Some Comments on "A Denotational Semantics for Prolog"

BIJAN ARBAB
IBM
and
DANIEL M. BERRY
Technion

Two independently derived denotational semantics for Prolog are contrasted, Arbab and Berry's for the full language and Nicholson and Foo's for a databaseless language. Using the ideas suggested by the former, the latter can be easily extended to include the database operations.

Categories and Subject Descriptors: D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*semantics;* D.3.2 [**Programming Languages**]: Language Classifications; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*denotational semantics;* F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*logic programming*

General Terms: Languages, Theory

Additional Key Words and Phrases: Denotational semantics, logic programming, operational semantics

We read with interest the article "A Denotational Semantics for Prolog" by Nicholson and Foo in the October 1989 *TOPLAS*. We worked on that problem independently, in a largely overlapping time period. We refer the reader to our article, "Operational and Denotational Semantics of Prolog" in the November 1987 issue of the *Journal of Logic Programming*. In this article, we have given both an operational and a denotational semantics of the *full* Prolog, including the database operations, *assert* and *retract*, that were purposely not covered in Nicholson's and Foo's semantics.

We started with an operational semantics patterned after the standard Prolog interpreter. It is no particular problem to define these database operations in the operational semantics. Our denotational semantics was obtained by systematic translation from the operational semantics; this translation had no problems dealing with the database operations.

Authors' addresses: B. Arbab, IBM, Los Angeles Scientific Center, Santa Monica, CA 90406; D. M. Berry, Faculty of Computer Science, Technion, Haifa 32000, Israel.

It is interesting to note our article's conclusion, "...it is easy in retrospect to see how to define the database... feature... denotationally. One simply lets the meaning of the feature be the application of the current continuation to the components of the state that get modified by the feature." To apply this idea to Nicholson's and Foo's semantics, we would have to add the domain of continuations with databases,

$$D_c = D_b \to Q_c,$$

and would have to change the goal valuation function to have one of these new continuations as an argument

$$\mathbf{G}: \mathit{Goals} \to D_b \to D_c \to \mathit{Env} \to \mathit{Res} \to Q_c.$$

Then, the semantics of a database operation, D, either *assert* or *retract*, would be defined by an equation of the form

$$\mathbf{G}[\![D]\!]\delta\gamma\rho\phi\theta = \gamma\delta'\rho\phi\theta$$

where δ' is the extended or restricted database constructed from δ using the arguments of D.