Alternating Refinement Relations[⋆] [⋆⋆]

Rajeev Alur[1] Thomas A. Henzinger[2] Orna Kupferman[2] Moshe Y. Vardi[3]

[1] Department of Computer and Information Science,
University of Pennsylvania, Philadelphia, PA 19104, USA
Email: alur@cis.upenn.edu
URL: www.cis.upenn.edu/˜alur

[2] Department of Electrical Engineering and Computer Sciences,
University of California, Berkeley, CA 94720-1770, USA
Email: {tah,orna}@eecs.berkeley.edu
URL: www.eecs.berkeley.edu/˜{tah,orna}

[3] Department of Computer Science,
Rice University, Houston, TX 77005-1892, USA
Email: vardi@cs.rice.edu
URL: http://www.cs.rice.edu/˜vardi

Abstract. *Alternating transition systems* are a general model for composite systems which allows the study of collaborative as well as adversarial relationships between individual system components. Unlike in labeled transition systems, where each transition corresponds to a possible step of the system (which may involve some or all components), in alternating transition systems, each transition corresponds to a possible move in a game between the components. In this paper, we study refinement relations between alternating transition systems, such as "Does the implementation refine the set A of specification components without constraining the components not in A?" In particular, we generalize the definitions of the simulation and trace containment preorders from labeled transition systems to alternating transition systems. The generalizations are called *alternating simulation* and *alternating trace containment*. Unlike existing refinement relations, they allow the refinement of individual components within the context of a composite system description. We show that, like ordinary simulation, alternating simulation can be checked in polynomial time using a fixpoint computation algorithm. While ordinary trace containment is PSPACE-complete, we prove alternating trace containment to be EXPTIME-complete. Finally, we present logical characterizations for the two preorders in terms of ATL, a temporal logic capable of referring to games between system components.

[⋆] An abbreviated version of this paper appeared in the *Proceedings of the Tenth International Conference on Concurrency Theory* (CONCUR), Lecture Notes in Computer Science 1466, Springer-Verlag, 1998, pp. 163–178.

[⋆⋆] This work is supported in part by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the NSF grants CCR-9504469, CCR-9628400, and CCR-9700061, by the DARPA/NASA grant NAG2-1214, by the ARO MURI grant DAAH-04-96-1-0341, by the SRC contract 97-DC-324.041, and by a grant from the Intel Corporation.

1 Introduction

A central issue in a formal approach to design and analysis of reactive systems is the notion of *refinement*. The relation "A_I refines A_S" is intuitively meant to say that "system A_S has more behavioral options than system A_I," or equivalently, "every behavioral option realized by implementation A_I is allowed by specification A_S." Broadly speaking, there are two kinds of interpretations for "behavioral options": *global* interpretations as sequences of observables, and *local* interpretations as successor observables at individual states. The former leads to refinement as *trace containment*, or one of its relatives; the latter leads to refinement as *simulation* [Mil71], or one of its relatives.

Consider now a composite implementation $A_I||B$ and specification $A_S||B$. Suppose we want to check that the A-component A_I of the implementation refines the A-component A_S of the specification. The traditional refinement preorders are inappropriate in this setting, because they allow A_I to achieve refinement by constraining its environment B, for example, by refusing certain inputs from B. This problem is well-known, and has led to more complicated versions of refinements such as ready simulation and failures containment. These variants have been defined on the labeled transition graphs of components in such a way that they are congruent with respect to parallel composition; then, it suffices to prove that A_I refines A_S in order to conclude that $A_I||B$ refines $A_S||B$. However, now the burden is on A_S to allow all possible behaviors of B, such as permitting at any time any input from B. If more complicated assumptions are required about B, they also need to be folded into A_S. We suggest a different, potentially more general route, of modeling the environment B explicitly. In this way, we can keep all assumptions about the environment separate from the models of A. The main problem then is to specify, and verify, the relation "A_I refines A_S *without constraining the environment* B." For this purpose, we propose definitions of simulation and trace containment that are parameterized by names of components. The resulting *alternating* refinement preorders allow us to check refinement with respect to any subset of the system components.

Composite systems can be viewed as multi-agent systems [Sha53, HF89]. While in labeled transition systems, each transition corresponds to a possible step of the system (which may involve some or all components), in multi-agent systems each transition corresponds to a possible move in a game between the components (which are called *agents*). We model multi-agents systems by *alternating transition systems* (ATS), proposed in [AHK97]. In each move of the game between the agents of an ATS, the choice of an agent at a state is a set of states, and the successor state is determined by considering the intersection of the choices made by all agents. Unlike labeled transition systems, ATS can distinguish between collaborative and adversarial relationships among components. For example, the environment is typically viewed adversarially, meaning that a component may be required to meet its specification *no matter how the environment behaves*. Then, a refinement of the component must not constrain the environment. By contrast, if two components collaborate to meet a specification, then a refinement of one component may constrain the other component.

Before we explain alternating refinement relations, let us consider the simulation refinement that is defined via games played on the graphs of labeled transition systems. To determine whether the initial state s of system A_I is simulated by the initial state t of system A_S, consider the following two-player game between protagonist and antagonist. With each move of the game, the antagonist updates the state of A_I applying any transition of A_I, and then, the protagonist must update the state of A_S using a transition of A_S so that the observables of the updated states match. If the protagonist fails to produce a match, the antagonist wins; if the game continues forever, the protagonist wins. The state s is simulated by t if the protagonist has a winning strategy in this game.

For a subset A of agents, the alternating A-simulation relation is defined via a similar two-player game, except that the game is played on the graph of an ATS and a move now consists of four parts. Consider the game scenario with antagonist at state s and protagonist at state t. First, the antagonist makes choices for the the agents in A at state s. Second, the protagonist makes choices for the agents in A at state t. Third, the antagonist updates the state t in a way that is consistent with the choices made in the second part. Fourth, the protagonist updates the state s consistent with the choices made in the first part so that the observables of the updated states match. Thus, compared to the simulation game for labeled transition systems, protagonist and antagonist play the same roles for the choices of the agents in A, while their roles are reversed for the choices of the agents not in A.

