The inverse method and backward chaining in ordered logic
Request For Logic (RFL) #5
Robert J. Simmons
December 6, 2009

1 Introduction

Chaudhuri et al. have observed that, given a set of Horn clauses Ψ and a query, g, the set of SLD-derivations (Ψ ⊢ g) is isomorphic to the set of focused linear logic proofs (Ψ; · ⊢ g) where the bias of every atom is set to positive [CPP]. You could argue that this is lucky: the inverse method happens to start with the initial sequent (g ⊢ g) and then grow down on the left. The derivation of a sequent (a1...an ⊢ g) represents the possibility that SLD search can get to the point were g is provable if the subgoals a1... an are all shown to be provable.

To be clear, the “luck” here is that the multiset-like resource management of linear logic maps correctly onto SLD resolution. Persistent logic would correspond to SLD with a degree of loop-checking, and ordered logic and a bit of creativity could model either SLD resolution or strict, left-to-right Prolog style proof search.

In this note, I explore backwards and forwards proof search in ordered logic with two small-but-critical extensions. One is that Horn clauses can have two forms, expansive (A -• B • C) or contractive (A • B -• C). The other is that goals themselves get to be Horn clauses, not individual atomic propositions. This lends itself well to representing Chomsky normal form grammars, which will be our running example. There are eight possibilities to consider, corresponding to:

* Positive or negative atoms
* Contractive or expansive
* Backward proof search or forward inverse-method search

As it turns out, the negative fragment doesn’t seem terribly interesting, and there is no longer the sharp contrast between using positive and negative polarities for atoms - though in almost all cases it seems more reasonable to use positive atoms. Rather, the interesting relationship between backward chaining proof search and the inverse method plays out between contractive and expansive rules - forward chaining on contractive rules is related to backward chaining on expansive rules, and vice versa. This should be generalizable to an extended Horn fragment that includes multiple-premise and multiple-conclusion rules like (A • B -• C • D).

2 Extended example

Our running example will be an ambiguous parsing for arithmetic expressions given by the following rules:

```
e -> n
e -> e+e
```

We’ll represent numbers N with the atomic proposition N and the operator “+” with the atomic proposition P. These can be interpreted by expansive rules like (E -• E • P • E) or by contractive rules (E • P • E -• E).

Expansive rules

```
E  -•  N
E  -•  E • P • E
Goal : E  -•  N • P • N
```

All atoms positive

```
E  -•  ↑(N)
E  -•  ↑(E • P • E)
Goal:  E  -•  ↑(N • P • N)
```

Synthetic rules:

```
ΩL, E, ΩR ⊢ J
-------------- r1
ΩL, N, ΩR ⊢ J

ΩL, E, P, E, ΩR ⊢ J
-------------------- r2
ΩL, E, ΩR ⊢ J

-------------------- rgoal
N, P, N ⊢ N • P • N
```

Synthetic proof:

```
-------------------- rgoal
N, P, N ⊢ N • P • N
-------------------- r1
N, P, E ⊢ N • P • N
-------------------- r1
E, P, E ⊢ N • P • N
-------------------- r2
E ⊢ N • P • N
```

Backward chaining (bad idea!)

1) E ⊢ N • P • N   (goal)
2) E, P, E ⊢ N • P • N   (rule r2 on 1)
3) N, P, E ⊢ N • P • N   (rule r1 on 2)
4) N, P, N ⊢ N • P • N   (rule r1 on 3)
5) Done!   (rule rgoal on 4)

There are an unlimited number of sequent sets that can be derived in this way, such as (N, P, E, P, E ⊢ N • P • N), which can be derived by rule r2 on 3.

Inverse method (good idea!)

1) N, P, N ⊢ N • P • N   (initial sequent derived from the goal)
2) N, P, E ⊢ N • P • N   (rule r1 on 1)
3) E, P, E ⊢ N • P • N   (rule r1 on 2)
4) E ⊢ N • P • N   (rule r2 on 3 - done!)

There are a limited number of additional sequents that can be derived - (E, P, N ⊢ N • P • N) is the only one we didn’t mention here.

