Section 3: Foundations of computation and recursion theory
Dale MILLER, University of Pennsylvania
Philadelphia, PA 19104, USA

HEREDITARY HARROP FORMULAS AND LOGIC PROGRAMMING

1. Introduction. Logic programming is a style of computer programming in which the programs are a collection of quantificational formulas and computation involves the search for proofs. Current logic programming languages use positive Horn clauses as programs and search for proofs within first-order classical logic. Although this programming language has many interesting features, it lacks many other features commonly found in other programming languages. For example, functional programming, which is based on the logically simple notions of one-way equational rewriting and reduction, is capable of representing block structures, modules, and abstract data types. None of these are available with the classical theory of positive Horn clauses.

In this paper, we suggest that a reinterpretation of logic programming using *hereditary Harrop* formulas for it programs and an intuitionistic logic for its interpretation provides a natural extension to the conventional approach. This extension also provides for the above mentioned programming features. Our main focus in this abstract is to show that hereditary Harrop formulas admit *uniform* proofs, possess a Herbrand theorem, and have a minimal Kripke model semantics. For these reasons, hereditary Harrop formulas can be seen as having a metatheory which naturally extends the metatheory of positive Horn clauses as presented in [1].

2. Logic Programming. A logic program is a specification of a search and a proof is the result or documentation of a successful search. Proofs, however, are generally very large objects which contain far more details than are generally usable. Such large objects are not desirable kinds of value in many computational settings. As a result, only certain aspects of a proof are actually retained and reported as the result of a successful search. The most commonly retained aspect is the substitution used to instantiate quantifiers when building a proof. For this approach to be sensible, proofs must contain simple and uniform rules for all the logical connectives. If there were significant decisions on how these connectives are used in a proof then the substitution information would not reflect some of the significant aspects of the proof's structure.

One approach to simplifying the structure of proofs is to assume that all subproofs of a given proof treat the introduction of logical connectives in the same, predictable fashion. Assume that we have an L-system (sequent-style) proof system. In particular, let LK, LJ, LM be the L-systems for classical, intuitionistic, and minimal logics in [4]. Let T be some proof in such a proof system. We say that T is a *uniform* proof if all its sequents contain succedents of at most one formula, and if the following condition is also meet. Let $\mathcal{P} \to G$ be an arbitrary non-axiom occurrence of a sequent in T, for some formula G and some set of formulas $\mathcal{P}$. This sequent is the lower sequent of an inference figure with either one or two

upper sequents. Let T_1 and, if necessary, T_2 be subproofs which prove these upper sequents. We require the following conditions to be satisfied.

(1) If G is $G_1 \vee G_2$ then T_1 is a proof of either $\mathcal{P} \to G_1$ or $\mathcal{P} \to G_2$.

(2) If G is $G_1 \wedge G_2$ then T_1 is a proof of $\mathcal{P} \to G_1$ and T_2 is a proof of $\mathcal{P} \to G_2$.

(3) If G is $\exists x\, G'$ then T_1 is a proof of $\mathcal{P} \to [x/t]G$ for some closed term t.

(4) If G is $\forall x\, G'$ then T_1 is a proof of $\mathcal{P} \to [x/c]G$ for some parameter c with no occurrences in $\mathcal{P}$ or in G.

(5) If G is $H \supset G'$ then T_1 is a proof of $\mathcal{P} \cup \{H\} \to G'$.

Let $\mathcal{G}$ be a set of formulas over some logical language and let $\vdash_P$ be some provability predicate determined by a given L-system form of deduction. We say that the pair $(\mathcal{G}, P)$ is a *logic programming language* if the following is true: for all $G \in \mathcal{G}$ if $\vdash_P G$ then there is a uniform $\vdash_P$-proof of the sequent $\to G$.

In an operational sense, the sequent $\mathcal{P} \to G$ is though of as a specification of a search: from the facts in $\mathcal{P}$ try to establish goal G. The top-level connectives of G are used to indicate exactly how the search should proceed. Hence, the use of the connectives $\wedge$ and $\vee$ are meant to denote *and* and *or* nodes in the search of a proof. That is, if we are searching for a uniform proof of the sequent $\mathcal{P} \to G_1 \wedge G_2$ we must search for similar proofs for both $\mathcal{P} \to G_1$ *and* $\mathcal{P} \to G_2$. A similar observation holds for disjunction. Universal and existential quantifiers work as a kind of infinite *and* and *or* node in the the search. Finally, implication is designed to augment the current facts in order to a attempt a new goal.

A well known example of a logic programming language is $(\mathcal{G}_1, LK)$ where $\mathcal{G}_1$ is the set of closed formulas $H \supset \exists \bar{x}(A_1 \wedge \ldots \wedge A_n)$, where H is a conjunction of positive Horn clauses, $A_1, \ldots, A_n$ are atomic formulas, and $\bar{x}$ is a list of variables. It is easy to show that $(\mathcal{G}_1, LJ)$ and $(\mathcal{G}_1, LM)$ essentially describing the same logic programming language. It should be noted, however, that positive Horn theories are weak logic programming languages in the sense that in any uniform proof in $(\mathcal{G}_1, LK)$, conditions (1) and (4) are vacuously true and the only occurrences of implication or existential introduction rules are at the root of the proof's tree.

