

## Chapter 2

# Related Work

Rule programs as handled by Business Rules Management Systems (BRMS) are the focus of this book. In the present chapter we relate these programs to other programming paradigms, whether rule-based or not. We concentrate on the similarities and differences that have an impact on our objective, which is to help understand the effects, and prove correctness properties on the executions of rule programs as handled by BRMS.

The first section of this chapter reviews several rule-based paradigms, which allows us to relate them more or less closely to rule programs in BRMS. Then we discuss some formalization and verification approaches of related rule-based paradigms, but also of concurrent programs.

### 2.1 Rule-Based Paradigms: Model vs. State

Rule programs as handled by Business Rules Management Systems are by far not the only occurrence of the concept of rules in computer science. To sort out the numerous paradigms that give an important role to rules, we adopt a schematic classification between those where rules are used to model knowledge about a given domain, and those where they are used to perform a computation. We refer to the former as *model-oriented* rule-based paradigms, and to the latter as *state-oriented* ones.

This classification, which follows a similar one proposed by Victor Vianu [79], has the advantage of concentrating on the intent behind using rules. This contrasts, for example, with a common distinction based on whether the rules are used in backward- or forward-chaining. In the latter classification, the focus is on the way in which rules are put into action: this gives little, if any, information on their actual role.

The distinction between model- and state-oriented rule-based paradigms should not hide the fact that the usages of rules are not clear cut. In particular, and as

suggested by their genealogy briefly presented in Sect. 1.2.2, BRMS inherit from paradigms in both categories.

We conclude this section with a brief review of some other cases where the two categories are combined.

### 2.1.1 *Model-Oriented Rules*

In rule-based paradigms to which we refer as model-oriented, the intent is to build a description of a given body of knowledge. Rules are used to provide parts of this model intensionally. They are commonly gathered in rule *bases*, in contrast with state-oriented paradigms which often collect them in rule *programs*.

Model-oriented rules are of course abundantly used in the large area of artificial intelligence that studies knowledge representation; see for example [42] or [75] for a survey. Deriving from this area are business rules in the sense of the description given in Sect. 1.2.2, which are used to implement models of business policies.

As a consequence, Business Rules Management Systems naturally inherit from these rules. This inheritance materializes primarily in the authoring components. By authoring components, we mean the tools that provide business experts with rule languages that are as close as possible to their domain-specific vocabulary. BRMS will for example use standards such as the OMG one on Semantics of Business Vocabulary and Business Rules (SBVR) [77].

Another important occurrence of model-oriented rules is to be found in deductive databases, or deductive rules in databases [47, 50]. Here also rules are used to provide parts of the model intensionally. Examples are rules used to enrich the data stored in the base (for instance through recursive views), or rules used to specify integrity constraints in a concise way.

The increasing work on the semantic web [16] in the last decade has leveraged several concepts from artificial intelligence, with a renewed interest for ontologies. An extensive survey of the combination of rules and ontologies in the context of the semantic web is given in [6]. This remains an area of active research with an annual Conference on Web Reasoning and Rule Systems [21]. As mentioned in [6, p. 20], the kind of rules that are considered are model-oriented ones.

It is interesting to note also that the semantic web community is showing interest [25] in the SBVR standard, which had initially been promoted by the business rules community.

### 2.1.2 *State-Oriented Rules*

In rule-based paradigms to which we refer as state-oriented, the role of rules is not to build a model intensionally for a body of knowledge, but rather to solve a problem. In this, rules are used to evolve a state of the system towards a goal, by performing

a series of computations. Particularly in this category of rule-based paradigms, both forward- and backward-chaining approaches are represented.

Again, artificial intelligence provides several occurrences of state-oriented rules. As mentioned in Chap. 1, production systems were introduced through the work of Allen Newell on heuristics reproducing the human approach [53, 60]. Business Rules Management Systems have inherited from production systems the basis for their execution components: the engines that are in charge of selecting the rules to apply, and actually perform their actions on the current state of the system.

The most obvious contribution is the Rete algorithm [29], the variants and evolutions of which are implemented in the rule engines of BRMS. This influence of production systems on BRMS is the one that is the most relevant to us, since we are interested in the execution of rule programs as handled by Business Rules Management Systems. Despite the seminal role of Rete, alternatives to Rete-based execution have emerged in the last decade [27, 37, 65]. However, in Chap. 5 we exhibit a structure in the execution of rule programs that is common to all execution algorithms. We achieve this by dissociating applicability and eligibility of rules, and exploit it to define a semantics of rule programs formally.

