POPL 2 (2020-04-15)

Srijith P K

Induction

- Logic programs are particularly amenable to formal reasoning.
  - prove that our programs won’t crash, or that they terminate, or that they satisfy given specifications
- logic program has multiple interpretations.
- set of inference rules to deduce logical truths.
  - order in which the rules are written down, or the order in which the premisses to a rule are listed, are completely irrelevant:
- proof search follows a fixed strategy
  - both the order of the rules and the order of the premisses of the rules play a significant role in program termination
  - Operational semantics

Soundness and completeness

- Soundness of operational semantics : if a query A succeeds with a substitution θ, then the result of applying substitution θ to A (A θ) is true under logical semantics.
  - Negation is an exception :
  - ?- \+(X = a) always fails => there is no term t such that t != a for the constant a.
- *completeness* of the operational semantics: if there is an instance of the query A that has a proof, then the query should succeed.
  - This does not hold, since logic programs do not necessarily terminate even if there is a proof.
- *non-deterministic completeness* says that if the interpreter were always allowed to choose which rule to use next rather than having to use the first applicable one, then the interpreter would be complete.

Soundness and completeness

- *completeness* of the operational semantics: if there is an instance of the query A that has a proof, then the query should succeed.
  - This does not hold, since logic programs do not necessarily terminate even if there is a proof.
- *non-deterministic completeness* says that if the interpreter were always allowed to choose which rule to use next rather than having to use the first applicable one, then the interpreter would be complete.
- Pure logic programs are complete in this sense. This is important because it allows us to interpret finite failure as falsehood: if the interpreter returns with the answer ‘no’ it has explored all possible choices.

Rule Induction

$$\frac{}{\mathsf{even}(\mathsf{z})}\ \mathsf{evz} \qquad \frac{\mathsf{even}(N)}{\mathsf{even}(\mathsf{s}(\mathsf{s}(N)))}\ \mathsf{evs}$$

$$\frac{}{\mathsf{plus}(\mathsf{z}, N, N)}\ \mathsf{pz} \qquad \frac{\mathsf{plus}(M, N, P)}{\mathsf{plus}(\mathsf{s}(M), N, \mathsf{s}(P))}\ \mathsf{ps}$$

- Prove sum of 2 even numbers is even
  - *For any* m*,* n*, and* p*, if* even(m) *and* even(n) *and* plus(m, n, p) *then* even(p)*.*
  - *For any* m*,* n*, if* even(m) *and* even(n) *then there exists a* p *such that* plus(m, n, p) *and* even(p)*.*

Rule Induction

- *For any* m, n, *if* even(m) *and* even(n) *then there exists a* p *such that* plus(m, n, p) *and* even(p).

Case: $\mathcal{D} = \dfrac{}{\mathsf{even}(\mathsf{z})}$ evz where $m = \mathsf{z}$.

$$\begin{array}{lr} \mathsf{even}(n) & \text{Assumption} \\ \mathsf{plus}(\mathsf{z}, n, n) & \text{By rule pz} \\ \text{There exists } p \text{ such that } \mathsf{plus}(\mathsf{z}, n, p) \text{ and } \mathsf{even}(p) & \text{Choosing } p = n \end{array}$$

Case: $\mathcal{D} = \dfrac{\begin{array}{c}\mathcal{D}' \\ \mathsf{even}(m')\end{array}}{\mathsf{even}(\mathsf{s}(\mathsf{s}(m')))}$ evs where $m = \mathsf{s}(\mathsf{s}(m'))$.

$$\begin{array}{lr} \mathsf{even}(n) & \text{Assumption} \\ \mathsf{plus}(m', n, p') \text{ and } \mathsf{even}(p') \text{ for some } p' & \text{By ind. hyp. on } \mathcal{D}' \\ \mathsf{plus}(\mathsf{s}(m'), n, \mathsf{s}(p')) & \text{By rule ps} \\ \mathsf{plus}(\mathsf{s}(\mathsf{s}(m')), n, \mathsf{s}(\mathsf{s}(p'))) & \text{By rule ps} \\ \mathsf{even}(\mathsf{s}(\mathsf{s}(p'))) & \text{By rule evs} \\ \text{There exists } p \text{ such that } \mathsf{plus}(\mathsf{s}(\mathsf{s}(m')), n, p) \text{ and } \mathsf{even}(p) & \\ & \text{Choosing } p = \mathsf{s}(\mathsf{s}(p')). \end{array}$$

Operational semantics as proof

- logic programming search has some severe restrictions so that it is usable as a programming language
- restrictions are placed both on the forms of programs and the forms of queries.
- queries are purely *existential*: we ask whether there *exists* some instantiation of the variables ?- plus(s(z), s(s(z)), P)
- theorem is primarily *universal*

Inversion

- if the proposition matches the conclusion of several rules, we can split the proof into cases, considering each one in turn
- (a) missed cases that should have been considered, and (b) incorrect applications of inversion.

Inversion

$$\frac{}{\mathsf{append}([\,], ys, ys)}\ \mathsf{apnil} \qquad \frac{\mathsf{append}(xs, ys, zs)}{\mathsf{append}([x|xs], ys, [x|zs])}\ \mathsf{apcons}$$

- *For all* xs *and* zs *and for all* ys *and* ys’, *if* append(xs, ys, zs) *and*
- append(xs, ys’, zs) *then* ys = ys.’

Case: $\mathcal{D} = \dfrac{}{\mathsf{append}([\,], ys, ys)}$ where $xs = [\,]$ and $zs = ys$.

$$\begin{array}{lr} \mathsf{append}([\,], ys', ys) & \text{Given deduction } \mathcal{E} \\ ys' = ys & \text{By inversion on } \mathcal{E} \text{ (rule } \mathsf{apnil}\text{)} \end{array}$$

Case: $\mathcal{D} = \dfrac{\begin{matrix}\mathcal{D}_1 \\ \mathsf{append}(xs_1, ys, zs_1)\end{matrix}}{\mathsf{append}([x|xs_1], ys, [x|zs_1])}$ where $xs = [xs|xs_1]$, $zs = [xs|zs_1]$.

$$\begin{array}{lr} \mathsf{append}([x|xs_1], ys', [x|zs_1]) & \text{Given deduction } \mathcal{E} \\ \mathsf{append}(xs_1, ys', zs_1) & \text{By inversion on } \mathcal{E} \text{ (rule } \mathsf{apcons}\text{)} \\ ys = ys' & \text{By ind. hyp. on } \mathcal{D}_1 \end{array}$$

Operational properties

- specification of the predicate digit for decimal digits in unary notation, that is, natural numbers between 0 and 9.

$$\frac{}{\mathsf{digit}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{z}))))))))))} \qquad \frac{\mathsf{digit}(\mathsf{s}(N))}{\mathsf{digit}(N)}$$

- deduce that z is a digit by using the second rule nine times (working bottom up) and then closing of the deduction with the first rule.
- *Any query* ?- digit(n) *for* n > 9 *will not terminate.*

Operational properties

- specification of the predicate digit for decimal digits in unary notation, that is, natural numbers between 0 and 9.

$$\frac{}{\mathsf{digit}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{z}))))))))))} \qquad \frac{\mathsf{digit}(\mathsf{s}(N))}{\mathsf{digit}(N)}$$

Any query ?- digit(n) *for* n > 9 *will not terminate.*

- **Proof:** By induction on the computation. If n > 9, then the first clause cannot apply. Therefore, the goal digit(n)must be reduced to the subgoal digit(s(n)) by the second rule. But s(n) > 9 if n > 9, so by induction hypothesis the subgoal will not terminate. Therefore the original goal also does not terminate.