intuitionistic logic. Therefore, $\mathscr{UC}$-provability turns out to be more constructive than CLP's model-theoretic semantics, and thus closer to constrained resolution. This is the ultimate reason why our completeness Theorem 5.5 involves no disjunction of computed answers.

As an illustration of the goal solving procedure, we show next the detailed resolution of the second goal from Example 2.3.

Example 5.3

Let us recall the program and goal from Example 2.3. As usual in programming practice, we write program clauses $\forall x_1 \ldots \forall x_n (G \Rightarrow A)$ in the form $A \Leftarrow G$[3].

$$\begin{aligned}\Delta \equiv \{\ & mortgage(P,T,I,M,B) \Leftarrow 0 \leqslant T \wedge T \leqslant 3 \wedge \\ & \qquad TotalInt \approx T * (P * I/1200) \wedge B \approx P + TotalInt - (T * M), \\ & mortgage(P,T,I,M,B) \Leftarrow T > 3 \wedge QuartInt \approx 3 * (P * I/1200) \wedge \\ & \qquad mortgage(P + QuartInt - 3 * M, T - 3, I, M, B) \\ \}& \\ G \equiv \forall M \forall P (0.9637 \leqslant P/(6 * M) \leqslant 0.97 \Rightarrow & \\ \exists I (0 \leqslant Imin \leqslant I \leqslant Imax \ \wedge & mortgage(P,6,I,M,0))).\end{aligned}$$

We present a resolution of G from Δ, using the state transformation rules (1)–(8) from Definition 5.2:

$$\begin{aligned}& [\top \Box \langle \Delta, \top, G \rangle] \\ & \quad \Vdash_{(6)} \\ & \quad \forall M \forall P [\top \Box \langle \Delta, \top, 0.9637 \leqslant P/(6 * M) \leqslant 0.97 \Rightarrow \exists I (0 \leqslant Imin \leqslant I \leqslant Imax \wedge \\ & \qquad\qquad mortgage(P,6,I,M,0)) \rangle] \\ & \quad \Vdash_{(4)} \\ & \quad \forall M \forall P [\top \Box \langle \Delta, 0.9637 \leqslant P/(6 * M) \leqslant 0.97, \\ & \qquad\qquad \exists I (0 \leqslant Imin \leqslant I \leqslant Imax \wedge mortgage(P,6,I,M,0)) \rangle] \\ & \quad \Vdash_{(5)} \\ & \quad \forall M \forall P \exists I [\top \Box \ \langle \Delta, 0.9637 \leqslant P/(6 * M) \leqslant 0.97, \\ & \qquad\qquad 0 \leqslant Imin \leqslant I \leqslant Imax \wedge mortgage(P,6,I,M,0) \rangle] \\ & \quad \Vdash_{(1),(8)} \\ & \quad \forall M \forall P \exists I [0.9637 \leqslant P/(6 * M) \leqslant 0.97 \Rightarrow 0 \leqslant Imin \leqslant I \leqslant Imax \Box \\ & \qquad\qquad \langle \Delta, 0.9637 \leqslant P/(6 * M) \leqslant 0.97, mortgage(P,6,I,M,0) \rangle] \\ & \quad \Vdash_{(8)} \\ & \quad \forall M \forall P \exists I [0.9637 \leqslant P/(6 * M) \leqslant 0.97 \Rightarrow 0 \leqslant Imin \leqslant I \leqslant Imax \Box \\ & \qquad\qquad \langle \Delta, 0.9637 \leqslant P/(6 * M) \leqslant 0.97, \\ & \qquad\qquad \underbrace{\exists P' \exists T' \exists I' \exists M' \exists B' \exists QuartInt (P \approx P' \wedge 6 \approx T' \wedge I \approx I'}\end{aligned}$$

[3] In fact, we have already followed this convention in Section 2.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

$$\underbrace{\wedge M \approx M' \wedge 0 \approx B' \wedge T' > 3 \wedge QuartInt \approx 3*(P'*I'/1200)}$$
$$\underbrace{\wedge mortgage(P'+QuartInt-3*M', T'-3, I', M', B'))}\rangle]$$

Simplifying the underbraced formula in the constraint system $\mathscr{R}$, we obtain:

$$\forall M \forall P \exists I\,[0.9637 \leqslant P/(6*M) \leqslant 0.97 \Rightarrow 0 \leqslant Imin \leqslant I \leqslant Imax \Box$$
$$\langle \Delta, 0.9637 \leqslant P/(6*M) \leqslant 0.97,$$
$$mortgage(P + 3*(P*I/1200) - 3*M, 3, I, M, 0)\rangle]$$
$$\Vdash_{(8)}$$
$$\forall M \forall P \exists I\,[0.9637 \leqslant P/(6*M) \leqslant 0.97 \Rightarrow 0 \leqslant Imin \leqslant I \leqslant Imax \Box$$
$$\langle \Delta, 0.9637 \leqslant P/(6*M) \leqslant 0.97,$$
$$\underbrace{\exists P'' \exists T'' \exists I'' \exists M'' \exists B'' \exists TotalInt\,(P'' \approx P + 3*(P*I/1200) - 3*M}$$
$$\underbrace{\wedge\ T'' \approx 3 \wedge I'' \approx I \wedge M'' \approx M \wedge B'' \approx 0 \wedge 0 \leqslant T'' \wedge T'' \leqslant 3}$$
$$\underbrace{\wedge\ TotalInt \approx T'' * (P'' * I''/1200) \wedge B'' \approx P'' + TotalInt - (T'' * M''))}\rangle]$$

Simplifying anew the underbraced formula in $\mathscr{R}$:

$$\forall M \forall P \exists I\,[0.9637 \leqslant P/(6*M) \leqslant 0.97 \Rightarrow 0 \leqslant Imin \leqslant I \leqslant Imax \Box$$
$$\langle \Delta, 0.9637 \leqslant P/(6*M) \leqslant 0.97,$$
$$0 \approx P + 3*(P*I/1200) - 3*M +$$
$$3*(P + 3*(P*I/1200) - 3*M)*I/1200 - 3*M\rangle]$$

Applying now transformation (7), we obtain the following answer constraint:

$$\forall M \forall P \exists I((0.9637 \leqslant P/(6*M) \leqslant 0.97 \Rightarrow 0 \leqslant Imin \leqslant I \leqslant Imax)) \wedge$$
$$(0.9637 \leqslant P/(6*M) \leqslant 0.97 \Rightarrow 0 \approx P + 3*P*I/1200 - 3*M +$$
$$3*(P + 3*P*I/1200 - 3*M)*I/1200 - 3*M))$$
$$\vdash_{\mathscr{R}}$$
$$\forall M \forall P \exists I\,\big(0.9637 \leqslant P/(6*M) \leqslant 0.97 \Rightarrow 0 \leqslant Imin \leqslant I \leqslant Imax \wedge$$
$$0 \approx P * \big(1 + 3*\tfrac{I}{1200} + 3*\tfrac{I}{1200} + 9*\tfrac{I^2}{1200^2}\big) - M*\big(6 + 9*\tfrac{I}{1200}\big)\big)$$
$$\vdash_{\mathscr{R}}$$
$$\forall M \forall P \exists I\,\big(0.9637 \leqslant P/(6*M) \leqslant 0.97 \Rightarrow 0 \leqslant Imin \leqslant I \leqslant Imax \wedge$$
$$0 \approx P * \big(1 + \tfrac{I}{200} + \tfrac{I^2}{400^2}\big) - M*\big(6 + 3*\tfrac{I}{400}\big)\big)$$
$$\vdash_{\mathscr{R}}$$
$$\forall M \forall P \exists I\,\big(0.9637 \leqslant P/(6*M) \leqslant 0.97 \Rightarrow 0 \leqslant Imin \leqslant I \leqslant Imax \wedge$$
$$\tfrac{P}{6*M} \approx \tfrac{1+\frac{I}{800}}{1+\frac{I}{200}+\frac{I^2}{400^2}}\big) \equiv C_1$$

We prove $C_1 \vdash_{\mathscr{R}} Imin \approx 8.219559$ (approx.) $\wedge\, Imax \approx 10$. In effect, let

$$f(I) = \frac{1 + \frac{I}{800}}{1 + \frac{I}{200} + \frac{I^2}{400^2}},$$

we observe that $f(I)$ is a strictly decreasing continuous function of I for any $I \geqslant 0$,

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

and also that

$$f(I) \approx 0.9637(\text{approx.}) \vdash_{\mathscr{R}} I \approx 10, \text{ and}$$

$$f(I) \approx 0.97 \vdash_{\mathscr{R}} I \approx 8.219559 \text{ (approx.)}.$$

Then, C_1 is true iff for any M and P such that

$$P/(6 * M) \in [0.97..0.9637 \text{ (approx.)}],$$

there exists $I \in [Imax..Imin]$ such that $f(I) \approx P/(6 * M)$ (f strictly decreasing continuous function), and this is true iff I has its maximum value for $f(I) \approx 0.9637$ (approx.) and its minimum for $f(I) \approx 0.97$, or equivalently $Imax \approx 10 \wedge Imin \approx 8.219559$ (approx.).