Databases, which as just seen use model-oriented rules through deductive rules, also use state-oriented rules in active databases [80]. While the rules in production systems are also called condition-action rules, the active rules in databases are often called event-condition-action rules, because they react to events that are internal or external to the database. The actions taken by these rules range from checking integrity constraints to updating data in the base, to triggering broader operations outside of the database. Even when the purpose of these rules is to ensure the integrity of the database, the fact that they do so by applying updates to the base gives them a state orientation.

The use of event-condition-action rules, not in the context of databases but in that of the semantic web, has been investigated in [15, 18].

### ***2.1.3 Combining the Two Approaches***

Although the classification between model- and state-oriented rules reflects an actual difference in the intent for using rules, there are several cases where a combination of the two approaches has been investigated.

One of them is automated planning and scheduling, another occurrence of rules in artificial intelligence. In this field, rules can be used to infer additional information on a given state from domain knowledge; this usage of rules is called axiomatic inference and is rather model-oriented. Rules are also extensively used in the state space approach to planning, to evolve the state of the system towards a goal, that is, with a state orientation. See [33] for an overview of automated planning and scheduling.

The deductive and active rules in databases give another illustration of the use of both approaches in a common area, with deductive rules oriented towards

knowledge modeling and active rules oriented towards computation. In [47] their seemingly diverging capabilities are integrated by the introduction of the Statelog language, which Bertram Ludäscher defines as a state-oriented extension of Datalog. This goes further than the work of Vianu, who provides alternate semantics for the same syntax in [79]. Indeed, Ludäscher simultaneously puts into play features of both kinds of rules to build Statelog.

The FLORID engine [30] was developed to evaluate rules in deductive databases, thus in a model-oriented usage of rules. In [68] the use of the Rete algorithm, which was introduced for the implementation of production rule systems, is investigated as an evaluation technique.

Another example of combining the two approaches is given by the work on performing rule-based computations on the semantic web, that is, on universes described by ontologies. This usage of state-oriented rules on the semantic web is addressed in [6], and is one of the subjects of study for the REVERSE network [66]. The specific question of how Business Rules Management Systems could be enabled to handle data modeled by such intensional mechanisms as ontologies is one objective of the ONTORULE project [58], addressed for example in [22, 35].

## 2.2 Formalization and Verification of Rule Programs

### 2.2.1 Formalization

Business Rules Management Systems inherit from both model-oriented and state-oriented rules, although the influence of each category expresses itself in distinct components. On the one hand, the model orientation has an influence in the rule authoring components, through the design and implementation of vocabularies targeted at business experts. On the other hand, the state orientation shows in execution components, through the semantics given to rules. We can therefore expect that the formalization of rule programs as handled by BRMS will show the influence of formalisms adopted for both orientations of rule-based paradigms.

Databases also make extensive use of both kinds of rules with deductive and active rules being the respective representatives of model- and state-oriented rules. The Datalog language [31, 50, 78] is at the heart of deductive and active databases, and formalizations of numerous variants of this language have been proposed [1, 2, 50, 79]. These formalisms, which provide several semantics for Datalog, are all logic-based.

Among them, F-logic [39] has the peculiarity of having originally been developed for deductive databases, and of being used more recently in the context of the semantic web as a formalism for ontologies [5], together with description logic [11] and order-sorted features logic [3, 4]. There is no definitive formalism for rules in the context of the semantic web. The choice depends primarily on the formalism

adopted for ontologies, and on whether the rules are used in a model orientation to provide parts of the knowledge, or in a state orientation to perform computations.

The principal point of effort when formalizing state-oriented rules, such as active rules in databases or production rules, is the fact that they perform updates on states. To this end, [20] uses  $\mu$ -calculus to represent propositional production systems, and fixed-point logic to represent production systems with variables. This approach leads to using first-order logic structures as we do. A formalization of production systems with situation calculus is given by Baral and Lobo [13], while Baralis and Widom [14] represents active rules in databases by means of an extended relational algebra.