We present several results that support the claim that our definition of alternating simulation is a natural, and useful, generalization of Milner's simulation. First, when restricted to ATS with a single agent—i.e., to labeled transition systems—the two notions coincide. Second, we show that for finite ATS, deciding A-simulation, for a given set A of agents, is solvable in polynomial time. Third, we present a logical characterization of alternating simulation. In [AHK97], we proposed *alternating temporal logic* as a language for specifying properties of system components. In particular, ATL and ATL* are the alternating versions of the branching temporal logics CTL and CTL*. Besides universal (do all computations satisfy a property?) and existential (does some computation satisfy a property?) requirements of CTL, in ATL one can specify alternating requirements: can a component resolve its choices so that the satisfaction of a property is guaranteed no matter how the environment resolves its choices? We show that an ATS $\mathcal{S}$ is A-simulated by an ATS $\mathcal{I}$ precisely when every ATL (or ATL*) formula with path quantifiers parameterized by A that holds in $\mathcal{S}$, also holds in $\mathcal{I}$. This result, which generalizes the relationship between ordinary simulation and the universal fragment of CTL (or CTL*), allows us to carry over alternating temporal logic properties from the specification to the implementation.

The second refinement relation studied in this paper is trace containment. For labeled transition systems, the specification $\mathcal{S}$ trace-contains the implementation $\mathcal{I}$ if for every global computation of $\mathcal{I}$ chosen by the antagonist, the protagonist can produce a computation of $\mathcal{S}$ with the same sequence of observables. The corresponding generalization to ATS is *alternating trace containment.* For ATS $\mathcal{S}$ and $\mathcal{I}$ and a set A of agents, the relation $\mathcal{I}$ A-trace-contains $\mathcal{S}$ is determined as follows: the antagonist first chooses a strategy in $\mathcal{S}$ for the agents in A, the protagonist then chooses a strategy in $\mathcal{I}$ for the agents in A, the antagonist then determines a computation of $\mathcal{I}$ by resolving in $\mathcal{I}$ the choices for the agents not in A, and finally, the protagonist must produce a computation of $\mathcal{S}$ with the same sequence of observables by resolving in $\mathcal{S}$ the choices for the agents not in A.

As is the case for the corresponding relations on labeled transition systems, alternating simulation implies alternating trace containment, but not vice versa. Checking alternating trace containment corresponds to checking inclusion between *sets* of ω-languages (i.e., between sets of sets of traces). Our solution is based on a novel application of tree automata in which ω-languages are represented as trees. We show that the problem of deciding alternating trace containment is EXPTIME-complete, and we give a logical characterization of alternating trace containment using a fragment of ATL* .

2 Alternating Transition Systems

In ordinary transition systems, each transition corresponds to a possible step of the system. In *alternating transition systems* (ATS, for short), introduced in [AHK97], each transition corresponds to a possible move in the game between the underlying components of the system. We refer to the components as *agents.* In each move of the game, every agent chooses a set of successor states. The game then proceeds to the state in the intersection of the sets chosen by all agents. Equivalently, each agent puts a constraint on the choice of the successor state, and the game proceeds to a state that satisfies the constraints imposed by all the agents.

Formally, an alternating transition system is a 6-tuple $\mathcal{S} = \langle \Pi, \Omega, Q, q_{in}, \pi, \delta \rangle$ with the following components:

- Π is a finite set of propositions.
- Ω is a finite set of agents.
- Q is a finite set of states.
- q_{in} is an initial state.
- $\pi : Q \rightarrow 2^{\Pi}$ maps each state to the set of propositions that are true in the state.
- $\delta : Q \times \Omega \rightarrow 2^{2^Q}$ is a transition function that maps a state and an agent to a nonempty set of choices, where each choice is a set of possible next states. Whenever the system is in state q, each agent a chooses a set $Q_a \in \delta(q, a)$. In this way, an agent a ensures that the next state of the system will be in its choice Q_a. However, which state in Q_a will be next depends on the choices made by the other agents, because the successor of q must lie in the intersection $\bigcap_{a \in \Omega} Q_a$ of the choices made by all the agents. We require that the transition function is nonblocking and that the agents together choose

a unique next state: assuming $\Omega = \{a_1, \ldots, a_n\}$, for every state $q \in Q$ and every set $Q_1, \ldots, Q_n$ of choices $Q_i \in \delta(q, a_i)$, the intersection $Q_1 \cap \ldots \cap Q_n$ is a singleton.

The number of transitions of $\mathcal{S}$ is defined to be $\sum_{q \in Q, a \in \Omega} |\delta(q, a)|$. For two states q and q' and an agent a, we say that q' is an *a-successor* of q if there exists a set $Q' \in \delta(q, a)$ such that $q' \in Q'$. For two states q and q', we say that q' is a *successor* of q if for all agents $a \in \Omega$, the state q' is an a-successor of q. Thus, q' is a successor of q iff whenever the system $\mathcal{S}$ is in state q, the agents in Ω can cooperate so that q' will be the next state. A *computation* of $\mathcal{S}$ is an infinite sequence $\eta = q_0, q_1, q_2, \ldots$ of states such that for all positions $i \geq 0$, the state q_{i+1} is a successor of the state q_i. We refer to a computation starting at state q as a *q-computation*. For a computation η and a position $i \geq 0$, we use $\eta[i]$, $\eta[0, i]$, and $\eta[i, \infty]$ to denote the i-th state in η, the finite prefix $q_0, q_1, \ldots, q_i$ of η, and the infinite suffix $q_i, q_{i+1}, \ldots$ of η, respectively. Each computation $\eta = q_0, q_1, q_2, \ldots$ induces a *trace* $\pi(\eta) = \pi(q_0) \cdot \pi(q_1) \cdot \pi(q_2) \cdots$ in $(2^{\Pi})^{\omega}$.

