Approaching a Formal Definition of Fairness in Electronic Commerce

Felix C. Gärtner* Henning Pagnia Holger Vogt†
Computer Science Department
Darmstadt University of Technology
D-64283 Darmstadt, Germany
{felix|pagnia|holgervo}@informatik.tu-darmstadt.de

Abstract

The notion of fairness *is a very general concept and can be used to coin terms in many different application areas. Recently, the term fairness has appeared in the context of electronic commerce. Here, the term* fair exchange *refers to the problem that two parties want to swap some distinct items in a way which ensures that no participant can gain advantage over the other. Many protocols for fair exchange have been proposed but comparing or formally verifying them has remained rather difficult. The reason for this is that the notion of fairness they use is often different, and exact (i.e. formal) fairness definitions do not exist. In this paper, we make a first attempt to approach a formal definition of fairness in electronic commerce. We do this by reviewing the established terminology regarding the notion of fairness in concurrency theory and adapting the formal apparatus to derive three precisely separable definitions of fairness in electronic commerce which we call* strong, eventually strong *and* weak fairness.

1. Introduction

Fairness is a word with several meanings. The latest edition of Longman's Dictionary of Contemporary English [1] contains at least eight distinct entries for the adjective *fair* alone, not counting idioms such as "have more than your fair share". It is therefore not surprising that the notion of fairness has been incorporated into academic terminology in many different and seemingly unrelated ways.

When surveying some of the different usages of the term *fair* in computer science, there are mainly two slightly different linguistic meanings associated with it:

1. The first refers to behaviors which seem to be *proper, decent, reasonable* or *generally acceptable*.
2. The second refers to systems or actions which *treat* every affected item *equally* and play *according to "the rules"* (note that this must not generally be the same as the first meaning).

We have come across an adaption of the second meaning in the field of electronic commerce. The vision of this branch of computer science is to build and enable technologies [2] allowing to perform maintenance, provide services, or advertise, sell and even distribute commercial products through global computer networks like the Internet. A major problem which must be solved in this scenario is to permit a *fair exchange* of payment and the electronic goods in consideration between two participating parties (or players). Different notions of fairness have been introduced in this area (such as *strong* and *weak fairness* [6]), but their definitions have remained rather informal. We believe that it is important to have formal definitions of these terms, because without a precise terminology it is hardly possible to compare or verify fair exchange protocols.

There exists an established and completely formalized terminology regarding the notion of fairness in concurrency theory [11, 15]. Generally spoken, these notions refer to the timeliness constraints imposed onto the scheduled execution of concurrent actions in certain system models.

In this paper we try to approach a formal definition of the term *fairness* in electronic commerce. We briefly compare the distinct terminologies regarding fairness in electronic commerce and concurrency theory. It turns out that the benefits of such a comparison do not only lie in showing the differences of terminology; by adapting formality and rigor of the formal apparatus of concurrency theory it is possible to refine and clarify the definitions of fairness in electronic commerce.

The remainder of this paper is structured as follows: First we present the two distinct terminologies concerning fairness in the fields of electronic commerce (Section 2) and concurrency theory (Section 3). We compare both concepts in Sec-

*This author's work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the PhD program (Graduiertenkolleg) "Intelligente Systeme für die Informations- und Automatisierungstechnik" at Darmstadt University of Technology.

†This author's work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the PhD program "Enabling Technologies for Electronic Commerce" at Darmstadt University of Technology.

tion 4 and use the results of the comparison to derive a new set of fairness concepts for electronic commerce in Section 5. Section 6 concludes this paper.

2. Fairness in Electronic Commerce

Asokan [6] defines a *fair* system as being a system "that does not discriminate against a correctly behaving player. [...] As long as a player behaves correctly, a fair system must ensure that other players will not gain any advantage over the correctly behaving players." [6, p. 4] In the context of fair exchange of items between two parties A and B, a system must *not* allow, for example, that B can obtain access to A's item while A cannot access B's item. More specifically, a system which implements a *fair exchange protocol* between A and B must ensure three conditions [6, p. 9]:

1. Effectiveness: If A and B behave correctly and if the protocol terminates, then A will have B's item and vice versa.
2. Timeliness: The protocol will eventually terminate.
3. Fairness.

