U N I V E R S I T Y   O F   C O P E N H A G E N

A survey of formal languages for contracts

Hvitved, Tom

Published in:
Formal Languages and Analysis of Contract-Oriented Software - 4th International Workshop, FLACOS 2010

Publication date:
2010

Document version
Early version, also known as pre-print

Document license:
Unspecified

A Survey of Formal Languages for Contracts*

Tom Hvitved

Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen, Denmark
hvitved@diku.dk

In this short paper we present the current status on formal languages and models for contracts. By a formal model is meant an unambiguous and rigorous representation of contracts, in order to enable their automatic validation, execution, and analysis — activates that are collectively referred to as *contract lifecycle management* (CLM). We present a set of formalism requirements, which represent features that any *ideal* contract model should support, based on which we present a comparative survey of existing contract formalisms.

Empirical studies conducted by the Aberdeen Group [16, 17] suggest that *contract lifecycle management* (CLM) will be a critical key to success for businesses in the near future. CLM is a broad term used to cover the activities of systematically and efficiently managing contract *creation*, contract *negotiation*, contract *approval*, contract *execution*, and contract *analysis*. In the Aberdeen Group studies it is reported that around 80 percent of the surveyed enterprises (220 participants) are exercising only manual, or partially automated contract management activities, the implication of which is a lower rate of compliant transactions (i.e., actions according to contract). This in turn implies potential financial penalties:

"... the average savings of transactions that are compliant with contracts is 22%." [16, p. 1]

The conclusion of the Aberdeen Group studies is a list of *required actions* [17], which serve as recommendations for enterprises seeking to utilize CLM. The most important recommendations are:

(A1) Establish standardized and formal contract management processes, including a standard language for contracts accessible via libraries and templates.

(A2) Clearly define protocols for the complete contracting process and contract administration (e.g., contract signing and contract execution).

(A3) Use reporting and analytic capabilities on contract data to gain competitive advantage.

Requirements A1 and A2 capture the essence of what a CLM system should support, and A3 why automated CLM is of interest to enterprises. However, A1 — which is the topic of our survey — is the *enabling technology* for achieving A2, and in particular for achieving A3, since analyses need representations of the actual contracts. As a complement to our survey, Tan et al. provide an overview of CLM features [21], which focuses on the aspects of A2 and A3, rather than formal languages for contracts. For a comparative analysis of existing contract formalisms we need a set of representative features, which any *ideal* contract model should support. Such benchmark provides a basis for comparing existing contract formalisms, and indeed for constructing new formalisms as well, but we must first make it clear what kind of contracts we consider: in this survey a contract is a legally binding agreement between two or more parties, and it constitutes a normative description of the commitments, i.e., *expected actions*, of the

---

*Extended abstract. Work in progress extension of the author's original survey [9].

contract participants. This restriction means that we consider contracts to be *ought-to-do* [19] rather than *ought-to-be*, meaning that a contract is a description of what may/must/must not be *performed*, rather than what may/must/must not be *the state of affairs*.

In order to gather contract features, we need to consider some actual example contracts. However, due to lack of space we consider here only one contract in Figure 1. From the example contract we derive the following requirements: *(R1) modelling of contract participants; (R2) parametrized contract templates; (R3) (conditional) commitments, i.e., obligations, permissions, and prohibitions; (R4) absolute and relative temporal constraints; (R5) history-sensitive commitments; and (R6) basic arithmetic.* Modelling of contract participants (R1) is needed in order to make explicit whom commitments are pertained to, and parametrized contract templates (R2) are desirable similar to code reuse via macros and functions (this requirement was also mentioned by the Aberdeen Group, cf. A1). (Conditional) commitments (R3), and absolute and relative temporal constraints (R4) are at the heart of contracts [15], and history sensitivity (R5) refers to the aspect that commitments may depend on the actual events that have taken place earlier: in the example contract this is evident in Paragraph 3, where the remaining balance in each month depends what has been paid so far, and in-place arithmetic expressions (R6) are needed to calculate such values dynamically.

The next requirements are not present in the example (R7 and R8) or of a more *fundamental* character (R9–R13): *(R7) contrary-to-duty (reparation clauses); (R8) potentially infinite and repetitive contracts; (R9) compositionality; (R10) deterministic contract execution (run-time monitoring); (R11) blame assignment; (R12) the isomorphism principle; and (R13) subject to analysis.* We will not go into detail with R7–R10 and R13 as they are also identified by Pace and Schneider [15]. Run-time monitoring (R10) and blame assignment (R11) are closely related: in case of a contract violation, the monitoring algorithm should not only report a violation, but also *who* among the contract participants are responsible for the violation. This aspect is the main differences between contracts and more conventional *workflows*: workflows usually only describe the "happy-path" executions, and in case the execution does not follow an accepting path, it is called a violation. Saying that a contract is violated however, is insufficient unless there is also information about who violated the contract (in general, *who* may be more than one party). The isomorphism principle (R12) was originally introduced in the context of legal text formalization [2], and the principle refers to the idea of a formal encoding that is in a one-to-one correspondence with the informal paper contract; one paragraph in the paper contract corresponds to a separate "component" in the formalization, and any dependencies between paragraphs in the paper contract are also present between the corresponding components in the formalization. The reason why the isomorphism principle is desirable is immediate: (local) changes in the paper contract correspond to (local) changes in the formalized contract (and vice versa), and a formal encoding which is reminiscent of the paper contract, should be easier for domain experts (i.e., lawyers) to maintain.

