Project title

Implementation of reducibility process for a set-theoretical Δ-language

Supervisor

Vladimir Sazonov, Logic and Computations Group,
`http://www.csc.liv.ac.uk/~sazonov/`

Subject Areas

Logic, set theory and corresponding foundations of database theory.

Brief description

The goal of this Project is *to implement reduction rules* of set-theoretical Delta-formulas. Delta-language (Δ) is considered as a query language under a set-theoretic approach to simi-tructured databases. Thus, this project would be a first step towards reduction-based implementation of a simple, basic fragment of the language (and possibly of its full version). This would also complement another already existing [4] experimental implementation of Delta of a different kind. For a student working on this project this would be a good and challenging opportunity to apply his/her programming skills and to shed new, more practical light on this direction of research on semi-structured databases.

More detailed description (cf. also [5, 6, 3, 7])

Usually set theory is formulated on the base of primitives $\in$ and $=$ with set variables as the only terms. We consider set-theoretic languages with a richer class of terms, because we are interested in explicitly definable operations (queries) over hereditarily-finite sets (database states).

We call the languages of various Bounded Set Theories (BST's) by a generic name Δ and define a *basic fragment* of them by the clauses

$$\begin{array}{rcl} \Delta\text{-}\textit{formulas} & ::= & a = b \,|\, a \in b \,|\, \varphi \,\&\, \psi \,|\, \varphi \vee \psi \,|\, \neg\varphi \,|\, \varphi \Rightarrow \psi \,|\, \forall x \in a.\varphi \,|\, \exists x \in a.\varphi \,|\, \mathbf{true} \,|\, \mathbf{false} \\ \Delta\text{-}\textit{terms} & ::= & \text{set variables } x, y, \ldots \,|\, \emptyset \,|\, \{a_1, \ldots, a_n\} \,|\, \bigcup s \,|\, \{t(x) \mid x \in a \,\&\, \varphi(x)\} \end{array}$$

where a, b, t are Δ-terms and φ, ψ are Δ-formulas, set-variable x is not free in a. Only *bounded* quantifiers are allowed in Δ.

The above clauses define the language of *basic* = *rudimentary* operations of R.O.Gandy [1] and R.B.Jensen [2], respectively. Formulas of a Δ-language involving only atomic Δ-terms (i.e. variables) are known as A.Lévy's Δ_0-formulas.

For simplicity we considered above only so called *pure* case of sets $\{a_1, \ldots, a_n\}$ consisting of elements a_i which themselves are sets, etc. More generally, for

database applications, it is important *labelled* case of sets $\{l_1 : a_1, \ldots, l_n : a_n\}$ consisting of labelled elements $l_i : a_i$, e.g., `{name:Smith,DOB:1981}`.

Consider a *reduction relation* $\twoheadrightarrow$ based on the following rules for Δ-formulas:

$\neg\mathbf{false} \twoheadrightarrow \mathbf{true}$,

$\neg\mathbf{true} \twoheadrightarrow \mathbf{false}$,

$\varphi \vee \mathbf{true} \twoheadrightarrow \mathbf{true}$,

$\mathbf{true} \vee \varphi \twoheadrightarrow \mathbf{true}$,

$\mathbf{false} \vee \mathbf{false} \twoheadrightarrow \mathbf{false}$,

$\varphi \,\&\, \mathbf{false} \twoheadrightarrow \mathbf{false}$,

$\mathbf{false} \,\&\, \varphi \twoheadrightarrow \mathbf{false}$,

$\mathbf{true} \,\&\, \mathbf{true} \twoheadrightarrow \mathbf{true}$,

$\varphi \Rightarrow \psi \twoheadrightarrow \neg\varphi \vee \psi$,

$s = t \twoheadrightarrow \forall x \in s.(x \in t) \,\&\, \forall x \in t.(x \in s)$,

$s \in t \twoheadrightarrow \exists x \in t.(s = x)$ if s is not an atomic term,

$s \in \emptyset \twoheadrightarrow \mathbf{false}$,

$s \in \{t_1, \ldots, t_n\} \twoheadrightarrow s = t_1 \vee \cdots s = t_n$,

$s \in \bigcup t \twoheadrightarrow \exists x \in t.(s \in x)$,

$s \in \{t(x) \mid x \in a \,\&\, \varphi(x)\} \twoheadrightarrow \exists x \in a.(\varphi(x) \,\&\, s = t(x))$,

$\exists y \in \emptyset.\psi(y) \twoheadrightarrow \mathbf{false}$

$\exists y \in \{t_1, \ldots t_n\}.\psi(y) \twoheadrightarrow \psi(t_1) \vee \cdots \vee \psi(t_n)$,

