Idea: Towards a Vision of Engineering Controlled Interaction Execution for Information Services[*]

Joachim Biskup and Cornelia Tadros

Fakultät für Informatik, Technische Universität Dortmund, Germany
{joachim.biskup,cornelia.tadros}@cs.tu-dortmund.de

Abstract. To protect an agent's own knowledge or belief against unwanted information inferences by cooperating agents, Controlled Interaction Execution offers a variety of control methods to confine the information content of outgoing interaction data according to agent-specific confidentiality policies, assumptions and reaction specifications. Based on preliminary experiences with a prototype implementation as a frontend to a relational DBMS, in this article we outline the architectural design and the parameterized construction of specific tasks to uniformly shield all information services in need of confinement, potentially comprising query answering, update processing with refreshments, belief revision, data publishing and data mining. Refraining from any intervention at the cooperating agents, which are also seen as intelligently attacking the defending agent's own interest in preserving confidentiality, the engineering solely aims at self-confinement when releasing information.

Keywords: agent, a priori knowledge, attacker assumption, belief, belief revision, confidentiality policy, constraint, censor, data mining, data publishing, formal semantics, frontend, group, inference control, inference-usability confinement, information engineering, information flow, information integration, invariant, logic, lying, overestimation, permission, possibilistic secrecy, prohibition, query answering, reasoning, refreshment, refusal, simulation, state, theorem-proving, update processing.

1 Introduction

People are communicating by using their computing devices – profiting from external facilities while purposely either sharing or protecting own informational resources. We consider the people's devices as a kind of intelligent agents which are interacting within a multiagent system. Accordingly, while being designed to cooperatively share its data with another agent in general, each agent also has to keep its own sensitive information confidential. In this report, we outline a specific approach to engineer such a "defending" agent, for the sake of supporting privacy as informational self-determination and protecting business assets.

[*] This work has been supported by the Deutsche Forschungsgemeinschaft (German Research Council) under grant SFB 876/A5 within the framework of the Collaborative Research Center "Providing Information by Resource-Constrained Data Analysis".

Our approach can be motivated by the case of Bob living with his family and running his own business, gathering any information he needs and maintaining the plans he pursues by means of his computing devices. This includes XML documents with personal details about him and his children, their electronic health records, a relational database about commercial offers and customers, and an AI system for assembling and evaluating ideas on further projects.

Bob is cooperating with a large variety of relatives, friends, business partners, officials and so on. Communicating with any of them, according to the agreed purposes behind the specific contact, he is willing to discretionarily share some pieces of information and selected details of his plans. At the same time, however, depending on the social relationship and guided by personal preferences, Bob might want each of the persons involved not to learn parts of the information and plans that appear to be sensitive regarding the specific situation.

So, Bob is facing the challenge to uniformly and consistently shield the information services he offers to others, balancing his and their interests in availability and integrity of data to be shared with his potentially conflicting own interest of context-specific confidentiality. Accordingly, he would like to employ a single control mechanism to confine the outgoing information flow from all his computing devices according to his personal or business needs and the nature of the social contact to the recipient envisioned. To be sure, this mechanism should not at all be invasive to the computing devices of the others, but only regulate the functionality of his own devices.

In other words, Bob seeks for installing and parameterizing a personal intelligent computing agent that securely mediates the interactions with the devices of the others such that the specifically expressed interests are actually automatically enforced, in each single interaction and over the time as well.

More abstractly, we aim at constructing a *defending agent*, i.e., an intelligent computing agent that will enable its owner to both conveniently and effectively deal with formal requirements about the following high-level issues:

- general *permission* to share data, for the sake of *availability*: declared as *interface* language;
- dedicated *prohibition* to acquire information, for the sake of *confidentiality*: declared as a *policy*, expressing *security constraints/invariants* on "released information";
- application-oriented *quality guarantees* to reflect aspects of the "real-world", for the sake of *integrity*: declared as *functional constraints/invariants* on own information.

