APPENDIX
GLOSSARY OF TERMS

This appendix contains a glossary of terms related to logic programming and deductive database terms. It enjoys hyperlinks for the terms in the definitions that are included in the glossary (if the same term occurs several times in the same definition, only the first occurrence is hyperlinked). Clicking in an hyperlink (a term in a red box) redirects to the referenced term in the glossary. References to the attached bibliography are also enclosed with green boxes as hyperlinks. Finally, hyperlinks are also added in bibliographic entries themselves (as blue boxes) linking to the on-line resources (either papers or books) if available. The printed appendix does not include such coloured boxes. Some definitions or parts of them are taken from public repositories. A good and classical reference for logic programming and Prolog is [1], while for deductive databases are [2] and [3].

- **Aggregate Predicate**. An aggregate predicate is a meta-predicate that has, at least, a predicate as an argument.
- **Answer**. Outcome for a goal in an interactive system. An answer in a Prolog system consists of a substitution for each alternative. An answer in a deductive database system consists of a (possibly multi-)set of ground terms. In a system with fuzzy features it is usually accompanied by a truth degree.
- **Arity**. Number of arguments of either a term or a predicate.
- **Assertion**. An assertion is either a program directive or a constraint.
- **Atom**. An atom is a term with a relation name for its functor name.
- **Body**. Right hand side (antecedent) of the implication in a rule.
- **Clause**. A first order logic formula which is a disjunction of literals. Variables occurring in a clause are assumed universally quantified. *Horn clauses* are clauses with at most one positive literal. Clauses can be written in an implication form: $h \leftarrow g_1 \wedge \cdots \wedge g_n$, where h is known as the head of the clause and $g_1 \wedge \cdots \wedge g_n$ is known as the body of the clause. A textual clause in a system implementation uses the neck symbol (`:-`) for the implication, uses commas (`,`) for the conjunctions, and ends with a dot (`.`).
- **Constraint**. A query that, if can be successfully solved, indicates an inconsistent database instance. A constraint is a clause with a false consequent. A textual constraint in a system implementation is written as a textual clause with no head. They are also known as specific cases of assertions.
- **Database Instance**. Facts (*aka* table rows in relational databases) and clauses (*aka* views in relational databases) defining all the terms for every relation, excluding the database schema.
- **Database Schema**. Constraints for a given database (types, primary keys, ...).
- **Deductive Database**. A deductive database is a logic program with syntactic restrictions to ensure safety (finiteness and termination).
- **Deductive Database System**. A database management system which incorporates a deductive engine to make deductions from stored rules and facts.
- **Derivation**. A formal proof or derivation is a sequence of sentences each of which is an axiom or follows from the preceding sentences in the sequence by a rule of inference.
- **Expression**. A term that can be evaluated.
- **Extensional Database**. Part of the database which defines relations in terms of tuples. *See also Intensional Database.*
- **Fact**. A clause with a true antecedent. A textual fact in a system implementation omits the implication and the antecedent, and ends with a dot.
- **Functor**. Name and arity of the syntax tree root of a term, written as *Name/Arity*.
- **Goal**. A clause of the form $\leftarrow\ g_1 \wedge \cdots \wedge g_n$ (i.e., a clause without a head). It is understood as a query $\exists(g_1 \wedge \cdots \wedge g_n)$ submitted to a logic system. So, it is the root of a resolution as known in Prolog. Each g_i is called a subgoal. It is also known as a *query* in a deductive database setting.
- **Ground**. Adjective referred to terms and instantiations so that no substitution can make them less general.
- **Head**. Left hand side (consecuent) of the implication in a clause.
- **Instantiation**. The application of a substitution to a syntactic expression to make it more specific. For example, given a substitution $\theta = \{x/a\}$, the application of θ to an atomic formula $A(x, y)$, denoted $A(x, y)\theta$, is the atomic formula $A(a, y)$, which is called an *instance* of $A(x, y)$.
- **Intensional Database**. Part of the database which defines relations in terms of other relations. *See also Extensional Database.*
- **Literal**. A positive (i.e., non-negative) or negative atomic formula.
- **Logic Program**. Collection of clauses. Also known as (deductive) database in the deductive database setting.
- **Logic Variable**. A logic variable is a variable that can be instantiated only once in a derivation.
- **Meta-predicate**. A meta-predicate is a predicate in higher-order logic. Whereas pure logic programming deals only with first-order logic, most current logic programming systems deal with meta-predicates.
- **Predicate**. A set of clauses with the same symbol and arity at the root of the head of the rules or facts which define it.
- **Predicate Dependency Graph**. A predicate dependency graph (PDG) [4] shows both the positive and negative dependencies between predicates in the program. Each node in this graph is a program predicate symbol. Arcs come from each predicate in a rule body to its rule predicate. If a predicate occurs as an aggregate argument, its outcoming arc is labelled as negative, and positive otherwise.
- **Query**. A goal as known in a deductive database setting.
- **Relation**. A predicate as known in the deductive database

setting.