$\exists y \in \bigcup s.\psi(y) \twoheadrightarrow \exists x \in s.\exists y \in x.\psi(y)$,

$\exists y \in \{t(x) \mid x \in a \,\&\, \varphi(x)\}.\psi(y) \twoheadrightarrow \exists x \in a.(\varphi(x) \,\&\, \psi(t(x)))$,

$\forall y \in \emptyset.\psi(y) \twoheadrightarrow \mathbf{true}$

$\forall y \in \{t_1, \ldots, t_n\}.\psi(y) \twoheadrightarrow \psi(t_1) \,\&\, \cdots \,\&\, \psi(t_n)$,

$\forall y \in \bigcup s.\psi(y) \twoheadrightarrow \forall x \in s.\forall y \in x.\psi(y)$,

$\forall y \in \{t(x) \mid x \in a \,\&\, \varphi(x)\}.\psi(y) \twoheadrightarrow \forall x \in a.(\varphi(x) \Rightarrow \psi(t(x)))$.

Note, that formulas are considered up to renaming closed (local) variables which allows to avoid collisions between free (global) and closed (local) variables. Moreover, substitution of t_i in $\psi(x)$ for the variable x above may also assume renaming some closed variables in ψ to avoid collisions with free variables of substituted terms. Evidently, $\varphi \twoheadrightarrow \psi$ implies $\varphi \Leftrightarrow \psi$. Reduction, in a sense, simplifies ("unravels") a formula (by eliminating the complex subformulas and subterms) without changing its meaning.

As a simple illustration, consider the following reduction terminating by Δ_0-formula:

$$\begin{aligned}\bigcup z = \{x, y\} \quad &\twoheadrightarrow \quad \forall u \in \bigcup z.(u \in \{x, y\}) \;\&\; \forall u \in \{x, y\}.(u \in \bigcup z) \\ &\twoheadrightarrow \quad \forall u \in \bigcup z.(u = x \vee u = y) \;\&\; (x \in \bigcup z \;\&\; y \in \bigcup z) \\ &\twoheadrightarrow \quad \forall v \in z. \forall u \in v.(u = x \vee u = y) \;\&\; (\exists v \in z. x \in v \;\&\; \exists v \in z. y \in v).\end{aligned}$$

Theorem 1 (cf. [6]) *Any Δ-formula φ is reducible to a unique Δ_0-formula φ^0, a* **normal form** *of φ (φ^0 coincides with φ for φ in Δ_0). In particular, if φ has no free variables, it is reducible to* **true** *or* **false**.

Thus, **the concrete goal of implementation consists in automated and machine-human interacting process of reducing any Δ-formula to Δ_0-normal form, or just to true or false**. The user may choose one of the suggested parts of a current Δ-formula to reduce. In particular, an appropriate interface allowing to the user to make this choice could be designed. As an option the user could decide that the system would make a random choice automatically, or a particular strategies of making a choice could be opted. **More advanced and interesting goal could be to use these reduction process for evaluating Δ-queries to semistructured databases represented set-theoretically.**

Background requirements

Familiarity with elements of logic, simplest set-theoretic concepts, and having sufficiently advanced programming skills.

References

[1] Gandy, R.O.: 1974, 'Set-theoretic functions for elementary syntax', in: *Proc. Symp. in Pure Math.* **Vol. 13, Part II**, pp. 103–126.

[2] Jensen, R.B.: 1972, 'The fine structure of the constructible hierarchy', *Ann. Math. Logic* **4**, pp. 229-308.

[3] Lisitsa, A., and Sazonov, V., Bounded Hyper-set Theory and Web-like Data Bases. *Computational Logic and Proof Theory, 5th Kurt Gödel Colloquium, KGC'97*, Springer LNCS Vol. 1289, 1997, pp. 172–185.

[4] Molyneux R., Hyperset Approach to Semi-structured Databases and the Experimental Implementation of the Query Language Delta, PhD Thesis, 2008, `http://www.csc.liv.ac.uk/~molyneux/t/`

[5] Sazonov, V.Yu.: 1993, 'Hereditarily-finite sets, data bases and polynomial-time computability', *Theoretical Computer Science* **Vol. 119**, Elsevier, pp. 187–214.

[6] Sazonov, V., On Bounded Set Theory. Invited talk on the *10th International Congress on Logic, Methodology and Philosophy of Sciences, in Volume I: Logic and Scientific Method*, Kluwer Academic Publishers, 1997, pp. 85–103

[7] V.Yu.Sazonov, Querying Hyperset/Web-Like Databases, Logic Journal of IGPL, 2006; 14(5): 785-814. doi:10.1093/jigpal/jzl010