Piotr Dembiński, Wojciech Jamroga
Antoni Mazurkiewicz, Wojciech Penczek

Towards Partial Order Reductions for
Fragments of Alternating-Time Temporal
Logic

Nr 1036

Warsaw, December 2016
Abstract

A general semantics of strategic abilities of agents in asynchronous systems with and without perfect information is proposed, and some general complexity results for verification of strategic abilities in asynchronous systems are presented. A methodology for partial order reduction (POR) in verification of agents with imperfect information is developed, based on the notion of traces introduced by Mazurkiewicz. Two semantics of $\text{ATL}^*_X$ are considered and it is shown that for memoryless imperfect information ($|=_{ir}$) contrary to memoryless perfect information ($|=_{Ir}$), one can apply techniques known for $\text{LTL}_X$.

Keywords: Alternating-Time Temporal Logic, asynchronous systems, partial order reduction, traces

Streszczenie

O redukcjach częścio-porządkowych dla fragmentów logiki temporalnej czasu alternującego

Raport definiuje ogólną semantykę dla strategicznych umiejętności agentów w systemach asynchronicznych z pełną i niepełną informacją, oraz prezentuje ogólne wyniki dotyczące złożoności weryfikacji strategicznych umiejętności w systemach asynchronicznych. Metoda redukcji częścio-porządkowych, wykorzystująca ślady Mazurkiewicza, została zastosowana do weryfikacji agentów z niepełną informacją. Dla rozważanych semantyk logiki $\text{ATL}^*_X$ zostało pokazane, że dla bezpamięciowej niepełnej informacji ($|=_{ir}$) w przeciwieństwie do bezpamięciowej pełnej informacji ($|=_{Ir}$), można zastosować metody znane dla $\text{LTL}_X$.

Słowa kluczowe: Logika temporalna czasu alternującego, systemy asynchroniczne, redukcje częściowo-porządkowe, ślady
Contents

1 Introduction 4

2 Preliminaries 7
   2.1 Asynchronous Multi-Agent Systems 7
   2.2 Interleaved Interpreted Systems 8

3 Reasoning about Agents’ Abilities 9
   3.1 Alternating-Time Temporal Logic 10
   3.2 Strategies and Outcomes 11
   3.3 Asynchronous Semantics of ATL and ATL* 12

4 Model Checking sATL and sATL* for Asynchronous MAS 13
   4.1 Model checking 1ATL_{ir} 14
   4.2 Model checking sATL_{ir} 15
   4.3 Model checking sATL*_{ir} and 1ATL*_{ir} 16
   4.4 Discussion 17

5 Equivalences for LTL_{¬X} 18
   5.1 Stuttering (trace) equivalence 18
   5.2 Preserving LTL_{¬X} 19

6 Partial Order Reductions for sATL*_{ir} 19
   6.1 Traces 19
   6.2 Algorithms for Partial Order Reduction 22
   6.3 Preserving sATL*_{ir} 22

7 Conclusions and Future Work 24
1 Introduction

Multi-agent systems describe interactions of multiple entities called *agents*, often assumed to be intelligent and autonomous. *Alternating-time temporal logic* ATL* and its fragment ATL [3, 6] are logics which allow for reasoning about strategic interactions in such systems. The main idea is to extend the framework of temporal logic with the game-theoretic notion of *strategic ability*. Hence, ATL* enables to express statements about what agents (or groups of agents) can achieve. For example, $\langle \langle a \rangle \rangle F \text{win}_a$ says that agent $a$ has the ability to eventually win no matter what the other agents do, while $\langle \langle a, b \rangle \rangle G \text{safe}$ expresses that agents $a$ and $b$ together can force the system to always remain in a safe state. Such properties can be useful for specification, verification, and reasoning about interaction in agent systems. They have become especially relevant due to active development of algorithms and tools for verification where the “correctness” property is given in terms of strategic ability [4, 5, 36, 45, 14, 28, 11, 59, 50]. However, there are two caveats.

First, most of the tools and algorithmic solutions focus on agents with perfect information, i.e., agents who at any point of the game know exactly the global state of the game. This assumption is clearly unrealistic in all but the simplest multi-agent scenarios. Still, the tendency is somewhat easy to understand, since model checking of ATL variants with imperfect information has been proved $\Delta_2^P$- to PSPACE-complete for agents playing memoryless (a.k.a. positional) strategies [62, 33, 10] and EXPTIME-complete to undecidable for agents with perfect recall of the past [18, 26]. Moreover, the imperfect information semantics of ATL does not admit alternation-free fixpoint characterizations [8, 19, 20], which makes incremental synthesis of strategies impossible, or at least difficult to achieve. Some early attempts at verification of imperfect information strategies made their way into the MC-MAS model-checker [46, 60, 47, 50], but the issue has never been at the heart of the tool. More dedicated attempts have begun to emerge only recently [59, 11, 28, 12, 35]. Up until now, experimental results confirm that the initial intuition was right: model checking strategic modalities for imperfect information is hard, and dealing with it requires innovative algorithms and verification techniques.

Secondly, the semantics of strategic logics are almost exclusively based...
on synchronous concurrent game models. That is, one implicitly assumes the existence of a global clock that triggers subsequent global events in the system. At each tick of the clock, all the agents choose their actions, and the system proceeds accordingly with the corresponding global transition. However, many real-life systems are inherently asynchronous, and do not operate on a global clock that perfectly synchronizes the atomic steps of all the components. As an example, consider robots interacting in an environment with faulty communication and/or non-negligible delays in execution of actions. No less importantly, many systems whose implementation may be synchronous at the implementation level (say, the level of the virtual machine) can be conveniently modeled as asynchronous on a more abstract level, because when we abstract away the implementation details it is not clear anymore how transitions initiated by different agents are exactly arranged in a particular temporal order. For instance, the actual implementation of a soccer match in the simulated RoboCup competition can be executed on a single computer with a global clock ticking every 0.3 ns, but the corresponding synchronous model would be huge and in consequence useless for any kind of analysis. Instead, one can remove a lot of unnecessary details by assuming that the players execute their actions asynchronously – without clear temporal relationship between their execution times – and synchronize only when a particular event has to be executed jointly. In many scenarios, both aspects combine. For example, when modeling a national election, one must take into account both the truly asynchronous nature of events happening at different polling stations, and the best level of granularity for modeling the events happening within a single polling station.

In this paper, we make the first step towards strategic analysis of such systems. Our contribution is threefold:

1. 