3. Hereditary Harrop formulas. To consider a stronger logic programming language, consider the class of formulas denoted by the following recursive definition of the syntactic variable D:

$$D := A \mid G \supset A \mid \forall x\, D \mid D_1 \wedge D_2.$$

Here we are assuming that A denotes atomic formulas and G are arbitrary first-order formulas. This class represents the set of all Harrop formula [2], in the sense that for any Harrop formula, there is a formula D which is intuitionistically equivalent to it. Let $\mathcal{G}'$ be the set of formulas $D \supset G$, where D and G denote formulas as above. It is well known that the pair $(\mathcal{G}', LJ)$ satisfies the conditions on uniformity only for the initial part of a proof. In general, however, proofs in this setting are not uniform. There is, however, an interesting subset of Harrop formulas which in fact admit uniform proofs.

In order to restrict Harrop formulas, we need to restrict the formulas G so that immediate negative subformula occurrences are actually D formulas. More precisely, we introduce the following restricted definition of G.

$$G := A \mid G_1 \wedge G_2 \mid G_1 \vee G_2 \mid \forall x\, G \mid \exists x\, G \mid D \supset G$$

The definitions of G and D are now mutually recursive. We shall call any D formula a *hereditary Harrop* formula and any G formula a *goal formula*. Let $\mathcal{G}_2$ be the set of goal formulas. The following theorem can be proved.

Theorem A. $(\mathcal{G}_2, LJ)$ and $(\mathcal{G}_2, LM)$ are logic programming languages.

In neither of these cases are any of the clauses defining uniform proofs vacuously true. Notice that these languages properly include the logic programming languages based on positive Horn clauses, *i.e.* the languages determined by $\mathcal{G}_1$. This extension can be used to provide logic programming languages with notions of block structuring, modules, and abstract data types. These have been demonstrated in [3].

Finally, it is easy to show that the pair $(\mathcal{G}_2, LK)$ is not a logic programming language. For example, $(p \supset q) \vee p$ has an LK proof but has no uniform LK proof.

4. A Herbrand Theorem. Full intuitionistic logic has not admitted a Herbrand theorem. Because of the existence of uniform proofs, however, the hereditary Harrop formula fragment of intuitionistic logic does admit such a theorem. First, we define *existential* goals and hereditary Harrop formulas as those G and D formulas obtained by deleting the case $\forall x\, G$ in the definition of G above. We now define g-instances of existential goal formulas and d-instances of existential hereditary Harrop formulas.

- If A is an atom, then A is a g- and d-instance of A.
- If E_1 and E_2 are d-instances of D_1 and D_2, then $E_1 \wedge E_2$ is a d-instance of $D_1 \wedge D_2$.
- If $t_1, \ldots, t_n$ $(n \geq 1)$ is a list of terms and if for all $i \in \{1, \ldots, n\}$, E_i is a d-instance of $[x/t_i]D$, then $E_1 \wedge \ldots \wedge E_n$ is a d-instance of $\forall x\, D$.
- If H_1 and H_2 are g-instances of G_1 and G_2, then $H_1 \wedge H_2$ is a g-instance of $G_1 \wedge G_2$, and $H_1 \vee H_2$ is a g-instance of $G_1 \vee G_2$.
- If H is a g-instance of $[x/t]G$ then H is a g-instance of $\exists x\, G$.
- If H is a g-instance of G then $H \supset A$ is a d-instance of $G \supset A$.
- If E is a d-instance of D and H is a g-instance of G then $E \supset H$ is a g-instance of $D \supset G$.

The following is a form of Herbrand's theorem for this fragment of intuitionistic logic.

Theorem B. The goal formula G is provable in LJ (resp. LM) if and only if there is a g-instance of the Skolem form of G which is a tautology in LJ (resp. LM).

5. A Model Theory. The above Herbrand result can be used to build a Kripke model such that an existential goal formula is valid in this model if and only if it is provable in LM. This

model can be simply described as the least fixed point of a continuous operator over a class of Kripke models. Furthermore, this model is connected by its accessibility relation.

Let $\mathcal{W}$ be the set of all finite sets of existential hereditary Harrop formulas. An interpretation I is a function from $\mathcal{W}$ to the set of atoms such that $\forall w_1, w_2 \in \mathcal{W}[w_1 \subseteq w_2 \supset I(w_1) \subseteq I(w_2)]$. The triple $\langle \mathcal{W}, \subseteq, I \rangle$ can be viewed as a Kripke model. The set of all interpretations is a complete lattice under world-wise inclusion. Let T be the function from interpretations to interpretations such that

$$T(I)(w) := \{A \mid \text{there is a formula in } w \text{ with either } A \text{ or } G \supset A \text{ as an instance and, in the latter case, } I, w \models G\ \}.$$

This operator is a continuous, order-preserving map from the lattice of interpretations to itself.

Theorem C. The least fixed point of T is a Kripke model which is connected by its accessibility relation and is such that G is any LM-provable existential goal formula if and only if G is satisfiable at the root world of this model.

With slight modifications, a similar model can be constructed to characterize provability in LJ instead of LM.

6. Acknowledgements. We are grateful to Gopalan Nadathur, Andre Scedrov, and Scott Weinstein for helpful comments and suggestions.

7. References.

[1] Krzysztof R. Apt, M. H. van Emden, "Contributions to the Theory of Logic Programming" Journal of the ACM **29** (1982), 841 – 862.

[2] Ronald Harrop, "Concerning Formulas of the types $A \to B \vee C$, $A \to (Ex)B(x)$ in Intuitionistic Formal Systems," Journal of Symbolic Logic, **25** (1), 1960, 27 — 32.

[3] Dale Miller, "A Theory of Modules for Logic Programming," Third Annual IEEE Symposium on Logic Programming, Salt Lake City, Utah, September 1986.

[4] Dag Prawitz, *Natural Deduction*, Almqvist & Wiksell, Uppsala, 1965.

Abstract of the Eigtht International Congress of Logic, Methodology and Philosphy of Science, Moscow, 17 – 22 August 1987.