On the computational complexity of logic programs with nested implications

by

Jörg Hudelmaier
WSI, Universität Tübingen
Sand 13, D72076 Tübingen
Tel. (49)7174 297361
joerg@logik.informatik.uni-tuebingen.de

On the computational complexity of logic programs with nested implications

by

Jörg Hudelmaier
WSI, Universität Tübingen
Sand 13, D72076 Tübingen
joerg@logik.informatik.uni-tuebingen.de

We consider propositional programs in a logic programming language which allows implications in the bodies of rules. Given a program P in such a language and a goal g the relation "*P implies g*" may be conceived as derivability in intuitionistic logic of the formula g from the set P of premisses. In general, however, logic programming languages do not admit the full syntax of intuitionistic logic for writing program rules. N-Prolog, for instance, has rules of the form $B \to h$, where h is an atom or the constant $\bot$ and B is a conjunction of an arbitrary number of atoms and rules. Goals are atoms or the constant $\bot$ (cf. [1]). In order to determine, whether a program P implies a goal g ("*P ? g succeeds*") a so called goal directed calculus is used, which in essence is a version of Gentzen's calculus NJ of natural deduction. It is built on the observation that in an NJ-deduction of an atom a from a set of N-Prolog rules the final inference has to be an application of modus ponens. Therefore P must contain a rule of the form $B \to a$ and B must be derivable from P. But B is a conjunction, therefore each of its conjuncts has to be derivable from P and in turn each of the conjuncts is either an atom or an implication $C \to i$. In the latter case we know that i must be derivable from P,C. But this set is again a program, so we are back in the situation we had started with, having obtained a new program and a new goal to derive from it. Combining these three steps into one we are able to show completeness of the following N-Prolog calculus:
The calculus consists of axioms of the form

$P,g\ ?g$ *succeeds*
$P,\bot\ ?\ g$ *succeeds*

and the single rule

$P,(B_1{\to}h_1 \wedge \ldots \wedge B_n{\to}h_n){\to}g\ \ ?\ g$ *succeeds if for all i*
$P,(B_1{\to}h_1 \wedge \ldots \wedge B_n{\to}h_n){\to}g\ ,B_i\ ?\ h_i$ *succeeds*

(where some of the B_i may also be empty; if $P\ ?\ g$ does not succeed we say that it fails.)

Now it is well known that for every formula of intuitionistic propositional logic with connectives $\wedge$ and $\to$ there exists an intuitionistically equivalent conjunction of implications $P \to g$, where P is an N-Prolog program and g is a goal. This correspondence is established using the intuitionistically valid equivalences $a{\to}(b{\to}c) \equiv (a{\wedge}b){\to}c$ and $a{\to}(b{\wedge}c) \equiv (a{\to}b){\wedge}(a{\to}c)$. But according to [2] we may even find such implications to any arbitrary formula in the full language of intuitionistic propositional logic including disjunction and absurdity. But in this case we cannot achieve equivalence of the two formulas, but only *equideducibility*, i.e.

the fact that one of these formulae is intuitionistically derivable if and only if the other one is. This, however, already suffices to show that propositional N-Prolog is PSPACE-complete. But we may further restrict the form of programs by noting the well known

Lemma 1:
a) A formula $M \to g$ is derivable in intuitionistic propositional logic iff the formula $(M \wedge g \to p) \to p$ is derivable, where p is an atom occurring only at the indicated positions.
b) A formula $(M \wedge (B \wedge C) \to h) \to g$ is derivable in intuitionistic propositional logic iff the formula $(M \wedge B \to p \wedge (p \wedge C) \to h) \to g$ is derivable, where p occurs only at the indicated places.
c) A formula $(M \wedge (B \to i) \to h) \to g$ is derivable in intuitionistic propositional logic iff the formula $(M \wedge p \to B \wedge (p \to i) \to h) \to g$ is derivable, where p occurs only at the indicated places.

This lemma allows us to construct to every formula of intuitionistic propositional logic an equiderivable formula of the form $P \to g$ where g is an atom and P is a conjunction of formulae of the form a, $a \to b$, $(a \wedge b) \to c$, and $(a \to b) \to c$, where a, b, and c are atoms. Moreover we may do this transformation in such a way that for any two formulas $B \to a$ and $C \to a$ with the same right hand side the formulas B and C are atoms. This means that for the formulae $(a \wedge b) \to c$ resp. $(a \to b) \to c$ the atom c is unique as a right hand side of an implication in P. According to [2] the length of such an implication $P \to g$ depends at most quadratically on the length of the original formula; therefore derivability of goals from such restricted programs is still PSPACE-complete.