2. We propose a general semantics of strategic abilities of agents in asynchronous systems, with and without perfect information.

3. We present some general complexity results for verification of strategic abilities in asynchronous systems.

4. We develop a methodology for partial order reduction (POR) in ver-
ification of agents with imperfect information, based on the notion of traces introduced by Mazurkiewicz in [53]. We also observe that, interestingly, the scheme does not work for verification of agents with perfect information.

*Partial order reduction* is one of the most widely known techniques in verification of reactive systems. Our main goal is to obtain a variant of POR that can be used for verification of strategic properties in multi-agent systems.

**Related work.** Related relevant work is relatively scarce. Alur, Henzinger and Kupferman mentioned asynchronous systems in their seminal paper on ATL [6], but they modeled them as a special case of synchronous systems. Reactive modules [2], the class of representations behind the Mocha model checker [4, 5], feature several modes of asynchronous execution, but – to the best of our knowledge – this aspect has never been given a more systematic analysis.

Asynchronous semantics and partial order reduction were extensively studied in [54, 55, 56, 38, 24, 23, 57, 22, 40, 58]. The most recent approaches include dynamic POR [21, 1, 13] and combine POR with symbolic methods [37, 39]. Still, the only efficient approach to partial order reduction in a MAS context [48, 49] presents results for standard epistemic-temporal logics (LTLK\(\neg X\), CTL*K\(\neg X\)) interpreted over interleaved interpreted systems. It is by no means immediately clear how those approaches extend to modeling and verification of strategic play from autonomous, rational, and purposeful players.

The work that comes closest to our new proposal is [17] where a variant of ATL was proposed for the special case of agent-oriented agent programs written in 2APL with asynchronous execution semantics. A very crude and rather impractical notion of stuttering equivalence was also proposed there, as the first step towards a partial order reduction scheme.

It should also be mentioned that our complexity results largely coincide with the pattern already known for model checking of synchronous systems, cf. e.g. [10].
2 Preliminaries

We begin with defining asynchronous multi-agent systems and their semantics in terms of interleaved interpreted systems.

2.1 Asynchronous Multi-Agent Systems

We will model multi-agent systems as networks of automata that execute asynchronously by interleaving of local transitions, and synchronize their moves whenever a shared action is executed. Formally, consider a MAS composed of \( n \) agents \( A = \{1, \ldots, n\} \). Each agent \( i \in A \) is associated with a set of possible local states \( L_i = \{l_{i1}^1, l_{i2}^2, \ldots, l_{in_i}^{nl_i}\} \) and a set of actions \( \text{Act}_i = \{\epsilon, a_{i1}^1, a_{i2}^2, \ldots, a_{ina_i}^{nai}\} \). The special action \( \epsilon \), called the “null” action of agent \( i \), does not change the local state of \( i \), for each \( i \in A \). Notice that the sets of actions of the agents do not need to be disjoint also for actions different than \( \epsilon \). \( \text{Act} = \bigcup_{i \in A} \text{Act}_i \) is the union of all the sets \( \text{Act}_i \), whereas \( \text{Loc} = \bigcup_{i \in A} L_i \) is the union of all the sets \( L_i \). For each action \( a \in \text{Act} \) the set \( \text{Agent}(a) \subseteq A \) contains these agents \( i \) for which \( a \in \text{Act}_i \), i.e., these which can potentially execute \( a \).

We define the independency \( I \) on \( \text{Act} \) as follows: \( I = \{(a, b) \in \text{Act} \times \text{Act} | \text{Agent}(a) \cap \text{Agent}(b) = \emptyset\} \). Notice that \( \epsilon \) is dependent on all the other actions of \( \text{Act} \).

Following the interpreted system model, for each agent, a local protocol is defined, which models the program the agent is executing. Formally, a local protocol \( P_i : L_i \rightarrow 2^{\text{Act}_i} \) for each agent \( i \), selects actions which can be executed at each local state. For each agent \( i \), we define an evolution (partial) function \( t_i : L_i \times A_i \rightarrow L_i \), where \( t_i(l_i, \epsilon) = l_i \) if \( \epsilon \in P_i(l_i) \), for each \( l_i \in L_i \).

A global state \( q = (l_1, \ldots, l_n) \in L_1 \times \ldots \times L_n \) is a tuple of local states for all the agents in the MAS. By \( q^i = l_i \) we mean the local component of agent \( i \in A \) in \( q \). Now the global transitions are defined.

\textbf{Definition 1 (Interleaved semantics).} Let \( St = L_1 \times \ldots \times L_n \) be a set of global states. The global interleaved evolution function \( T : St \times \text{Act} \rightarrow St \) is defined

\footnote{Note that we do not consider the environment component, which may be added with no technical difficulty.}
as follows: \( T(q, a) = q_1 \) iff \( t_i(q_i, a) = q_i^1 \) for all \( i \in \text{Agent}(a) \), and \( q_i^1 = q_1^1 \) for all \( i \in \mathcal{A} \setminus \text{Agent}(a) \). The above is denoted as \( q \stackrel{a}{\longrightarrow} q_1 \).

Notice that \( q \stackrel{\epsilon}{\longrightarrow} q \) if \( \epsilon \in P_i(q_i) \) for each \( i \in \mathcal{A} \). The global transition relation is assumed to be total, i.e., for each \( q \in St \) there exists an \( a \in \text{Act} \) such that \( q \stackrel{a}{\longrightarrow} q_1 \), for some \( q_1 \in St \). An infinite sequence of global states and actions \( \pi = q_0 a_0 q_1 a_1 q_2 \ldots \) is called an (interleaved) path, starting at \( q_0 \) if there is a sequence of global transitions from \( q_0 \) onwards, i.e., if \( q_i \stackrel{a_i}{\longrightarrow} q_{i+1} \) for every \( i \geq 0 \). By \( \text{Act}(\pi) = a_0 a_1 a_2 \ldots \) we denote the sequence of actions of \( \pi \), while by \( \Pi(q) \) - the set of all interleaved paths starting at \( q \).

2.2 Interleaved Interpreted Systems

In order to define a semantics of ATL* we need a notion of model.

**Definition 2** (Interleaved Interpreted System). Let \( PV \) be a set of propositional variables. An interleaved interpreted system (IIS) or a model, is a 4-tuple \( M = (St, \iota, T, V) \), where \( St \) is a set of global states, \( \iota \in St \) is an initial (global) state, \( T \) is the global transition relation, and \( V : St \rightarrow 2^{PV} \) is a valuation function.