The formalizations proposed by François Fages and Rémi Lissajoux in [26], then by Claude Kirchner, Pierre-Étienne Moreau et al. in [24], are also based on first-order logic. They are restricted to the Rete algorithm. Furthermore, the former explicitly discards the control strategy from its scope, and the latter gives no detail of how it affects the execution of a rule program. On the other hand, the RIF-PRD recommendation by the W3C [67] does provide a rigorous description of the control strategy of the Rete algorithm and its influence on rule program execution.

Our formalization goes one step further towards the verification of properties of rule programs as handled by BRMS, in particular by considering the execution strategy as a parameter (Chap. 5), and by allowing the description of a saturation semantics that extends to parallel or nondeterministic programs (Chap. 7).

### 2.2.2 *Verification of Model-Oriented Rule Programs*

As seen in Sect. 2.1.1, model-oriented rules are primarily used in knowledge management applications and in deductive databases. In these contexts the verification concerns are almost exclusively focused on consistency and completeness of the knowledge base, respectively of the database. See [49, 62, 76] for a survey of problems and techniques in the verification of knowledge bases, and [46] for an illustration on a specific knowledge management system. Deductive databases reuse these techniques, or develop specific ones [41, 50].

The verification of the consistency of an ontology can be seen as a particular case of consistency of a knowledge base. Specific investigations have been carried out when ontologies are used in the semantic web context [64], and in particular when they are implemented with description logic [63].

The verification of business rules in the sense of Sect. 1.2.2, that is, rules modeling business policies, has been the subject of extensive study by Rik Gerrits and Silvie Spreeuwenberg [32]. It has given birth to the commercial product LibRT VALENS, which it is claimed is used by a few Business Rules Management Systems. However the description of the product in [32] and on the company website, furthermore confirmed by an in-depth study from the standpoint of a BRMS [52], shows that LibRT VALENS considers rules with a model orientation, and proves to be out of focus for a BRMS. For example, a self-contradiction is signaled

when a rule is not applicable in the state that results from its execution, which is a rather common case in state-oriented rule applications as they are nonmonotonic. More generally, the verification concerns of LibRT VALENS focus on rule base consistency and completeness, and transpose badly to production rules.

Consistency and completeness of a rule base are not our primary focus here. Rather, we are interested in correctness properties of executions of a rule program. However, we can provide an interpretation of consistency and completeness in the context of Business Rules Management Systems. For example, a rule that can never be applicable, no matter the values of the attributes of the objects in the working memory, can be seen as an inconsistency in the rule program. Additional examples of such interpretations of consistency for rule programs in BRMS are given in App. A and in [23, 64]. Completeness can also be relevant for BRMS: checking the completeness of a rule program means verifying whether the program can be executed from all possible working memories—or from all working memories in a subspace of interest.

### 2.2.3 *Verification of State-Oriented Rule Programs*

While the verification concerns for model-oriented rules are focused on consistency and completeness, those for state-oriented rules primarily concentrate on confluence and termination.

Several approaches to the verification of confluence and termination in active databases are reviewed in [14]. Among them, Karadimce and Urban [38] reduces event-condition-action rules to conditional term rewriting, and Hellerstein and Hsu [34] restricts the rule language to guarantee confluence. Baralis and Widom [14] itself proposes an approach based on an extension of the relational algebra. On the other hand, Ludäscher and Lausen [48] focuses on the termination problem in active databases.

The connection between production rules and term-rewriting systems has been explored with the intention of using the techniques of rewriting systems to analyze properties, and eventually verify the correctness, of production systems. In [69–72, 74], James Schmolze and Wayne Snyder propose a rewrite semantics for OPS5. This proposal includes a form of constrained rewriting for negative tests (“there is no object such that...”, which we do not address). In addition, techniques such as Knuth-Bendix completion [40] are applied to test the confluence of production systems and repair it in case of failure, and to detect design issues such as redundancy. This approach has shortcomings, which are listed by the authors themselves [74, pp. 514–515]. Some of them can probably be addressed by appropriate extensions of the formalism: for example the limitation of expressions to Boolean ones, and of actions to additions to, and removals from, the working memory. Other limitations however, such as not modeling the selection and eligibility strategies, may prove too severe for this approach to be applied to rule programs as handled by BRMS.