Example 1. Consider a system with two processes a and b. The process a assigns values to the boolean variable x. When $x = \mathit{false}$, then a can leave the value of x unchanged or change it to true. When $x = \mathit{true}$, then a leaves the value of x unchanged. In a similar way, the process b assigns values to the boolean variable y. When $y = \mathit{false}$, then b can leave the value of y unchanged or change it to true. When $y = \mathit{true}$, then b leaves the value of y unchanged. The initial value of both x and y is false. We model the composition of the two processes by the following ATS $\mathcal{S} = \langle \Pi, \Omega, Q, q, \pi, \delta \rangle$:

- $\Pi = \{x, y\}$.
- $\Omega = \{a, b\}$.
- $Q = \{q, q_y, q_x, q_{xy}\}$. The state q corresponds to $x = y = \mathit{false}$, the state q_x corresponds to $x = \mathit{true}$ and $y = \mathit{false}$, and similarly for q_y and q_{xy}.
- The labeling function $\pi : Q \rightarrow 2^{\Pi}$ is therefore as follows:
  - $\pi(q) = \emptyset$.
  - $\pi(q_x) = \{x\}$.
  - $\pi(q_y) = \{y\}$.
  - $\pi(q_{xy}) = \{x, y\}$.
- The transition function $\delta : Q \times \Omega \rightarrow 2^{2^Q}$ is as follows:
  - $\delta(q, a) = \{\{q, q_y\}, \{q_x, q_{xy}\}\}$.
  - $\delta(q, b) = \{\{q, q_x\}, \{q_y, q_{xy}\}\}$.
  - $\delta(q_x, a) = \{\{q_x, q_{xy}\}\}$.
  - $\delta(q_x, b) = \{\{q, q_x\}, \{q_y, q_{xy}\}\}$.
  - $\delta(q_y, a) = \{\{q, q_y\}, \{q_x, q_{xy}\}\}$.
  - $\delta(q_y, b) = \{\{q_y, q_{xy}\}\}$.
  - $\delta(q_{xy}, a) = \{\{q_x, q_{xy}\}\}$.
  - $\delta(q_{xy}, b) = \{\{q_y, q_{xy}\}\}$.

  Consider, for example, the transition $\delta(q, a)$. As the process a controls only the value of x, and can change its value from false to true, the agent a can determine whether the next state of the system will be some q' with $x \in \pi(q')$ or some q' with $x \notin \pi(q')$. It cannot, however, determine the value of y. Therefore, $\delta(q, a) = \{\{q, q_y\}, \{q_x, q_{xy}\}\}$, letting a choose between $\{q, q_y\}$ and $\{q_x, q_{xy}\}$, yet leaving the choice between q and q_y, in the first case, and between q_x and q_{xy}, in the second case, to process b.

Consider the state q_x. While the state q_y is a b-successor of q_x, the state q_y is not an a-successor of q_x. Therefore, the state q_y is not a successor of q_x: when the system is in state q_x, the processes a and b cannot cooperate so that the system will move to q_y. On the other hand, the agents can cooperate so that the system will stay in state q_x or move to q_{xy}. By similar considerations, it follows that the infinite sequences $q, q, q_x, q_x, q_x, q_{xy}^\omega$ and $q, q_y, q_y, q_{xy}^\omega$ and q, q_{xy}^ω are three possible q-computations of the ATS $\mathcal{S}$. $\square$

An ordinary *labeled transition system*, or Kripke structure, is the special case of an ATS where the set $\Omega = \{sys\}$ of agents is a singleton set. In this special case, the sole agent sys can always determine the successor state: for all states $q \in Q$, the transition $\delta(q, sys)$ must contain a nonempty set of choices, each of which is a singleton set.

Often, we are interested in the cooperation of a subset $A \subseteq \Omega$ of the agents. Given A, we define

$$\delta(q, A) = \{T : \text{ for each } a_i \in A \text{ there exists } Q_i \in \delta(q, a_i) \text{ and } T = \bigcap_{a_i \in A} Q_i\}.$$

Intuitively, whenever the system is in state q, the agents in A can choose a set $T \in \delta(q, A)$ such that, no matter what the other agents do, the next state of the system is in T. Correspondingly, we define $\delta(q, \emptyset)$ to contain the single set of all successors of q. When all agents cooperate, they can decide the next state; that is, $\delta(q, \Omega)$ is a set of singletons.

A *strategy* for an agent $a \in \Omega$ is a mapping $f_a : Q^+ \to 2^Q$ such that for $\rho \in Q^*$ and $q \in Q$, we have $f_a(\rho \cdot q) \in \delta(q, a)$. Thus, the strategy f_a maps a finite nonempty prefix $\rho \cdot q$ of a computation to a set in $\delta(q, a)$. This set contains possible extensions of the computation as suggested to agent a by the strategy. Each strategy f_a induces a set of computations that agent a can enforce. For a set A of agents, and a set $F_A = \{f_a : a \in A\}$ of strategies for the agents in A, we sometimes refer to F_A as a strategy $F_A : Q^+ \to 2^Q$ where $F_A(\rho) = \bigcap_{a \in A} f_a(\rho)$. Note that $F_A(\rho \cdot q) \in \delta(q, A)$. Given a state q, and a strategy F_A, we define the *outcomes* in $\mathcal{S}$ of F_A from q to be the set $out_{\mathcal{S}}(q, F_A)$ of all q-computations that the agents in A can enforce when they cooperate and follow the strategies in F_A; that is, a q-computation η is in $out_{\mathcal{S}}(q, F_A)$ iff η always proceeds according to the strategies in F_A. Formally, $\eta = q_0, q_1, \ldots$ is in $out_{\mathcal{S}}(q, F_A)$ iff $q_0 = q$, and for all positions $i \geq 0$, the state q_{i+1} is a successor of q_i satisfying $q_{i+1} \in F_A(\eta[0, i])$.

3 Alternating Simulation

We generalize simulation between labeled transition systems to *alternating simulation* between ATS. Consider fixed sets Π of propositions and Ω of agents, and two ATS $\mathcal{S} = \langle \Pi, \Omega, Q, q_{in}, \pi, \delta \rangle$ and $\mathcal{S}' = \langle \Pi, \Omega, Q', q'_{in}, \pi', \delta' \rangle$. For a subset $A \subseteq \Omega$ of the agents, a relation $H \subseteq Q \times Q'$ is an *A-simulation from $\mathcal{S}$ to $\mathcal{S}'$* if for all states q and q' with $H(q, q')$ the following conditions hold:

(1) $\pi(q) = \pi'(q')$.
(2) For every set $T \in \delta(q, A)$, there exists a set $T' \in \delta'(q', A)$ such that for every set $R' \in \delta'(q', \Omega \setminus A)$, there exists a set $R \in \delta(q, \Omega \setminus A)$ so that $(T \cap R) \times (T' \cap R') \subseteq H$.