Figure 1 presents the three agents of the interleaved interpreted system (the untimed version of the original Train-Gate-Controller (TGC) [4, 27]): a controller and two trains. Each train runs on a circular track and both tracks pass through a narrow tunnel (state “T”), allowing one train only to go through it (to state “A” - (Away) at any time. The controller operates the signal (Green (“G”) and Red (“R”)) to let trains enter and leave the tunnel. In the figure, the initial states of the controller and the train are “G” and “W” (Waiting) respectively, and the transitions with the same label are synchronised. Silent \( \epsilon \) actions are omitted in the figure.

**Definition 3** (Reduced Model). Given two IIS (models) \( M = (St, \iota, T, V) \) and \( M' = (St', \iota', T', V') \). If \( St' \subseteq St \), \( \iota' = \iota \), \( T \) is an extension of \( T' \), and \( V' = V | St' \), then we write \( M' \subseteq M \) and call \( M' \) a submodel of \( M \), and \( M' \) - a reduced model of \( M \).

It is easy to show that for each \( q \in St' \) we have \( \Pi'(q) \subseteq \Pi(q) \), where \( \Pi(q) \) (\( \Pi'(q) \)), is the set of all interleaved paths of \( M \) (\( M' \), resp.) starting at \( q \).
In order to generate reduced models we will need a notion of invisibility of actions with respect a subset $PV'$ of $PV$.

**Definition 4.** Given a model $M = (St, \iota, T, V)$. An action $a \in Act$ is invisible for $PV'$ if for each two global states $q, q' \in St$: if $q \xrightarrow{a} q'$, then $V(q) \cap PV' = V(q') \cap PV'$.

The set of all actions invisible for $PV'$ is denoted by $Invis_{PV'}$ and its closure by $Vis_{PV'} = Act \setminus Invis_{PV'}$.

Intuitively, invisible actions do not change valuations of the propositions of $PV'$ in $M$.

## 3 Reasoning about Agents’ Abilities

Many important properties in a MAS can be specified in reference to the existence strategic ability (or inability) of some agents to achieve a given goal. Examples include the ability of a voter to cast her vote according to her intent in an election, the inability of a coercer to disturb the outcome of the election by coercion and/or vote buying, the ability of two parties to successfully communicate or sign a contract, and the existence of a suitable collective strategy for trains in a railway network so that neither a crash nor a deadlock can occur. Such properties can be specified by formulae of alternating-time logic ($ATL$). The semantics of $ATL$ is typically defined for models of synchronous systems. In this section, we show how the semantics can be redefined for interleaved interpreted systems.
3.1 Alternating-Time Temporal Logic

Alternating-time temporal logic [3, 6] generalizes the branching-time temporal logic CTL [15] by replacing path quantifiers $\mathcal{E}, \mathcal{A}$ with strategic modalities $\langle \langle A \rangle \rangle$. Informally, $\langle \langle A \rangle \rangle \gamma$ expresses that the group of agents $A$ has a collective strategy to enforce temporal property $\gamma$. Formulas of ATL make use of temporal operators: “$X$” (“in the next state”), “$G$” (“always from now on”), “$F$” (“now or sometime in the future”), $U$ (“strong until”), and $R$ (“release”). The logic comes in several syntactic variants, the most popular of which are ATL* and “vanilla ATL” (the latter often called simply “ATL”).

**Definition 5** (Syntax of ATL*). Let $PV$ be a set of propositional variables and $A$ the set of all agents. The language of ATL* is defined by the following grammar:

$$
\varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \langle \langle A \rangle \rangle \gamma,
$$

$$
\gamma ::= \varphi \mid \neg \gamma \mid \gamma \land \gamma \mid X \gamma \mid \gamma U \gamma.
$$

where $p \in PV$ and $A \subseteq A$.

Disjunction and Boolean constants are defined as usual. The “release” operator can be defined as $\gamma_1 R \gamma_2 \equiv \neg((\neg \gamma_1) U (\neg \gamma_2))$. The “sometime” and “always” operators can be defined as $F \gamma \equiv \top U \gamma$ and $G \gamma \equiv \bot R \gamma$.

**Definition 6** (Syntax of ATL). In “vanilla ATL,” every occurrence of a strategic modality is immediately followed by a temporal operator. Note that, in that case, “release” is not definable from “until” anymore [44], and it must be added to the syntax as another primitive operator. Formally, the language of ATL is defined by the following grammar:

$$
\varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \langle \langle A \rangle \rangle X \varphi \mid \langle \langle A \rangle \rangle G \varphi \mid \langle \langle A \rangle \rangle \varphi U \varphi.
$$

In the rest of the paper, we will be mainly interested in formulae that do not use the next step operator $X$, and do not contain nested strategic modalities. We denote the corresponding subsets of ATL* and ATL by sATL* (“simple ATL*”) and sATL (“simple ATL”), respectively. Moreover, 1ATL* denotes the fragment of sATL* that admits only formulae consisting of a strategic modality, followed by an LTL formula (i.e., $\langle \langle A \rangle \rangle \gamma$, where $\gamma \in LTL$).
3.2 Strategies and Outcomes

Let $M = (St, i, T, V)$ be an IIS and $i \in A$. A strategy of agent $i$ is a conditional plan that specifies what $i$ is going to do in any potential situation. A number of semantic variations are possible. Here, we follow Schobbens [62], and adopt his taxonomy of four “canonical” strategy types: IR, iR, Ir, and ir. In the notation, $R$ (resp. $r$) stands for perfect (resp. imperfect) recall, and $I$ (resp. $i$) refers to perfect (resp. imperfect) information.

In this paper, we consider strategies of types Ir and ir. Formally:

- A memoryless perfect information strategy for agent $i$ is a function $\sigma_i : St \rightarrow A_i$ such that $\sigma_i(q) \in P_i(q^i)$ for each state $q \in St$. We denote the set of such strategies by $\Sigma_{Ir}$.

- A memoryless imperfect information strategy for agent $i$ is a function $\sigma_i : St \rightarrow A_i$ such that $\sigma_i(q) \in P_i(q^i)$ for each state $q \in St$, and for each $q, q_1 \in St$ if $q^i = q_1^i$, then $\sigma_i(q) = \sigma_i(q_1)$. Equivalently, the strategy can be defined as $\sigma_i : L_i \rightarrow A_i$ such that $\sigma_i(l) \in P_i(l)$. We denote the set of such strategies by $\Sigma_{ir}$.