2 From a Required Vision to Available Ideas

Our *vision* of a defending agent is driven by widespread requirements to enable individuals to discretionarily control the sharing of their information with others, which are treated as cooperation partners on the one hand and nevertheless perceived as potential attackers against wanted confidentiality on the other hand. While the requirements appear to be socially accepted, actually offered

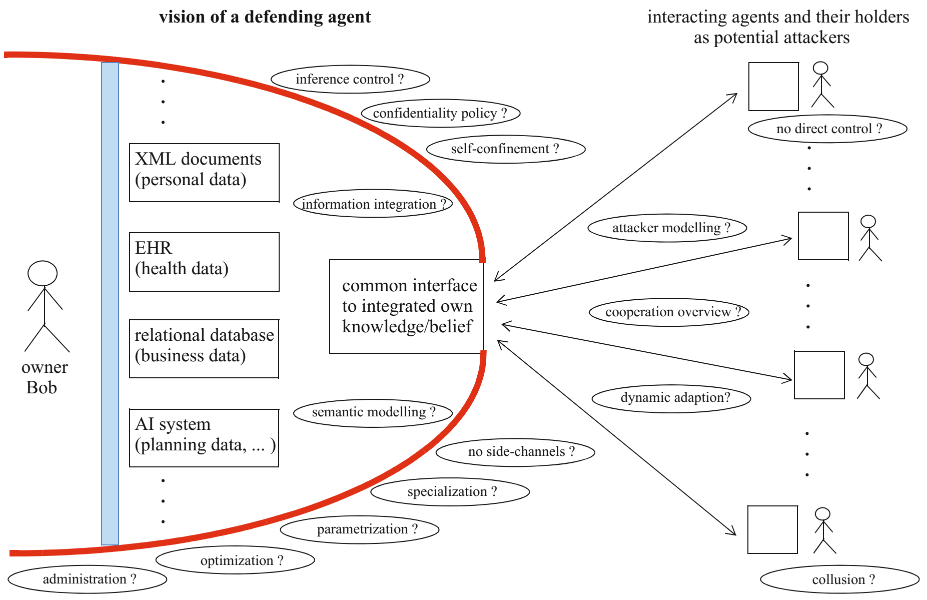

Fig. 1. The vision and major challenges

IT-systems rarely comply with them, and some people might even consider our vision to be just a dream, nice to have but impossible to achieve. We deal with the latter concern in a gradual way, rather than giving a strict yes-no answer.

In fact, in this article we argue for the thesis that our vision is approximately realizable indeed, by exhibiting an architectural design that exploits or partly is justified by *ideas* about actually *available technologies*. Trying a somehow bold correspondence, realizing the vision of controlling the release of own information is comparable to traveling to the moon: having been a dream for a long time, at a specific stage of scientific development, a concrete top-down plan was possible based on ideas from various fields. Clearly, just reaching the moon has left many further issues open, and so will our design be in need of many further features.

Figure 1 illustrates the vision and indicates major challenges. In the following, we gather most of these challenges into three main groups, and for each group we identify a basic idea about available technologies to solve the problems involved.

The challenges of inference control according to a confidentiality policy by self-confinement of the information supplier include the problems of attacker modelling and a complete cooperation overview. As a main idea towards a solution, we follow the approach of *Controlled Interaction Execution*, as summarized in Section 3. Though this approach is mainly logic-oriented, additional features are expected to be smoothly integrable, in particular numerical information like outputs of statistical functions and data mining support and confidence numbers. Attacker modelling in a large scale and, in particular, establishing an overview about cooperations are in need of adapting insight from, e.g., adversarial reasoning [18] and normative reasoning [1] about an inaccessible environment which does not provide complete knowledge about the dissemination of information.

