Concept Abduction and Contraction in Description Logics

Simona Colucci[1], Tommaso Di Noia[1], Eugenio Di Sciascio[1], Francesco Donini[2], Marina Mongiello[1]

1: Dipartimento di Elettrotecnica ed Elettronica,

Politecnico di Bari, BARI, Italy

Email: {s.colucci,t.dinoia,disciascio,mongiello}@poliba.it

2: Facoltà di Scienze Politiche

Università della Tuscia, VITERBO, Italy

Email: donini@unitus.it

Abstract

Motivated by matchmaking in Peer-to-Peer electronic marketplaces, we study abduction and contraction in description logics. We devise suitable definitions of the problem, and prove some simple complexity results.

1 Motivation

Several recent proposals try to formalize with Description Logics (DLs) the description of supplies and demands in Peer-to-Peer electronic marketplaces (see [7, 15, 14, 6, 11, 5] among others). Usually, proposals tend to use the standard reasoning services of a DL system — subsumption and (un)satisfiability — to classify potential partners. In brief, if a supply is described by a concept C and a demand by a concept D, unsatisfiability of $C \sqcap D$ identifies the incompatible proposals, satisfiability identifies potential partners — that still have to agree on underspecified constraints — and subsumption $C \sqsubseteq D$ means that requirements on D are completely fulfilled by C.

However, we believe that *ranking* of potential counteroffers is fundamental to make a matchmaking service useful for an end user — similarly to "good" rankings for search engines in the WWW. Moreover, the ranking should be as transparent as possible, in case a user wants to know why a given proposal has been ranked before another. This transparency is crucial for our electronic business scenario, where we must give the user reasons to trust the system.

Therefore, we want to base our ranking functions on logical properties in DLs, which call for more sophisticated reasoning services in DLs.

2 Concept Abduction

We tend to follow the notation of [9] for propositional abduction whenever possible, and modify it as needed. A Propositional Abduction Problem is a triple $\langle H, M, T\rangle$ where H (Hypotheses) and M (Manifestations) are set of literals, and T (Theory) is a set of formulae. A solution for $\langle H, M, T\rangle$ is an Explanation $E \subseteq H$ such that $T \cup E$ is consistent, and $T \cup E \models M$. We adapt this framework to DLs as follows.

Definition 1 *Let $\mathcal{L}$ be a DL, C, D, be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$, where both C and D are satisfiable in $\mathcal{T}$. A* Concept Abduction Problem *(CAP), denoted as $\langle \mathcal{L}, C, D, \mathcal{T}\rangle$, is finding a concept $H \in \mathcal{L}$ such that $\mathcal{T} \not\models C \sqcap H \equiv \bot$, and $\mathcal{T} \models C \sqcap H \sqsubseteq D$.*

We use $\mathcal{P}$ as a symbol for a CAP, and we denote with $SOLCAP(\mathcal{P})$ the set of all solutions to a CAP $\mathcal{P}$. Observe that in the definition, we limit to satisfiable C and D, since C unsatisfiable implies that the CAP has no solution at all, while D unsatisfiable leads to counterintuitive results ($\neg C$ would be a solution in that case).

As propositional abduction extends implication, a CAP extends concept subsumption. But differently from propositional abduction, we do not make any distinction between manifestations and hypotheses, which is usual when abduction is used for diagnosis. In fact, when making hypotheses about properties of goods in e-marketplaces, there is no point in making such a distinction. This uniformity implies that there is always the trivial solution D to a non-trivial CAP $\langle \mathcal{L}, C, D, \mathcal{T}\rangle$, as stated more formally as follows.

Proposition 1 *If $C \sqcap D$ is satisfiable in $\mathcal{T}$, then $D \in SOLCAP(\langle \mathcal{L}, C, D, \mathcal{T}\rangle)$.*

Interpreted in our e-marketplace application domain, it means that if I hypothesize for the counteroffer C exactly all my specifications D, then the counteroffer trivially meets my specifications — if it was compatible anyway. However, in case $C \sqsubseteq D$ no hypothesis is really necessary, while in other cases the entire D must be supposed. Clearly, in order to start the transaction the first case is in a much better shape than the second one. Hence, if we want to use abduction to highlight most promising counteroffers, "minimal" hypotheses must be defined. In the following definition, we denote with $\sqsubseteq_{\mathcal{T}}$ the subsumption relation between concepts w.r.t. a TBox $\mathcal{T}$.