Now we turn to an even stronger restriction on programs viz. so called *well founded programs* and we shall show that derivability of goals from programs of this seemingly benign class is still of the same complexity as full intuitionistic propositional logic.

Definition:
a) Let P be a program obeying the previous restriction and consider a relation $<_P$ on the set of atoms of P defined by $a <_P b$ iff P contains a rule $a \to b$ or a rule $(c \to a) \to b$ or a rule $(c \wedge a) \to b$ or $(a \wedge c) \to b$. Then the program P is *well founded* iff the relation $<_P$ is well founded.
(In this case we denote its transitive closure still by $<_P$.)
b) Let P be a program obeying the previous restriction. Then we call an atom a of P a c-atom iff P contains a rule $(c \wedge b) \to a$ and we call a a d-atom iff P contains two rules $c \to a$ and $b \to a$.
(Note that the sets of c-atoms and d-atoms are disjoint.)
c) Let P be a well founded program and g a goal. Then $P\ ?\ g$ is called a C_n-sequent iff g is a c-atom of P and n is the maximal number of changes between c-atoms and d-atoms along the order $<_P$ restricted to the atoms which are $<_P g$. Similarly $P\ ?\ g$ is a D_n-sequent iff g is a d-atom of P and n is the maximal number of changes between c-atoms and d-atoms along the order $<_P$ restricted to the atoms which are $<_P g$.

Note that the rules of the form $(a \to b) \to c$ in P do not contribute to the size of n in part c) of this definition. Thus C_0 is a strictly larger class than the usual class of Horn programs. In fact we do not know, whether provability of C_0-sequents is polynomially decidable. We can only show:

Proposition 1:

a) The set of failing C_0-sequents is in **NP**.

b) The set of successful D_0-sequents is in **NP**.

Proof: We define the sets CACCEPT and DACCEPT by the following mutually exclusive clauses:

CACCEPT(P,$a{\to}g$? g)	IFF	CACCEPT(P,$a{\to}g$? a)
CACCEPT(P,$(a{\wedge}b){\to}g$? g)	IFF	CACCEPT(P,$(a{\wedge}b){\to}g$? a) OR CACCEPT(P,$(a{\wedge}b){\to}g$? b)
CACCEPT(P,$(a{\to}b){\to}g$?g)	IFF	CACCEPT(P,a,$(a{\to}b){\to}g$? b)
CACCEPT(P ? g)	OTHERWISE	

DACCEPT(P,g ? g)		
DACCEPT(P,$a{\to}g$? g)	IFF	DACCEPT(P,$a{\to}g$? a)
DACCEPT(P,$a{\to}g$,$b{\to}g$?g)	IFF	DACCEPT(P,$a{\to}g$,$b{\to}g$? a) OR DACCEPT(P,$a{\to}g$,$b{\to}g$? b)
DACCEPT(P,$(a{\to}b){\to}g$?g)	IFF	DACCEPT(P,a,$(a{\to}b){\to}g$? b)

The sets CACCEPT resp. DACCEPT obviously coincide with the sets of failing C_0-sequents resp. successful D_0-sequents. Moreover for any two successive calls to CACCEPT resp. DACCEPT with arguments P ? g resp. Q ? h for the atoms g and h the relation $h <_P g$ holds. Thus there may at most be as many successive such calls as there are different atoms in P. Therefore the sets CACCEPT and DACCEPT are in **NP** and thus the sets of successful D_0-sequents and failing C_0-sequents are also in **NP**.

Unfortunately is not known, whether these sets are **NP**-complete. All we can show is

Proposition 2:

a) The set of failing C_2-sequents is **NP**-hard.

b) The set of successful D_2-sequents is **NP**-hard.