Note that since $\delta(q, \Omega)$ is a set of singletons, the product $(T \cap R) \times (T' \cap R')$ contains a single pair. If there exists an A-simulation H from $\mathcal{S}$ to $\mathcal{S}'$ with $H(q_{in}, q'_{in})$, we say that $\mathcal{S}'$ *A-simulates* $\mathcal{S}$, and we write $\mathcal{S} \leq_A \mathcal{S}'$ (when $A = \{a\}$ is a singleton, we call H an a-simulation and write $\mathcal{S} \leq_a \mathcal{S}'$). It is easy to check that $\leq_A$ is a preorder on ATS.

Intuitively, $H(q, q')$ means that for every choice T of the agents in A from q, there exists a matching choice T' of the agents in A from q' such that for every choice R' of the agents in $\Omega \setminus A$ from q', there exists a choice R of the agents in $\Omega \setminus A$ from q so that the successor of q' that follows from the choices T' and R' is in a simulation relation with the successor of q that follows from the choices T and R. This intuition is captured in the following game-theoretic interpretation of alternating simulation. Consider a two-player game whose positions are pairs $\langle q, q' \rangle \in Q \times Q'$ of states. The initial position is $\langle q_{in}, q'_{in} \rangle$. The game is played between an antagonist and a protagonist and it proceeds in a sequence of rounds. Each round consists of four steps as follows. Assume that the current position is $\langle q, q' \rangle$.

1. The antagonist chooses a set $T \in \delta(q, A)$.
2. The protagonist choose a set $T' \in \delta(q', A)$.
3. The antagonist chooses a state $u' \in T'$ such that u' is a successor of q'.
4. The protagonist chooses a state $u \in T$ such that u is a successor of q and $\pi(u) = \pi'(u')$.

If the game proceeds ad infinitum, then the antagonist loses. Otherwise the game reaches a position from which the protagonist cannot chose u as required, and the antagonist wins. It can be shown that $\mathcal{S}'$ A-simulates $\mathcal{S}$ iff the protagonist has a winning strategy.

Another way to understand alternating simulation is to observe that $\mathcal{S}'$ A-simulates $\mathcal{S}$ iff each behavior that the agents in A can induce in $\mathcal{S}$, they can also induce in $\mathcal{S}'$. In Lemma 1 below, we make this observation formal. For two computations (or prefixes of computations) $\eta = q_0, q_1, \ldots$ of $\mathcal{S}$ and $\eta' = q'_0, q'_1, \ldots$ of $\mathcal{S}'$, we write $H(\eta, \eta')$ to abbreviate $H(q_i, q'_i)$ for all positions $i \geq 0$.

Lemma 1. *Consider two ATS $\mathcal{S}$ and $\mathcal{S}'$, and a set A of agents. If H is an A-simulation from $\mathcal{S}$ to $\mathcal{S}'$, then for every two states q and q' with $H(q, q')$ and for every set F_A of strategies in $\mathcal{S}$ for the agents in A, there exists a set F'_A of strategies in $\mathcal{S}'$ for the agents in A such that for every computation $\rho' \in out_{\mathcal{S}'}(q', F'_A)$, there exists a computation $\rho \in out_{\mathcal{S}}(q, F_A)$ so that $H(\rho, \rho')$.*

Recall that a labeled transition system corresponds to an ATS with the single agent *sys*. Our definition of alternating simulation then coincides with Milner's definition of simulation between labeled transition systems [Mil71]. This is because $\mathcal{S} \leq_{sys} \mathcal{S}'$ iff there exists a relation H where $H(q,q')$ implies that $\pi(q) = \pi'(q')$ and for every $\{t\} \in \delta(q, sys)$ there exists $\{t'\} \in \delta'(q', sys)$ such that $H(t,t')$. Note also that $\mathcal{S} \leq_{\emptyset} \mathcal{S}'$ iff there exists a relation H where $H(q,q')$ implies that $\pi(q) = \pi'(q')$ and for every $\{r'\} \in \delta(q', sys)$ there exists $\{r\} \in \delta(q, sys)$ such that $H(r,r')$. Thus, $\mathcal{S} \leq_{\emptyset} \mathcal{S}'$ iff $\mathcal{S}' \leq_{sys} \mathcal{S}$. It follows that alternating simulation can be used on labeled transition systems to specify both directions of Milner's simulation.

Example 2. In Example 1, we described an ATS $\mathcal{S}$ for a system with two processes a and b, which assign values to the boolean variables x and y. Consider a variant of the system in which whenever the value of both x and y is *false*, process b assigns values to both x and y. We can model the composition of the two processes by the following ATS $\mathcal{S}' = \langle \Pi, \Omega, Q', q', \pi', \delta' \rangle$:

- $Q' = \{q', q'_y, q'_x, q'_{xy}\}$.
- The labeling function $\pi' : Q' \to 2^{\Pi}$ is as in Example 1.
- The transition function $\delta' : Q' \times \Omega \to 2^{2^{Q'}}$ is as follows:
  - $\delta'(q', a) = \{\{q', q'_y, q'_x, q'_{xy}\}\}$.
  - $\delta'(q'_x, a) = \{\{q'_x, q'_{xy}\}\}$.
  - $\delta'(q'_y, a) = \{\{q', q'_y\}, \{q'_x, q'_{xy}\}\}$.
  - $\delta'(q'_{xy}, a) = \{\{q'_x, q'_{xy}\}\}$.
  - $\delta'(q', b) = \{\{q'\}, \{q'_x\}, \{q'_y\}, \{q'_{xy}\}\}$.
  - $\delta'(q'_x, b) = \{\{q', q'_x\}, \{q'_y, q'_{xy}\}\}$.
  - $\delta'(q'_y, b) = \{\{q'_y, q'_{xy}\}\}$.
  - $\delta'(q'_{xy}, b) = \{\{q'_y, q'_{xy}\}\}$.

Intuitively, while the joint behavior of a and b is the same in $\mathcal{S}$ and $\mathcal{S}'$, process b is more powerful in system $\mathcal{S}'$ than in system $\mathcal{S}$. Every behavior that b can induce in $\mathcal{S}$, it can also induce in $\mathcal{S}'$ (formally, $\mathcal{S} \leq_b \mathcal{S}'$). On the other hand, there are behaviors that b can induce in $\mathcal{S}'$ but cannot induce in $\mathcal{S}$ (formally, $\mathcal{S}' \not\leq_b \mathcal{S}$). Dually, process a is more powerful in $\mathcal{S}$ than in $\mathcal{S}'$: we have $\mathcal{S}' \leq_a \mathcal{S}$ and $\mathcal{S} \not\leq_a \mathcal{S}'$.