Definition 2 *Let $\mathcal{P}$=$\langle \mathcal{L}, C, D, \mathcal{T}\rangle$ be a CAP. The set $SOLCAP_{\sqsubseteq}(\mathcal{P})$ is the subset of $SOLCAP(\mathcal{P})$ whose concepts are maximal under $\sqsubseteq_{\mathcal{T}}$. The set $SOLCAP_{\leq}(\mathcal{P})$ is the subset of $SOLCAP(\mathcal{P})$ whose concepts have minimum length.*

We note that being maximal under $\sqsubseteq_{\mathcal{T}}$ is still a minimality criterion, since it means that no unnecessary hypothesis is assumed. It can be proved that the two measures are incomparable.

Proposition 2 *There exists a CAP $\mathcal{P}$ such that the two sets $SOLCAP_{\sqsubseteq}(\mathcal{P})$ and $SOLCAP_{\leq}(\mathcal{P})$ are incomparable.*

Proof. It is sufficient to consider $D = A_1 \sqcap A_2 \sqcap A_3$, $C = A_1$, and $\mathcal{T} = \{B \sqsubseteq A_2 \sqcap A_3\}$. The logic is even propositional. Then $A_2 \sqcap A_3 \in SOLCAP_{\sqsubseteq}(\langle \mathcal{L}, C, D, \mathcal{T} \rangle)$, $B \in SOLCAP_{\leq}(\langle \mathcal{L}, C, D, \mathcal{T} \rangle)$, and neither solution is in the other set. □
The proof highlights that, although $\leq$-minimality could be preferable for conciseness, it is heavily dependent on $\mathcal{T}$. In fact, for every concept $H \in SOLCAP(\mathcal{P})$, it is sufficient to add the axiom $A = H$ to get a $\leq$-minimal solution A.

Observe also that the length-minimal solution B could not be obtained by applying concept rewriting techniques — as defined in [1] — to $C \sqcap D = A_1 \sqcap A_1 \sqcap A_2 \sqcap A_3$, because $A_1 \sqcap B$ is not a rewriting of the former concept.

A third minimality criterion is possible for DLs which admit a normal form as a conjunction of concepts, that is, every concept C in $\mathcal{L}$ can be rewritten as an equivalent concept $C_1 \sqcap \cdots \sqcap C_n$. This is the case for $\mathcal{L} = \mathcal{ALN}$, and for the DL of the CLASSIC KR system. We call such a normal form CNF, in analogy with propositional logic.

Definition 3 *Let $\mathcal{P}{=}\langle \mathcal{L}, C, D, \mathcal{T} \rangle$ be a CAP in which $\mathcal{L}$ admits a normal form with conjunctions of concepts. The set $SOLCAP_{\sqcap}(\mathcal{P})$ is the subset of $SOLCAP(\mathcal{P})$ whose concepts are minimal conjunctions,* i.e., *if $C \in SOLCAP_{\sqcap}(\mathcal{P})$ then no sub-conjunction of C is in $SOLCAP(\mathcal{P})$. We call such solutions* irreducible.

It turns out that $\sqcap$-minimality subsumes both $\sqsubseteq_{\mathcal{T}}$-minimality and $\leq$-minimality. This is not a surprise, since $\sqcap$-minimality is a form of $\sqsubseteq_{\emptyset}$-minimality, *i.e.*, maximality for subsumption w.r.t. an empty TBox.

Proposition 3 *For every CAP $\mathcal{P}$ in which $\mathcal{L}$ admits a CNF, both $SOLCAP_{\sqsubseteq}(\mathcal{P})$ and $SOLCAP_{\leq}(\mathcal{P})$ are included in $SOLCAP_{\sqcap}(\mathcal{P})$.*

Proof. If a concept C is not $\sqcap$-minimal, then it is not length-minimal, and the same for $\sqsubseteq_{\mathcal{T}}$. □

2.1 Computational Complexity

Since Concept Abduction extends Concept Subsumption w.r.t. a TBox, complexity lower bounds of the latter problem carry over to decision problems related to a CAP.