6 Conclusions and future work

We have proposed a novel combination of Constraint Logic Programming (CLP) with first-order Hereditary Harrop Formulas (HH). Our framework includes a proof system with the uniform proofs property and a sound and complete goal solving procedure. Our results are parametric w.r.t. a given constraint system $\mathscr{C}$, and they can be related to previously known results for CLP and HH. Therefore, we can speak of a scheme whose expressivity sums the advantages of CLP and HH.

As far as we know, our work is the first attempt to combine the full expressivity of HH and CLP. A related, but more limited approach, can be found in Darlington and Guo (1994). This paper presents an amalgamated logic that combines the Horn fragment of intuitionistic logic with the entailment relation of a given constraint system, showing the existence of uniform proofs as well as soundness and completeness of constrained *SLD* resolution w.r.t. the proof system. The more general case of HH is not studied. Moreover, the presentation of constrained *SLD* resolution is not fully satisfactory, because the *backchaining* transition rule (see Darlington and Guo, 1994), guesses an arbitrary instance of a program clause, instead of adding unification constraints to the new goal, as done in our state transition rule (8).

Several interesting issues remain for future research. First, more concrete evidence on potential application areas should be found. We are currently looking for CLP applications where greater HH expressivity may be useful, as well as for typical HH applications that can benefit from the use of numeric and/or symbolic constraints. Secondly, tractable fragments of our formalism (other than CLP and HH separately) should be discovered. Otherwise, constraint satisfiability and constraint entailment may become intractable or even undecidable. Our broad notion of constraint system includes any first-order theory based on arbitrary equational axiomatization. Such theories are sometimes decidable (see Comon (1993) and Comon, Haberstrau and Jouannaud (1994)), but most often restricted fragments must be chosen to ensure decidability. Last but not least, our framework should be extended to higher-order HH as used in many λ-Prolog applications.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Acknowledgements

We are grateful to the anonymous referees for their constructive criticisms.

Appendix

Proofs of results from Section 4.1

Lemma 4.1
For any Δ, Γ, G, x and t, if $\Delta; \Gamma \vdash_{\mathscr{IC}} G$, then there is a proof of the same size of $\Delta[t/x]; \Gamma[t/x] \vdash G[t/x]$.

Proof
By induction on the size l of the proof of the sequent $\Delta; \Gamma \vdash G$.

If $l = 1$, then (C_R) or $(Atom)$ have been applied. In the first case, $G \equiv C$ for some constraint C and $\Gamma \vdash_{\mathscr{C}} C$. Hence $\Gamma[t/x] \vdash_{\mathscr{C}} C[t/x]$, by the properties of $\vdash_{\mathscr{C}}$. Therefore the sequent $\Delta[t/x]; \Gamma[t/x] \vdash C[t/x]$ has a proof of size 1, by applying (C_R). In the second case, $G \equiv A$, for some predicate formula A, $\Delta = \Delta' \cup \{A'\}$, with A' beginning with the same predicate symbol as A, and $\Gamma \vdash_{\mathscr{C}} A' \approx A$. Hence $\Gamma[t/x] \vdash_{\mathscr{C}} (A' \approx A)[t/x]$. Therefore, applying $(Atom)$, $\Delta'[t/x], A'[t/x]; \Gamma[t/x] \vdash A[t/x]$ has a proof of size 1, and $\Delta[t/x] = \Delta'[t/x] \cup \{A'[t/x]\}$.

If $l > 1$, we distinguish cases in accordance with the last rule applied in the deduction of $\Delta; \Gamma \vdash G$. Let us analyze some cases (the omitted ones are similar).

$(\Rightarrow C_R)$ In this case $G \equiv C \Rightarrow G'$, and the last step of the proof has the form:

$$\frac{\Delta; \Gamma, C \vdash G'}{\Delta; \Gamma \vdash C \Rightarrow G'} \ (\Rightarrow C_R)$$

By the induction hypothesis, $\Delta[t/x]; \Gamma[t/x], C[t/x] \vdash G'[t/x]$ has a proof of size $l - 1$. Then, applying $(\Rightarrow C_R)$, we obtain that $\Delta[t/x]; \Gamma[t/x] \vdash (C \Rightarrow G')[t/x]$ has a proof of size l.

$(\forall_R)$ In this case $G \equiv \forall z G'$ and the last step of the proof has the form:

$$\frac{\Delta; \Gamma \vdash G'[y/z]}{\Delta; \Gamma \vdash \forall z G'} \ (\forall_R)$$

where y does not appear free in the sequent of the conclusion. We can assume, without loss of generality, that $z \neq x$ and z does not appear in t. If this were not the case, the induction hypothesis could be applied another time, in order to rename coincident variables. Also we can assume that y is different from x and that y does not occur in t. By the induction hypothesis, $\Delta[t/x]; \Gamma[t/x] \vdash G'[t/x][y/z]$ has a proof of size $l - 1$, because under our hypothesis, $G'[y/z][t/x] \equiv G'[t/x][y/z]$. Now, applying $(\forall_R)$, $\Delta[t/x]; \Gamma[t/x] \vdash \forall z(G'[t/x])$ has a proof of size l, but this is the expected result because $\forall z(G'[t/x]) \equiv (\forall z G')[t/x]$.

$(\forall_L)$ In this case $\Delta = \Delta' \cup \{\forall z D\}$. As before, we can assume that $z \neq x$ and does not appear in t, and the last step of the proof has the form:

$$\frac{\Delta', D[y/z]; \Gamma, C \vdash G \qquad \Gamma \vdash_{\mathscr{C}} \exists y C}{\Delta, \forall z D; \Gamma \vdash G} \ (\forall_L)$$

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

where y does not appear free in the sequent of the conclusion. We can assume without loss of generality that y is different from x and that y does not occur in t. Then, by the induction hypothesis,

$$\Delta'[t/x], D[t/x][y/z]; \Gamma[t/x], C[t/x] \vdash G[t/x] \quad (\dagger)$$

has a proof of size $l-1$, because under our hypothesis, $D[y/z][t/x] \equiv D[t/x][y/z]$. Now $\Gamma \vdash_{\mathscr{C}} \exists y C$ implies

$$\Gamma[t/x] \vdash_{\mathscr{C}} \exists y(C[t/x]) \quad (\ddagger),$$

by the properties of $\vdash_{\mathscr{C}}$ and the fact that $(\exists y C)[t/x] \equiv \exists y(C[t/x])$. Then applying $(\forall_L)$ to $(\dagger)$ and $(\ddagger)$, $\Delta[t/x]; \Gamma[t/x] \vdash G[t/x]$ has a proof of size l, because $\forall z(D[t/x]) \equiv (\forall z D)[t/x]$ and $\Delta[t/x] = \Delta'[t/x] \cup \{(\forall z D)[t/x]\}$. □

Lemma 4.2
For any Δ, Γ, G, if Γ' is a set of constraints such that $\Gamma' \vdash_{\mathscr{C}} \Gamma$, and $\Delta; \Gamma \vdash_{\mathscr{JC}} G$, then $\Delta; \Gamma' \vdash G$ has a proof of the same size.