We can further profit from the idea of *information engineering* [14] to deal with information integration requiring semantic modelling of diverse sources including their dynamic adaption. In particular, research on multi-context systems [13] facilitates the integration of knowledge with heterogeneous logical representations such as envisioned in the sketched example scenario. Presumably, an information owner may employ not only structured or at least semi-structured information services but also ad-hoc facilities like email conversation led in natural language. To avoid the resulting opening of side-channels, we would need additional expertise, e.g., to convert "freely" expressed information into "more structural" one, as explored in the field of information extraction [21].

Guided by the two ideas sketched above and adapting the respective technologies, we come up with the architectural design described in Section 4. To actually build a manageable system we suggest to employ the idea of modern *software development and maintenance*, in particular to cope with specialization and parametrization as well as automatic optimization and administration, as further exemplified in Section 5. Additional insight can be provided by the field of multiagent systems [23], in particular for the integration of agent systems with other technologies, including standards for communication protocols [16].

We willingly accept to exercise no direct control at all on the cooperating agents and, accordingly, we only indirectly deal with options of collusion among those agents. These evident shortcomings will remain unsolved.

We intend to complement other approaches to overcome their restrictions and shortcomings sketched as follows. *Access control* and *encryption* applied to single data items are helpful but in general not sufficient to protect against an intelligent agent that might combine data received and already available before and intelligently *infer* consequences. *Usage control* is conceptually mandatory in general, but requires to implant trusted components into the interacting devices, see, e.g., [20]. Cryptographic *multiparty computations* are powerful means for protecting numerically encoded information but in most cases are rather costly and not applicable for logic-oriented information, see, e.g., [17].

3 Summary of Controlled Interaction Execution

Specifically realizing *security automata*, see, e.g., [19], for a logic-oriented view on information services, our own approach of inference-usability confinement by *Controlled Interaction Execution*, CIE, has been summarized in [3,4]. This approach originated from a seminal proposal of Sicherman/de Jonge/van de Riet [22] in 1983, which later has been resumed and extended by Biskup/Bonatti, e.g., [5,7] and then has further been elaborated by Biskup et al, e.g., [11,8,12,2,9,10]. CIE has been proved to be in accordance with fundamental notions of secrecy, as unified by Halpern/O'Neill [15], while adding dedicated logic-oriented enforcement mechanisms.

We briefly summarize the main characteristics of CIE concepts:

- *cooperativeness:*
  - no intervention whatsoever at other agents (seen as potential attackers),
  - only self-confinement when releasing own information,
  - confidentiality requirements as exceptions from permissions to share data;
- *logic-orientation:*
  - information represented by (sets of) sentences of a suitable logic, coming along with formal semantics of formulas, to precisely capture notions of knowledge and belief,
  - information acquired either explicitly/directly from communication data or implicitly/indirectly inferred by intelligent reasoning,
  - focus on possibilistic secrecy of a sentence to be kept confidential, roughly meaning, belief in the possibility of the sentence being not valid from an attacker's point of view;
- *statefulness:*
  - rich supported functionality for interaction to share data, including query answering, update processing with refreshments, belief revision, and data publishing, potentially as well as related services like, e.g., data mining,
  - unlimited interaction sequences,
  - keeping track of interaction history and thus state-based reactions;
- *modelling of "attacking" agent:*
  - agent-specific assumptions and agent-specific policy,
  - several approaches to attacker-specific reactions on potentially harmful requests, namely refusal, weakening, lying and combinations thereof,
  - simulating an attacking agent's postulated reasoning about candidates for reactions on an attacker's request to determine potential harmfulness, before deciding how to actually react;
- *formal assurances:*
  - formally proved compliance with confidentiality requirements under specified assumptions and policies.

In the remainder of this note, we elaborate our vision of a defending agent of the wanted kind, based on the broad theoretical work on CIE and preliminary experiences with an ongoing prototype implementation for a less ambitious situation, only requiring a frontend to a single relational DBMS.