The fairness condition comes in two flavors:

- *strong fairness*: When the protocol has completed, either A has B's item, or B has gained no additional information about A's item, or vice versa [6, p. 9]. In other words: A node which behaves correctly does not suffer a disadvantage, i.e., it is never the case that A receives B's item and B does not receive A's item, or vice versa.
- *weak fairness*: Either strong fairness is achieved, or a correctly behaving node A can prove to an arbiter that B has received (or still can receive) A's item without any further intervention from A [6, p. 9] (no matter whether B behaves correctly or not), and vice versa.

The reason for having two versions of fairness lies in the fact that guaranteeing strong fairness is often very costly and sometimes even impossible to achieve. In these cases, the weak version of fairness is useful. If a node has suffered a disadvantage, it should be possible to start a dispute outside of the system. If a node can prove its disadvantage, strong fairness can be re-established, for example, by order of a court.

Protocols have been presented that are able to guarantee strong fairness with the help of a trusted third party [7–9, 20]. Apart from Bürk and Pfitzmann [9] they all refer to the fairness definition of Asokan [6]. Other protocols (like those of Jakobsson [12], Pagnia and Jansen [19], and Sandholm and Lesser [21]) do not exactly specify the type of fairness achieved, which makes the task of comparing them difficult. While other fairness definitions have been presented (like the notions of *money atomicity* and *goods atomicity* of Tygar [25]), they have not been stated precisely enough to attempt a formalization. This is why we take the definitions of Asokan as a basis for the formalizations in the following sections.

3. Fairness in Concurrency Theory

In concurrency theory, fairness is a concept to define different levels of timeliness constraints placed on concurrent actions. To explain this, we must introduce a formal system model in Section 3.1 and some additional terminology in Section 3.2 before we can present the most prominent definitions in Section 3.3.

3.1. System Model

We consider a system to be an automaton with a possibly infinite number of states (determined by the values of some set of variables). The automaton has a set of initial states I and a state transition relation δ. The relation δ is given as a finite set of *guarded commands* [10] (or *actions*) of the following form:

$$\begin{array}{ccc} \langle \text{guard}_1 \rangle & \rightarrow & \langle \text{command}_1 \rangle \\ \langle \text{guard}_2 \rangle & \rightarrow & \langle \text{command}_2 \rangle \\ \vdots & & \vdots \\ \langle \text{guard}_n \rangle & \rightarrow & \langle \text{command}_n \rangle \end{array}$$

Every *guard* is a boolean expression over the system state. If for some state c a guard evaluates to the value true, we say that the guarded command is *enabled*. The *command* is basically an assignment to the state variables which in effect changes the state.

An *execution* of the system is an infinite sequence $c_0, c_1, c_2, \ldots$ of states where

- $c_0 \in I$ and
- for every state c_i, c_{i+1} is obtained by nondeterministically choosing a guarded command which is enabled in state c_i and executing the command.

It is assumed that state transitions occur atomically.

The nondeterministic choice of which guarded command to execute is attributed to some hypothetical *daemon* or *scheduler* [11]. Restrictions placed on such a scheduler are usually called *fairness conditions*. However, before discussing the different forms of fairness, we must introduce some additional terms.

3.2. Properties: Safety and Liveness

A system is sometimes identified with the set of executions which it might produce. Due to the nondeterminism inherent in the state transition relation more than one execution is possible. A *property* of a system is a set of system executions. There are two special kinds of properties called *safety* and *liveness*. A *safety property* informally states that "something bad will never happen". Safety properties can be thought of as invariants, i.e. sets of states which usually describe the "safe", "legal" or "good" states of the system. Formally, a safety property S is a set of executions which is prefix-closed, i.e. for every execution $e \in S$ and every prefix α of e, α must also be in S. Intuitively, if a property is finitely refutable, then it is a safety property. Partial correctness, deadlock freedom and the effectiveness condition of fair exchange are examples of safety properties.

On the other hand, a *liveness property* informally states that "something good will eventually happen". To discuss liveness it is necessary to stress that we have defined executions as *infinite* state sequences (a finite execution is technically turned into an infinite execution by infinitely repeating the final state). A partial execution α is said to be live for a property L, if there exists an execution $e \in L$ of which α is a prefix. A property L is called a *liveness property* if every partial execution in L is live. Intuitively, a liveness property states that any partial system execution α is "not lost", i.e. it is always possible to extend α to still reach some goal. Termination, starvation freedom, eventual delivery and the timeliness requirement of fair exchange are examples of liveness properties.