For rule programs as handled by Business Rules Management Systems, confluence can be defined as the property of a rule program that all its executions from a given state lead to the same final state. This property is described in [17]. We do not study it here, mainly because general confluence involves several concurrent executions and hence cannot be expressed with correctness formulas of the form  $\{p\} \mathcal{R} \{q\}$ . However, in App. A, we present examples of particular cases of confluence checking.

In addition to confluence and termination verification, analyses have been designed in the context of production systems to find design flaws. For example, Prakash et al. [61] describes a method to analyze OPS5 programs statically, in order to detect relations between production rules that would suggest errors in the design of the program.

The verification of safety properties in BRMS rule programs is within our focus. Berstel and Leconte [17] addresses it by translating this verification task into a constraint satisfiability problem. However, the technique described in this paper for proving these safety properties lacks compositionality, contrary to the one introduced in Part IV.

## 2.3 Verification of Pointer and Concurrent Programs

Verification of programs with a Hoare logic is a widespread technique. Its origins are in the work of Robert Floyd [28] and Tony Hoare [36]. It has since been extended to numerous programming paradigms. Surveys can be found for example in [7, 8, 12, 54].

We have seen that rule programs as handled by Business Rules Management Systems have common points with databases, in particular because production rules can be seen as a special case of event-condition-action rules in active databases, although the execution semantics of the two kinds of rule programs are different. In addition they share features with two other, non-rule-based paradigms: programs with pointers and concurrent programs.

The similarities with programs with pointers come from the fact that business applications integrating with Business Rules Management Systems are typically object-oriented applications. As a result, rules in BRMS handle objects with attributes and methods. As mentioned in Chap. 1 (and formally described in Chap. 5) the objects on which the rules reason are handled through a set called the working memory.

In many aspects the working memory is similar to a heap: it is finite but not bounded, it is accessed randomly, in particular when rule instances are formed, and it is dynamic in two ways. Firstly, the links between the objects can be updated by the execution of the rules (this aspect is covered in the present work). Secondly, rules can add objects to the working memory or remove objects from it (we do not cover this aspect). In contrast with programs with pointers however, there is usually no concern such as shape analysis when verifying rule programs.

Beyond the similarities between the working memory and a heap, the fact that rules handle objects brings in the aliasing problem, that is, the fact that two distinct variables can hold the same object. This has to be taken into account, in particular, in the way updates of attributes are interpreted (Sect. 4.4) or rule execution is defined (Sect. 5.3). We adopt the approach of Joseph Morris [51], which consists in including a conditional expression in the formula that defines the effect of assignment. We present it on p. 57; it is also presented in detail in [9, pp. 42–45].

Rule programs in BRMS also have important similarities with concurrent programs, which is a well-studied topic [19, 44, 45, 55–57]. Clearly the working memory can be seen as a shared memory. Furthermore, the execution of a rule program as described in Chap. 5 shows two degrees of concurrency, which result from the Cartesian product between rules and object tuples that is performed by the rule engine. One way to look at this is to consider each rule in the program as a process, within which all objects are concurrent; the symmetric vision holds as well, where all objects are processes and run the same program made of the rules concurrently. In addition, both viewpoints must account for the fact that rules do not match individual objects but object tuples, which introduces synchronization constraints. Chapter 7 handles this complexity by transposing the saturated semantics of rule engines onto parallel programs.

In the light of the similarities between rule programs in BRMS and concurrent programs, the question naturally arises of whether verification methods for concurrent programs could apply to rule programs. Such methods were initially designed by Edward Ashcroft [10] using a global invariant, then independently made compositional by Leslie Lamport [43], and by Susan Owicki and David Gries [59]. These methods are also described or discussed in [9, 44, 73].

We shall see in Chap. 7 that in general a rule program as handled by a BRMS cannot be simulated by a concurrent program. As a consequence, the verification of a rule program cannot be performed by translating it into a concurrent one and then applying the verification methods for concurrent programs.

## References

All URLs were successfully accessed on January 22, 2013.