4 Architectural Design

As a starting point, we assume that there are one or more existing functional components for information services – like a local DBMS for a private company, XML documents for personal and family data, and electronic health records. Figure 2 then shows the overall design of an agent implementing CIE. That agent should uniformly shield the functional components as a common control frontend to confine the outgoing flow of information to each of the cooperating agents according to the pertinent agent-specific policy and assumptions.

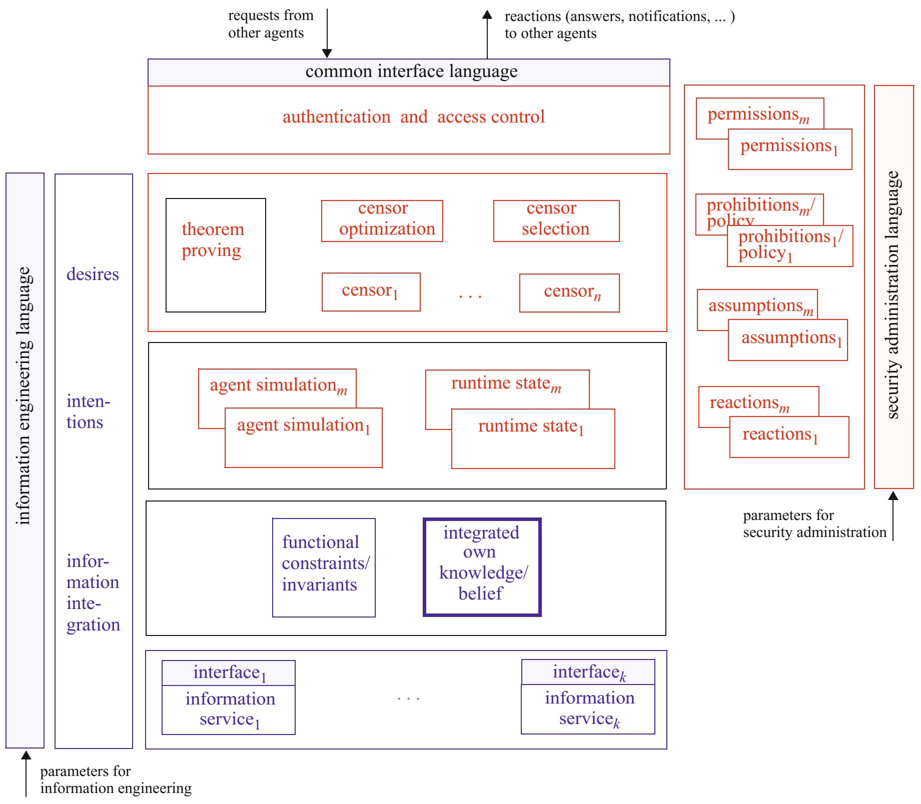
requests from other agents
reactions (answers, notifications, ...) to other agents
common interface language
authentication and access control
theorem proving
censor optimization
censor selection
censor$_1$
. . .
censor$_n$
agent simulation$_m$
agent simulation$_1$
runtime state$_m$
runtime state$_1$
functional constraints/ invariants
integrated own knowledge/ belief
interface$_1$
information service$_1$
. . .
interface$_k$
information service$_k$
information engineering language
desires
inten- tions
infor- mation inte- gration
parameters for information engineering
permissions$_m$
permissions$_1$
prohibitions$_m$/ policy
prohibitions$_1$/ policy$_1$
assumptions$_m$
assumptions$_1$
reactions$_m$
reactions$_1$
security administration language
parameters for security administration

Fig. 2. Controlled Interaction Execution uniformly shielding a defending agent's integrated own knowledge or belief by means of, e.g., n available censors regarding, e.g., k existing information services offered to, e.g., m currently cooperating agents

As a prerequisite, we need a technology of *information system integration*, e.g., [14,13], to treat the existing functional components in a uniform manner:

- embedding each individual interface into a *common interface*;
- forming the integrated own *knowledge/belief* – whether explicitly or implicitly – , part of which is the target of the policy for prohibitions;
- evaluating an incoming request expressed in the common interface language in terms of evaluations of subrequests to pertinent *information services*.