Consider the relation $H = \{\langle q, q'\rangle, \langle q_x, q'_x\rangle, \langle q_y, q'_y\rangle, \langle q_{xy}, q'_{xy}\rangle\}$. It is easy to see that H is an A-simulation from $\mathcal{S}$ to $\mathcal{S}'$ for $A \in \{\{a,b\}, \{b\}, \emptyset\}$. It follows that $\mathcal{S} \leq_{\{a,b\}} \mathcal{S}'$, $\mathcal{S} \leq_b \mathcal{S}'$, and $\mathcal{S} \leq_{\emptyset} \mathcal{S}'$. We now prove that $\mathcal{S} \not\leq_a \mathcal{S}'$. Assume, by way of contradiction, that an a-simulation relation H' from $\mathcal{S}$ to $\mathcal{S}'$ exists, and let H be as above. By the definition of a-simulation, it must be that $H' \subseteq H$ and $\langle q, q'\rangle \in H'$. Since $\delta'(q', a) = \{\{q', q'_x, q'_y, q'_{xy}\}\}$, by condition (2) for an a-simulation, for every set $T \in \delta(q,a)$ and for every set $R' \in \delta'(q', b)$, there exists a set $R \in \delta(q, b)$ such that $(T \cap R) \times R' \subseteq H'$. Consider the sets $\{q, q_y\} \in \delta(q,a)$ and $\{q'_x\} \in \delta'(q', b)$. Since for every $R \in \delta(q,b)$ we have $q_x \notin R \cap \{q, q_y\}$, it follows that $(T \cap R) \times R' \not\subseteq H'$, and we reach a contradiction. □

Proposition 2. *Consider two ATS $\mathcal{S}$ and $\mathcal{S}'$, and two sets A and B of agents from Ω. Then:*

1. *$\mathcal{S} \leq_A \mathcal{S}'$ does not imply $\mathcal{S}' \leq_{\Omega \setminus A} \mathcal{S}$.*
2. *$\mathcal{S} \leq_A \mathcal{S}'$ does not imply $\mathcal{S} \leq_{A'} \mathcal{S}'$ for $A' \subseteq A$.*
3. *$\mathcal{S} \leq_A \mathcal{S}'$ and $\mathcal{S} \leq_B \mathcal{S}'$ does not imply $\mathcal{S} \leq_{A \cup B} \mathcal{S}'$.*

The properties studied in Proposition 2 describe the power of cooperation between agents in $\mathcal{S}$ and $\mathcal{S}'$. Intuitively, the properties can be understood as follows.

1. It may be that every behavior that the agents in A can induce in $\mathcal{S}$, they can also induce in $\mathcal{S}'$, yet still there are behaviors that the agents in A can avoid in $\mathcal{S}$ but cannot avoid in $\mathcal{S}'$. Technically, it follows that in the definition of A-simulation, the order in which the sets T, T', R, and R' are selected is important, as R and R' may depend on T and T'. We note that in the special cases $A = \Omega$ and $A = \emptyset$, the property does hold. Thus, $\mathcal{S} \leq_{\Omega} \mathcal{S}'$ iff $\mathcal{S}' \leq_{\emptyset} \mathcal{S}$.
2. It may be that every behavior that the agents in A can induce in $\mathcal{S}$, they can also induce in $\mathcal{S}'$, but the cooperation of all agents in A is required.
3. It may be that every behavior that the agents in A can induce in $\mathcal{S}$, they can also induce in $\mathcal{S}'$, every behavior that the agents in B can induce in $\mathcal{S}$, they can also induce in $\mathcal{S}'$, and still the cooperation of the agents in A and B is stronger in $\mathcal{S}$ than their cooperation in $\mathcal{S}'$. The special case of $A \cup B = \Omega$ does not hold either.

Checking alternating simulation

Given two ATS $\mathcal{S}$ and $\mathcal{S}'$ and a set A of agents, the *alternating-simulation problem* is to determine whether $\mathcal{S} \leq_A \mathcal{S}'$. The local definition of ordinary simulation for labeled transition systems makes its decidability easy. Specifically, given two labeled transition systems $\mathcal{S}$ and $\mathcal{S}'$, it is possible to determine whether $\mathcal{S} \leq \mathcal{S}'$ in time that is quadratic in the sizes of $\mathcal{S}$ and $\mathcal{S}'$ [HHK95], and a witnessing relation for simulation can be computed using a symbolic fixpoint procedure [Mil90]. We show that alternating simulation can also be computed in polynomial time, as well as symbolically.

Theorem 3. *The alternating-simulation problem is PTIME-complete.*

Proof. Consider two ATS $\mathcal{S} = \langle \Pi, \Omega, Q, q_{in}, \pi, \delta \rangle$ and $\mathcal{S}' = \langle \Pi, \Omega, Q', q'_{in}, \pi', \delta' \rangle$. For a set A of agents, a relation $H \subseteq Q \times Q'$, and a pair $\langle q, q' \rangle \in Q \times Q'$, we say that $\langle q, q' \rangle$ is *A-good* in H iff conditions (1) and (2) from the definition of A-simulation hold for $\langle q, q' \rangle$. Following [Mil90], we characterize alternating simulation as a greatest fixpoint. Let

$$H_0 = \{\langle q, q' \rangle : q \in Q, q' \in Q', \text{ and } \pi(q) = \pi(q')\}.$$

Thus, H_0 is the maximal relation whose pairs satisfy condition (1) of A-simulation. Consider the monotonic function $f : 2^{Q \times Q'} \to 2^{Q \times Q'}$, where

$$f(H) = H \cap \{\langle q, q' \rangle : \langle q, q' \rangle \text{ is } A\text{-good in } H\}.$$

Thus, $f(H)$ contains all pairs in H that are A-good in H. Let $H^\star$ be the greatest fixpoint of f when restricted to pairs in H_0; that is, $H^\star = (\nu z)(H_0 \cap f(z))$. Then, $\mathcal{S} \leq_A \mathcal{S}'$ iff $H^\star(q_{in}, q'_{in})$. Since $Q \times Q'$ is finite, we can calculate $H^\star$ by iterative application of f, starting with H_0 until we reach a fixpoint. There can be at most $|Q \times Q'|$ many iterations. Checking whether a pair $\langle q, q' \rangle$ is A-good in H_i can be performed in time polynomial in $\delta(q, A)$ and $\delta'(q', A)$. Since the number of checks for each H_i is bounded by $|Q \times Q'|$, the overall effort is polynomial in $\mathcal{S}$ and $\mathcal{S}'$.