Thus, an Ir strategy can assign different actions of agent $i$ to any two global states, while in an ir strategy the agent’s choices can only depend on the local state (i.e., the location) of the agent.

A joint strategy $\sigma_A$ for a coalition $A \subseteq A$ is a tuple of strategies, one per agent $i \in A$. Let $A = \{1, \ldots, k\}$ for some $k \in N$ and $\sigma_A = (\sigma_1, \ldots, \sigma_k)$ be a joint strategy for $A$. For each $i \in N$ and $q \in St$ we denote $\sigma_A(q) = (\sigma_1(q), \ldots, \sigma_k(q))$. By $Act(\sigma_A)$ we denote the set of actions assigned to the states by the strategies of $\sigma_A$. We will consider two types of joint strategies: Ir-joint strategies consisting of memoryless perfect information strategies and ir-joint strategies consisting of memoryless imperfect information strategies.

Definition 7 (Outcome). The outcome of a joint strategy $\sigma_A$ in a state $q \in St$ is the set $\text{out}_M(q, \sigma_A) \subseteq St^\omega$ such that $\pi \in \text{out}_M(q, \sigma_A)$ iff $\pi[0] = q$ and for each $i \in N$:

- $\pi[i] \xrightarrow{a_i} \pi[i + 1]$ for some $a_i \in Act$,

- for each $j \in A$ if $j \in \text{Agent}(a_i)$, then $a_i \in \sigma_j(\pi[i])$,
Intuitively, the outcome of a joint strategy $\sigma_A$ in a state $q$ is the set of all the possible paths that can occur when the agents of coalition $A$ execute the strategy $\sigma_A$ and each other agent, in $\overline{A}$, follows its own protocol.

3.3 Asynchronous Semantics of ATL and ATL*

The semantics of $\text{ATL}^*$, parameterised with the strategy type $Y$, is defined below.

**Definition 8 (Semantics of $\text{ATL}^*$).** Let $M = (St, i, T, V)$ be an IIS, $q \in St$, $A \subseteq A$, and $Y \in \{\text{Ir}, \text{ir}\}$. The $Y$-semantics of $\text{ATL}^*$ and ATL is given by the following clauses:

$M, q |=_Y p$ iff $p \in V(q)$;

$M, q |=_Y \neg \varphi$ iff $M, q \not|=_Y \varphi$;

$M, q |=_Y \varphi_1 \land \varphi_2$ iff $M, q |=_Y \varphi_1$ and $M, q |=_Y \varphi_2$;

$M, q |=_Y \langle\langle A \rangle\rangle \gamma$ iff there is a joint $Y$-strategy $\sigma_A$ for agents $A$ such that, for each path $\pi \in \text{out}_M(q, \sigma_A)$, we have $M, \pi |=_Y \gamma$;

$M, \pi |=_Y \varphi$ iff $M, \pi[0] |=_Y \varphi$;

$M, \pi |=_Y \neg \gamma$ iff $M, \pi \not|=_Y \gamma$;

$M, \pi |=_Y \gamma_1 \land \gamma_2$ iff $M, \pi |=_Y \gamma_1$ and $M, \pi |=_Y \gamma_2$;

$M, \pi |=_Y X \gamma$ iff $M, \pi[1, \infty] |=_Y \gamma$;

$M, \pi |=_Y \gamma_1 U \gamma_2$ iff there is $i \geq 0$ such that $M, \pi[i, \infty] |=_Y \gamma_2$ and $M, \pi[j, \infty] |=_Y \gamma_1$ for all $0 \leq j < i$.

The semantics of “vanilla ATL” can be given entirely with respect to states.

**Definition 9 (State-based semantics of ATL).** The $Y$-semantics of ATL can be equivalently defined by the following clauses:

$M, q |=_Y p$ iff $p \in V(q)$;

$M, q |=_Y \neg \varphi$ iff $M, q \not|=_Y \varphi$;

$M, q |=_Y \varphi_1 \land \varphi_2$ iff $M, q |=_Y \varphi_1$ and $M, q |=_Y \varphi_2$;
\[ M, q \models_Y \langle A \rangle X \varphi \] iff there is a \( Y \)-strategy \( \sigma_A \) such that, for each \( \pi \in \text{out}_M(q, \sigma_A) \), we have \( M, \pi[1] \models_Y \varphi \);

\[ M, q \models_Y \langle A \rangle \varphi_1 \lor \varphi_2 \] iff there is a \( Y \)-strategy \( \sigma_A \) such that, for each \( \pi \in \text{out}_M(q, \sigma_A) \), there exists \( i \geq 0 \) with \( M, \pi[i] \models_Y \varphi_2 \) and \( M, \pi[j] \models_Y \varphi_1 \) for all \( 0 \leq j < i \);

\[ M, q \models_Y \langle A \rangle \varphi_1 \lor \varphi_2 \] iff there is a \( Y \)-strategy \( \sigma_A \) such that, for all \( \pi \in \text{out}_M(q, \sigma_A) \) and \( i \geq 0 \), either \( M, \pi[i] \models_Y \varphi_2 \) or \( M, \pi[j] \models_Y \varphi_1 \) for some \( 0 \leq j < i \).

For a syntax \( \mathcal{L} \) and the semantic relation \( \models_Y \), we will denote the logical system \( (\mathcal{L}, \models_Y) \) by \( \mathcal{L}_Y \). Thus, \( \text{ATL}_{ir} \) is the “vanilla ATL” with memoryless perfect information semantics, \( \text{sATL}^*_{ir} \) is the “simple ATL**” with memoryless imperfect information strategies, and so on.

**Remark 1.** We observe that the relation \( \models_{it} \) captures the “objective” notion of ability under imperfect information [32, 9]. That is, \( \langle A \rangle \gamma \) holds iff agents in \( A \) have a collective strategy to enforce \( \gamma \) from the current global state of the system. We expect to obtain analogous results for the semantic variant based on “subjective” ability [62, 34, 9], but a detailed study of the latter case is outside the scope of this report.

### 4 Model Checking sATL and sATL* for Asynchronous MAS

In this work, we focus on simple specifications of strategic ability, i.e., ones that can be written formally without nesting strategic modalities. We believe that an overwhelming majority of properties, relevant in actual application domains, follow that pattern. We usually want to require (or ask if) a given player has a strategy to eventually win the game, two trains can persistently avoid the crash, Alice and Bob can exchange a secret without Cathy learning it on the way, etc. The three example properties can be tentatively specified by the following formulae:

1. \( \langle i \rangle \text{Fwin}_i \),
2. \( \langle t_1, t_2 \rangle \text{G}\neg\text{crash} \),
3. $\langle a, b \rangle (F(\text{knowsSecr}_a \land \text{knowsSecr}_b) \land G\neg \text{knowsSecr}_c)$.