Safety and liveness were introduced by Lamport [13] and later formally refined by Alpern and Schneider [4]. It has been argued that almost all important system properties can be expressed as a combination of a safety and a liveness property [4, 14]. We will try to make use of this fact when specifying fairness in electronic commerce.

3.3. Fairness Definitions

Now we present the three most common definitions of fairness in concurrency theory; they are usually called *unconditional*, *weak* and *strong fairness* [11]. In *unconditional fairness* the scheduler must ensure that every guarded command is executed infinitely often. This is a rather strong condition and requires that every guarded command must be continuously enabled. More interesting concepts of fairness take an *enabledness* of guarded commands into account.

We will try to illustrate the different fairness conditions using a "real world" example (attributed to Francez [11]). Consider a company which delivers items to customers over the day using a single car. The company employs a set of n drivers. Every morning all drivers show up for work and the company owner selects one of the drivers who will do the job for the day. All others go home again. This procedure is repeated every workday (and therefore infinitely often). The drivers stand for the guarded commands, the company owner is the scheduler and the selected driver is the chosen action. Drivers can become sick sometimes and thus do not show up for work. This represents enabledness of an action. For example, an unconditionally fair company owner would ensure that every driver will eventually take a turn in driving (i.e., every driver will drive infinitely often).

In *weak fairness*, it is ensured that once a guarded command becomes enabled and once it remains *continuously* enabled it will eventually be executed. In terms of the example, a driver may become sick, but as long as a driver shows up on enough consecutive workdays, he will take his turn in driving.

Strong fairness guarantees that as long as a guarded command is enabled infinitely often, it will eventually be executed. This is a stronger guarantee than weak fairness. To see this, consider the car example again: Say some driver is sick every other day. With a weakly fair company owner, he or she might never be chosen to drive. However, because the driver turns up often enough, a strongly fair company owner will eventually give him or her the chance to drive.

The fairness definitions described above seem to be generally accepted [11, 16, 24] but have been always tied to the automata-based system models of concurrent and distributed systems with interleaved execution sequences. In such a model, fairness represents a limitation on the relative speeds of system components. Fairness conditions rule out executions where some action is always (or infinitely often) enabled but is denied to actually take place. Fairness assumptions are central to problems that have to ensure some liveness property in the presence of faulty behavior of components. Take as an example the following problem: Some process A wants to detect whether a remote process B has crashed (i.e., has stopped to take further steps). If there are no fairness assumptions and B does not think of crashing, B may take arbitrarily many steps before A is allowed to execute an action. Now A cannot distinguish between the two cases in which (1) B has crashed and thus does not react to A's messages, or (2) B is favored by the scheduler infinitely more often than A. So the properties of algorithms critically depend on the level of fairness inherent in the system.

There are other fairness notions, too. For example *finitary fairness* [5] requires that for every execution of the system there is an unknown bound k such that no enabled action is postponed more than k consecutive times. However, the important thing to note here is that unconditional, weak and strong fairness describe liveness properties, while the other forms of fairness (e.g. finitary fairness) are mostly safety properties.

4. Comparison

In this section we compare terminologies. As to the authors' knowledge, such a comparison has not yet appeared in the literature. The only other work which has at least remarked the problem statement is Asokan [6, p. 136].

In concurrency theory, fairness properties are mostly "eventuality", i.e. liveness properties. In electronic commerce, fairness seemingly is related to a notion of "non-discrimination" or "treat equally". In terms of safety and liveness, what kind of property is this? To answer this question, it is necessary to state precisely what fairness is in electronic commerce. Pagnia and Gärtner [18] have attempted to do this concerning *strong fairness*. They show that strong fairness can be formalized as the following safety property:

- (strong fairness) it is never the case that A receives B's item and B does not receive A's item, or vice versa.

The inherent assumption behind this formalization [18] is that items are atomic, i.e. they can only be exchanged in their entirety. The property is a safety property because we can tell in finite time whether it is violated.

How can *weak fairness* in electronic commerce be described in terms of safety and liveness? To answer this question we should ask: Can we tell in finite time whether weak fairness is violated? In terms of the definition by Asokan cited in Section 2 this would mean to ask the *weak fairness property question*:

Q1 Is there a point in time where (a) strong fairness is violated, and (b) a participating party (say, A) loses its ability to prove that the other party has received or still can receive A's item?

Obviously, if strong fairness is violated, some participating party has achieved (or can achieve) an advantage over the other party and there is always a point in time when this becomes visible. So the answer to this question depends on the definition of when a party is able to prove the required fact.