Proof: We use the following remark:
A sequent $P,B{\to}h$? g, where $h <_P g$ does not hold, succeeds iff the sequent P ? g succeeds. This remark follows directly from the fact that the for any later goal h evaluated during evaluation of the goal g the relation $h <_P g$ must hold.
Now to show the first claim we will obviously have to encode provability of formulae in classical propositional logic using suitable well founded sequents. But for classical propositional logic it is well known that we may retsrict ourselves to formulae in disjunctive normal form, i.e. formulae of the form $v = C_1 \vee \dots \vee C_n$, where the C_i are conjunctions $p_{i,1} \wedge \dots \wedge p_{i,n(i)}$ and the formulas $p_{i,j}$ are Boolean literals. For such a formula v we let $\Pi(v)$ depend from the propositional variables and from the disjunctive clauses of v as follows:

Suppose that the variables of v are numbered from 1 to t; then for any variable x_i the program $\Pi(v)$ has one rule $((x_i\to r_{i-1})\wedge(y_i\to r_{i-1}))\to r_i$, where the r_i and the y_i are new pairwise different atoms. Moreover for any clause C_i of v the program has one rule $B_i\to r_0$, where B_i results from C_i by replacing any negative literal $\neg x_i$ by the corresponding new atom y_i.
Now induction on t shows that v is provable in classical logic if and only if the goal r_t succeeds from the program $\Pi(v)$:
If $t = 1$, then $\Pi(v)$ can only contain the rules $x_1\to r_0$, $y_1\to r_0$, and $((x_1\to r_0)\wedge(y_1\to r_0))\to r_1$ and r_1 succeeds from $\Pi(v)$ iff $\Pi(v)$ contains all these three rules, i.e. iff v contains two clauses x_1 and $\neg x_1$. Thus for $t = 1$ the formula v is provable if $\Pi(v)$? r_1 succeeds.
For $t > 1$ the formula v is provable iff both the formulas $v(x_t)$ and $v(\neg x_t)$ are provable, where $v(p)$ results from v by deleting all clauses of v which contain p and deleting $\neg p$ from the remaining clauses. By the induction hypothesis these formulas are provable if the goal r_{t-1} succeeds from the programs $\Pi(v\text{–}x_t)$ and $\Pi(v\text{—}\neg x_t)$. Therefore the following lemma completes the induction:

Lemma 2: A goal r_t succeeds from a program
$P = (x_t\wedge C_1)\to r_0, ...,(x_t\wedge C_l)\to r_0,(y_t\wedge D_1)\to r_0, ..., (y_t\wedge D_m)\to r_0, E_1\to r_0, ..., E_n\to r_0,((x_1\to r_0)\wedge(y_1\to r_0))\to r_1, ..., ((x_t\to r_{t-1})\wedge(y_t\to r_{t-1}))\to r_t$ iff r_{t-1} succeeds from the programs $P(x_t) = D_1\to r_0, ..., D_m\to r_0, E_1\to r_0, ..., E_n\to r_0,((x_1\to r_0)\wedge(y_1\to r_0))\to r_1, ..., ((x_{t-1}\to r_{t-2})\wedge(y_{t-1}\to r_{t-2}))\to r_{t-1}$ and $P(y_t) = C_1\to r_0, ..., C_l\to r_0,E_1\to r_0, ..., E_n\to r_0,((x_1\to r_0)\wedge(y_1\to r_0))\to r_1, ..., ((x_{t-1}\to r_{t-2})\wedge(y_{t-1}\to r_{t-2}))\to r_{t-1}$.

Proof: First we show: if r_t succeeds from P, then r_{t-1} succeeds from $P(x_t)$:
Suppose that a program contains a rule $(b\wedge c)\to h$ and a goal g succeeds from this program. Then a fortiori the program which contains, instead of $(b\wedge c)\to h$, the rule $b\to h$ implies the goal g, too. Therefore if r_t succeeds from P, then it also succeeds from the program P' which results from P by deleting y_t from all clauses $(y_t\wedge D_i)\to r_0$; and thus r_{t-1} succeeds from P',x_t. Moreover by the above Remark the program which results from P',x_t by deleting the rule $((x_t\to r_{t-1})\wedge(y_t\to r_{t-1}))\to r_t$ also implies r_{t-1}. This program now does not have any occurrence of x_t as head of a rule; therefore all rules which have x_t in the body may be deleted, thereby obtaining the program $P(x_t)$.
That r_{t-1} succeeds from $P(y_t)$ is shown analogously.
Suppose now that $P(x_t)$ and $P(y_t)$ imply r_{t-1}. Then it suffices to remark that in intuitionistic logic the formula r_t is derivable from the formulae $((C_1\to r_0)\wedge ... \wedge(C_l\to r_0))\to r_{t-1},((D_1\to r_0)\wedge ... \wedge(D_m\to r_0))\to r_{t-1}$, and $(x_t\wedge C_1)\to r_0, ... ,(x_t\wedge C_l)\to r_0,(y_t\wedge D_1)\to r_0,... ,(y_t\wedge D_m)\to r_0$, and the formula $((x_t\to r_{t-1})\wedge(y_t\to r_{t-1}))\to r_t$ and by applying the cut rule we see that P itself implies r_t, completing the proof of the lemma.