1. Abiteboul, S., Vianu, V.: Procedural and declarative database update languages. In: PODS, pp. 240–250. ACM, New York (1988)
2. Abiteboul, S., Vianu, V.: Datalog extensions for database queries and updates. *J. Comp. Syst. Sci.* **43**(1), 62–124 (1991)
3. Aït-Kaci, H.: Description logic vs. order-sorted feature logic. In: Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Turhan, A.Y., Tessaris, S. (eds.) *Description Logics. CEUR Workshop Proceedings*, vol. 250. CEUR-WS.org (2007). [http://ceur-ws.org/Vol-250/paper\\_2.pdf](http://ceur-ws.org/Vol-250/paper_2.pdf)
4. Aït-Kaci, H., Podelski, A., Copen Goldstein, S.: Order sorted feature theory unification. *J. Logic Program.* **30**(2), 99–124 (1997)



5. Angele, J., Kifer, M., Lausen, G.: Ontologies in F-logic. In: Bernus, P., Blazewics, J., Schmidt, G., Shaw, M., Staab, S., Studer, R. (eds.) *Handbook on Ontologies*. International Handbooks on Information Systems, pp. 45–70. Springer, Berlin (2009)
6. Antoniou, G., Damásio, C.V., Grosz, B., Horrocks, I., Kifer, M., Maluszynski, J., Patel-Schneider, P.F.: Combining Rules and Ontologies: A Survey. Tech. Rep. IST506779/Linköping/I3-D3/D/PU/a1, Linköping University (2005). IST-2004-506779 REWERSE Deliverable 2005-I3-D3. <http://rewerse.net/publications/rewerse-description/REWERSE-DEL-2005-I3-D3.html>
7. Apt, K.R.: Ten years of Hoare’s logic: A survey—part I. *ACM Trans. Program. Lang. Syst.* **3**(4), 431–483 (1981)
8. Apt, K.R.: Ten years of Hoare’s logic: A survey—part II: Nondeterminism. *Theor. Comp. Sci.* **28**, 83–109 (1984)
9. Apt, K.R., de Boer, F.S., Olderog, E.R.: *Verification of Sequential and Concurrent Programs*, 3rd edn. Texts in Computer Science. Springer, Berlin (2009)
10. Ashcroft, E.A.: Proving assertions about parallel programs. *J. Comp. Syst. Sci.* **10**(1), 110–135 (1975)
11. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, London (2003)
12. de Bakker, J.W., de Bruin, A., Zucker, J.: *Mathematical Theory of Program Correctness*. Prentice-Hall International Series in Computer Science. Prentice Hall, Englewood Cliffs (1980)
13. Baral, C., Lobo, J.: Characterizing production systems using logic programming and situation calculus (2007). <http://cs.utep.edu/chitta/papers/char-prod-systems.ps>
14. Baralis, E., Widom, J.: An algebraic approach to rule analysis in expert database systems. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) *VLDB*, pp. 475–486. Morgan Kaufmann, Los Altos (1994)
15. Behrends, E., Fritzen, O., May, W., Schenk, F., Schubert, D.: A framework and components for ECA rules in the web. In: *Proceedings of Second International Conference on Rules and Rule Markup Languages for the Semantic Web*, Athens, GA, USA (RuleML) (2006). <http://rewerse.net/publications/rewerse-description/REWERSE-RP-2006-165.html>
16. Berners-Lee, T., Hendler, J.: The semantic web. *Sci. Am.* **284**, 34–43 (2001)
17. Berstel, B., Leconte, M.: Using constraints to verify properties of rule programs. In: *Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops, ICSTW’10*, pp. 349–354. IEEE Computer Society, Silver Spring (2010)
18. Berstel, B., Bonnard, P., Bry, F., Eckert, M., Pătrânjan, P.L.: Reactive rules on the web. In: Antoniou, G., Almann, U., Baroglio, C., Decker, S., Henze, N., Pătrânjan, P.L., Tolsdorf, R. (eds.) *Reasoning Web. Lecture Notes in Computer Science*, vol. 4636, pp. 183–239. Springer, Berlin (2007)
19. Broy, M., Olderog, E.R.: Trace-oriented models of concurrency. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) *Handbook of Process Algebra*, pp. 101–195. Elsevier Science, New York (2001)
20. de Bruijn, J., Rezk, M.: A logic based approach to the static analysis of production systems. In: Polleres, A., Swift, T. (eds.) *RR. Lecture Notes in Computer Science*, vol. 5837, pp. 254–268. Springer, Berlin (2009)
21. Calvanese, D., Lausen, G. (eds.): *Web Reasoning and Rule Systems*. *Proceedings of Second International Conference, RR 2008, Karlsruhe, Germany, 31 October–1 November 2008*. *Lecture Notes in Computer Science*, vol. 5341. Springer, Berlin (2008)
22. Chniti, A., Dehors, S., Albert, P., Charlet, J.: Authoring business rules grounded in OWL ontologies. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) *RuleML. Lecture Notes in Computer Science*, vol. 6403, pp. 297–304. Springer, Berlin (2010)
23. Chniti, A., Albert, P., Charlet, J.: Gestion de la cohérence des règles métier éditées à partir d’ontologies OWL. In: *Conférence Francophone d’Apprentissage 2011* (2011)