Proof
By induction on the size of the proof of the sequent $\Delta; \Gamma \vdash G$, by case analysis on the last rule applied, and using the properties of entailment in constraint systems. It is obvious for proofs of size 1. For proofs of size $l > 1$, let us analyse the case $(\forall_L)$ (the others are similar). In this case, the last step of the proof is of the form:

$$\frac{\Delta', D[y/x]; \Gamma, C \vdash G \qquad \Gamma \vdash_{\mathscr{C}} \exists y C}{\Delta', \forall x D; \Gamma \vdash G} \ (\forall_L)$$

where y does not appear free in the sequent of the conclusion, and $\Delta = \Delta' \cup \{\forall x D\}$. By the induction hypothesis

$$\Delta', D[y/x]; \Gamma', C \vdash G \quad (\dagger)$$

has a proof of size $l-1$. We know that $\Gamma \vdash_{\mathscr{C}} \exists y C$, and by the hypothesis $\Gamma' \vdash_{\mathscr{C}} \Gamma$, so

$$\Gamma' \vdash_{\mathscr{C}} \exists y C \quad (\ddagger).$$

We can assume that y does not appear free in Γ', in other case, by Lemma 4.1, we can work with $\Delta', D[y'/x]; \Gamma', C[y'/y] \vdash G$ (y' new), instead of $(\dagger)$, and with $\Gamma' \vdash_{\mathscr{C}} \exists y' C[y'/y]$, instead of $(\ddagger)$, by the properties of $\vdash_{\mathscr{C}}$. Then we finish by applying $(\forall_L)$ to $(\dagger)$ and $(\ddagger)$. □

Lemma 4.4
For any Δ, Γ, C, G, if $\Delta; \Gamma, C \vdash_{\mathscr{JC}} G$ and x is a variable that does not appear free in Δ, Γ, G, then $\Delta; \Gamma, \exists x C \vdash G$ has a proof of the same size.

Proof
By induction on the size of the proof. We will assume that x appears free in C, if not $\exists x C \vdash_{\mathscr{C}} C$, and the proof is immediate due to Lemma 4.2.

If $\Delta; \Gamma, C \vdash G$ has a proof of size 1, (*Atom*) or (C_R) has been applied. In both cases $\Gamma, C \vdash_{\mathscr{C}} C'$ for certain constraint C'. Both C' and Γ do not contain free

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

occurrences of x, hence $\Gamma, \exists x C \vdash_{\mathscr{C}} C'$, and therefore $\Delta; \Gamma, \exists x C \vdash\!\!- G$ has a proof of size 1. If $\Delta; \Gamma, C \vdash\!\!- G$ has a proof of size $l > 1$, let us discuss some of the possible cases.

($\exists_R$) Then $G \equiv \exists z G'$ and the last step of the proof is of the form:

$$\frac{\Delta; \Gamma, C, C' \vdash\!\!- G'[y/z] \qquad \Gamma, C \vdash_{\mathscr{C}} \exists y C'}{\Delta; \Gamma, C \vdash\!\!- \exists z G'} \ (\exists_R)$$

where y does not appear free in the sequent of the conclusion. Hence, by Lemma 4.2, $\Delta; \Gamma, C \wedge C' \vdash\!\!- G'[y/z]$ has a proof of size $l - 1$. Now, the conditions on y imply that $x \neq y$, so x is not free in $G'[y/z]$, because it is not free in $\exists z G'$. Then, by the induction hypothesis and again using Lemma 4.2,

$$\Delta; \Gamma, \exists x C, \exists x (C \wedge C') \vdash\!\!- G'[y/z] \quad (\dagger)$$

has a proof of size $l - 1$. On the other hand, $\Gamma, C \vdash_{\mathscr{C}} \exists y C'$ implies that $\Gamma, C \vdash_{\mathscr{C}} C \wedge \exists y C'$ so $\Gamma, \exists x C \vdash_{\mathscr{C}} \exists x (C \wedge \exists y C')$, since x is not free in Γ, thus

$$\Gamma, \exists x C \vdash_{\mathscr{C}} \exists y \exists x (C \wedge C') \quad (\ddagger),$$

since y is not free in C. Therefore the desired result is obtained by applying ($\exists_R$) to (†) and (‡).

($\forall_R$) Then $G \equiv \forall z G'$, and the last step of the proof has the form:

$$\frac{\Delta; \Gamma, C \vdash\!\!- G'[y/z]}{\Delta; \Gamma, C \vdash\!\!- \forall z G'} \ (\forall_R)$$

where y does not appear free in the sequent of the conclusion. Then y does not occur free in C, so it is different from x. Applying the induction hypothesis to the sequent $\Delta; \Gamma, C \vdash\!\!- G'[y/z]$, we obtain that $\Delta; \Gamma, \exists x C \vdash\!\!- G'[y/z]$ has a proof of size $l - 1$. Then $\Delta; \Gamma, \exists x C \vdash\!\!- G$ has a proof of size l by ($\forall_R$). □

Proofs of results from Section 4.2

Lemma 4.5 (*Proof transformation*)
If G is a goal, Δ a program and Γ a set of constraint formulas, such that $\Delta; \Gamma \vdash\!\!- G$ has a proof of size l, then:

1. For $G \equiv A$, there are n constraint formulas $C_1, \ldots, C_n$ ($n \geqslant 0$) and a formula $\forall x_1 \ldots \forall x_n \ (G' \Rightarrow A')$ that is a variant of some formula in $elab(\Delta)$ such that $x_1, \ldots, x_n$ are new distinct variables not appearing free in Δ, Γ, A, where x_i does not appear free in $C_1, \ldots, C_{i-1}$, for $1 < i \leqslant n$, and A' begins with the same predicate symbol as A. In addition it holds:
   (a) $\Gamma \vdash_{\mathscr{C}} \exists x_1 C_1; \quad \Gamma, C_1 \vdash_{\mathscr{C}} \exists x_2 C_2; \ldots; \quad \Gamma, C_1, \ldots, C_{n-1} \vdash_{\mathscr{C}} \exists x_n C_n.$
   (b) $\Gamma, C_1, \ldots, C_n \vdash_{\mathscr{C}} A' \approx A.$
   (c) $\Delta; \Gamma, C_1, \ldots, C_n \vdash\!\!- G'$ has a proof of size less than l, or $G' \equiv \top$.
2. If $G \equiv C$, then $\Gamma \vdash_{\mathscr{C}} C$.
3. If $G \equiv G_1 \wedge G_2$, then $\Delta; \Gamma \vdash\!\!- G_1$ and $\Delta; \Gamma \vdash\!\!- G_2$ have proofs of size less than l.
4. If $G \equiv G_1 \vee G_2$, then $\Delta; \Gamma \vdash\!\!- G_i$ has a proof of size less than l for $i = 1$ or 2.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

5. If $G \equiv D \Rightarrow G_1$, then $\Delta, D; \Gamma \vdash G_1$ has a proof of size less than l.
6. If $G \equiv C \Rightarrow G_1$, then $\Delta; \Gamma, C \vdash G_1$ has a proof of size less than l.
7. For $G \equiv \exists x G_1$, if y is a variable not appearing free in Δ, Γ, G, then there is a constraint formula C such that:
   (a) $\Gamma \vdash_{\mathscr{C}} \exists y C$.
   (b) $\Delta; \Gamma, C \vdash G_1[y/x]$ has a proof of size less than l.
8. If $G \equiv \forall x G_1$, then $\Delta; \Gamma \vdash G_1[y/x]$ has a proof of size less than l, where y is a variable that does not appear free in Δ, Γ, G.

Proof

We reason by induction on the size l of a given $\mathscr{IC}$-proof of $\Delta; \Gamma \vdash G$.

If l is 1, then G has been proved by a single application of axiom (C_R) or axiom (*Atom*). In the former case, G is a constraint and item 2 of the lemma holds. In the latter case G is an atomic formula A and there is $A' \in \Delta$, beginning with the same predicate symbol that A such that $\Gamma \vdash_{\mathscr{C}} A' \approx A$. But $A' \in \Delta$ implies $\top \Rightarrow A' \in elab(\Delta)$, then conditions (a), (b) and (c) of item 1 are satisfied with $n = 0$, $G' \equiv \top$.

If $l > 1$, let us analyse cases according to the last inference rule applied in the proof of the sequent $\Delta; \Gamma \vdash G$. The lemma is obviously true by induction hypothesis if the last inference rule introduces on the right the main connective or quantifier of the goal. So the problem is reduced to the rules $(\wedge_L), (\Rightarrow_L)$ and $(\forall_L)$. For each of these three rules, we must analyse cases according to the structure of G. In each case, it is possible to transform the proof by permuting the application of right and left-introduction rules, in the same way as in Miller *et al.* (1991). In our setting, however, the treatment of $(\forall_L)$ gives rise to some new situations. We analyse the most interesting cases; those we omit can be treated analogously.

$(\wedge_L)$ Then we can decompose Δ as $\Delta = \Delta' \cup \{D_1 \wedge D_2\}$, and the last step of the proof is of the form:

$$\frac{\Delta', D_1, D_2; \Gamma \vdash G}{\Delta', D_1 \wedge D_2; \Gamma \vdash G}\ (\wedge_L)$$

- If $G \equiv G_1 \vee G_2$, then by the induction hypothesis, there is a proof of size less than $l - 1$ of $\Delta', D_1, D_2; \Gamma \vdash G_i$. Applying $(\wedge_L)$ we obtain a proof of size less or equal $l - 1$, so less than l, of $\Delta', D_1 \wedge D_2; \Gamma \vdash G_i$ for $i = 1$ or 2.