Hardness in PTIME follows from the PTIME-hardness of ordinary simulation on labeled transition systems [BGS92, KV98]. □

Recall that alternating simulation can be used on labeled transition systems to specify both directions of simulation. Since the complexity of the simulation problem $\mathcal{S} \leq_{sys} \mathcal{S}'$ for labeled transition systems $\mathcal{S}$ and $\mathcal{S}'$ is hard for PTIME already for a fixed specification $\mathcal{S}'$ [KV98], it follows that the alternating-simulation problem is PTIME-complete even when either $\mathcal{S}$ or $\mathcal{S}'$ is fixed.

4 Alternating Trace Containment

We now study the refinement relation on ATS that corresponds to trace containment on labeled transition systems. Consider an ATS $\mathcal{S} = \langle \Pi, \Omega, Q, q_{in}, \pi, \delta \rangle$. For a set A of agents and a set F_A of strategies for the agents in A, let $trace_{\mathcal{S}}(F_A) \subseteq \Omega^\omega$ be the set of traces that the agents in A can enforce when they follow the strategies in F_A; that is,

$$trace_{\mathcal{S}}(F_A) = \{w : \text{ there exists } \eta \in out_{\mathcal{S}}(q_{in}, F_A) \text{ such that } \pi(\eta) = w\}.$$

Using different strategies, the agents in A can enforce different trace sets. Let $\mathcal{L}_{\mathcal{S}}(A)$ denote the trace sets that the agents in A can enforce; that is,

$$\mathcal{L}_{\mathcal{S}}(A) = \{L : \text{ there exists a set } F_A \text{ of strategies for } A \text{ with } L = trace_{\mathcal{S}}(F_A)\}.$$

For two ATS $\mathcal{S}$ and $\mathcal{S}'$ over the same set Ω of agents, and a subset $A \subseteq \Omega$ of the agents, we say that $\mathcal{S}'$ *A-trace contains* $\mathcal{S}$, denoted $\mathcal{S} \subseteq_A \mathcal{S}'$, iff for every trace set $L \in \mathcal{L}_{\mathcal{S}}(A)$, there exists a trace set $L' \in \mathcal{L}_{\mathcal{S}'}(A)$ such that $L' \subseteq L$. The relation $\leq_A$ is again a preorder on ATS.

Example 3. Consider the ATS $\mathcal{S}$ and $\mathcal{S}'$ from Examples 1 and 2. The trace sets that the agents a and b can enforce in $\mathcal{S}$ and $\mathcal{S}'$ are as follows:

- $\mathcal{L}_{\mathcal{S}}(a) = \{\emptyset^\omega + \emptyset^+ \cdot \{y\}^\omega\} \cup$
  $\{\emptyset^{i+1} \cdot \{y\}^j \cdot \{x,y\}^\omega + \emptyset^{n+1} \cdot \{x\}^* \cdot \{x,y\}^\omega : i + j = n \geq 0\}$.
- $\mathcal{L}_{\mathcal{S}'}(a) = \{\emptyset^\omega + \emptyset^+ \cdot \{y\}^\omega + \emptyset^+ \cdot \{x\}^\omega + \emptyset^+ \cdot \{y\}^* \cdot \{x,y\}^\omega + \emptyset^+ \cdot \{x\}^* \cdot \{x,y\}^\omega\}$.
- $\mathcal{L}_{\mathcal{S}}(\{a,b\}) = \mathcal{L}_{\mathcal{S}'}(\{a,b\}) = \{\emptyset^\omega\} \cup \{\emptyset^i \cdot \{x\}^\omega : i \geq 1\} \cup \{\emptyset^i \cdot \{y\}^\omega : i \geq 1\} \cup$
  $\{\emptyset^i \cdot \{x\}^j \cdot \{x,y\}^\omega : i \geq 1, j \geq 0\} \cup \{\emptyset^i \cdot \{y\}^j \cdot \{x,y\}^\omega : i \geq 1, j \geq 0\}$.

It follows that $\mathcal{S} \subseteq_{\{a,b\}} \mathcal{S}'$ and $\mathcal{S} \not\subseteq_a \mathcal{S}'$. On the other hand, it is not hard to see that $\mathcal{S} \subseteq_b \mathcal{S}'$. □

Recall that a labeled transition system corresponds to an ATS with the single agent *sys*. Our definition of alternating trace containment then coincides with ordinary trace containment between labeled transition systems. This is because when *sys* is the only agent, then for every strategy f_{sys} for *sys*, the set $\mathit{traces}_{\mathcal{S}}(f_{sys})$ contains a single trace. Hence, $\mathcal{S} \subseteq_{sys} \mathcal{S}'$ iff for every computation η of $\mathcal{S}$, there exists a computation η' of $\mathcal{S}'$ such that $\pi(\eta) = \pi'(\eta')$.

Remark. In the definition of alternating trace containment, the strategy of an agent may depend on an unbounded amount of information, namely, the full history of the game up to the current state. If we consider instead memoryless strategies —that is, strategies $f_a : Q \to 2^Q$, which depend only on the current state of the game— then the alternating trace-containment relation we obtain is different. On the other hand, as the definition of alternating simulation is local, Lemma 1 holds also for memoryless strategies. $\square$

As in the nonalternating case, while alternating simulation implies alternating trace containment, the converse direction does not hold.