24. Cirstea, H., Kirchner, C., Moossen, M., Moreau, P.É.: Production systems and Rete algorithm formalisation. Research report, LORIA, Nancy (2004). <http://hal.inria.fr/inria-00280938/PDF/rete.formalisation.pdf>
25. Demuth, B., Liebau, H.B.: An approach for bridging the gap between business rules and the semantic web. In: Proceedings of the 2007 International Conference on Advances in Rule Interchange and Applications, RuleML'07, pp. 119–133. Springer, Berlin (2007)
26. Fages, F., Lissajoux, R.: Sémantique opérationnelle et compilation des systèmes de production. *Revue d'Intelligence Artificielle* **6**(4), 431–456 (1992)
27. Fair, Isaac, and Company: High-volume batch processing with Blaze Advisor. Computer World UK (2007). <http://www.computerworlduk.com/white-paper/business-process/5092/high-volume-batch-processing-with-blaze-advisor/>
28. Floyd, R.W.: Assigning meaning to programs. In: Proceedings of the Symposium on Applied Math, vol. 19, pp. 19–32. American Mathematical Society, Providence (1967)
29. Forgy, C.: Rete: A fast algorithm for the many patterns/many objects match problem. *Artif. Intell.* **19**(1), 17–37 (1982)
30. Frohn, J., Himmeröder, R., Kandzia, P.T., Lausen, G., Schlepphorst, C.: FLORID: A prototype for F-logic. In: Gray, W.A., Larson, P.Å. (eds.) ICDE, p. 583. IEEE Computer Society, Silver Spring (1997)
31. Gallaire, H., Minker, J. (eds.): Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse. *Advances in Data Base Theory*. Plenum Press, New York (1977)
32. Gerrits, R., Spreeuwenberg, S.: VALENS: A knowledge based tool to validate and verify an Aion knowledge base. In: Horn, W. (ed.) ECAI, pp. 731–738. IOS Press, Amsterdam (2000)
33. Ghallab, M., Nau, D.S., Traverso, P.: *Automated Planning: Theory and Practice*. Morgan Kaufmann, Los Altos (2004)
34. Hellerstein, J.M., Hsu, M.: Determinism in partially ordered production systems. Research report, IBM Almaden Research Center (1991)
35. Heymans, S., de Bruijn, J., Rezk, M., Aït-Kaci, H., Citeau, H., Korf, R., Pührer, J., Feier, C., Eiter, T.: Initial combinations of rules and ontologies. Public Deliverable 3.2, ONTORULE Project (2009). <http://ontorule-project.eu/outcomes?func=fileinfo&id=18>
36. Hoare, C.A.R.: An axiomatic basis for computer programming. *Comm. ACM* **12**, 576–580 (1969)
37. IBM: IBM Operational Decision Manager v8.0 User's Manual (2012). <http://publib.boulder.ibm.com/infocenter/dmanager/v8r0/>
38. Karadimce, A.P., Urban, S.D.: Conditional term rewriting as a formal basis for active database rules. In: RIDE-ADS, pp. 156–162. IEEE Computer Society (1994)
39. Kifer, M., Lausen, G.: F-logic: A higher-order language for reasoning about objects, inheritance, and scheme. In: Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, SIGMOD'89, pp. 134–146. ACM, New York (1989)
40. Knuth, D., Bendix, P.: Simple word problems in universal algebra. In: *Computational Problems in Abstract Algebra*, pp. 263–297. Pergamon Press, New York (1970)
41. Kowalski, R.A., Sadri, F., Soper, P.: Integrity checking in deductive databases. In: Proceedings of the 13th International Conference on Very Large Data Bases, VLDB'87, pp. 61–69. Morgan Kaufmann, Los Altos (1987)
42. Lakemeyer, G., Nebel, B. (eds.): *Foundation of Knowledge Representation and Reasoning. Lecture Notes in Computer Science*, vol. 810. Springer, Berlin (1994)
43. Lamport, L.: Proving the correctness of multiprocess programs. *IEEE Trans. Software Eng.* **3**(2), 125–143 (1977)
44. Lamport, L.: Verification and specification of concurrent programs. In: de Bakker, J.W., de Roever, W.P., Rozenberg, G. (eds.) REX School/Symposium. *Lecture Notes in Computer Science*, vol. 803, pp. 347–374. Springer, Berlin (1993)
45. Lamport, L.: *Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, Reading (2002). <http://research.microsoft.com/en-us/um/people/lamport/tla/book.html>