Now $\Pi(v)$ is not a well founded program, but it may easily be transformed into such a program by introducing some new atoms according to Lemma 1 and using the fact that a program $P,b\to h$ implies a goal g iff the program $P,b\to y,y\to h$ implies g, where y is an atom not occurring in the original goal. The program and goal transformed in such a way then turns out to be a C_2-sequent. This means that the set of classically provable formulae is reduced to the set of successful C_2-sequents, and therefore the set of classically unprovable formulae is reduced to the set of failing such sequents. Thus this latter set is **NP**-hard.

For part b) we proceed in a similar manner, encoding unprovability of our formula v as success of the goal r_t from a program $\Sigma(v)$, which for every variable x_i of v now has two rules $(x_i \to r_{i-1}) \to r_i$ and $(y_i \to r_{i-1}) \to r_i$ and for every clause $C_i = p_{i,1} \wedge ... \wedge p_{i,n(i)}$ of v has $n(i)$rules $b_{i,1} \to q_i$, ..., $b_{i,n(i)} \to q_i$, where b_{ij} is defined from p_{ij} as B_{i} was defined from C_i and q_i is a new atom. Finally $\Sigma(v)$ has a single clause $(q_1 \wedge ... \wedge q_n) \to r_0$. Then $\Sigma(v)$? r_t may be transformed into a D_2-sequent, showing that the set of successful D_2-sequents is also **NP**-hard.

To be able to generalize these results, we have to introduce the familiar so called *polynomial hierarchy*:
Let the complexity classes $\mathbf{NP}_n$ be given by $\mathbf{NP}_1$ = **NP**, $\mathbf{NP}_{n+1}$ = the class of languages accepted by non deterministic Turing machines with oracles from the class $\mathbf{NP}_n$ (cf. [3].) Then we can show:

Proposition 3:

a) The set of failing C_n -sequents is in $\mathbf{NP}_{n+1}$
b) The set of successful D_n-sequents is in $\mathbf{NP}_{n+1}$.

Proof: The case $n = 0$ has been treated above.
For $n > 0$ we define:

CACCEPT(P ? g,n)	IFF	NOT DACCEPT(P ? g,n-1), if P ? g has fewer than n alternations between c- and d-atoms.
CACCEPT(P,$(a \wedge b) \to g$? g,n)	IFF	CACCEPT(P,$(a \wedge b) \to g$? a,n) OR CACCEPT(P,$(a \wedge b) \to g$? b,n)
CACCEPT(P,$(a \to b) \to g$?g)	IFF	CACCEPT(P,a,$(a \to b) \to g$? b,n)
DACCEPT(P ? g,n)	IFF	NOT CACCEPT(P ? g,n-1), if P ? g has fewer than n alternations between c- and d-atoms.
DACCEPT(P,$a \to g$? g,n)	IFF	DACCEPT(P,$a \to g$? a,n)
DACCEPT(P,$a \to g$,$b \to g$?g,n)	IFF	DACCEPT(P,$a \to g$,$b \to g$? a,n) OR DACCEPT(P,$a \to g$,$b \to g$? b,n)
DACCEPT(P,$(a \to b) \to g$?g,n)	IFF	DACCEPT(P,a,$(a \to b) \to g$? b,n)

Again CACCEPT(P ? g,n) holds iff P is in C_{n} and g does not succeed from P and DACCEPT(P ? g,n) holds iff P is in D_{n} and g succeeds from P. But the definitions of the new relations CACCEPT(_,n) and DACCEPT(_,n) coincide with the former definitions except for their first clauses. Thus the new relations may be implemented on non deterministic Turing machines with oracles for CACCEPT(_,n-1) resp. DACCEPT(_,n-1). But by the induction hypothesis the latter relations are in Σ_{n-1}; therefore the relations CACCEPT(_,n) and DACCEPT(_,n) are in Σ_n.