- **Resolution**. Proof method for logic programs developed by Robinson in 1965 [5]. Resolution is a rule of inference leading to a refutation theorem-proving technique for sentences in propositional logic and first-order logic.
- **Rule**. A clause written as a conditional formula with a head and a body. This denomination is commonly used both in logic programming and in the deductive database setting.
- **Safety**. Property of rules that obey syntax restrictions for ensuring finiteness and termination [6], [7].
- **SLD-Resolution**. Basic inference system for logic programming. SLD-resolution stands for Selection-function-driven Linear resolution for Definite clauses resolution [8]. Practical systems are based on a variant of SLDNF-resolution (SLD-resolution with negation as failure) [9].
- **Stratification**. A stratification collects predicates into numbered strata so that, given the function $str(\Pi, \mathtt{p})$ which assigns a stratum number to predicate $\mathtt{p}$ in a database Π, then for a positive arc $\mathtt{p} \leftarrow \mathtt{q}$, $str(\Pi, \mathtt{p}) \leq str(\Pi, \mathtt{q})$, and for a negative arc $\mathtt{p} \overset{\neg}{\leftarrow} \mathtt{q}$, $str(\Pi, \mathtt{p}) < str(\Pi, \mathtt{q})$.
- **Substitution**. Set of pairs x/y, indicating that the symbol x is syntactically substituted by y.
- **Subsumption**. A concept closely related to instantiation. An atom A subsumes another B if there exists a substitution σ such that $A\sigma = B$. That is, if B is an instance of A.
- **Tabling**. Tabling is an implementation technique based on dynamic programming for enhancing performance and termination properties of logic programs [10]. With its roots in memoization [11], this technique is based on storing already-computed results to be reused for the same or similar calls instead of recomputing them.
- **Term**. A term is a structure $f(t_1, \ldots, t_n)$ with arity greater than 0, whose arguments t_i can be constants, logic variables or other terms. f/n is known as the functor of the term.
- **Unification**. Unification is an algorithmic process of solving equations between symbolic expressions. A unification problem is an equation set for which a substitution is a solution.

REFERENCES

[1] K. Apt, *From logic programming to Prolog*, ser. Prentice-Hall international series in computer science. Prentice Hall, 1997. [Online]. Available: https://books.google.es/books?id=8Y1QAAAAMAAJ

[2] S. Ceri, G. Gottlob, and L. Tanca, "What you Always Wanted to Know About Datalog (And Never Dared to Ask)," *IEEE Transactions on Knowledge Data Engineering*, vol. 1, no. 1, pp. 146–166, 1989. [Online]. Available: http://dblp.uni-trier.de/db/journals/tkde/tkde1.html#CeriGT89

[3] T. J. Green, S. S. Huang, B. T. Loo, and W. Zhou, "Datalog and recursive query processing." *Foundations and Trends in Databases*, vol. 5, no. 2, pp. 105–195, 2012. [Online]. Available: http://blogs.evergreen.edu/sosw/files/2014/04/Green-Vol5-DBS-017.pdf

[4] C. Zaniolo, S. Ceri, C. Faloutsos, R. T. Snodgrass, V. S. Subrahmanian, and R. Zicari, *Advanced Database Systems*. Morgan Kaufmann, 1997.

[5] J. A. Robinson, "A machine-oriented logic based on the resolution principle," *J. ACM*, vol. 12, no. 1, pp. 23–41, Jan. 1965. [Online]. Available: http://doi.acm.org/10.1145/321250.321253

[6] J. D. Ullman, *Database and Knowledge-Base Systems, Vols. I (Classical Database Systems) and II (The New Technologies)*. Computer Science Press, 1988.

[7] S. Cohen, J. Y. Gil, and E. Zarivach, *Datalog Programs over Infinite Databases, Revisited*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 32–47. [Online]. Available: https://pdfs.semanticscholar.org/2d5a/accbb6bfe6e4a20b02ca9ff75105c43e61bb.pdf

[8] R. Kowalski and D. Kuehner, "Linear resolution with selection function," *Artificial Intelligence*, vol. 2, 1971.

[9] K. R. Apt and R. N. Bol, "Logic programming and negation: A survey," *J. Log. Program.*, vol. 19/20, pp. 9–71, 1994.

[10] S. Verbaeten, K. Sagonas, and D. De Schreye, "Termination proofs for logic programs with tabling," *ACM Transactions on Computational Logic*, vol. 2, no. 1, pp. 57–92, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?doid=371282.371357

[11] S. W. Dietrich, "Extension tables: Memo relations in logic programming," in *IEEE Symp. on Logic Programming*, 1987, pp. 264–272. [Online]. Available: http://www.public.asu.edu/~dietrich/publications/ExtensionTablesMemoRelations.pdf