Note that (1) and (2) are formulae of $\text{sATL}$ (in fact, $\text{1ATL}$), while (3) is a formula of $\text{sATL}^\ast$. Also, specification (3) suggests that many interesting properties can be more conveniently specified with a combination of strategic and epistemic modalities, which seems an interesting path for future work.

Note also that, in all realistic scenarios, players have only partial knowledge of the current global state of the world during the interaction. Thus, a semantics with imperfect information strategies must be used. Since verification of $\text{ATL}$ with imperfect information and perfect recall is undecidable [18], we focus on the memoryless imperfect information semantics $|=_{\text{ir}}$.

In this section, we establish the complexity of model checking for some relevant fragments of $\text{sATL}_{\text{ir}}^\ast$. We observe that the complexity can be given with respect to the logical model of the system (i.e., an interleaved interpreted system, cf. Section 2.2), or the usual representation of the system (in our case, an asynchronous automata network, cf. Section 2.1). We give both kinds of results here. We also briefly look at the program complexity of model checking, i.e., the complexity of the problem when the input formulae are of fixed or bounded length.

4.1 Model checking $\text{1ATL}_{\text{ir}}$

We begin by looking at the verification complexity for simplest specifications, consisting of a single strategic modality $\langle A \rangle$ immediately followed by a single temporal modality.

**Proposition 2.** Model checking $\text{1ATL}_{\text{ir}}$ is $\text{NP}$-complete in the size of the model and the length of the formula. It remains $\text{NP}$-complete even for formulae of bounded length.

**Proof.** (sketch) Analogous to the result in [62] for $\langle \Gamma \rangle$-$\text{ATL}_{\text{ir}}$.

For the upper bound, observe that model checking of $\langle A \rangle \gamma$ in $M, q$ can be done by (1) guessing a uniform strategy $s_A$, (2) pruning $M$ according to $s_A$, and (3) model checking the $\text{CTL}$ formula $A \gamma$ (“for all paths, $\gamma$”) in state $q$ of the resulting model. Since $s_A$ is of at most linear size with respect to $|M|$, and model checking of $A \gamma$ can be done in deterministic polynomial time w.r.t. $|M|$, we obtain the bound.
For the lower bound, we use the reduction from [62] of the Boolean satisfiability problem (SAT) to model checking formula \( \langle\langle 1 \rangle\rangle F_{\text{yes}} \) in a single-agent model (note that single-agent systems can be seen as special cases of both synchronous and asynchronous systems). Note that the lower bound does not rely on the length of the formula, as formulae of length 3 are sufficient to construct the reduction.

**Proposition 3.** Model checking 1ATL_{ir} is PSPACE-complete in the size of the representation (even for formulae of bounded length).

**Proof.** (sketch) For the upper bound, observe that model checking of formula \( \langle\langle A \rangle\rangle \gamma \) in representation (automata network) \( R \) can be done by: (1) guessing a uniform \( s_A \) as a deterministic restriction of the protocols \( P_i, i \in A \), (2) pruning \( M \), and (3) model checking the LTL formula \( \gamma \) in the resulting representation \( R' \). Since the size of \( s_A \) is linear with respect to \( |R| \), and model checking LTL is in PSPACE with respect to \( |R| \) (cf. [61]), we obtain the bound.

To prove the lower bound, we adapt the construction from [41, Theorem 6.1]. Given a Turing machine \( T \) with space complexity \( s(n) \), we construct the concurrent program \( P(T) \) as in [41, Theorem 6.1]. To obtain an asynchronous MAS \( P' \) with a similar behavior, we sequentialize the concurrent actions of modules in \( P(T) \) by adding an extra “synchronizer” module which enforces that each agent \( i \in \{1, \ldots, n\} \) takes the \( i \)th step in each “execution cycle.” That is, every concurrent transition in \( P(T) \) is decomposed into a sequence of \( n \) asynchronous transitions in \( P' \). Now, there exists a computation of \( T \) on the empty tape which eventually reaches an accepting state iff \( P(T) \models_{\text{CTL}} EF_{\text{accept}} \) iff \( P' \not\models_{\text{ir}} \langle\langle \emptyset \rangle\rangle G \neg_{\text{accept}} \). This way we obtain the co-PSPACE-hardness for 1ATL_{ir}. Since co-PSPACE = PSPACE, the lower bound follows.

Again, we note that the reduction does not rely on the length of the formula.

**4.2 Model checking sATL_{ir}**

The verification complexity for Boolean combinations of formulae from 1ATL is almost the same.
Proposition 4. Model checking $\text{sATL}_{\text{ir}}$ is NP-hard and in $\Theta_2^P$ in the size of the model and the length of the formula (even for formulae of bounded length).\(^2\)

Proof. (sketch) The lower bound follows from Proposition 2.

The following algorithm for checking $\varphi$ in $M, q$ demonstrates the upper bound. First, the nondeterministic algorithm in Proposition 2 is used as an oracle that determines the truth value for each subformula $\langle \langle A \rangle \rangle^\gamma$ of $\varphi$. It is easy to see that the oracle is called at most $|\varphi|$ times, and the input in the next call does not depend on the output of the preceding calls. Finally, based on the output of the calls, the value of $\varphi$ is calculated according to the truth tables of Boolean operators.

\[
\Box
\]

Proposition 5. Model checking $\text{sATL}_{\text{ir}}$ is PSPACE-complete in the size of the representation and the length of the formula (even for formulae of bounded length).

Proof. (sketch) The lower bound follows from Proposition 3. For the upper bound, we use the algorithm from Proposition 4, but with the algorithm from Proposition 3 as the oracle. Since $P^{\text{PSPACE}} = \text{PSPACE}$, we obtain the result.

\[
\Box
\]

4.3 Model checking $\text{sATL}^*_{\text{ir}}$ and $\text{1ATL}^*_{\text{ir}}$

Finally, we examine the complexity of verification for specifications with arbitrary LTL subformulæ.

Proposition 6.

1. Model checking $\text{sATL}^*_{\text{ir}}$ and $\text{1ATL}^*_{\text{ir}}$ is PSPACE-complete in the size of the model and the length of the formula.