Proposition 4. *Alternating simulation is stronger than alternating trace containment:*

(1) *For all ATS $\mathcal{S}$ and $\mathcal{S}'$, and every set A of agents, $\mathcal{S} \leq_A \mathcal{S}'$ implies $\mathcal{S} \subseteq_A \mathcal{S}'$.*
(2) *There exist ATS $\mathcal{S}$ and $\mathcal{S}'$ and a set A of agents such that $\mathcal{S} \subseteq_A \mathcal{S}'$ and $\mathcal{S} \not\leq_A \mathcal{S}'$.*

For deterministic labeled transition systems, simulation and trace containment coincide. This motivates the following definition. An ATS $\mathcal{S} = \langle \Pi, \Omega, Q, q, \pi, \delta \rangle$ is *A-deterministic*, for a subset $A \subseteq \Omega$ of the agents, iff for all states $q \in Q$ and sets $\Psi \subseteq \Pi$ of propositions, there exists at most one choice $T \in \delta(q, A)$ such that $T \cap \pi^{-1}(\Psi) \neq \emptyset$. Intuitively, $\mathcal{S}$ is A-deterministic iff fixing the propositions that are true in the next state uniquely determines the choice of the agents in A. For all A-deterministic ATS $\mathcal{S}$ and $\mathcal{S}'$, the relations A-simulation and A-trace containment coincide: $\mathcal{S} \leq_A \mathcal{S}'$ iff $\mathcal{S} \subseteq_A \mathcal{S}'$.

Checking alternating trace containment

Given two ATS $\mathcal{S}$ and $\mathcal{S}'$ and a set A of agents, the *alternating trace-containment problem* is to determine whether $\mathcal{S} \subseteq_A \mathcal{S}'$. Checking alternating trace containment requires us to consider sets of trace sets —i.e., sets of ω-languages. We first show that for every set F_A of strategies, the ω-language $\mathit{traces}_{\mathcal{S}}(F_A)$ can be represented as a tree, and consequently, the desired set $\mathcal{L}_{\mathcal{S}}(A)$ of ω-languages can be represented by a tree automaton. This leads to a reduction of the alternating trace-containment problem to the language-containment problem for tree automata.

Given a finite set Υ, an *Υ-tree* is a set $\tau \subseteq \Upsilon^*$ such that if $(x \cdot \upsilon) \in \tau$, where $x \in \Upsilon^*$ and $\upsilon \in \Upsilon$, then also $x \in \tau$. The elements of τ are called *nodes*, and the empty word ϵ is the *root* of τ. Each node x of τ has a *direction* in Υ. The direction of the root is υ_0, for some designated element $\upsilon_0 \in \Upsilon$. The direction of each node $x \cdot \upsilon$ is υ. A *path* η of the tree τ is a set $\eta \subseteq \tau$ such that $\epsilon \in \eta$ and for every $x \in \eta$, there exists a unique node $\upsilon \in \Upsilon$ with $(x \cdot \upsilon) \in \eta$. Given two finite sets Υ and Σ, a *Σ-labeled Υ-tree* is a pair $\langle \tau, \lambda \rangle$, where τ is an Υ-tree and the labeling function $\lambda : \tau \rightarrow \Sigma$ maps each node of τ to a letter in Σ.

A *language tree* is a $\{\top, \bot\}$-labeled Υ-tree $\langle \tau, \lambda \rangle$. The labeled tree $\langle \tau, \lambda \rangle$ encodes an ω-language $L(\langle \tau, \lambda \rangle) \subseteq \Upsilon^\omega$. An infinite word w is in $L(\langle \tau, \lambda \rangle)$ iff for every finite prefix x of w, the finite word x is a node of the tree τ labeled with $\top$; that is, $x \in \tau$ and $\lambda(x) = \top$. Note that the ω-language $L(\langle \tau, \lambda \rangle)$ is limit-closed (i.e., an infinite word w belongs to the language iff every finite prefix of w can be extended to some infinite word in the language). Conversely, every limit-closed ω-language over the alphabet Υ can be represented as a language tree. It follows that we can encode the trace set $trace_{\mathcal{S}}(F_A)$ that is consistent with a set F_A of strategies as a language tree.

An *alternating Büchi tree automaton* [MS87] $\mathcal{A} = \langle \Sigma, d, S, s_{in}, M, \alpha \rangle$ runs on Σ-labeled Υ-trees with $|\Upsilon| = d$, say, $\Upsilon = \{1, \ldots, d\}$. The automaton $\mathcal{A}$ consists of a finite set S of states, an initial state $s_{in} \in S$, a transition function M, and an acceptance condition $\alpha \subseteq S$. Let $\mathcal{B}^+(\Upsilon \times S)$ be the set of positive boolean formulas over $\Upsilon \times S$; that is, formulas built from elements in $\Upsilon \times S$ using $\wedge$ and $\vee$, where we also allow the formulas *true* and *false*. The transition function $M : S \times \Sigma \rightarrow \mathcal{B}^+(\Upsilon \times S)$ maps a state and an input letter to a formula that suggests a new configuration for the automaton. For example, when $d = 2$,

$$M(s_0, \sigma) = ((1, s_1) \wedge (1, s_2)) \vee ((1, s_2) \wedge (2, s_2) \wedge (2, s_3))$$

means that when the automaton is in state s_0 and reads the letter σ, it can either send two copies, in states s_1 and s_2, to direction 1 of the tree, or send a copy in state s_2 to direction 1 and two copies, in states s_2 and s_3, to direction 2. Thus, the transition function may require the automaton to send several copies to the same direction, or allow it not to send any copy to some directions.

A *run* of the alternating automaton $\mathcal{A}$ on the input Σ-labeled Υ-tree $\langle \tau, \lambda \rangle$ is a labeled tree $\langle \tau_r, r \rangle$ (without fixed branching degree) in which the root is labeled by s_{in} and every other node is labeled by an element of $\Upsilon^* \times S$. Each node of τ_r corresponds to a node of τ. A node in τ_r, labeled by (x, s), describes a copy of the automaton that reads the node x of τ and visits the state s. Note that many nodes of τ_r can correspond to the same node of τ. The labels of a node and its children have to satisfy the transition function. For example, if $\langle \tau, \lambda \rangle$ is a $\{1, 2\}$-tree with $\lambda(\epsilon) = a$ and $M(s_{in}, a) = ((1, s_1) \vee (1, s_2)) \wedge ((1, s_3) \vee (2, s_2))$, then the nodes of $\langle \tau_r, r \rangle$ at level 1 include the label $(1, s_1)$ or $(1, s_2)$, and include the label $(1, s_3)$ or $(2, s_2)$. Each (infinite) path η of $\langle \tau_r, r \rangle$ is labeled by a word $r(\eta)$ in S^ω. Let $inf(\eta)$ denote the set of states in S that appear in $r(\eta)$ infinitely often. The path η satisfies the Büchi acceptance condition α iff iff $inf(r(\eta)) \cap \alpha \neq \emptyset$. The run $\langle \tau_r, r \rangle$ is accepting iff all paths of $\langle \tau_r, r \rangle$ satisfy the acceptance condition. The automaton $\mathcal{A}$ *accepts* the labeled tree $\langle \tau, \lambda \rangle$ iff there exists an accepting run

$\langle \tau_r, r\rangle$ on $\langle \tau, \lambda\rangle$. We denote by $\mathcal{L}(\mathcal{A})$ the language of the automaton $\mathcal{A}$ —i.e., the set of labeled trees that are accepted by $\mathcal{A}$.