The generalization of proposition 2 now reads:

Proposition 4:

a) The set failing C_{n+1}-sequents is hard for Σ_n.

b) The set of successful D_{n+1}-sequents is hard for Σ_n.

Proof: We consider canonical complete sets for the stages Σ_n of the polynomial hierarchy, viz. the formulas v_n provable in second order classical propositional logic of the form

$$v_n = \exists X_1 \forall X_2 \ldots \forall X_{2n}\, D \text{ resp. } v_n = \exists X_1 \forall X_2 \ldots \exists X_{2n+1} C$$

where the X_i are sequences of propositional variables, every variable of C resp. D occurs exactly once in this prefix and D is in disjunctive normal form and C is in conjunctive normal form (cf. [3]). Provability of such formulae is encoded as validity of implications of type b) as follows:

For a formula $v = C_1 \vee \ldots \vee C_n$ in disjunctive normal form we define $\Pi(v)$ as above to be $B_1 \rightarrow r_0, \ldots, B_n \rightarrow r_0$ and for $v = C_1 \wedge \ldots \wedge C_n$, where C_i is $p_{i1} \vee \ldots \vee p_{in(i)}$ we define $\Pi(v)$ to consist of all rules $b_{i1} \rightarrow q_i$ together with the one rule $(q_1 \wedge \ldots \wedge q_n) \rightarrow r_0$. Then for every quantifier $\exists x_i$ we add to $\Pi(v)$ two rules $(x_i \rightarrow r_{i-1}) \rightarrow r_i$ and $(y_i \rightarrow r_{i-1}) \rightarrow r_i$ and for every $\forall x_i$ we add to $\Pi(v)$ one rule $((x_i \rightarrow r_{i-1}) \wedge (y_i \rightarrow r_{i-1})) \rightarrow r_i$. Then induction on the length t of the quantifier prefix shows that v is provable in second order propositional logic iff $\Pi(v)$ implies r_t.

The case $t = 1$ has been treated in proposition 2. For $t > 1$, t even, we note that $v = \exists x_t\, Q_{t-1} \ldots Q_1 D$ is provable iff either $v_0 = Q_{t-1} \ldots Q_1 D(x_t)$ or $v_1 = Q_{t-1} \ldots Q_1 D(\neg x_t)$ is provable and $v = \forall x_t\, Q_{t-1} \ldots Q_1 D$ is provable iff both v_0 and v_1 are provable. To these formulas the induction hypothesis applies, so we just have two show that success of $\Pi(v)$? r_t is equivalent to success of either $\Pi(v_0)$? r_{t-1} or $\Pi(v_1)$? r_{t-1} resp. success of both $\Pi(v_0)$? r_{t-1} and $\Pi(v_1)$? r_{t-1}:

The second equivalence has already been proved as lemma 2; the proof of the first equivalence is a dual to that proof using the fact that in intuitionistic logic the formula r_n is derivable from the formulae $((C_1 \rightarrow r_0) \wedge \ldots \wedge (C_n \rightarrow r_0)) \rightarrow r_{n-1}$, $(x \rightarrow r_{n-1}) \rightarrow r_n$, and $(x \wedge C_1) \rightarrow r_0, \ldots, (x \wedge C_n) \rightarrow r_0$. The proof for odd t again is a dual to the previous proof.

Our results established so far finally yield the

Theorem:

The relation P ? g, where P is a well founded program is PSPACE-complete.

Proof: The collection of all provable prenex formulas of second order propositional logic is PSPACE-hard (cf. [3]) and these formulas are all coded at some stage of the above construction using well founded programs.

Thus we showed that even a class of very perspicuous well founded programs embodies the full strength of the entire language and exhausts all of the polynomial hierarchy. The image of this hierarchy in the hierarchy of fragments of our programming language, however, is not one-to-one but somewhat fuzzy—the two hierarchies, although being cofinal differ by two stages.

References:
[1] Gabbay, D.M & U. Reyle: N-Prolog. Part 1. In: *Journal of Logic Programming* **2**(1984), 319–355
[2] Statman, R.: Intuitionistic logic is polynomial-space complete. In: *Theoretical Computer Science* **9**(1979), 67–72
[3] Stockmeyer, L.J.: The polynomial time hierarchy. In: *Theoretical computer science* **3**(1976), 1–22