2. For formulae of bounded length, the problem is NP-complete for $\text{1ATL}^*_{\text{ir}}$ and between NP and $\Theta_2^P$-complete for $\text{sATL}^*_{\text{ir}}$.

\(^2\)Where $\Theta_2^P = P^{\text{||NP}}$ is the class of problems solvable by a deterministic polynomial-time Turing machine that can make polynomially many nonadaptive calls to an NP oracle.
Proof. (sketch) For (1), the lower bound follows from the \( \text{PSPACE} \)-completeness of \( \text{LTL} \) model checking [61]. The upper bound can be obtained by polynomially many calls to an \( \text{LTL} \) model checking algorithm, one per strategic subformula in \( \varphi \) (recall that \( \text{P}^{\text{PSPACE}} = \text{PSPACE} \)), and computing the Boolean combination.

For (2), the lower bound follows from Proposition 2. The inclusion in \( \text{NP} \) for \( \text{1ATL}^\ast_{ir} \) can be obtained by an algorithm similar to that in Proposition 2, only an \( \text{LTL} \) rather than \( \text{CTL} \) model checker is called. Since \( \text{LTL} \) model checking is \( \text{NLOGSPACE} \)-complete for formulae of bounded size [61], and \( \text{NLOGSPACE} \subseteq \text{P} \), the upper bound follows.

The upper bound for \( \text{sATL}^\ast_{ir} \) is obtained by an algorithm similar to that in Proposition 4, only an \( \text{LTL} \) rather than \( \text{CTL} \) model checker is called inside the oracle.

\[ \Box \]

Proposition 7. Model checking \( \text{sATL}^\ast_{ir} \) and \( \text{1ATL}^\ast_{ir} \) is \( \text{PSPACE} \)-complete in the size of the representation and the length of the formula (even for formulae of bounded length).

Proof. (sketch) The lower bounds follow from Proposition 3. The upper bounds can be obtained by the same algorithm as for Proposition 5.

\[ \Box \]

4.4 Discussion

Models (interleaved interpreted systems) are usually exponentially larger than the automata network representations from which they arise. Thus, for practical verification it is essential to provide the input to the model checker using the latter rather than the former. Unfortunately, the above complexity results strongly suggest that model checking fragments of \( \text{sATL}^\ast_{ir} \) on automata networks is hard, and the size of the representation is the main factor for this hardness. Thus, it is essential to use as small a representation as possible. If the input is given beforehand (e.g., prepared by the user), then any reduction of the representation increases the likelihood that the verification task becomes feasible.

In the next sections, we recall the idea of partial order reduction, very important for verification of asynchronous systems, and show how it can be used to model-check \( \text{sATL}^\ast_{ir} \) and its fragments.
5 Equivalences for LTL\(_{-X}\)

Partial order reductions have been defined for temporal logics without the next step operator \(X\) since it is known how to generate reduced models preserving either stuttering trace equivalence (for LTL\(_{-X}\)) or stuttering bisimulation (for CTL\(_{*-X}\)). Since stuttering trace equivalence is less discriminative than stuttering bisimulation, partial order reductions preserving LTL\(_{-X}\) are more efficient than these for CTL\(_{*-X}\). Since ATL\(_{*-X}\) has a more distinguishing power than CTL\(_{*-X}\), one can expect that the equivalence preserving ATL\(_{*-X}\) would be very discriminative, which would likely result in inefficient reductions. Therefore, aware of the above and motivated by practical applications, in this paper we take another route. We are investigating subsets of ATL\(_{*-X}\) for which known partial order reduction methods apply [54, 58, 22, 48, 49].

5.1 Stuttering (trace) equivalence

So, we start with definitions of stuttering equivalence and stuttering trace equivalence.

**Definition 10** (Stuttering Equivalence). A path \(\pi\) in \(M\) and a path \(\pi'\) in \(M'\) are called stuttering equivalent, denoted \(\pi \equiv_s \pi'\), if there exists a partition \(B_1, B_2 \ldots\) of the states of \(\pi\), and a partition \(B'_1, B'_2 \ldots\) of the states of \(\pi'\) such that for each \(j \geq 0\) we have that \(B_j\) and \(B'_j\) are nonempty and finite, and for every state \(q\) in \(B_j\) and every state \(q'\) in \(B'_j\) we have \(V(q) = V'(q')\).

Notice that in the above definition in each block \(B\) all the states share the same valuation.

**Definition 11** (Stuttering Trace Equivalence). Two states \(q\) in \(M\) and \(q'\) in \(M'\) are said to be stuttering trace equivalent, denoted \(q \equiv_s q'\), if

1. for each path \(\pi\) in \(M\) starting at \(q\), there is a path \(\pi'\) in \(M'\) starting at \(q'\) such that \(\pi \equiv_s \pi'\);

2. for each path \(\pi'\) in \(M'\) starting at \(q'\), there is a path \(\pi\) in \(M\) starting at \(q\) such that \(\pi' \equiv_s \pi\).

Two models \(M\) and \(M'\) are stuttering trace equivalent, denoted \(M \equiv_s M'\), if \(\iota \equiv_s \iota'\).
5.2 Preserving LTL−X

The following theorem connects LTL−X with stuttering trace equivalence:

**Theorem 8.** Let M and M′ be two stuttering trace equivalent models, where M′ ⊆ M. Then, M,ι |= φ iff M′,ι′ |= φ, for any LTL−X formula φ over PV.

**Proof.** See [25]. □

6 Partial Order Reductions for sATL∗ir

In what follows we propose how to obtain partial order reduction for sATL∗ir and its fragments, with very promising results. Interestingly, it turns out that our approach does not apply to sATL∗ir; we show it the end of this section. This suggests that ATL with imperfect information, besides conceptual advantage, can also offer some technical advantage over ATL with perfect information.

6.1 Traces

Partial order reductions are based on Mazurkiewicz traces as introduced in [51] and used in e.g. [52, 53]. Consider two words w, w′ ∈ Act*. We say that w I w′ iff w = w1abw2 and w′ = w1baw2, for some w1, w2 ∈ Act* and (a, b) ∈ I. Let ≡I be the reflexive and transitive closure of I. By (finite) traces we mean equivalence classes of ≡I, denoted by [w]≡I. Formally, a trace [w]≡I = {w′ ∈ Act* | w′ ≡I w}.

