Constructive Logic (15-317), Spring 2021
Recitation 13: Datalog with chains (4-28-2021)

1 Forward Logic with Inference Rules

Task 1. Give the inference rules for a forward logic program length(l, n) which derives the atom **no** if and only if n (which satisfies the *nat* predicate) is not the length of list l. You may assume that n and l are ground.

Solution 1:

$$\frac{length([],\ s(N))}{no}\ zero_l \qquad \frac{length([X \mid L],\ z)}{no}\ zero_n \qquad \frac{length([X \mid L],\ s(N))}{length(L,\ N)}\ dec$$

Task 2. Recall the grammar representing natural numbers:

$$n ::= z \mid s(n)$$

Give the inference rules for a forward logic program factor(m, n) which derives the atom **no** if and only if m does not evenly divide n. You may assume that m and n are ground.

Solution 2:

$$\frac{factor(m,n)}{div(m,n,m)}\ def \qquad \frac{div(s(m),s(n),d)}{div(m,n,d)}\ decr \qquad \frac{div(z,s(n),d)}{div(d,s(n),d)}\ rep \qquad \frac{div(s(m),z,d)}{no}\ nz$$

2 Functional Evaluation with Forward Chaining

Consider the language of the untyped lambda calculus.

$$e ::= x \mid \lambda x.e \mid e_1 e_2$$

We can write a set of rules using three predicates

$\mathsf{eval}(e)$	evaluate e
$e \mapsto^* e'$	e reduces to e'
$e \hookrightarrow v$	e evaluates to v

so that we can evaluate e with forward chaining, by seeding the system with $\mathsf{eval}(e)$ and waiting for a fact of the form $e \hookrightarrow v$ to appear.

Task 3. Define such a set of rules.

Solution 3: See `http://www.cs.cmu.edu/~fp/courses/lp/lectures/20-bottomup.pdf`.

3 Return to Focusing

Recall the rules for even and odd natural numbers:

$$\frac{}{even(z)}\,ev_z \qquad \frac{odd(N)}{even(s(N))}\,ev_s \qquad \frac{even(N)}{odd(s(N))}\,od_s$$

Task 4. Give a proof, using the focusing rules, of odd($s(s(s(z)))$).

Solution 4: Define Γ_{eo} = even(z), $\forall n$. odd(n) $\supset$ even(s(n)), $\forall n$. even(n) $\supset$ odd(s(n)).
Our goal sequent is $\Gamma_{eo} \longrightarrow$ even(s(s(z)))).

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{id}
\UnaryInfC{$\Gamma_{eo}, [even(s(s(z)))] \longrightarrow even(s(s(z)))$}
\AxiomC{}
\RightLabel{id}
\UnaryInfC{$\Gamma_{eo}, [odd(s(z))] \longrightarrow odd(s(z))$}
\AxiomC{}
\RightLabel{id}
\UnaryInfC{$\Gamma_{eo}, [even(z)] \longrightarrow even(z)$}
\RightLabel{$focusL$}
\UnaryInfC{$\Gamma_{eo} \longrightarrow even(z)$}
\RightLabel{$blurR$}
\UnaryInfC{$\Gamma_{eo} \longrightarrow [even(z)]$}
\RightLabel{$\supset L$}
\BinaryInfC{$\Gamma_{eo}, [even(z) \supset odd(s(z))] \longrightarrow odd(s(z))$}
\RightLabel{$\forall L$}
\UnaryInfC{$\Gamma_{eo}, [\forall n.even(n) \supset odd(s(n))] \longrightarrow odd(s(z))$}
\RightLabel{$focusL$}
\UnaryInfC{$\Gamma_{eo} \longrightarrow odd(s(z))$}
\RightLabel{$blurR$}
\UnaryInfC{$\Gamma_{eo} \longrightarrow [odd(s(z))]$}
\RightLabel{$\supset L$}
\BinaryInfC{$\Gamma_{eo}, [odd(s(z)) \supset even(s(s(z)))] \longrightarrow even(s(s(z)))$}
\RightLabel{$\forall L$}
\UnaryInfC{$\Gamma_{eo}, [\forall n.odd(n) \supset even(s(n))] \longrightarrow even(s(s(z)))$}
\RightLabel{$focusL$}
\UnaryInfC{$\Gamma_{eo} \longrightarrow even(s(s(z)))$}
\end{prooftree}
$$