46. von Luck, K., Nebel, B., Peltason, C., Schmiedel, A.: BACK to consistency and incompleteness. In: Stoyan, H. (ed.) *GWAI. Informatik-Fachberichte*, vol. 118, pp. 245–256. Springer, Berlin (1985)
47. Ludäscher, B.: *Integration of Active and Deductive Database Rules*. DISDBIS, vol. 45. Infix Verlag, St. Augustin (1998)
48. Ludäscher, B., Lausen, G.: Handling termination in a logical language for active rules. *Informatica (Lithuanian Academy of Sciences)* **9**(1), 65–84 (1998)
49. Menzies, T., Pecheur, C.: Verification and validation and artificial intelligence. *Adv. Comp.* **65**, 154–203 (2005)
50. Minker, J.: Logic and databases: A 20 year retrospective. In: Pedreschi, D., Zaniolo, C. (eds.) *Logic in Databases*. Lecture Notes in Computer Science, vol. 1154, pp. 3–57. Springer, Berlin (1996)
51. Morris, J.M.: A general axiom of assignment. In: Bauer, F.L., Dijkstra, E.W., Hoare, C.A.R. (eds.) *Theoretical Foundations of Programming Methodology: Lecture Notes of an International Summer School*. Reidel, Dordrecht (1982)
52. Nakoulima, S.D.: Added value of LibRT VALENS for ILOG JRules. Rapport de stage, École des Mines de Nancy (2004). Available from Bruno Berstel-Da Silva
53. Newell, A., Simon, H.A.: *Human Problem Solving*. Prentice Hall, Englewood Cliffs (1972)
54. Olderog, E.R.: Hoare's logic for programs with procedures—What has been achieved? In: Clarke, E.M., Kozen, D. (eds.) *Logics of Programs*. Lecture Notes in Computer Science, vol. 164, pp. 383–395. Springer, Berlin (1984)
55. Olderog, E.R.: Correctness of concurrent processes. *Theor. Comp. Sci.* **80**, 263–288 (1991)
56. Olderog, E.R.: *Nets, Terms and Formulas: Three Views of Concurrent Processes and Their Relationship*. Cambridge University Press, London (1991). Paperback Edition 2005
57. Olderog, E.R., Rössig, S.: A case study in transformational design of concurrent systems. In: Gaudel, M.C., Jouannaud, J.P. (eds.) *Theory and Practice of Software Development (TAPSOFT'93)*. Lecture Notes in Computer Science, vol. 668, pp. 90–104. Springer, Berlin (1993)
58. ONTORULE Project. <http://ontorule-project.eu>
59. Owicki, S.S., Gries, D.: An axiomatic proof technique for parallel programs I. *Acta Informatica* **6**, 319–340 (1976)
60. Piccinini, G.: Allen Newell. In: *New Dictionary of Scientific Biography*. Thomson Gale (2007). <http://www.umsi.edu/~piccininig/Newell%205.htm>
61. Prakash, G.R., Subrahmanian, E., Mahabala, H.N.: A methodology for systematic verification of OPS5-based AI applications. In: *IJCAI*, pp. 3–8. Morgan Kaufmann, Los Altos (1991)
62. Preece, A.D., Shinghal, R.: Foundation and application of knowledge base verification. *Int. J. Intell. Syst.* **9**(8), 683–701 (1994)
63. Pührer, J., Heymans, S., Eiter, T.: Dealing with inconsistency when combining ontologies and rules using DL-programs. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) *ESWC (1)*. Lecture Notes in Computer Science, vol. 6088, pp. 183–197. Springer, Berlin (2010)
64. Pührer, J., Ghali, A.E., Chniti, A., Korf, R., Schwichtenberg, A., Lévy, F., Heymans, S., Xiao, G., Eiter, T.: Consistency Maintenance. Public Deliverable 2.3, ONTORULE Project (2011). <http://ontorule-project.eu/outcomes?func=fileinfo&id=44>
65. Red Hat: JBoss Rules 5 Reference Guide (2012). [https://access.redhat.com/knowledge/docs/en-US/JBoss\\_Enterprise\\_BRMS\\_Platform/5/html-single/JBoss\\_Rules\\_5\\_Reference\\_Guide/index.html#Sequential\\_Mode](https://access.redhat.com/knowledge/docs/en-US/JBoss_Enterprise_BRMS_Platform/5/html-single/JBoss_Rules_5_Reference_Guide/index.html#Sequential_Mode)
66. REVERSE Network of excellence. <http://reverse.net>
67. de Sainte Marie, C., Hallmark, G., Paschke, A.: Rule Interchange Format, Production Rule Dialect. Recommendation, W3C (2010). <http://www.w3.org/TR/rif-prd/>
68. Schmedding, F., Sawas, N., Lausen, G.: Adapting the Rete-algorithm to evaluate F-logic rules. In: Paschke, A., Biletskiy, Y. (eds.) *Advances in Rule Interchange and Applications*. Lecture Notes in Computer Science, vol. 4824, pp. 166–173. Springer, Berlin (2007)