Proposition 4 *Let $\mathcal{P}$=$\langle\mathcal{L}, C, D, \mathcal{T}\rangle$ be a CAP. If Concept Subsumption w.r.t. a TBox in $\mathcal{L}$is a problem $\mathcal{C}$-hard for a complexity class $\mathcal{C}$, then deciding whether a concept belongs to $SOLCAP(\mathcal{P})$ is $\mathcal{C}$-hard.*

Proof. Hardness for $\mathcal{C}$ comes from the fact that C subsumes D in $\mathcal{T}$ iff $\top \in SOLCAP(\mathcal{P})$. □
Hence, if $\mathcal{L}$ contains the DL $\mathcal{AL}$, then deciding whether a concept belongs to $SOLCAP(\mathcal{P})$ is EXPTIME-hard [8] for a general TBox $\mathcal{T}$, but it is PSPACE-hard if $\mathcal{T}$ contains only acyclic concept axioms [3].

Regarding upper bounds, a simple result can be derived from the fact that D is always a solution of the CAP $\langle\mathcal{L}, C, D, \mathcal{T}\rangle$ — although not always a minimal one. First of all, a total length-lexicographic order $\prec$ can be defined over concepts as follows: given two concepts $C, D \in \mathcal{L}$, let $C \prec D$ if either $|C| < |D|$, or both $|C| = |D|$ and C is lexicographically before D. Based on this total order, one can easily devise a simple-minded algorithm for finding a $\leq$-minimal solution of a CAP, using polynomial space relatively to an oracle for subsumption in $\mathcal{L}$. In fact, it is sufficient to try all concepts with less symbols than D, and return D if a shorter solution is not found. This provides an upper bound on the complexity of CAP, depending on the complexity class to which subsumption in $\mathcal{L}$ belongs to.

Theorem 1 *Let $\mathcal{P}$=$\langle\mathcal{L}, C, D, \mathcal{T}\rangle$ be a CAP. If subsumption in $\mathcal{L}$ belongs to a complexity class $\mathcal{C}$ that is included in PSPACE, then finding a concept in $SOLCAP_{\leq}(\mathcal{P})$ is a problem in PSPACE. Otherwise if PSPACE is included in $\mathcal{C}$, then finding a concept in $SOLCAP_{\leq}(\mathcal{P})$ is a problem in $\mathcal{C}$.*

Given that the problem of finding a solution cannot be simpler than the corresponding decision problem, we can conclude with some general results about $\leq$-minimal abduction.

Theorem 2 *Let $\mathcal{P}$=$\langle\mathcal{L}, C, D, \mathcal{T}\rangle$ be a CAP, with $\mathcal{L}$ a DL whose expressiveness is between $\mathcal{AL}$ and the DL containing concept constructors $\sqcap, \sqcup, \neg, \exists R.C, \forall R.C$ and role constructors $\sqcup$, role chain, transitive-reflexive closure of roles, role identity, role inverse, and $\mathcal{T}$ is a TBox with general axioms of the form $E \sqsubseteq F$. Then deciding whether a concept is in $SOLCAP_{\leq}(\mathcal{P})$ is a problem EXPTIME-complete.*

Proof. Hardness results for $\mathcal{AL}$ are in [8]. Membership result for the most expressive logic comes from converse-Propositional Dynamic Logic [16]. □
Hence, for a general TBox the best known algorithms require exponential time and also exponential space (unless unless one proves PSPACE = EXPTIME).

When the TBox is acyclic, complexity results for subsumption imply that finding a concept in $SOLCAP_{\leq}(\mathcal{P})$ is a problem PSPACE-complete for DLs whose expressiveness is between $\mathcal{ALE}$ [3] and $\mathcal{ALC}$ [12]. Even for the simplest logic $\mathcal{AL}$, the problem is co-NP-hard [3].