$(\Rightarrow_L)$ Then we can decompose Δ as $\Delta = \Delta' \cup \{G' \Rightarrow A\}$, and the last step of the proof is of the form:

$$\frac{\Delta'; \Gamma \vdash G' \quad \Delta', A; \Gamma \vdash G}{\Delta', G' \Rightarrow A; \Gamma \vdash G}\ (\Rightarrow_L)$$

- If $G \equiv \forall x G_1$, then $\Delta', A; \Gamma \vdash \forall x G_1$ has a proof of size $l_1 < l$, and by the induction hypothesis there is a proof of size less than l_1 of $\Delta', A; \Gamma \vdash G_1[y/x]$, where y is a new variable. Then, using that $\Delta'; \Gamma \vdash G'$ has a proof of size l_2, $l_1 + l_2 = l - 1$, and applying $(\Rightarrow_L)$, $\Delta', G' \Rightarrow A; \Gamma \vdash G_1[y/x]$ has a proof of size less or equal $l_1 + l_2$ so less than l, as we wanted to prove.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

- If $G \equiv D \Rightarrow G_1$, then $\Delta', A; \Gamma \vdash D \Rightarrow G_1$ has a proof of size $l_1 < l$, so by the induction hypothesis there is a proof of size less than l_1 of $\Delta', A, D; \Gamma \vdash G_1$. Then, since $\Delta'; \Gamma \vdash G'$ has a proof of size l_2, obviously $\Delta', D; \Gamma \vdash G'$ also has a proof of size l_2, and $l_1 + l_2 < l$. Therefore, using $(\Rightarrow_L)$, we obtain that $\Delta', G' \Rightarrow A, D; \Gamma \vdash G_1$ has a proof of size less or equal $l_1 + l_2$, so less than l, as we wanted to prove.

$(\forall_L)$ Then we can decompose Δ as $\Delta = \Delta' \cup \{\forall x D\}$, and the last step of the proof is of the form:

$$\frac{\Delta', D[y/x]; \Gamma, C' \vdash G \qquad \Gamma \vdash_{\mathscr{C}} \exists y C'}{\Delta', \forall x D; \Gamma \vdash G} \ (\forall_L)$$

where y is not free in the sequent of the conclusion, and the sequent

$$Q \equiv \Delta', D[y/x]; \Gamma, C' \vdash G$$

has a proof of size $l - 1$.

- If $G \equiv C$, then by the induction hypothesis applied to Q, we know that $\Gamma, C' \vdash_{\mathscr{C}} C$. Since $\Gamma \vdash_{\mathscr{C}} \exists y C'$ and y is not free in Γ, C, we conclude that $\Gamma \vdash_{\mathscr{C}} C$, due to the properties of $\vdash_{\mathscr{C}}$, that coincides with item 2 of the lemma.
- If $G \equiv C \Rightarrow G_1$, then by the induction hypothesis applied to Q, the sequent

  $$\Delta', D[y/x]; \Gamma, C', C \vdash G_1$$

  has a proof of size less than $l - 1$. Therefore, since $\Gamma \vdash_{\mathscr{C}} \exists y C'$ implies $\Gamma, C \vdash_{\mathscr{C}} \exists y C'$, and y is not free in C, applying $(\forall_L)$, $\Delta', \forall x D; \Gamma, C \vdash G_1$, has a proof of size less or equal than $l - 1$ so less than l.
- If $G \equiv \exists w G_1$, then by applying the induction hypothesis to Q we conclude that there is C such that $\Gamma, C' \vdash_{\mathscr{C}} \exists z C$, where z is not free in $\Delta', D[y/x], \Gamma, C', \exists w G_1$, and

  $$\Delta', D[y/x]; \Gamma, C', C \vdash G_1[z/w] \quad (\dagger)$$

  has a proof of size less than $l - 1$. Since y is not free in Δ', $G_1[z/w]$, applying Corollary 4.3 to (†) we obtain that $\Delta', D[u/x]; \Gamma, C', C, u \approx y \vdash G_1[z/w]$, where u is a new variable, has a proof of the same size, so by Lemma 4.2,

  $$\Delta', D[u/x]; \Gamma, C' \wedge C, u \approx y \vdash G_1[z/w] \quad (\ddagger)$$

  still with a proof of size less than $l - 1$. Now by the properties of the constraint entailment, $\Gamma, C' \wedge C \vdash_{\mathscr{C}} \exists u (u \approx y)$ (§). Then, since u is not free in $\Delta', \forall x D, \Gamma, C' \wedge C, G_1[z/w]$, we apply $(\forall_L)$ to (‡) and (§), obtaining that

  $$\Delta', \forall x D; \Gamma, C' \wedge C \vdash G_1[z/w]$$

  has a proof of size less than or equal $l - 1$. Hence using Lemma 4.4

  $$\Delta', \forall x D; \Gamma, \exists y (C' \wedge C) \vdash G_1[z/w]$$

  has a proof of size less than or equal $l - 1$, because, by the assumptions, y is not free in $\Delta', \forall x D, \Gamma, G_1[z/w]$. Therefore we can conclude the result for this case (item 7), taking $\exists y (C' \wedge C)$ as auxiliary constraint. In fact, $\Gamma, C' \vdash_{\mathscr{C}} \exists z C$ implies

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

$\Gamma, C' \vdash_{\mathscr{C}} \exists z(C' \wedge C)$, since z is not free in Γ, C'. Hence $\Gamma, \exists y C' \vdash_{\mathscr{C}} \exists z \exists y(C' \wedge C)$, since y is not free in Γ. Finally, $\Gamma \vdash_{\mathscr{C}} \exists z \exists y(C' \wedge C)$, because $\Gamma \vdash_{\mathscr{C}} \exists y C'$.

- If $G \equiv A$, then the induction hypothesis for the sequent Q assures that there are constraints $C_1, \ldots, C_n$ $(n \geqslant 0)$ and a formula $\forall x_1 \ldots \forall x_n(G' \Rightarrow A')$ that is a variant of a formula in $elab(\Delta' \cup \{D[y/x]\})$, where $x_1, \ldots, x_n$ are new variables, x_i not appearing free in $C_1, \ldots, C_{i-1}$, for $1 < i \leqslant n$, A' begins with the same predicate symbol as A, and such that:

(i) $\Gamma, C' \vdash_{\mathscr{C}} \exists x_1 C_1$; $\Gamma, C', C_1 \vdash_{\mathscr{C}} \exists x_2 C_2; \ldots$; $\Gamma, C', C_1, \ldots, C_{n-1} \vdash_{\mathscr{C}} \exists x_n C_n$.

(ii) $\Gamma, C', C_1, \ldots, C_n \vdash_{\mathscr{C}} A' \approx A$.

(iii) $\Delta', D[y/x]; \Gamma, C', C_1, \ldots, C_n \vdash G'$ has a proof of size less than $l - 1$, or $G' \equiv \top$.

In order to establish item 1 of the lemma, we distinguish two cases:

(I) $\forall x_1 \ldots \forall x_n(G' \Rightarrow A')$ is a variant of a formula in $elab(\Delta')$, or

(II) $\forall x_1 \ldots \forall x_n(G' \Rightarrow A')$ is a variant of a formula in $elab(D[y/x])$.

(I). If $\forall x_1 \ldots \forall x_n(G' \Rightarrow A')$ is a variant of a formula in $elab(\Delta')$, then $\forall x_1 \ldots \forall x_n(G' \Rightarrow A')$ is a variant of a formula in $elab(\Delta)$. Taking the following n auxiliary constraints $\exists y(C' \wedge C_1), \ldots, \exists y(C' \wedge C_1 \wedge \ldots \wedge C_n)$, we will prove conditions (a), (b) and (c).

- For condition (a) we need to prove:

$$\Gamma \vdash_{\mathscr{C}} \exists x_1 \exists y(C' \wedge C_1) \qquad (1)$$

$$\Gamma, \exists y(C' \wedge C_1) \vdash_{\mathscr{C}} \exists x_2 \exists y(C' \wedge C_1 \wedge C_2) \qquad (2)$$

$$\vdots \qquad \vdots$$

$$\Gamma, \exists y(C' \wedge C_1), \ldots, \exists y(C' \wedge C_1 \wedge \ldots \wedge C_{n-1}) \vdash_{\mathscr{C}} \exists x_n \exists y(C' \wedge C_1 \wedge \ldots \wedge C_n) \qquad (n)$$

This can be deduced from condition (i) above, as follows:

(1). By (i), $\Gamma, C' \vdash_{\mathscr{C}} \exists x_1 C_1$, then $\Gamma, C' \vdash_{\mathscr{C}} C' \wedge \exists x_1 C_1$. Hence

$$\Gamma, \exists y C' \vdash_{\mathscr{C}} \exists y(C' \wedge \exists x_1 C_1),$$

since y is not free in Γ. Therefore

$$\Gamma, \exists y C' \vdash_{\mathscr{C}} \exists x_1 \exists y(C' \wedge C_1),$$

since x_1 is not free in C'. Now we can conclude (1) because $\Gamma \vdash_{\mathscr{C}} \exists y C'$.