Paragraph 1. Seller agrees to transfer and deliver `[goods]` to Buyer, on or before `[delivery date]`.

Paragraph 2. Buyer agrees to accept the goods and pay the sum of `[prepayment]` upon receipt, and `[closing payment]` is to be paid at closing, and the balance of `[balance]` shall be paid as follows:

Paragraph 3. `[installment]` or more per month on the first day of each and every month hereafter commencing on `[first month]`, and continuing until the entire balance, including both principal and interest, shall be paid in full; provided, however, that the entire balance due plus accrued interest and any other amounts due hereunder shall be paid in full on or before the first day of `[final month]`. Monthly payments shall include both principle and interest with interest at the rate of `[rate]`% per annum, computed monthly on the remaining balance from time to time unpaid.

Figure 1: Installment sale.

We have now gathered a total of 13 contract formalism requirements, which are most likely not representative for *all* possible contracts, however they each represent important aspects. Due to lack of space, we will not present the full comparative analysis of contract languages here, but we do provide references to the relevant work, and consider a few of the benchmark requirements. Existing work on contract formalisms roughly fall into three categories: (deontic) logic based formalisms [3, 11, 22], *event-condition-action* based formalisms [5, 7, 12], and action/trace based formalisms [1, 10, 19]. However, we have also encountered formalisms based on combinators [18], defeasible reasoning [6, 8, 20], commitment graphs [23, 24], finite state machines [13], and more informal approaches [4, 14]. Most formalisms are applicable to contracts in general, however, the formalisms by Peyton-Jones and Eber [18], Andersen et al. [1], and Tan and Thoen [20], represent more specialized contract models (for financial, commercial, and trade contracts, respectively). Not to our surprise, none of the existing formalisms cover all the gathered requirements R1–R13, but more to our surprise, only few models come with a clear, formal semantics [1, 10, 23]: most approaches only have incomplete semantics [6, 7, 8, 11, 12, 13, 14, 22], and some languages are only described in plain text [3, 4, 5, 18, 20]. We find this surprising, since one of the purposes of defining a contract language, is to have a formal, unambiguous semantics. Explicit modelling of contract participants (R1) is supported by some models [1, 3, 5, 12, 13, 23], but only Xu [23] considers a model, in which blame is assigned in the case of non-conformance (R11). Modelling of (conditional) commitments (R3) is present in virtually all models, yet the deontic-logic inspired models deal with explicit obligations, permissions, and prohibitions [7, 11, 12, 13, 19, 22], where as others do not use the deontic modalities directly. Temporal constraints (R4) are fully supported by few formalisms [1, 11], and partially supported (i.e., relative temporal constraints only) by most formalisms [7, 12, 13, 18, 19]. Contrary-to-duty clauses (R7) are supported as primitives by [6, 7, 8, 12, 19], and encodable as disjunctions in [1, 11]. Algorithms for run-time monitoring of contracts (R10) are found in [1, 10, 13, 24], but only formally defined by Andersen et al. [1] and Kyas et al. [10]. In the former approach, run-time monitoring is performed via *residuation* (i.e., rewrite rules), and in the latter approach, finite state machines are derived from contract specifications in the language of Prisacariu and Schneider [19]. Potentially infinite contracts (R8) can be expressed in [1, 11, 19], but only [1, 11] are able to handle more complex repetitive patterns involving dynamic parameters (R5) and arithmetic expressions (R6). Finally, we have generally found a lack of analyses applicable to contracts (R13), with the exception of the valuation analysis for financial contracts by Peyton-Jones and Eber [18].

References

[1] Jesper Andersen, Ebbe Elsborg, Fritz Henglein, Jakob Grue Simonsen & Christian Stefansen (2006): *Compositional specification of commercial contracts*. International Journal on Software Tools for Technology Transfer (STTT) 8(6), pp. 485–516.

[2] T. J. M. Bench-Capon & F. P. Coenen (1992): *Isomorphism and Legal Knowledge Based Systems*. Artificial Intelligence and Law 1(1), pp. 65–86.