2.2 Irreducible solutions in $\mathcal{ALN}$

In this section, we assume that $\mathcal{T}$ of a CAP $\mathcal{P}= \langle \mathcal{L}, C, D, \mathcal{T} \rangle$ is always acyclic. Finding an irreducible solution is easier than finding a $\leq$-minimal or a $\sqsubseteq$-minimal solution, since a greedy approach can be used to minimize the set of conjuncts in the solution: starting from $C \sqcap D$, delete one redundant conjunct at a time from D. However, instead of starting from $C \sqcap D$, we adapt a structural subsumption algorithm [2] that collects all concepts that should be conjoined to C to be subsumed by D. The algorithm operates on concepts in the well-known normal form for Classic, which puts all $\forall R.C$ with the same role in one concept, simplifies redundant and inconsistent number restrictions, and expands all concept definitions and inclusions. In the following algorithm, we denote the fact that a concept A appears as a conjunct of a concept C with $A \in C$.

Algorithm $findIrred(\mathcal{P})$;
input: a CAP $\mathcal{P}= \langle \mathcal{L}, C, D, \mathcal{T} \rangle$, with $\mathcal{L}$=$\mathcal{ALN}$, acyclic $\mathcal{T}$
output: concept $H \in SOLCAP_{\sqcap}(\mathcal{P})$
(where $H = \top$ means that C is already subsumed by D)
variables: concept H
begin
  $H := \top$;
  for every concept name y in D
    if y is not in C
    then $H := H \sqcap y$;
  for every concept $(\geq\ n\ R) \in D$
  such that there is no concept $(\geq\ m\ R) \in C$ with $m \geq n$
    $H := H \sqcap (\geq\ n\ R)$;
  for every concept $(\leq\ n\ R) \in D$
  such that there is no concept $(\leq\ m\ R) \in C$ with $m \leq n$
    $H := H \sqcap (\leq\ n\ R)$;
  for every concept $\forall R.E \in D$
    if there exists $\forall R.F \in C$
      then $H := H \sqcap \forall R.findIrred(\langle \mathcal{L}, F, E, \mathcal{T} \rangle)$;
      else $H := H \sqcap \forall R.findIrred(\langle \mathcal{L}, \top, E, \mathcal{T} \rangle)$;
  /* now $H \in SOLCAP(\mathcal{P})$, but it might be reducible */
($\diamond$)**for every** concept $H_i \in H$
    if H without H_i is in $SOLCAP(\mathcal{P})$
    then delete H_i from H;
  return H;
end.

It can be proved that the concept H returned by $findIrred()$ is indeed an irreducible solution of $\mathcal{P}$. We explain the need for the reducibility check following

($\diamond$) with the help of an example. Let $\mathcal{T} = \{A_1 \sqsubseteq A_2, A_3 \sqsubseteq A_4\}$, and let $C = A_3$, $D = A_1 \sqcap A_4$. Then $\mathcal{L}$ is the propositional part of $\mathcal{AL}$. The normal form for C is $C' = A_3 \sqcap A_4$, while $D' = A_1 \sqcap A_2 \sqcap A_4$. Then before ($\diamond$) the algorithm computes $H = A_1 \sqcap A_2$, which must still be reduced to A_1.

As for complexity, the expansion of the TBox in the construction of the normal form can lead to an exponential blow-up, as demonstrated by Nebel in [13]. And anyway, a polynomial algorithm cannot be expected since subsumption in $\mathcal{AL}$ with an acyclic $\mathcal{T}$ is co-NP-hard [3]. However, in the cited paper Nebel argues that the expansion is exponential in the depth of the hierarchy $\mathcal{T}$; if the depth of $\mathcal{T}$ is $O(\log |\mathcal{T}|)$, then the expansion is polynomial, and so is the above algorithm.

Theorem 3 *Let $\mathcal{P}= \langle \mathcal{L}, C, D, \mathcal{T} \rangle$ be a CAP, with $\mathcal{L}=\mathcal{AL}$, and $\mathcal{T}$ an acyclic TBox whose depth is always bounded by $O(\log |\mathcal{T}|)$. Then finding an irreducible solution to $\mathcal{P}$ is a problem solvable in polynomial time.*

In order to rank the proposals in a marketplace according to how "near" they are to a given proposal D, we take the number of concept names in the irreducible solution returned by the above algorithm. The fact that a concept H is actually computed makes easy to devise an explanation facility, in case a user wants to know why a given proposal has been ranked before another. This transparency is crucial for our electronic business scenario, where we must give the user reasons to trust the system.