(2). By (i), $\Gamma, C', C_1 \vdash_{\mathscr{C}} \exists x_2 C_2$, then $\Gamma, C' \wedge C_1 \vdash_{\mathscr{C}} C' \wedge C_1 \wedge \exists x_2 C_2$. Hence

$$\Gamma, \exists y(C' \wedge C_1) \vdash_{\mathscr{C}} \exists y(C' \wedge C_1 \wedge \exists x_2 C_2),$$

since y is not free in Γ. Therefore

$$\Gamma, \exists y(C' \wedge C_1) \vdash_{\mathscr{C}} \exists x_2 \exists y(C' \wedge C_1 \wedge C_2),$$

since x_2 is not free in C', C_1.

By a similar reasoning, we can prove (3) to $(n-1)$.

(n). By (i), $\Gamma, C', C_1, \ldots, C_{n-1} \vdash_{\mathscr{C}} \exists x_n C_n$, then $\Gamma, C' \wedge C_1 \wedge \ldots \wedge C_{n-1} \vdash_{\mathscr{C}} C' \wedge C_1 \wedge \ldots \wedge C_{n-1} \wedge \exists x_n C_n$. Hence

$$\Gamma, \exists y(C' \wedge C_1 \wedge \ldots \wedge C_{n-1}) \vdash_{\mathscr{C}} \exists y(C' \wedge C_1 \wedge \ldots \wedge C_{n-1} \wedge \exists x_n C_n),$$

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

since y is not free in Γ. Therefore

$$\Gamma, \exists y(C' \wedge C_1 \wedge \ldots \wedge C_{n-1}) \vdash_{\mathscr{C}} \exists x_n \exists y(C' \wedge C_1 \wedge \ldots \wedge C_{n-1} \wedge C_n),$$

since x_n is not free in $C', C_1, \ldots, C_{n-1}$. Then we deduce (n) obviously.

- For condition (b) we need:

$$\Gamma, \exists y(C' \wedge C_1), \ldots, \exists y(C' \wedge C_1 \wedge \ldots \wedge C_n) \vdash_{\mathscr{C}} A' \approx A.$$

To deduce this from (ii), we note that y is not free in Δ', Γ, A by assumption. Moreover, y is not free in A', or else it would be free in Δ'. Therefore, (ii) implies that

$$\Gamma, \exists y(C' \wedge C_1 \wedge \ldots \wedge C_n) \vdash_{\mathscr{C}} A' \approx A,$$

which amounts to what we needed.

- Finally, for condition (c) we assume the interesting case where G' is not $\top$. We need a proof of size less than l for the sequent

$$\Delta', \forall xD; \Gamma, \exists y(C' \wedge C_1), \ldots, \exists y(C' \wedge C_1 \wedge \ldots \wedge C_n) \vdash\!\!\!- G' \quad (\dagger)$$

To deduce this, we first choose a fresh variable u, and we apply Corollary 4.3 to (iii), thus obtaining that

$$\Delta', D[u/x]; \Gamma, C', C_1, \ldots, C_n, u \approx y \vdash\!\!\!- G'$$

has a proof of size less than $l - 1$. Since u is new and $\Gamma, C', C_1, \ldots, C_n \vdash_{\mathscr{C}} \exists u(u \approx y)$, we can apply $(\forall_L)$ obtaining that

$$\Delta', \forall xD; \Gamma, C', C_1, \ldots, C_n \vdash\!\!\!- G'$$

has a proof of size less than l. From this, Lemma 4.2 and Lemma 4.4 (note that y is not free in $\Delta', \forall xD, \Gamma, G'$) lead to a proof of size less than l for

$$\Delta', \forall xD; \Gamma, \exists y(C' \wedge C_1 \wedge \ldots \wedge C_n) \vdash\!\!\!- G'.$$

Another application of Lemma 4.2 leads from this to a proof of size less than l for the sequent (†).

(II). If $\forall x_1 \ldots \forall x_n(G' \Rightarrow A')$ is a variant of a formula in $elab(D[y/x])$, then $\forall y \forall x_1 \ldots \forall x_n(G' \Rightarrow A')$ is a variant of a formula in $elab(\forall xD)$, and so it is a variant of a formula in $elab(\Delta)$. Then condition (a) coincides with (i) plus $\Gamma \vdash_{\mathscr{C}} \exists yC'$, and (b) is equivalent to (ii). Moreover from (iii) (assuming that G' is not $\top$) we can deduce that the sequent

$$\Delta', D[u/x]; \Gamma, C', C_1, \ldots, C_n, u \approx y \vdash\!\!\!- G'$$

has a proof of size less than $l - 1$, because of Corollary 4.3 (u is chosen as a new variable). Since $\Gamma, C', C_1, \ldots, C_n \vdash_{\mathscr{C}} \exists u(u \approx y)$, we can apply $(\forall_L)$ and we obtain a proof of size less than l for the sequent

$$\Delta', \forall xD; \Gamma, C', C_1, \ldots, C_n \vdash\!\!\!- G'.$$

That is precisely condition (c). □

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Proofs of results from Section 4.3

Lemma 4.7 (*Elaboration*)
For any Δ, Γ, A and $F \in elab(\Delta)$: if $\Delta, F; \Gamma \vdash_{\mathscr{JG}} A$, then $\Delta; \Gamma \vdash_{\mathscr{JG}} A$.

Proof
Since $F \in elab(\Delta)$, there will be $D \in \Delta$ such that $F \in elab(D)$. The proof of the lemma is by case analysis according to the structure of D.

- If $D \equiv A'$, then $F \equiv \top \Rightarrow A'$. We prove $\Delta; \Gamma \vdash_{\mathscr{JG}} A$ by induction on the size l of the proof of $\Delta, F; \Gamma \vdash\!\!\!- A$. If $l = 1$, the proof consists on the application of (*Atom*), the form of F implies that it does not take part in this proof. So there exists $A'' \in \Delta$ such that $\Gamma \vdash_{\mathscr{C}} A'' \approx A$. Therefore $\Delta; \Gamma \vdash_{\mathscr{JG}} A$, by (*Atom*). Assuming now the result for proofs of size less than l, $l > 1$, we proceed by case analysis on the last rule applied in the proof of $\Delta, F; \Gamma \vdash\!\!\!- A$. Note that it is only necessary to analyse the left-introduction rules, since the goal is an atomic formula. For $(\wedge_L)$ and $(\forall_L)$, we note that $F \equiv \top \Rightarrow A'$ cannot participate on this step of the proof, instead a formula of Δ has been introduced. For instance, for $(\wedge_L)$, if $D_1 \wedge D_2$ is the formula introduced, then Δ is of the form $\Delta' \cup \{D_1 \wedge D_2\}$, and the last step of the proof is:

$$\frac{\Delta', D_1, D_2, F; \Gamma \vdash\!\!\!- A}{\Delta', D_1 \wedge D_2, F; \Gamma \vdash\!\!\!- A} \ (\wedge_L).$$

  So $\Delta', D_1, D_2, F; \Gamma \vdash\!\!\!- A$ has a proof of size less that l, and since $F \in elab(\Delta' \cup \{D_1, D_2\})$, $\Delta', D_1, D_2; \Gamma \vdash_{\mathscr{JG}} A$, by induction hypothesis. The result can be obtained now using the rule $(\wedge_L)$.
  For the case $(\Rightarrow_L)$, if the introduced formula is F (other cases are proved as before), then the last step of the proof is:

$$\frac{\Delta; \Gamma \vdash\!\!\!- \top \quad \Delta, A'; \Gamma \vdash\!\!\!- A}{\Delta, F; \Gamma \vdash\!\!\!- A} \ (\Rightarrow_L).$$