Proving something means to collect sufficient evidence and to convince somebody of this fact by presenting this evidence. What "sufficient" actually is and to whom this evidence is presented is not clear in the context of electronic commerce. This is the point where it is possible to clarify definitions. There are two different possibilities now:

- An answer of "yes" to the weak fairness property question (Q1) means that executing the exchange may result in a permanent disadvantage of either party. (In effect, weak fairness would be understood as a safety property.) This means that there are no means within the system to refute the disadvantage.

- An answer of "no" would mean to assume that some form of disadvantage has occurred but that eventually the disadvantage can be refuted within the system. This is a liveness property.

To refute a disadvantage, there must exist some additional assumptions about a misbehaving party. Such an assumption could be, for example, that a misbehaving party will (or can be forced to) eventually cooperate. (Note that the means to force a party to cooperate are fully within the automated system.) However, such assumptions are not very realistic in the usual settings of electronic commerce. However, we meet these assumptions again in a different context.

Schneider [22] investigates system properties which can be automatically enforced onto a system by mechanical means. He presents a theoretical result which shows that liveness properties cannot be automatically enforced without making additional assumptions about the misbehaving node. Thus, specifying weak fairness as a liveness property has important practical implications. If we cannot guarantee a certain behavior of the participating nodes, weak fairness as "liveness" cannot be achieved within the automated system.[1]

Overall, the comparison of the concepts of fairness in concurrency theory and electronic commerce results in the following findings:

- There are two possible forms of (electronic commerce) weak fairness formalizations: a liveness form using special assumptions about participating nodes, and a safety form without special assumptions.

- (electronic commerce) fairness in its formalizations as safety properties is unrelated to (concurrency theory) fairness.

- (electronic commerce) weak fairness in its formalization as a liveness property is related to (concurrency theory) fairness in the sense that we need (concurrency theory) fairness assumptions to prove liveness properties in a specific system model.

The fact that different assumptions are necessary in the two possible definitions of weak fairness can be used to approach a refined terminology concerning fairness in electronic commerce, which is attempted in the following section.

5. Approaching a Formal Definition of Fairness in Electronic Commerce

In the previous section we have met three distinctive forms of fairness: strong fairness and two variants of weak fairness. On the basis of these three variants we now derive formal fairness definitions for electronic commerce.

5.1. Strong Fairness

We start with the strongest notion of fairness which we have come across in the context of electronic commerce. This is a form of fairness which can be ensured completely within the system without additional assumptions about participating nodes. We call this form of fairness *strong fairness*. This form of fairness is formalized as a safety property along the lines of Pagnia and Gärtner [18].

5.2. Eventually Strong Fairness

The next level of fairness can also be guaranteed automatically within the system but only if additional assump-

[1] This fact has been briefly discussed in the context of open and closed systems by Pagnia and Gärtner [18] and appears in a more formal context in a paper by Abadi and Lamport [3].

tions about the participating parties are made. Such assumptions generally state that eventually some state can be imposed onto a participating party or that such a party eventually cooperates. Note that these additional assumptions are formal assumptions which can be derived from investigating the automated system. In practice such assumptions can be, for example, that a remote party runs within a trusted computing environment [26]. In such an environment, state can be imposed onto such a party if it misbehaves. For example, it could be forced to reveal its item by order of a procedure invoked by some trusted authority, or it could be terminated and lose all of its information if it has been misbehaving [22].

Overall, if fairness can be guaranteed automatically by the system but only because of additional assumptions about the other party (e.g., eventual cooperation), we call this level of fairness *eventually strong fairness*. This corresponds to the "liveness" version of weak fairness in the previous section.

Note that assumptions made about components outside of the system are not treated as "additional assumptions" in the previous definition. For example, consider the following scenario: Alice wants to purchase some item from Bob using fair exchange and Bob happens to get hold of Alice's money without sending her his item. For some reason, Alice knows Bob's wife, so she calls her and explains the state of affairs. Subsequently, by some family pressure, his wife forces Bob to send Alice the item. This scenario results in an effectively executed fair exchange, but the level of fairness guaranteed by the used protocol is in no form strongly fair, since means outside of the system have been used.