All atoms negative

```
↓E  -•  ↑(↓N)
↓E  -•  ↑(↓E • ↓P • ↓E)
Goal:  ↓E  -•  ↑(↓N • ↓P • ↓N)

Synthetic rules:

Ω ⊢ E    ΩL, N, ΩR ⊢ J
----------------------- r1
ΩL, Ω, ΩR ⊢ J

Ω ⊢ E    ΩL, E, P, E, ΩR ⊢ J
------------------------------ r2
ΩL, Ω, ΩR ⊢ J

------ rN
N ⊢ N

------ rE
E ⊢ E

------ rP
P ⊢ P

Ω₁ ⊢ N     Ω₂ ⊢ P     Ω₃ ⊢ N
------------------------------- rgoal
Ω₁, Ω₂, Ω₃ ⊢ ↓N • ↓P • ↓N

Synthetic proof:

                                   ------- rN  ------- rP  ------- rN
                                    N ⊢ N      P ⊢ P       N ⊢ N
                       ------- rE  ------------------------------ rgoal
                        E ⊢ E      N, P, N ⊢ ↑N • ↑P • ↑N
           ------- rE  ------------------------------------- r1
            E ⊢ E      N, P, E ⊢ ↑N • ↑P • ↑N
------ rE  ------------------------------------ r1
E ⊢ E      E, P, E ⊢ ↑N • ↑P • ↑N
---------------------------------------- r2
E ⊢ ↑N • ↑P • ↑N
```

This is basically going to be a less-efficient version of the story with positive-polarity connectives.

Contractive rules

```
N  -•  E
E • P • E  -•  E
Goal : N • P • N  -•  E
```

All atoms positive

```
N  -•  ↑(E)
E • P • E  -•  ↑(E)
Goal : N • P • N  -•  ↑(E)
```

```
Synthetic connectives:

ΩL, E, ΩR ⊢ J
-------------- r1
ΩL, N, ΩR ⊢ J

ΩL, E, ΩR ⊢ J
--------------------- r2
ΩL, E, P, E, ΩR ⊢ J

------ rgoal
E ⊢ E

Synthetic proof:

------ rgoal
E ⊢ E
------------ r2
E, P, E ⊢ E
------------ r1
N, P, E ⊢ E
------------ r1
N, P, N ⊢ E

Backward chaining (good idea!)

1) N, P, N ⊢ E   (goal)
2) N, P, E ⊢ E   (rule r2 on 1)
3) E, P, E ⊢ E   (rule r1 on 2)
4) E ⊢ E   (rule r1 on 3)
5) Done!   (rule rgoal on 4)
```

There are a limited number of goal sets that can be derived in this way - (E, P, N ⊢ E) is the only one we didn't mention here.

```
Inverse method (bad idea?)

1) E ⊢ E   (rgoal)
2) E, P, E ⊢ E   (rule r2 on 1)
3) N, P, E ⊢ E   (rule r1 on 2)
4) N, P, N ⊢ E   (rule r1 on 2 - done!)
```

There are a seemingly infinite number of goal sets that can be derived in this way - unless some pretty strong intelligence about subformulas is applied, (N, P, E, P, E ⊢ E) seems to be derivable by rule r2 on 2.

All atoms negative

Here we had to make a decision: since we knew we had contractive rules we didn't add any shifts at the head, but if we wanted to generalize this then we would have rules like (↓N -• ↑(↓N)) instead.

```
↓N  -•  E
↓E • ↓P • ↓E  -•  E
Goal : ↓N • ↓P • ↓N  -•  E
```

Synthetic rules:

```
Ω ⊢ N
------ r1
Ω ⊢ E

Ω1 ⊢ E     Ω2 ⊢ P     Ω3 ⊢ E
------------------------------- r2
Ω1, Ω2, Ω3 ⊢ E

------ rN
N ⊢ N

------ rP
P ⊢ P
```

(I don't *think* that E ⊢ E is a derived initial sequent! Not certain.)

Synthetic proof:

```
------ rN              ------ rN
N ⊢ N                  N ⊢ N
------ r1  ------ rP  ------ r1
N ⊢ E      P ⊢ P      N ⊢ E
---------------------------- r2
N, P, N ⊢ E
```

Backward chaining will need to use resource management in a clever way, but because it is forced to split up the context will be able to explore only a limited number of possibilities.

The inverse method will need similar intelligence to before in order to avoid coming up with arbitrarily large contexts.

3 References

[CPP] K. Chaudhuri, F. Pfenning, and G. Price, "A logical characterization of forward and backward chaining in the inverse method," Journal of Automated Reasoning, vol. 40, no. 2, pp. 133-177, March 2008. [Online]. Available: http://dx.doi.org/10.1007/s10817-007-9091-0