  Since $A' \equiv D$ and $D \in \Delta$, the sequent $\Delta, A'; \Gamma \vdash\!\!\!- A$ can be also written as $\Delta; \Gamma \vdash\!\!\!- A$, and we are done.
- If $D \equiv D_1 \wedge D_2$, then $F \in elab(D_i)$ for $i = 1$ or 2. $\Delta, F; \Gamma \vdash_{\mathscr{JG}} A$, by hypothesis, then it is easy to prove that also $\Delta, D_1, D_2, F; \Gamma \vdash_{\mathscr{JG}} A$. Hence, applying structural induction hypothesis to D_i, $\Delta, D_1, D_2; \Gamma \vdash_{\mathscr{JG}} A$. Therefore $\Delta, D_1 \wedge D_2; \Gamma \vdash_{\mathscr{JG}} A$, in accordance with the rule $(\wedge_L)$. This is equivalent to $\Delta; \Gamma \vdash_{\mathscr{JG}} A$, since $D \equiv D_1 \wedge D_2$ and $D \in \Delta$.
- If $D \equiv G_1 \Rightarrow D_1$, then $F \equiv D$, so $F \in \Delta$ and we have $\Delta; \Gamma \vdash_{\mathscr{JG}} A$ directly.
- If $D \equiv \forall x D_1$, then $F \equiv \forall x F_1$ and $F_1 \in elab(D_1)$. We proceed by induction on the size l of the proof of $\Delta, F; \Gamma \vdash\!\!\!- A$. The case $l = 1$ is trivial because F cannot take part in the proof. Similarly, we can reason the inductive step for the cases $(\wedge_L)$ and $(\Rightarrow_L)$. The interesting case occurs when $(\forall_L)$ was the last rule applied and F was the introduced formula. In this case, the last proof step is of the form:

$$\frac{\Delta, F_1[y/x]; \Gamma, C \vdash\!\!\!- A \qquad \Gamma \vdash_{\mathscr{C}} \exists y C}{\Delta, F; \Gamma \vdash\!\!\!- A} \ (\forall_L),$$

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

where y is not free in the sequent of the conclusion.
$\Delta, D_1[y/x], F_1[y/x]; \Gamma, C \vdash_{\mathscr{IC}} A$ can be deduced from $\Delta, F_1[y/x]; \Gamma, C \vdash_{\mathscr{IC}} A$. Then $\Delta, D_1[y/x]; \Gamma, C \vdash_{\mathscr{IC}} A$, since the lemma holds for $D_1[y/x]$ –simpler than D– and $F_1[y/x] \in elab(D_1[y/x])$. Therefore $\Delta, \forall x D_1; \Gamma \vdash_{\mathscr{IC}} A$, by $(\forall_L)$, using the fact that y is not free in $\Delta, \forall x D_1, \Gamma, A$, and that $\Gamma \vdash_{\mathscr{C}} \exists y C$. We conclude because $D \equiv \forall x D_1$ and $D \in \Delta$. □

Lemma 4.9
For any Δ, Γ, G, if Γ' is a set of constraints such that $\Gamma' \vdash_{\mathscr{C}} \Gamma$, and $\Delta; \Gamma \vdash_{\mathscr{UC}} G$, then $\Delta; \Gamma' \vdash\!\!- G$ has a $\mathscr{UC}$-proof of the same size.

Proof
By induction on the size of the proof of the sequent $\Delta; \Gamma \vdash\!\!- G$, by case analysis on the last rule applied. Using the definition of the system $\mathscr{UC}$ and Lemma 4.2, the only interesting case is when the last step corresponds to rule (*Clause*). But the proof in this case is a direct consequence of the induction hypothesis. □

Lemma 4.10
For any Δ, Γ, C, G, if $\Delta; \Gamma, C \vdash_{\mathscr{UC}} G$ and x is a variable that does not appear free in Δ, Γ, G, then $\Delta; \Gamma, \exists x C \vdash\!\!- G$ has a $\mathscr{UC}$-proof of the same size.

Proof
As in the previous lemma, and due now to Lemma 4.4, we can focus the proof on the case (*Clause*). In this case $G \equiv A$ and the last step of the proof is of the form:

$$\frac{\Delta; \Gamma, C \vdash\!\!- \exists x_1 \ldots \exists x_n((A' \approx A) \wedge G')}{\Delta; \Gamma, C \vdash\!\!- A} \ (Clause)$$

where A, A' begin with the same predicate symbol, and $\forall x_1 \ldots \forall x_n(G' \Rightarrow A')$ is a variant of a formula of $elab(\Delta)$, $x_1, \ldots, x_n$ do not appear free in the sequent of the conclusion.

Since x is not free in Δ, A, and $\forall x_1 \ldots \forall x_n(G' \Rightarrow A')$ is a variant of a formula of $elab(\Delta)$, then x is not free in $\exists x_1 \ldots \exists x_n((A' \approx A) \wedge G')$. Note also, that x is not free in Γ, Δ, by assumption, so applying the induction hypothesis to the sequent $\Delta; \Gamma, C \vdash\!\!- \exists x_1 \ldots \exists x_n((A' \approx A) \wedge G')$,

$$\Delta; \Gamma, \exists x C \vdash\!\!- \exists x_1 \ldots \exists x_n((A' \approx A) \wedge G')$$

has a proof of the same size. Hence, applying (*Clause*), $\Delta; \Gamma, \exists x C \vdash\!\!- A$ has a $\mathscr{UC}$-proof of the same size that $\Delta; \Gamma, C \vdash\!\!- A$. □

Proofs of results from Section 5.1

Lemma 5.2
Assume $\mathscr{S} \equiv \Pi[S \square \mathscr{G}]$ and $\mathscr{S}' \equiv \Pi\Pi'[S' \square \mathscr{G}']$ are two states w.r.t. a set of variables V, such that $\mathscr{S} \Vdash\!\!- \mathscr{S}'$. If R' is a constraint with its free variables in $\Pi\Pi'$ or in V, and such that $R' \vdash_{\mathscr{C}} S'$ and for any $\langle \Delta', C', G' \rangle \in \mathscr{G}'$, $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$, then $\Pi' R' \vdash_{\mathscr{C}} S$ and for any $\langle \Delta, C, G \rangle \in \mathscr{G}$, $\Delta; \Pi' R', C \vdash_{\mathscr{UC}} G$.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Proof
We analyse the different cases, according to the transformation applied.

(2) *Disjunction.* Π' is empty and $S \equiv S'$ as above. Then let us check only the case $\langle \Delta, C, G\rangle \notin \mathscr{G}'$. This implies $G \equiv G_1 \vee G_2$ and $\langle \Delta, C, G_1\rangle \in \mathscr{G}'$ or $\langle \Delta, C, G_2\rangle \in \mathscr{G}'$. By hypothesis

$$\Delta; \Pi' R', C \vdash_{\mathscr{WC}} G_1 \text{ or } \Delta; \Pi' R', C \vdash_{\mathscr{WC}} G_2,$$

since $\Pi' R' \equiv R'$. Then $\Delta; \Pi' R', C \vdash_{\mathscr{WC}} G$, because of the rule $(\vee_R)$.

(3) *Implication with local clause.* As before the prefix and the partially calculated answer constraint do not change. If $\langle \Delta, C, G\rangle \notin \mathscr{G}'$, then $G \equiv D \Rightarrow G_1$ and $\langle \Delta \cup \{D\}, C, G_1\rangle \in \mathscr{G}'$. Hence, by hypothesis since $\Pi' R' \equiv R'$, it holds

$$\Delta, D; \Pi' R', C \vdash_{\mathscr{WC}} G_1$$

from which we conclude the result by applying $(\Rightarrow_R)$.

(4) *Implication with local constraint.* As in the preceding cases where there are no changes in S and Π, we check what happens if $\langle \Delta, C, G\rangle \in \mathscr{G} \setminus \mathscr{G}'$. In this case $G \equiv C' \Rightarrow G_1$ and $\langle \Delta, C \wedge C', G_1\rangle \in \mathscr{G}'$. By hypothesis, since $\Pi' R' \equiv R'$, we have $\Delta; \Pi' R', C \wedge C' \vdash_{\mathscr{WC}} G_1$ then in accordance with Lemma 4.9

$$\Delta; \Pi' R', C, C' \vdash_{\mathscr{WC}} G_1.$$

Now we conclude $\Delta; \Pi' R', C \vdash_{\mathscr{WC}} G$, by applying $(\Rightarrow C_R)$.