[3] Abdel Boulmakoul & Mathias Sall (2002): *Integrated Contract Management*. Technical Report, HP Laboratories Bristol.

[4] Content Reference Forum (2004): *Contract Expression Language (CEL) An UN/CEFACT BCF Compliant Technology*. Technical Report, Content Reference Forum.

[5] Andrew Goodchild, Charles Herring & Zoran Milosevic (2000): *Business Contracts for B2B*. In: *Proceedings of the CAISE '00 Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing (ISDO 2000)*, pp. 63–74.

[6] Guido Governatori (2005): *Representing Business Contracts in RuleML. International Journal of Cooperative Information Systems* 14, pp. 181–216.

[7] Guido Governatori & Zoran Milosevic (2006): *A Formal Analysis of a Business Contract Language. Int. J. Cooperative Inf. Syst.* 15(4), pp. 659–685.

[8] Guido Governatori & Duy Hoang Pham (2009): *DR-CONTRACT: An Architecture for e-Contracts in Defeasible Logic. International Journal of Business Process Integration and Management* 5(4).

[9] Tom Hvitved (2009): *Contracts in Programming and in Enterprise Systems.* Master's thesis, Department of Computer Science, University of Copenhagen.

[10] Marcel Kyas, Cristian Prisacariu & Gerardo Schneider (2008): *Run-Time Monitoring of Electronic Contracts.* In: *ATVA '08: Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis*, Springer-Verlag, Berlin, Heidelberg, pp. 397–407.

[11] Ronald M. Lee (1988): *A Logic Model for Electronic Contracting. Decision Support Systems* 4(1), pp. 27–44.

[12] Peter F. Linington, Zoran Milosevic, James B. Cole, Simon Gibson, Sachin Kulkarni & Stephen W. Neal (2004): *A unified behavioural model and a contract language for extended enterprise. Data Knowl. Eng.* 51(1), pp. 5–29.

[13] Carlos Molina-jimenez, Carlos Molina-jimenez, Santosh Shrivastava, Santosh Shrivastava, Ellis Solaiman, Ellis Solaiman, John Warne & John Warne (2004): *Run-time Monitoring and Enforcement of Electronic Contracts. Electronic Commerce Research and Applications* 3, p. 2004.

[14] Nir Oren, Sofia Panagiotidi, Javier Vázquez-Salceda, Sanjay Modgil, Michael Luck & Simon Miles (2009): *Towards a Formalisation of Electronic Contracting Environments.* In: *Coordination, Organizations, Institutions and Norms in Agent Systems IV*, Springer-Verlag, Berlin, Heidelberg, pp. 156–171.

[15] Gordon J. Pace & Gerardo Schneider (2009): *Challenges in the Specification of Full Contracts.* In: *IFM '09: Proceedings of the 7th International Conference on Integrated Formal Methods*, Springer-Verlag, Berlin, Heidelberg, pp. 292–306.

[16] Vishal Patel (2006): *The Contract Management Benchmark Report: Procurement Contracts.* Technical Report, Aberdeen Group.

[17] Vishal Patel & Christopher J. Dwyer (2007): *Contract Lifecycle Management and the CFO: Optimizing Revenues and Capturing Savings.* Technical Report, Aberdeen Group.

[18] Simon Peyton-Jones & Jean-Marc Eber (2003): *How to write a financial contract.* In: Jeremy Gibbons & Oege de Moor, editors: *The Fun of Programming*, chapter 6, Cambridge University Press, New York, NY, USA, pp. 105–130.

[19] Cristian Prisacariu & Gerardo Schneider (2007): *A Formal Language for Electronic Contracts.* In: *Formal Methods for Open Object-Based Distributed Systems*, Springer-Verlag, Berlin, Heidelberg, pp. 174–189.

[20] Yao-Hua Tan & Walter Thoen (2000): *INCAS: a legal expert system for contract terms in electronic commerce. Decision Support Systems* 29(4), pp. 389–411.

[21] Yao-Hua Tan, Walter Thoen & Somasundaram Ramanathan (2001): *A Survey of Electronic Contracting Related Developments.* In: *BLED 2001 Proceedings*, pp. 495–507.

[22] Hans Weigand & Lai Xu (2003): *Contracts in E-Commerce.* In: *Proceedings of the IFIP TC2/WG2.6 Ninth Working Conference on Database Semantics*, Kluwer, B.V., Deventer, The Netherlands, The Netherlands, pp. 3–17.

[23] Lai Xu (2004): *A multi-party contract model. SIGecom Exch.* 5(1), pp. 13–23.

[24] Lai Xu & Manfred A. Jeusfeld (2003): *Pro-active Monitoring of Electronic Contracts.* In: *Advanced Information Systems Engineering, Lecture Notes in Computer Science* 2681, Springer Berlin / Heidelberg, p. 1028.