Though highly demanding in its own right, it is indispensable to form a *unified own knowledge/belief* to which all security measures should refer, in order to achieve consistent enforcement of the owner's interests in confidentiality, independently of the information services involved and the interactions requested.

Each of the cooperating agents is specifically treated, as discretionarily specified by the defending agent's owner acting as security officer by means of context-specific parameters for the following components: *permissions* granted to the agent, the *confidentiality policy* as *prohibitions*, *assumptions* about the agent, and the kind of *reactions* in case of violating requests. Given these specifications, the *runtime state* and the *simulation* of the cooperating agent are initialized, and an initial *censor* is selected to control the reactions to that agent.

The runtime state represents the cooperating agent, in particular to capture original parameters, possibly their modifications, and past and current requests. Based on the runtime state, over the time, the component of *censor selection* determines the current *censor* for the agent. Together with the reactions shown to the agent, the runtime state also determines the *simulation* of the agent: basically, this simulation serves to model the cooperating agent's current belief about the defending agent's integrated own knowledge/belief, aiming to ensure invariantly that the former never contains any part of the latter that the policy requires to keep confidential.

In fact, the simulation of a cooperating agent('s belief of the defender's own knowledge/belief) is most crucial for an effective self-control as favored by CIE: though an agent is treated as cooperating in principle, it is also seen as a "curious attacker" and, as such, it cannot be supposed to frankly tell its belief about the defender's knowledge/belief. So, agent modelling by the control component is the only alternative, and obviously it can only be based on two features: the assumptions as specified and the requests and reactions having been occurred. Since the control component has observed the requests and even generated the reactions, it is completely certain about this feature; however, the assumptions are inherently uncertain. Accordingly, the parameters for the assumptions and their representation within the runtime state should be expressive enough to capture all relevant aspects of many and diverse situations; moreover, expressiveness should come along with best achievable orthogonality of dimensions to enable automatic translation into a runtime state and later uniform processing for censor selection and even *censor optimization* as well as agent simulation.

Besides the always postulated attacker's *system awareness* according to "no security by obscurity", i.e., knowledge of both the functional components and the confining frontend, the following parameter dimensions are most important:

– the attacker's *configuration awareness*: knowledge of specific declarations (interaction language, policy (security constraints), functional constraints);
– the attacker's applicable *common knowledge*: knowledge regarding the application, in particular schema declarations;
– the attacker's *specific knowledge*: knowledge resulting from other sources etc.;
– the attacker's *specific reasoning*: "procedural knowledge" to form beliefs etc.;
– the attacker's *guess* of the defender's hidden parameters (kind of defender's functional reasoning etc.).

Notably, in principle not only the defending agent to be implemented is uncertain about an attacking agent's belief, but also the attacking agent is uncertain how it is simulated by the defender, and thus there is mutually recursive uncertainty to be suitably resolved. Indeed, each particular censor working with a specific simulation has to be justified by a convincing postulate on the *coincidence* of the actual attacker with the defender's simulation.

5 Uniformity for Specific Engineering Tasks

Within the design, we will have to treat many specific engineering tasks. As exemplified in the following, for each task we envision to achieve *uniformity* across the anticipated variety of situations: conceptually, the situations are captured by a powerful *abstraction* which then, by an implementation, is strictly *encapsulated* into a single module; further, the selection of a specific situation is enabled by a powerful *parameterization* and embodied by a *specialization* of that module.