(5) *Existential quantification.* $\Pi' \equiv \exists w$ with w a new variable not in Π nor in V. Hence, by item (i) of Lemma 5.1, w is not free in the formulas of $\mathscr{G}$, nor in S. Therefore, using the facts $R' \vdash_{\mathscr{C}} S'$ and $S \equiv S'$, we can conclude $\exists w R' \vdash_{\mathscr{C}} S$.
Now let $\langle \Delta, C, G\rangle \in \mathscr{G}$, if $\langle \Delta, C, G\rangle \in \mathscr{G}'$, then $\Delta; R', C \vdash_{\mathscr{WC}} G$, by hypothesis. Then $\Delta; \exists w R', C \vdash_{\mathscr{WC}} G$ by Lemma 4.10, because w is not free in Δ, C, G.
If $\langle \Delta, C, G\rangle \notin \mathscr{G}'$, $G \equiv \exists x G_1$ and $\langle \Delta, C, G_1[w/x]\rangle \in \mathscr{G}'$. By hypothesis,

$$\Delta; R', C \vdash_{\mathscr{WC}} G_1[w/x]$$

and so also $\Delta; \exists w R', R', C \vdash_{\mathscr{WC}} G_1[w/x]$, by Lemma 4.9. Consequently, applying the rule $(\exists_R)$,

$$\Delta; \exists w R', C \vdash_{\mathscr{WC}} G$$

since $\exists w R', C \vdash_{\mathscr{C}} \exists w R'$, and w is new for the sequent of the conclusion.

(6) *Universal quantification.* $\Pi' \equiv \forall w$ with w a new variable w.r.t. Π and V, and $S \equiv S'$. So $\forall w R' \vdash_{\mathscr{C}} S$ holds directly from $R' \vdash_{\mathscr{C}} S'$.
Let $\langle \Delta, C, G\rangle \in \mathscr{G}$, if $\langle \Delta, C, G\rangle \in \mathscr{G}'$, then $\Delta; R', C \vdash_{\mathscr{WC}} G$, by hypothesis. Then $\Delta; \Pi' R', C \vdash_{\mathscr{WC}} G$ because $\Pi' R' \vdash_{\mathscr{C}} R'$ and Lemma 4.9.
If $\langle \Delta, C, G\rangle \notin \mathscr{G}'$, $G \equiv \forall x G_1$ and $\langle \Delta, C, G_1[w/x]\rangle \in \mathscr{G}'$. By the hypothesis, since $\forall w R' \vdash_{\mathscr{C}} R'$ and Lemma 4.9, we have

$$\Delta; \forall w R', C \vdash_{\mathscr{WC}} G_1[w/x]$$

Now, since w is not in Π nor in V, by item (i) of Lemma 5.1, it is not free in Δ, C, G, and obviously w is neither free in $\forall w R'$. Then we conclude

$$\Delta; \forall w R', C \vdash_{\mathscr{WC}} G$$

by applying $(\forall_R)$.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

(7) *Constraint*. In this case Π' is empty and $\Pi S' \equiv \Pi(S \wedge (C \Rightarrow C'))$ is $\mathscr{C}$-satisfiable. Trivially, $R' \vdash_{\mathscr{C}} S'$ implies $\Pi' R' \vdash_{\mathscr{C}} S$. Now let $\langle \Delta, C, G \rangle \in \mathscr{G}$, the case $\langle \Delta, C, G \rangle \in \mathscr{G}'$ is easily proved. If $\langle \Delta, C, G \rangle \notin \mathscr{G}'$, then $G \equiv C'$. $R' \vdash_{\mathscr{C}} C \Rightarrow C'$ because $R' \vdash_{\mathscr{C}} S'$ and $S' \equiv S \wedge (C \Rightarrow C')$. By the properties of the constraint entailment, we deduce $R', C \vdash_{\mathscr{C}} C'$. Then applying the rule (C_R),

$$\Delta; \Pi' R', C \vdash_{\mathscr{UC}} G,$$

because $\Pi' R' \equiv R'$.

(8) *Clause of the program.* Since Π' is empty and $S \equiv S'$, we only check the case $\langle \Delta, C, G \rangle \in \mathscr{G}$ and $\langle \Delta, C, G \rangle \notin \mathscr{G}'$. In such case $G \equiv A$ and there is $\forall x_1 \ldots \forall x_n (G_1 \Rightarrow A')$ a variant of a formula of $elab(\Delta)$ where:

- $x_1, \ldots, x_n$ are new variables not occurring in Π, V, and therefore not free in A, Δ, C and $\Pi' R'$.
- A and A' begin with the same predicate symbol.
- $\langle \Delta, C, \exists x_1 \ldots \exists x_n ((A' \approx A) \wedge G_1) \rangle \in \mathscr{G}'$.

By hypothesis, since $\Pi' R' \equiv R'$,

$$\Delta; \Pi' R', C \vdash_{\mathscr{UC}} \exists x_1 \ldots \exists x_n ((A' \approx A) \wedge G_1).$$

Using now the rule (*Clause*), we conclude $\Delta; \Pi' R', C \vdash_{\mathscr{UC}} G$. □

Proofs of results from Section 5.2

Lemma 5.4

Let $\mathscr{S} \equiv \Pi[S \square \mathscr{G}]$ be a non-final state w.r.t. a set of variables V, and let R be a constraint such that ΠR is $\mathscr{C}$-satisfiable and $R \vdash_{\mathscr{C}} S$. If $\Delta; R, C \vdash_{\mathscr{UC}} G$ for all $\langle \Delta, C, G \rangle \in \mathscr{G}$, then we can find a rule transforming $\mathscr{S}$ in a state $\mathscr{S}' \equiv \Pi'[S' \square \mathscr{G}']$ $(\mathscr{S} \Vdash \mathscr{S}')$ and a constraint R' such that:

1. $\Pi R \vdash_{\mathscr{C}} \Pi' R'$ and $R' \vdash_{\mathscr{C}} S'$.
2. $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$ for all $\langle \Delta', C', G' \rangle \in \mathscr{G}'$. Moreover $\mathscr{M}_{\mathscr{G}' R'} << \mathscr{M}_{\mathscr{G} R}$.

Proof

Let us choose any $\langle \Delta, C, G \rangle \in \mathscr{G}$; we reason by induction on the structure of G, analysing cases:

- If G has the form $G_1 \wedge G_2$, $G_1 \vee G_2$, $D \Rightarrow G_1$ or $C_1 \Rightarrow G_1$, then we apply respectively the transformation rules (1), (2), (3) or (4) to $\mathscr{S}$. Let $\mathscr{S}'$ be the state obtained after the transformation, and let $R' \equiv R$:
  1. $\Pi R \vdash_{\mathscr{C}} \Pi' R'$ and $R' \vdash_{\mathscr{C}} S'$ are obvious by the hypothesis and because $\Pi' \equiv \Pi$, $S' \equiv S$ and $R' \equiv R$.
  2. Let $\langle \Delta', C', G' \rangle \in \mathscr{G}'$. If $\langle \Delta', C', G' \rangle \in \mathscr{G}$, then $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$ trivially since $\Delta'; R, C' \vdash_{\mathscr{UC}} G'$ by hypothesis, and $R' \equiv R$. Moreover $\tau_{R'}(\Delta', C', G') = \tau_R(\Delta', C', G')$. If $\langle \Delta', C', G' \rangle \notin \mathscr{G}$ and (1), for example, was applied, then $\Delta' \equiv \Delta$, $C' \equiv C$, $G \equiv G_1 \wedge G_2$ and $G' \equiv G_1$ or $G' \equiv G_2$. By hypothesis $\Delta; R, C \vdash\!\!- G$ with a proof of size l, therefore by the definition of $\mathscr{UC}$, since $R' \equiv R$, $\Delta; R', C \vdash\!\!- G_1$ and $\Delta; R', C \vdash\!\!- G_2$ have proofs of size less than l.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Consequently $\tau_{R'}(\Delta', C', G_1) < \tau_R(\Delta, C, G)$ and $\tau_{R'}(\Delta', C', G_2) < \tau_R(\Delta, C, G)$, so, finally $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$ for all $\langle \Delta', C', G' \rangle \in \mathscr{G}'$ and $\mathscr{M}_{\mathscr{G}'R'} << \mathscr{M}_{\mathscr{G}R}$. The argument for transformations (2), (3) and (4) is similar. Note that, in the case of (2), we must choose G_1 (resp. G_2) if the shortest $\mathscr{UC}$-proof of $\Delta; R, C \vdash G_1 \vee G_2$ contains a subproof of $\Delta; R, C \vdash G_1$ (resp. G_2).