69. Schmolze, J.G., Snyder, W.: Using confluence to control parallel production systems. In: Kanal, L.N. (ed.) *Parallel Processing for Artificial Intelligence*. Elsevier Science, New York (1994)
70. Schmolze, J.G., Snyder, W.: A tool for testing confluence of production rules. In: Ayel, M., Rousset, M.C. (eds.) *EUROVAV*, pp. 91–104. ADERIAS-LIA, Université de Savoie (1995)
71. Schmolze, J.G., Snyder, W.: Detecting redundant production rules. In: *AAAI/IAAI*, pp. 417–423 (1997)
72. Schmolze, J.G., Snyder, W.: Detecting redundancy among production rules using term rewrite semantics. *Knowl. Base. Syst.* **12**(1–2), 3–11 (1999)
73. Schneider, F.B., Andrews, G.R.: Concepts for concurrent programming. In: de Bakker, J.W., de Roever, W.P., Rozenberg, G. (eds.) *Current Trends in Concurrency*. Lecture Notes in Computer Science, vol. 224, pp. 669–716. Springer, Berlin (1986)
74. Snyder, W., Schmolze, J.G.: Rewrite semantics for production rule systems: Theory and applications. In: McRobbie, M.A., Slaney, J.K. (eds.) *CADE*. Lecture Notes in Computer Science, vol. 1104, pp. 508–522. Springer, Berlin (1996)
75. Sowa, J.F.: *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, Belmont (1999)
76. Suwa, M., Scott, A.C., Shortliffe, E.H.: An approach to verifying completeness and consistency in a rule-based expert system. *AI Mag.* **3**(4), 16–21 (1982)
77. The Object Management Group: *Semantics of Business Vocabulary and Business Rules (SBVR 1.0)* (2008). <http://www.omg.org/spec/SBVR/1.0/>
78. Ullman, J.D.: *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, Rockville (1988)
79. Vianu, V.: Rule-based languages. *Ann. Math. Artif. Intell.* **19**(1–2), 215–259 (1997)
80. Widom, J., Ceri, S.: Introduction to active database systems. In: *Active Database Systems: Triggers and Rules For Advanced Database Processing*, pp. 1–41. Morgan Kaufmann, Los Altos (1996)

Verification of Business Rules Programs

Berstel-Da Silva, B.

2014, XVII, 236 p. 18 illus., 2 illus. in color., Hardcover

ISBN: 978-3-642-40037-7