Existing information services may considerably vary in the *underlying logic*, which, basically, defines the specific semantics. For example, an information service might be based on a *completeness* assumption – enabling reasoning about a "closed-world" [5,6,12]–, faced with partial or principle *incompleteness* – restricting or even disabling reasoning about "negative information" [11,9] –, or equipped with a concept of *preferences* – introducing essential differences between certain knowledge and uncertain belief and requiring non-monotonic reasoning for revisions and updates [10]. Given the pertinent parameters for the information services to be integrated, the totality of information available to the owner is abstracted into the integrated own knowledge/belief, and querying and anticipated manipulations, respectively, are encapsulated by specific modules, which have appropriate specializations for the potentially occurring variations.

Under each of these variations, a censor basically has to evaluate possible reactions on a request regarding *harmfulness* of adding the supplied information to the requester's current belief. This belief is abstracted as the pertinent agent simulation, and the needed evaluations are encapsulated by a module for determining harmfulness. Again, this module has appropriate specializations, which might employ the incorporated theorem prover to reduce the problem of harmfulness to a suitable entailment problem in the pertinent logic.

The abstraction of an agent simulation permits a large range of actual implementations. In a simple dynamic case, there is just a *logfile* containing sentences reflecting the a priori knowledge and the reactions provided so far [5,6,11]. For optimization, instead of keeping a logfile an *adapted version* of the original confidentiality policy might be maintained [2]. In the more advanced context of non-monotonic belief management, the agent simulation comprises both an *approximation* of the own knowledge/belief and *skeptical reasoning* regarding the aspects left only approximated [10]. For the static case of data publishing, a possibly distorted, "inference-free" *alternative view* on the own knowledge/belief is generated beforehand and later employed like an agent simulation, for which the associated censor does not need to perform any further dynamic control [12,9].

6 Experiences, Further Issues, and Concluding Remarks

We are implementing a *CIE-prototype* [3,4] for a simplified scenario, only shielding a single *relational DBMS*, Oracle, for somehow restricted interactions. This prototype provides a uniform treatment of all included interactions for various kinds of cooperating agents and the permissions, prohibitions (policies), assumptions and reactions specified for them. For each such a situation, the actual control is established by a dedicated specialization of a general censor component.

Successful CIE operation needs powerful tools and facilities for the *administration* of agent-specific parameters, and (semi-)automatic *optimization.* In particular, the initial selection of an appropriate censor instance followed by repeated reconsiderations appear to be crucial. As expected by theoretical insight, computational complexity and scalability remain a major issue. Accordingly, identification of parameters leading to feasible cases and their automatic recognition as part of optimization are further important topics. Our experiences also suggest to sometimes refrain from the principle of minimal distortion, but instead to look for efficiently computable overestimations of an agent's simulation.

A main concern is to achieve *robustness* of the defending agent's attempt to simulate the postulated behavior and reasoning of another agent seen as attacker: what assurances regarding preservation of confidentiality can the defending agent get if the attacker's situation actually differs from the defender's simulation?

There are several further issues whose solutions might have an impact on both the architectural design and specific tasks. Basically, in each case we would have to decide whether to include additional control components or only refined parameters for the already existing components, the latter found to be extremely worthwhile so far and hoped to be extendable. In the following, we briefly list selected issues: *collusion* among cooperating agents has to be made useless by treating a group of agents like a single agent; possibilistic confidentiality might be strengthened, for example to *k-confidentiality* demanding a stronger "negative belief", by ensuring the existence of at least $k > 0$ essentially different counterexamples, or to *probabilistic secrecy* or related notions; we might also weaken our current notion into a sort of *complexity-theoretic secrecy*; seen from a broader perspective, the *desires* and *intentions* of a BDI-like agent [23] might influence the agent-specific parameters, as do *normative concepts* [1]; as usual, production of *reliable software* and its appropriate *installation* are mandatory, leaving no options to circumvent the censoring or to exploit side-channels.