To define infinite traces we need more definitions. For v, v′ ∈ Actω the relation ≤I is defined as follows: v ≤I v′ iff ∀u ∈ Pref(v) ∃w ∈ Pref(v′) ∃z ∈ Act* (w ≡I z ∧ u ∈ Pref(z)). That means, each finite prefix of v is a prefix of a permutation (under commuting adjacent independent actions) of some prefix of v′.

Infinite traces are defined as equivalence classes of the following relation ≡ω, where v ≡ω v′ iff v ≤I v′ and v′ ≤I v, denoted by [v]≡ω, where v ∈ Actω.
Lemma 9. Let $M$ be an IIS and $PV' \subseteq PV$. Consider two paths $\pi, \pi' \in St_\omega$ starting at the same state such that $\text{Act}(\pi) = w$ and $\text{Act}(\pi') = w'$, and $w \equiv_\pi w'$. Assume that: **) for any two actions $a, b$ occurring in $w$ or $w'$ if $(a, b) \in I$, then either $a \in \text{Invis}_{PV'}$ or $b \in \text{Invis}_{PV'}$. Then, $\pi$ and $\pi'$ are stuttering equivalent wrt. $PV'$.

Proof. See [54].

We can always make $I$ to satisfy **) by restricting it to $I \setminus (\text{Vis}_{PV'} \times \text{Vis}_{PV'})$. Since stuttering equivalence preserves $\text{LTL}_X$, the above lemma implies that the paths over representatives of the same infinite trace cannot be distinguished by any $\text{LTL}_X$ formula using propositions of $PV'$. This fact is used for defining partial order reductions for $\text{LTL}_X$. Rather than generating the IIS (model) $M$ for a MAS, one can generate a reduced model $M'$ satisfying the following property:

(*) for each $\pi \in \Pi(\iota)$, there is $\pi' \in \Pi'(\iota)$ such that $\text{Act}(\pi) \equiv_\iota^\omega \text{Act}(\pi')$.

The reduced model $M'$ preserves the $\text{LTL}_X$ formulas over $PV'$.

Our aim is to show that $M'$ preserves also the $\text{sATL}_{ir}^*$ formulas over $PV'$. To this aim we show that each set $\text{out}_M(q, \sigma_A)$ is trace complete in the sense that with each path $\pi$ s.t. $\text{Act}(\pi) = w$, it contains a path over any $w' \in [w]_{\equiv_\iota^\omega}$.

Lemma 10. Let $\pi \in \text{out}_M(\iota, \sigma_A)$ and $\text{Act}(\pi) = w$. Then, for each $w' \in [w]_{\equiv_\iota^\omega}$ there is $\pi' \in \text{out}_M(\iota, \sigma_A)$ such that $\text{Act}(\pi') = w'$.

Proof. Consider a MAS in which the protocol of each agent $i \in A$ is equal to $\sigma_i$. Then, the set of paths $\Pi(\iota)$ of IIS for MAS is trace complete. But, this set of paths is equal to $\text{out}_M(\iota, \sigma_A)$, which ends the proof.

The above lemma suggests a method of partial order reduction for the formulas of $\text{sATL}_{ir}^*$.

Lemma 11. Let $M$ be a model, $PV' \subseteq PV$, and $M'$ be a reduced model satisfying the property *). Then, for each strategy $\sigma_A$, for each $\pi \in \text{out}_M(\iota, \sigma_A)$ there is $\pi' \in \text{out}_{M'}(\iota, \sigma_A)$ such that $\text{Act}(\pi) \equiv_\iota^\omega \text{Act}(\pi')$.

Proof. Follows easily from the property *) of $M$ and Lemma 10.
Theorem 12. Let $M$ be a model, $PV' \subseteq PV$, and $M'$ be a reduced model satisfying the property *). For each $\text{sATL}_{ir}^*$ formula $\varphi$ over $PV'$ we have:

\[ M, \iota \models \varphi \iff M', \iota' \models \varphi \]

Proof. The proof is by induction on the complexity of a formula. Consider $\varphi = \langle \langle A \rangle \rangle \gamma$.

$M, \iota \models \langle \langle A \rangle \rangle \gamma$ iff there is a joint strategy $\sigma_A$ such that for each $\pi \in \text{out}_M(\iota, \sigma_A)$ we have $M, \pi \models \gamma$ iff (by Lemmas 11 and 9)

there is a joint strategy $\sigma_A$ such that for each $\pi \in \text{out}_{M'}(\iota, \sigma_A)$ we have $M', \pi \models \gamma$ iff $M', \iota \models \langle \langle A \rangle \rangle \gamma$.

The cases of negation and conjunction are straightforward. \qed

The next example shows that Lemma 10 does not hold for memoryless perfect information strategies.

Example 1. Consider the MAS composed of two agents $\{1, 2\}$ such that

- $L_1 = \{l_1^1, l_1^2\}$, $L_2 = \{l_2^1, l_2^2\}$,
- $\text{Act}_1 = \{\epsilon, a\}$, $\text{Act}_2 = \{\epsilon, b\}$,
- $P_1(l_1^1) = \{a, \epsilon\}$, $P_1(l_1^2) = \{\epsilon\}$,
- $P_2(l_2^1) = \{b\}$, $P_2(l_2^2) = \{\epsilon\}$,
- $t_1(l_1^1, a) = l_1^2$, $t_2(l_2^1, b) = l_2^2$.

Define an Ir-strategy for each agent:

1: $\sigma_1(l_1^1, l_1^2) = a$, $\sigma_1(l_1^1, l_2^2) = \sigma_1(l_1^2, l_2^2) = \epsilon$,

2: $\sigma_2(l_1^1, l_2^1) = \sigma_2(l_1^2, l_1^1) = b$, $\sigma_2(l_1^2, l_2^2) = \epsilon$.

It is easy to see that $\text{out}((l_1^1, l_2^2), \sigma_{\{1,2\}})$ is not trace complete. Notice that $(a, b) \in I$, but while $\text{out}((l_1^1, l_2^2), \sigma_{\{1,2\}})$ contains the path over $ab(\epsilon)\omega$, it does not contain any path over $ba(\epsilon)\omega$. 
6.2 Algorithms for Partial Order Reduction

As mentioned above, the idea of verification by model checking with partial order reductions is to define an algorithm reducing the size of models while preserving satisfaction for a class of formulas. This requires a notion of equivalence between models. For the case of $s\text{ATL}_{ir}^*$, we know that the notion of stuttering trace equivalence presented above suffices. Traditionally, in partial order reduction the exploration is carried out either by depth-first-search (DFS) (see [22]), or double-depth-first-search (DDFS) [16].