5.3. Weak Fairness

If fairness cannot be guaranteed within the system (because for example no additional assumptions about the parties exist), but if sufficient evidence can be gathered to resolve the conflict outside of the system, then we will speak of *weak fairness*. (This version of fairness refers to the "safety" version of weak fairness from Section 4.) Obviously, the term "sufficient evidence" refers to enforcement mechanisms outside of the system and thus an exact definition of this aspect is a question of law.

5.4. Consequences

Recall that fairness is a property of a system or protocol. Usually, we come across statements of the form "protocol of the form x guarantees fairness of type y." The benefits of the fairness definitions from this section are that we can verify answers to this question rather mechanically now.

For example, the property of strong fairness can be verified using the standard proof techniques for reasoning about distributed algorithms (see for example the overview of Tel [24]). This involves invariance arguments to prove safety properties and well-foundedness arguments to prove liveness properties.

If a protocol in a particular system (provably) does not ensure strong fairness (e.g., because a party can gain an advantage by interrupting the protocol at some given point in time), it is sometimes possible to show that the protocol at least guarantees eventually strong fairness. To do this, we must identify the additional assumptions which make strong fairness attainable (e.g., that a misbehaving party may be terminated by a trusted authority). We can then use these properties within the preconditions and try to derive the strong fairness property by the usual means. If this can be done, the protocol ensures eventually strong fairness.

The case of weak fairness depends on the definition of "sufficient evidence". So the prerequisite to a formal proof that a protocol ensures weak fairness is to exactly specify what data must be collected. This could be, for example for party A, a receipt signed by B that B has received A's item. The term "signed" usually refers to the use of a digital signature scheme. Local law regulations must be consulted to exactly define the type and security level of the cryptosystem to use. If this "sufficient evidence" is exactly specified, we can then attempt a formal proof using the formulation of weak fairness as a safety property and applying the usual proof techniques as explained above.

With the proof obligations formally defined, we can now also embark on studying which level of fairness is at most achievable in a particular system model. For example, it can be shown that in asynchronous systems with only two processes there cannot be a protocol which ensures strong fairness if a party is allowed to act maliciously [18].

6. Conclusions

Fairness is a complex term and its different meanings have been attached to different concepts throughout computer science. We have attempted to compare the different uses of fairness in the long-used and well-established terminology of concurrency theory and in the strongly-emerging field of electronic commerce. By doing this we have observed that the common linguistic distinction between "play to the rules/treat equally" and "be acceptable/decent/reasonable" lends itself nicely to two different facets of fairness: those which can be described as safety properties and those which are liveness properties. As a result, we argued that the terminology in both fields is rather incompatible and the different usages must not be confused. However, we benefited from the formal treatment by stating more precisely what fairness is in the field of electronic commerce.

The contributions of this work are conceptual and terminological ones. We have tried to show that the use of formal methods can help introduce the rigor necessary to precisely state the meaning of and differences between different concepts, such as the strong and weak fairness definitions of Asokan [6]. By doing this, we have presented three precisely separable notions of fairness, the main contribution being a formalization of the previously rather "fuzzy" term of *weak fairness*. Apart from the precision of the terminology, we have yet argued too little that these definitions are actually useful and correspond to concepts which are met in practice. Future work must further examine their usefulness. However, because of the way in which these results were derived (based on the notions of safety and liveness) we conjecture that these definitions are rather complete, meaning that they define classes that can be further subdivided but not fully refuted.

Future work must also study the effect of malicious behavior onto the attainability of a certain amount of fairness. For example, the definition of eventually strong fairness depends very much on the actions which a node is allowed to perform. Do certain fault models (such as crash, unreliable communication, Byzantine) generally make a certain form of fairness unachievable?

In a formal analysis of a non-repudiation protocol, Schneider [23] solves the problem of imposing state onto a misbehaving party by requiring a trusted intermediate to ensure some liveness property. Also, latest research in concurrency theory has developed the idea of having different fairness levels in different parts of the system [17]. Can we benefit from these ideas, because obviously different requirements are placed on the trusted third party than on the other participants of the fair exchange? What are the minimal requirements of such a trusted intermediate to achieve a certain degree of fairness?

Also, we are interested in the question of how specific item properties affect the level of fairness which a protocol can attain? These and other questions still deserve some further attention.

References

[1] *Longman Dictionary of Contemporary English*. Longman Group, Harlow, U.K., third edition, 1995.

[2] Advanced services, architecture and design. SEMPER Deliverable D10; La Gaude, Mar. 1999. Available at http://www.semper.org/deliver/d10/d10.ps.gz

[3] M. Abadi and L. Lamport. Composing specifications. *ACM Trans. on Progr. Lang. and Sys.*, 15(1):73–132, Jan. 1993.