- If G has the form $\forall x G_1$, we apply then the transformation rule (6) and obtain $\mathscr{S}'$. Assume $R' \equiv R$:
  1. Trivial since the choice of w assures that $\Pi R \vdash\!\!\dashv_{\mathscr{C}} \Pi \forall w R \equiv \Pi' R'$; moreover, $S' \equiv S$.
  2. Let $\langle \Delta', C', G' \rangle \in \mathscr{G}'$, if $\langle \Delta', C', G' \rangle \in \mathscr{G}$, then we obtain $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$, being $\tau_{R'}(\Delta', C', G') = \tau_R(\Delta', C', G')$. If $\langle \Delta', C', G' \rangle \notin \mathscr{G}'$, this is the triple coming from the transformation of $\langle \Delta, C, G \rangle$, so $G' \equiv G_1[w/x]$, $C' \equiv C$ and $\Delta' \equiv \Delta$. By hypothesis $\Delta; R, C \vdash G$ has a proof of size l, then since w does not appear free in $\Delta, C, R' (\equiv R), G_1$, because of the form of the calculus $\mathscr{UC}$, $\Delta; R', C \vdash G_1[w/x]$ has a proof of size less than l, and for that reason $\tau_{R'}(\Delta', C', G') < \tau_R(\Delta, C, G)$, and thus we conclude that 2 is valid.
- If G is a constraint C_1, we apply the transformation (7) obtaining $\mathscr{S}'$. Assume $R' \equiv R$:
  1. $\Pi R \vdash_{\mathscr{C}} \Pi' R'$ is trivial since $\Pi' \equiv \Pi$. Furthermore, $\Delta; R, C \vdash_{\mathscr{UC}} C_1$ by hypothesis, so by the definition of $\mathscr{UC}$, $R, C \vdash_{\mathscr{C}} C_1$ and therefore $R \vdash_{\mathscr{C}} C \Rightarrow C_1$. Moreover $R \vdash_{\mathscr{C}} S$, then $R' \vdash_{\mathscr{C}} S'$, because $R' \equiv R$ and $S' \equiv S \wedge (C \Rightarrow C_1)$. Now, from $R' \vdash_{\mathscr{C}} S'$ and the $\mathscr{C}$-satisfiability of $\Pi' R' \equiv \Pi R$, we deduce that $\Pi' S'$ is also $\mathscr{C}$-satisfiable. Therefore the transformation step is allowed.
  2. $\mathscr{G}' \subset \mathscr{G}$, so $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$ for all $\langle \Delta', C', G' \rangle \in \mathscr{G}'$ and $\mathscr{M}_{\mathscr{G}'R'} << \mathscr{M}_{\mathscr{G}R}$.
- If G is atomic $G \equiv A$, by hypothesis $\Delta; R, C \vdash A$ has a proof of size l, then by reason of the form of $\mathscr{UC}$, if $x_1, \ldots, x_n$ are new variables not free in Δ, R, C neither in A, then there is a variant of a formula from $elab(\Delta)$, $\forall x_1 \ldots \forall x_n (G_1 \Rightarrow A')$, with A and A' beginning with the same predicate symbol, such that $\Delta; R, C \vdash \exists x_1 \ldots \exists x_n ((A' \approx A) \wedge G_1)(\dagger)$ has a proof of size less than l. We transform $\mathscr{S}$ in $\mathscr{S}'$ by means of the rule (8), using $\forall x_1 \ldots \forall x_n (G_1 \Rightarrow A')$. Assume now $R' \equiv R$. Since $S \equiv S'$ and $\Pi \equiv \Pi'$, the proof of 1. is immediate. 2. Let $\langle \Delta', C', G' \rangle \in \mathscr{G}'$, if $\langle \Delta', C', G' \rangle \in \mathscr{G}$, then $\Delta'; R, C' \vdash_{\mathscr{UC}} G'$ by hypothesis and therefore $\Delta'; R', C' \vdash_{\mathscr{UC}} G'$, besides $\tau_{R'}(\Delta', C', G') = \tau_R(\Delta', C', G')$.
  If $\langle \Delta', C', G' \rangle \notin \mathscr{G}$, then $G' \equiv \exists x_1 \ldots \exists x_n ((A' \approx A) \wedge G_1)$, $C' \equiv C$ and $\Delta' \equiv \Delta$. As we have noted in (†), $\Delta; R', C' \vdash G'$ has a proof of size less than l. So $\tau_{R'}(\Delta', C', G') < \tau_R(\Delta, C, G)$, and 2. is also proved in this case. □

References

Andréka, H. and Németi, I. (1978) The generalized completeness of Horn Predicate Logic as a programming language. *Acta Cybernetica*, **4**, 3–10.

Clark, K. L. (1978) Negation as failure. In: Gallaire, H. and Minker, J. (eds.), *Logic and Databases*, pp. 293–322. Plenum Press, New York.

https://doi.org/10.1017/S1471068401001041 Published online by Cambridge University Press

Comon, H., Haberstrau, M. and Jouannaud, J. P. (1994) Cycle-syntacticness, and shallow theories. *Information & Computation*, **111**, 154–191.

Cohen, J. (1990) Constraint Logic Programming languages. *Comm. ACM*, **35**(7), 52–68.

Comon, H. (1993) Complete axiomatizations of some quotient term algebras. *Theor. Comput. Sci.* **118**, 167–191.

Darlington, J. and Guo, Y. (1994) Constraint Logic Programming in the Sequent Calculus. In: Pfenning, F. (ed.), *LPAR'94. LNAI 822*, pp. 200–213. Springer-Verlag.

Dershowitz, N. and Manna, Z. (1979) Proving termination with multiset ordering. *Comm. ACM*, **22**(8), 465–476.

Hodas, J. S. (1994) Logic Programming in Intuitionistic Linear Logic: Theory, Design and Implementation. *PhD Thesis*, University of Pennsylvania.

Jaffar, J. and Lassez, J. L. (1987) Constraint Logic Programming. *Proc. 14th ACM Symposium on Principles of Programming Languages,* pp. 111–119.

Jaffar, J. and Michaylov, S. (1987) Methodology and implementation of CLP systems. In: Lassez, J. L. (ed.), *ICLP'87,* pp. 196–218. MIT Press.

Jaffar, J. and Maher, M. J. (1994) Constraint Logic Programming: A survey. *J. Logic Programming*, **19**(20), 503–581.

Jaffar, J., Maher, M. J., Marriott, K. and Stuckey, P. (1996) The Semantics of Constraint Logic Programs. *Technical Report TR 96/39*, Department of Computer Science, University of Melbourne.

Jaffar, J., Michaylov, S., Stuckey, P. and Yap, R. (1992) The CLP($\mathscr{R}$) Language and System. *ACM Trans. Programming Lang.*, **14**(3), 339–395.

Leach, J., Nieva, S. and Rodríguez-Artalejo, M. (1997) Constraint Logic Programming with Hereditary Harrop formulas. In: Małuszyński, J. (ed.), *ILPS'97,* pp. 307–321. MIT Press.

Maher, M. (1987) Logic semantics for a class of committed-choice programs. In: Lassez, J. L. (editor), *ICLP'87,* pp. 858–876. MIT Press.

Miller, D. (1989) A logical analysis of modules in logic programming. *J. Logic Programming*, **6**(1, 2), 79–108.

Miller, D. (1992) Unification under a mixed prefix. *J. Symbolic Computation*, **14**, 321–358.

Miller, D. and Nadathur, G. (1986) Higher-order logic programming. In: Shapiro, E. (ed.), *ICLP'86: LNCS 225*, pp. 448–462. Springer-Verlag.

Miller, D., Nadathur, G., Pfenning, F. and Scedrov, A. (1991) Uniform proofs as a foundation for logic programming. *Ann. Pure Appl. Logic*, **51**(1–2), 125–157.

Miller, D., Nadathur, G. and Scedrov, A. (1987) Hereditary Harrop formulas and uniform proof systems. In: Gries, D. (ed.), *LICS'87,* pp. 98–105. IEEE Press.

Nadathur, G. (1993) A proof procedure for the logic of Hereditary Harrop formulas. *J. Automated Reasoning*, **11**, 115–145.

Nadathur, G. and Miller, D. (1988) An overview of λ-Prolog. In: Bowen, K. A. and Kowalski, R. A. (eds.), *ICLP'88,* pp. 810–827. MIT Press.

Saraswat, V. (1992) The category of constraint systems is Cartesian closed. *LICS'92,* pp. 341–345. IEEE Press.

Smolka, G. and Treinen, R. (1994) Records for logic programming. *J. Logic Programming*, **18**(3), 229–258.

Van Emden, M. H. and Kowalski, R. H. (1976) The semantics of predicate logic as a programming language. *J. ACM*, **23**(4), 733–742.

Tarski, A. (1951) *A Decision Method for Elementary Algebra and Geometry*. University of California Press.