Concluding, we advocate the engineering of inference control as a *frontend* to existing functional components as a promising step to our vision, emphasizing a *uniform treatment* of various situations by means of strict *encapsulation* of powerful *abstractions*, expressive *parameterization*, and the concept of *specialization.*

References

1. Andrighetto, G., Governatori, G., Noriega, P., van der Torre, L.W.N. (eds.): Normative Multi-Agent Systems. Dagstuhl Follow-Ups, vol. 4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2013)
2. Biskup, J.: Dynamic policy adaption for inference control of queries to a propositional information system. Journal of Computer Security 20, 509–546 (2012)
3. Biskup, J.: Inference-usability confinement by maintaining inference-proof views of an information system. International Journal of Computational Science and Engineering 7(1), 17–37 (2012)

4. Biskup, J.: Logic-oriented confidentiality policies for controlled interaction execution. In: Madaan, A., Kikuchi, S., Bhalla, S. (eds.) DNIS 2013. LNCS, vol. 7813, pp. 1–22. Springer, Heidelberg (2013)
5. Biskup, J., Bonatti, P.A.: Controlled query evaluation for enforcing confidentiality in complete information systems. Int. J. Inf. Sec. 3(1), 14–27 (2004)
6. Biskup, J., Bonatti, P.A.: Controlled query evaluation for known policies by combining lying and refusal. Ann. Math. Artif. Intell. 40(1-2), 37–62 (2004)
7. Biskup, J., Bonatti, P.A.: Controlled query evaluation with open queries for a decidable relational submodel. Ann. Math. Artif. Intell. 50(1-2), 39–77 (2007)
8. Biskup, J., Gogolin, C., Seiler, J., Weibert, T.: Inference-proof view update transactions with forwarded refreshments. Journal of Computer Security 19, 487–529 (2011)
9. Biskup, J., Li, L.: On inference-proof view processing of XML documents. IEEE Trans. Dependable Sec. Comput. 10(2), 99–113 (2013)
10. Biskup, J., Tadros, C.: Preserving confidentiality while reacting on iterated queries and belief revisions. Ann. Math. Artif. Intell. (2013), doi:10.1007/s10472-013-9374-6
11. Biskup, J., Weibert, T.: Keeping secrets in incomplete databases. Int. J. Inf. Sec. 7(3), 199–217 (2008)
12. Biskup, J., Wiese, L.: A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. Theoretical Computer Science 412, 4044–4072 (2011)
13. Brewka, G.: Multi-context systems: Specifying the interaction of knowledge bases declaratively. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 1–4. Springer, Heidelberg (2012)
14. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: View-based query answering in description logics: Semantics and complexity. J. Comput. Syst. Sci. 78(1), 26–46 (2012)
15. Halpern, J.Y., O'Neill, K.R.: Secrecy in multiagent systems. ACM Trans. Inf. Syst. Secur. 12(1), 5.1–5.47 (2008)
16. Huget, M.-P., Poslad, S.: The Foundation of Intelligent Physical Agents, `http://www.fipa.org`
17. Kolesnikov, V., Sadeghi, A.-R., Schneider, T.: A systematic approach to practically efficient general two-party secure function evaluation protocols and their modular design. Journal of Computer Security 21(2), 283–315 (2013)
18. Kott, A., McEneaney, W.M. (eds.): Adversarial Reasoning: Computational Approaches to Reading the Opponent's Mind. Chapman & Hall/CRC, Boca Raton (2007)
19. Ligatti, J., Bauer, L., Walker, D.: Run-time enforcement of nonsafety policies. ACM Trans. Inf. Syst. Secur. 12(3) (2009)
20. Pretschner, A., Hilty, M., Basin, D.A.: Distributed usage control. Commun. ACM 49(9), 39–44 (2006)
21. Sarawagi, S.: Information extraction. Foundations and Trends in Databases 1(3), 261–377 (2008)
22. Sicherman, G.L., de Jonge, W., van de Riet, R.P.: Answering queries without revealing secrets. ACM Trans. Database Syst. 8(1), 41–59 (1983)
23. Wooldridge, M.J.: An Introduction to MultiAgent Systems, 2nd edn. Wiley, Hoboken (2009)