Abstract submitted for the 1st World Congress on Universal Logic (UNILOG 2005), Montreux, Switzerland, March 31 – April 3, 2005

Generalized Rules, Direct Negation, and Definitional Reflection

Peter Schroeder-Heister
Wilhelm-Schickard-Institut, Universität Tübingen
Sand 13, 72076 Tübingen, Germany
psh@informatik.uni-tuebingen.de

By "generalized rules" in logic (more specifically: in natural deduction) I mean uniform elimination rules for logical constants, given certain introduction rules. E.g., if

$$\frac{\Delta_1(p_1,\ldots,p_n)}{c(p_1,\ldots,p_n)} \quad \ldots \quad \frac{\Delta_m(p_1,\ldots,p_n)}{c(p_1,\ldots,p_n)}$$

are the introduction rules for an n-ary propositional connective c, the elimination rule may be presented uniformly as

$$\frac{c(p_1,\ldots,p_n) \qquad \begin{matrix}\Delta_1(p_1,\ldots,p_n)\\ C\end{matrix} \quad \ldots \quad \begin{matrix}\Delta_m(p_1,\ldots,p_n)\\ C\end{matrix}}{C}.$$

Obviously, this schema is modelled after the elimination rules for disjunction. Making this idea more precise requires specifying the exact form of the premisses $\Delta_i(p_1,\ldots,p_n)$ of the introduction rules. In [7] I proposed including some sort of structural implication which may be contained in the Δ's, leading to a theory of rules of higher levels.

The principle of *definitional reflection* ([3, 4, 8]) generalizes this approach. Here arbitrary clauses (with variables as in logic programming, and possibly also with embedded implication and quantification) are treated like introduction rules which can be inverted by means of this principle. Due to the presence of variables and function symbols, inversion is more complicated, the logical elimination rules just being a limiting case. At the same time it is more powerful, leading to a significant extension of logic programming, and allowing to deal with non-wellfounded phenomena such as semantical and mathematical paradoxes.

In this talk I consider the situation which arises when *direct negation* in the sense of Nelson's logic of constructible falsity [6] is added (the term "direct logic" is due to v. Kutschera [5]). Besides positive introduction rules we also have negative introduction rules for the rejection of logically compound formulas. In the more general case we would consider clauses with negated heads as in extended logic programs ([1, 2]). The logical case is relatively easy to deal with, as it is clear from the very beginning how the rejection rules for logically compound formulas should look like ([5, 9]). One would just have to add elimination rules for negated formulas. For example, in the case of implication, the rejection rule corresponding to implication introduction has the form

$$\frac{p \quad \sim q}{\sim(p \supset q)},$$

so that we would just add

$$\dfrac{\sim(p \supset q) \qquad \begin{matrix}[p & \sim q] \\ C\end{matrix}}{C}$$

as an elimination inference.

However, with generalized clauses, we would like to consider *arbitrary* positive and negative clauses in our database which are not related with each other in such a specific way. As in the theory of extended logic programming, this may even lead to inconsistent databases. In extended logic programming, no inversion principle like definitional reflection has been considered so far. If we want to add definitional reflection to a system containing both positive and negative clauses we have to address questions such as the following:

1. Due to the rejection operator we can dualize clauses, generating positive from negative clauses and vice versa. Should we distinguish between *primary* and *secondary* definitional clauses, the secondary ones being generated by dualization from the primary ones?

2. Are secondary clauses to be treated on par with primary ones, when it comes to definitional reflection?

3. How is dualization and inversion (definitional reflection) to be defined, if function constants are present, i.e. if not necessarily finitely many clauses are generated?

4. Which role do the "paradoxes of implication", in particular the absurdity principle, play in the context of dualizing definitional rules?

References

1. Damásio, C.V. & Pereira, L.M. A survey of paraconsistent semantics for logic programs. In: D.M. Gabbay & P. Smets (eds.), *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Vol. 2: Reasoning with Actual and Potential Contradictions*, Dordrecht: Kluwer 1998, 241–320.
2. Gelfond, M. & Lifschitz, V. Logic programs with classical negation. In: D.H.D. Warren & P. Szeredi (eds.), *Logic Programming: Proceedings of the 7th International Conference*, Cambridge Mass., London: MIT Press 1990, 579–597
3. Hallnäs, L. Partial inductive definitions. *Theoretical Computer Science* **87** (1991), 115-142.
4. Hallnäs, L. & Schroeder-Heister, P. A proof-theoretic approach to logic programming. II. Programs as definitions. *Journal of Logic and Computation* **1** (1990/91), 635–660.
5. Ein verallgemeinerter Widerlegungsbegriff für Gentzenkalküle, *Archiv für Mathematische Logik und Grundlagenforschung* **12** (1969), 104–118.
6. Nelson, D. Constructible falsity, *Journal of Symbolic Logic* **14** (1949), 16–26.
7. Schroeder-Heister, P. A natural extension of natural deduction, *Journal of Symbolic Logic* **49** (1984), 1284–1300.
8. Schroeder-Heister, P. Rules of definitional reflection. In: *8th Annual IEEE Symposium on Logic in Computer Science, LICS 1993*, Los Alamitos: IEEE Computer Society Press 1993, 222–232.
9. Wansing, H. *The Logic of Information Structures*, Berlin, Heidelberg: Springer 1993 (LNAI Vol. 681).