[4] B. Alpern and F. B. Schneider. Defining liveness. *Inf. Proc. Letters*, 21:181–185, 1985.

[5] R. Alur and T. Henzinger. Finitary fairness. In *Proc. of the 9th Annual IEEE Symp. on Logic in Computer Science*, pages 52–61, Paris, France, July 1994. IEEE Comp. Soc. Press.

[6] N. Asokan. *Fairness in electronic commerce*. PhD thesis, University of Waterloo, May 1998.

[7] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In T. Matsumoto, editor, *4th ACM Conf. on Computer and Comm. Security*, pages 8–17, Zurich, Switzerland, Apr. 1997. ACM Press.

[8] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proc. of the IEEE Symp. on Res. in Sec. and Priv.*, pages 86–99, May 1998.

[9] H. Bürk and A. Pfitzmann. Value exchange systems enabling security and unobservability. *Computers & Security*, 9(8):715–721, 1990.

[10] E. W. Dijkstra. Guarded commands, nondeterminacy, and formal derivation of programs. *Comm. of the ACM*, 18(8):453–457, Aug. 1975.

[11] N. Francez. *Fairness*. Texts and Monographs in Computer Science. Springer-Verlag, 1986.

[12] M. Jakobsson. Ripping coins for fair exchange. In L. C. Guillou and J.-J. Quisquater, editors, *Adv. in Cryptology (EUROCRYPT) 95*, volume 921 of *Lect. Notes in Comp. Science*, pages 220–230. Springer-Verlag, 21–25 May 1995.

[13] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Trans. on Softw. Eng.*, 3(2):125–143, Mar. 1977.

[14] L. Lamport. A simple approach to specifying concurrent systems. *Comm. of the ACM*, 32(1):32–45, Jan. 1989.

[15] L. Lamport and N. Lynch. Distributed computing: models and methods. In *Handbook of Theor. Computer Science (Volume B: Formal Models and Semantics)*, chapter 18, pages 1157–1199. Elsevier, 1990. J. van Leeuwen, Editor.

[16] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[17] M. Merritt and G. Taubenfeld. Fairness of shared objects. In *Proc. of the 12th Int. Symp. on Distr. Comp. (DISC'98)*, number 1499 in Lect. Notes in Comp. Science, pages 303–316, Andros, Greece, Sept. 1998.

[18] H. Pagnia and F. C. Gärtner. On the impossibility of fair exchange without a trusted third party. Tech. Report TUD-BS-1999-02, Darmstadt Univ. of Technology, Dep. of Comp. Science, Darmstadt, Germany, Mar. 1999.

[19] H. Pagnia and R. Jansen. Towards multiple-payment schemes for digital money. In R. Hirschfeld, editor, *Financial Cryptography (FC '97)*, volume 1318 of *Lect. Notes in Comp. Science*, pages 203–215, Anguilla, British West Indies, 24–28 Feb. 1997. Springer-Verlag.

[20] H. Pagnia and H. Vogt. Exchanging goods and payment in electronic business transactions. In *Proc. of the Third Europ. Res. Sem. on Adv. in Distr. Syst. (ERSADS'99)*, Madeira Island, Portugal, Apr. 1999.

[21] T. W. Sandholm and V. R. Lesser. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. In C. S. Mellish, editor, *Proc. of the 14th Int. Joint Conf. on Artif. Int.*, pages 694–703, San Mateo, Aug. 20–25 1995. Morgan Kaufmann.

[22] F. B. Schneider. Enforceable security policies. Tech. Report TR98-1664, Cornell Univ., Dep. of Comp. Science, Ithaca, New York, Jan. 1998.

[23] S. Schneider. Formal analysis of a non-repudiation protocol. In *Proc. of the 11th IEEE Computer Security Foundations Workshop*, pages 54–65, 1998.

[24] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 1994.

[25] J. D. Tygar. Atomicity in electronic commerce. In *Proc. of the 15th Annual ACM Symp. on Princ. of Distr. Comp. (PODC'96)*, pages 8–26, New York, May 1996.

[26] U. G. Wilhelm. Cryptographically protected objects, May 1997. A french version appeared in the Proceedings of RenPar'9, Lausanne, Switzerland. http://lsewww.epfl.ch/~wilhelm/CryPO.html.