3 Concept Contraction

Other proposals [15, 14] usually exclude the case in which the concept expressing a demand is inconsistent with the concept expressing a supply, assuming that all requirements are strict ones. However, proposals for matchmaking outside DLs (*e.g.*, Markus and Stolze's) are much more liberal on this subject, allowing a user to specify negotiable requirements — some of which could be bargained in favor of others.

The logical formalization of negotiable requirements calls, in our opinion, for definitions already encountered in belief revision. In particular, we tend to follow Gärdenfors' [10] formalization, in which a revision of a knowledge base $\mathcal{K}$ with a new piece of knowledge A is a *contraction* operation, which results in a new knowledge base $\mathcal{K}_A^-$ such that $\mathcal{K}_A^- \not\models \neg A$, followed by the addition of A to $\mathcal{K}_A^-$ — usually modeled by conjunction.

Contraction is the operation which we are interested in, since from the fact that $C \sqcap D$ is unsatisfiable in a TBox $\mathcal{T}$, we want to retract requirements in C to obtain a concept K (for Keep) such that $K \sqcap D$ is satisfiable in $\mathcal{T}$. Clearly, a user is interested in what she must trade to conclude the transaction — a concept

G (for Give up) such that C was made by G and K. We try to formalize these ideas as follows.

Definition 4 *Let $\mathcal{L}$ be a DL, C, D, be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$, where both C and D are satisfiable in $\mathcal{T}$. A* Concept Contraction Problem *(CCP), denoted as $\langle\mathcal{L}, C, D, \mathcal{T}\rangle$, is finding a pair of concepts $\langle G, K\rangle \in \mathcal{L} \times \mathcal{L}$ such that $\mathcal{T} \models C \equiv G \sqcap K$, and $K \sqcap D$ is satisfiable in $\mathcal{T}$. We call K a* contraction *of C according to D and $\mathcal{T}$.*

We use $\mathcal{Q}$ as a symbol for a CCP, and we denote with $SOLCCP(\mathcal{Q})$ the set of all solutions to a CCP $\mathcal{Q}$. We note that there is always the trivial solution $\langle G, K\rangle = \langle C, \top\rangle$ to a CCP. This solution corresponds to the most drastic contraction, that gives up everything of C. In our e-commerce setting, it would model the (infrequent) situation in which, in front of some very appealing counteroffer D, incompatible with mine, I just give up completely my specifications C in order to meet D. On the other hand, when $C \sqcap D$ is satisfiable in $\mathcal{T}$, the "best" possible solution is $\langle\top, C\rangle$, that is, give up nothing — if possible.

Observe that as for concept abduction, we rule out cases where either C or D are unsatisfiable, as they correspond to counterintuitive situations.

Since usually one wants to give up as few things as possible, some minimality in the contraction must be defined. As in the previous section, we denote with $\sqsubseteq_{\mathcal{T}}$ the subsumption relation between concepts w.r.t. a TBox $\mathcal{T}$.

Definition 5 *Let $\mathcal{Q}$=$\langle\mathcal{L}, C, D, \mathcal{T}\rangle$ be a CCP. The set $SOLCCP_{\sqsubseteq}(\mathcal{Q})$ is the subset of solutions $\langle G, K\rangle$ in $SOLCCP(\mathcal{Q})$ such that G is maximal under $\sqsubseteq_{\mathcal{T}}$. The set $SOLCCP_{\leq}(\mathcal{Q})$ is the subset of $SOLCCP(\mathcal{Q})$ such that G has minimum length.*

As Concept Abduction extends Subsumption, Concept Contraction extends satisfiability — in particular, satisfiability of a conjunction $K \sqcap D$.