In this context DFS is used to search states and transitions that will make up the reduced model by exploring systematically the possible computation tree and selecting only some of the possible states and transitions generated. In the following, a stack represents the path $\pi = g_0a_0g_1a_1 \cdots g_n$ currently being visited. For the top element of the stack $g_n$ the following three operations are computed in a loop:

1. The set $en(g_n) \subseteq Act$ of enabled actions (not including the $\epsilon$ action) is identified and a subset $E(g_n) \subseteq en(g_n)$ of possible actions is heuristically selected (see below).

2. For any action $a \in E(g_n)$ compute the successor state $g'$ such that $g_n \xrightarrow{a} g'$, and add $g'$ to the stack thereby generating the path $\pi' = g_0a_0g_1a_1 \cdots g_nag'$. Recursively proceed to explore the submodel originating at $g'$ in the same way by means of the present algorithm beginning at step 1.

3. Remove $g_n$ from the stack.

The algorithm begins with a stack comprising of the initial state and terminates when the stack is empty. The model generated by the algorithm is a submodel of the full one. Its size crucially depends on the ratio $E(g)/en(g)$. The choice of $E(q)$ is constrained to preserve the stuttering trace equivalence between the original and a submodel generated.

6.3 Preserving $s\text{ATL}_{ir}^*$

In the sequel, let $\phi$ be an $s\text{ATL}_{ir}^*$ formula to be checked over the model $M$ and let $M'$ be a submodel of $M$, generated by the algorithm. The states and
the actions connecting states in $M'$ define a directed state graph. We give conditions defining a heuristics for the selection of $E(g)$ (such that $E(g) \neq en(g)$) while visiting state $g$ in the algorithm below.

**C1** No action $a \in Act \setminus E(g)$ that is dependent (see Subsection 2.2) on an action in $E(g)$ can be executed before an action in $E(g)$ is executed.

**C2** For every cycle in the constructed state graph there is at least one node $g$ in the cycle for which $E(g) = en(g)$, i.e., for which all the successors of $g$ are expanded.

**C3** All actions in $E(g)$ are invisible (see Subsection 2.2).

The conditions **C1** – **C3** are inspired from [54].

**Theorem 13.** Let $M$ be a model and $M' \subseteq M$ be the reduced model generated by the DFS algorithm described above in which the choice of $E(g')$ for $g' \in G'$ is given by **C1**, **C2**, **C3** above. Then, $M$ and $M'$ are stuttering trace equivalent.

**Proof.** Although the setting is slightly different it can be shown similarly to Theorem 3.11 in [55] that the conditions **C1**, **C2**, **C3** guarantee that the models $M$ and $M'$ are stuttering trace equivalent. More precisely, for each path $\pi = g_0 a_0 g_1 a_1 \cdots$ with $g_0 = \iota$ in $M$ there is a stuttering equivalent path $\pi' = g'_0 a'_0 g'_1 a'_1 \cdots$ with $g'_0 = \iota$ in $M'$ and a partition $B_1, \ldots, B_j, \ldots$ of the states of $\pi$ and a partition $B'_1, \ldots, B'_j, \ldots$ of the states of $\pi'$ satisfying for each $i, j \geq 0$ the following two conditions:

* I. if $g_i \xrightarrow{a} g_{i+1}$ is a transition such that $g_i, g_{i+1} \in B_j$, then $a \in \text{Invis}$, and if $g'_i \xrightarrow{a'} g'_{i+1}$ is a transition such that $g'_i, g'_{i+1} \in B'_j$, then $a' \in \text{Invis}$,

* II. if $g_i \xrightarrow{a} g_{i+1}$ is a transition such that $g_i \in B_j$ and $g_{i+1} \in B_{j+1}$, and $g'_i \xrightarrow{a'} g'_{i+1}$ is a transition such that $g'_i \in B'_j$ and $g'_{i+1} \in B'_{j+1}$, then $a = a'$.

Algorithms generating reduced models in which the choice of $E(g')$ for $g' \in G'$ is given by **C1**, **C2**, **C3** can be found in many papers [55, 54, 58, 22, 48, 49].
7 Conclusions and Future Work

Many important properties of multi-agent systems are underpinned by the ability of some agents (or groups of agents) to achieve a given goal. Requirements of this kind can be conveniently specified using logics of strategic ability, such as ATL and ATL*. However, their semantics typically use concurrent synchronous models, whereas, for many systems, asynchronous modeling would be more appropriate. In this paper, we propose a general semantics of ATL and ATL* for asynchronous MAS, and consider the model checking problem for a relevant subset of ATL* formulae.

Model checking of strategic abilities under imperfect information is a notoriously hard problem, for which attempts at practical algorithms and tools started emerging only recently. We establish the theoretical complexity of the problem, and, more importantly, propose a partial order reduction scheme that can substantially decrease the practical complexity of verification. Interestingly, it turns out that the scheme does not work for perfect information strategies. Until now, virtually all the results have suggested that verification of strategic abilities is significantly easier for agents with perfect information. Thus, we identify an aspect of verification that might be in favor of imperfect information strategies in some contexts.

Considering potential practical applications, we can verify correctness of authentication protocols as specified and discussed in [31] as well as of timed authentication protocols [29, 30, 42] after extending our results to Timed ATL [43]. For example we can check the sATL property expressing that an intruder does not have a strategy to possess an ‘insecure’ information.

In the future, we plan also to extend our method to a larger subset of ATL* specifications. Adapting the POR scheme to combinations of strategic and epistemic modalities is another interesting path for future work. Finally, we would like to investigate whether our partial order reduction scheme can be combined with the bisimulation-based reduction for ATL_{ir}, proposed very recently in [7].
REFERENCES

References


REFERENCES


REFERENCES


REFERENCES


Pracę zgłosił Jan Mielniczuk

Adresy autorów: Piotr Dembiński, Wojciech Jamroga
Antoni Mazurkiewicz, Wojciech Penczek
Instytut Podstaw Informatyki PAN
ul. Jana Kazimierza 5
01-248 Warszawa, Polska

E-mail: piotrd, jamroga@ipipan.waw.pl
amaz, penczek@ipipan.waw.pl

Symbol klasyfikacji rzeczowej: D.2.4, F.3.1, F.4.1, H.2.4

Na prawach rękopisu
Printed as manuscript