Proposition 5 *Let $\mathcal{L}$ be a DL containing $\mathcal{AL}$, and let Concept Satisfiability w.r.t. a TBox in $\mathcal{L}$be a problem $\mathcal{C}$-hard for a complexity class $\mathcal{C}$. Then deciding whether a pair of concepts is a solution of a CCP $\mathcal{Q}$=$\langle\mathcal{L}, C, D, \mathcal{T}\rangle$ is $\mathcal{C}$-hard.*

Proof. A concept $E \in \mathcal{L}$ is satisfiable w.r.t. a TBox $\mathcal{T}$ if and only if the CCP $\langle\mathcal{L}, \forall R.E, \exists R.\top, \mathcal{T}\rangle$ has the solution $\langle\top, C\rangle$. □

This gives a lower bound on the complexity of Concept Contraction, for all DLs that include $\mathcal{AL}$. For DLs not including $\mathcal{AL}$ note that if the proof showing $\mathcal{C}$-hardness of satisfiability involves a concept with a topmost $\sqcap$ symbol, the same proof could be adapted for Concept Contraction.

Similarly to Concept Abduction, also for Concept Contraction some simpler definitions are possible when the DL admits a normal form made up of conjunctions, and in this case, an algorithm which is a simple modification of a "structural satisfiability" algorithm is possible. Basically, the algorithm collects all contradictory conjunctions while traversing the syntactic trees of the two concepts. See some recent papers [6, 4] for numerical versions of the algorithm.

Once a contraction has been made, the obtained requirements K might still not give an exact match. Hence, a Concept Abduction problem can be set up between K and D, to see how many things are to be hypothesized in D to fulfill K: *e.g.*, "even if I give up smoking, I still don't know whether the householder accepts my dog and has autonomous heating system…".

4 Conclusion

Motivated by Peer-to-Peer matchmaking in electronic marketplaces, we set up definitions for Concept Abduction and Concept Contraction in DLs. We analyzed minimality criteria, and showed some preliminary results on complexity. We also devised algorithms for the simple DL $\mathcal{ALN}$. Preliminary experimentation [5] shows a good accordance with end-users judgements. Future research is needed to devise algorithms for more expressive DLs.

Acknowledgements

We thank two anonymous referees for helpful comments and criticism. This work has been supported by MURST project CLUSTER22, by EU-POP project *Negotiation Agents for the Electronic Marketplace*, by project *Tecnologie innovative per la valorizzazione e la fruizione dei Beni Culturali*, by Italian CNR projects *LAICO*, and *Metodi di Ragionamento Automatico nella modellazione ed analisi di dominio*.

References

[1] Franz Baader, Ralf Küsters, and Ralf Molitor. Rewriting concepts using terminologies. In *KR 2000*, pages 297–308, 2000.

[2] A. Borgida and P. F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.

[3] D. Calvanese. Reasoning with inclusion axioms in description logics. In *ECAI'96*, pages 303–307. John Wiley & Sons, 1996.

[4] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. A system for principled matchmaking in an electronic marketplace. In *WWW'03*. ACM, 2003.

[5] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Abductive matchmaking in description logics. In *IJCAI'03*, 2003. To appear.

[6] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Semantic matchmaking in a P-2-P electronic marketplace. In *SAC'03*, pages 582–586. ACM, 2003.

[7] E. Di Sciascio, F.M. Donini, M. Mongiello, and G. Piscitelli. A knowledge-based system for person-to-person e-commerce. In *Workshop on Applications of Description Logics (KI 2001)*, 18 September 2001.

[8] F. M. Donini. Complexity of reasoning. In *Description Logics Handbook*, chapter 3. Cambridge University Press, 2003.

[9] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *JACM*, 42(1):3–42, 1995.

[10] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Bradford Books, MIT Press, Cambridge, MA, 1988.

[11] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *WWW'03*. ACM, 2003.

[12] C. Lutz. Complexity of terminological reasoning revisited. In *LPAR'99*, pages 181–200, 1999.

[13] B. Nebel. Terminological reasoning is inherently intractable. *Artif. Intell.*, 43:235–249, 1990.

[14] K. Sycara, S. Widoff, M. Klusch, and J. Lu. Larks: Dynamic matchmaking among heterogeneus software agents in cyberspace. *Autonomous agents and multi-agent systems*, 5:173–203, 2002.

[15] D. Trastour, C. Bartolini, and C. Priest. Semantic web support for the business-to-business e-commerce lifecycle. In *WWW'02*, pages 89–98. ACM, 2002.

[16] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. of Computer and System Sciences*, 32:183–221, 1986.