Theorem 5. *The alternating trace-containment problem is EXPTIME-complete.*

Proof. We start with the upper bound. Consider an ATS $\mathcal{S} = \langle \Pi, \Omega, Q, q_{in}, \pi, \delta\rangle$. For every observation $\ell \in 2^{\Pi}$, let $Q^{\ell} \subseteq Q$ be the set of states q with $\pi(q) = \ell$; that is, $Q^{\ell} = \pi^{-1}(\ell)$. Given $\mathcal{S}$ and a set $A \subseteq \Omega$ of agents, we define an alternating Büchi tree automaton $\mathcal{A}^A_{\mathcal{S}} = \langle \Sigma, d, S, s_{in}, M, \alpha\rangle$, where

- $\Sigma = \{\top, \bot\}$.
- $d = 2^{|\Pi|}$, and we assume $\Upsilon = 2^{\Pi}$; that is, the directions are observations of $\mathcal{S}$.
- $S = \alpha = Q$.
- $s_{in} = q_{in}$.
- The transition function M is defined only for the input letter $\top$, and for every state $q \in Q$ we have:

$$M(q, \top) = \bigvee_{P \in \delta(q,A)} \bigwedge_{\ell \in \Upsilon} \bigwedge_{q' \in P \cap Q^{\ell}} (\ell, q').$$

  That is, from the state q, the automaton chooses a set $P \in \delta(q, A)$ and then sends to every direction ℓ the states in P that are labeled by ℓ. Note that the automaton may send several states to the same direction and may also send no states to some directions.

Consider an accepting run $\langle \tau_r, r\rangle$ of $\mathcal{A}^A_{\mathcal{S}}$. Recall that the root of τ_r is labeled by q_{in} and every other node is labeled by an element of $\Upsilon^* \times Q$. For every set F_A of strategies for the agents in A, and every sequence $\rho \in \Upsilon^*$ of observations, we can define a set $\mathit{force}(F_A, \rho) \subseteq Q$ of states such that the strategies in F_A force $\mathcal{S}$ to one of the states in $\mathit{force}(F_A, \rho)$ after traversing a prefix of a computation that is labeled ρ. Intuitively, the run $\langle T_t, r\rangle$ of $\mathcal{A}^A_{\mathcal{S}}$ corresponds to a set F_A of strategies in which for every sequence of observations $\rho \in \Upsilon^*$, $\mathit{force}(F_A, \rho)$ is exactly the set of states that r visits as it reads the node ρ. Formally,

$$\mathit{force}(F_A, \rho) = \{q \in Q : \text{ there exists } y \in \tau_r \text{ with } r(y) = \langle \rho, q\rangle\}.$$

This relation between strategies for the agents in A and accepting runs of $\mathcal{A}^A_{\mathcal{S}}$ enables us to reduce the alternating trace-containment problem to the language containment problem $\mathcal{L}(\mathcal{A}^A_{\mathcal{S}}) \subseteq \mathcal{L}(\mathcal{A}^A_{\mathcal{S}'})$. Formally, we claim that $\mathcal{L}(\mathcal{A}^A_{\mathcal{S}}) \subseteq \mathcal{L}(\mathcal{A}^A_{\mathcal{S}'})$ iff for every set F_A of strategies there exists a set F'_A of strategies such that $\mathit{traces}_{\mathcal{S}'}(F'_A) \subseteq \mathit{traces}_{\mathcal{S}}(F_A)$. Since the size of the automata $\mathcal{A}^A_{\mathcal{S}}$ and $\mathcal{A}^A_{\mathcal{S}'}$ is linear in $\mathcal{S}$ and $\mathcal{S}'$, respectively, and the language containment problem for alternating Büchi tree automata can be solved in exponential time [VW86, MS95], the EXPTIME upper bound follows.

For the lower bound, we use a reduction from $\{\mathit{sys}\}$-LTL model checking (for the formal definition of $\{\mathit{sys}\}$-LTL see Section 5). A closed system can be viewed as an ATS $\mathcal{T}$ with the two agents *sys* and *env* in which the agent *env* is powerless; that is, for all states q of $\mathcal{T}$, we have $\delta(q, env) = \{Q\}$ (in Section 2, we noted that a closed system corresponds to an ATS with the single agent *sys*; here we add the agent *env* in order to compare $\mathcal{T}$ with a system in which the environment is not powerless). Given an LTL formula ψ, one can construct, following [VW94], a two-agent ATS $\mathcal{T}_\psi$, as above, such that $\mathcal{T}_\psi$ has exactly the traces that satisfy ψ (for this purpose, the ATS $\mathcal{T}_\psi$ is augmented with Büchi fairness constraints; the proof generalizes to ATS without fairness as in the nonalternating case). The size of $\mathcal{T}_\psi$ is exponential in ψ. Then, the model-checking problem $\mathcal{S} \models \langle\!\langle env \rangle\!\rangle \psi$ can be reduced to the alternating trace-containment problem $\mathcal{T}_\psi \subseteq_{env} \mathcal{S}$. Since the model-checking problem for A-LTL is 2EXPTIME-complete [AHK97], the EXPTIME lower bound follows. $\square$

5 Logical Characterizations

Simulation and trace containment between labeled transition systems can be logically characterized by temporal logics. We give logical characterizations of alternating simulation and alternating trace containment.

Alternating temporal logic

Alternating-time temporal logics are introduced in [AHK97] as a formalism for specifying properties of individual system components. The alternating-time temporal logic ATL* is defined with respect to a finite set Π of *propositions* and a finite set Ω of *agents*. Its syntax is very similar to the syntax of CTL*, only that each occurrence of a path quantifier is parameterized by a set of agents. There are two types of formulas in ATL*: *state formulas*, whose satisfaction is related to a specific state, and *path formulas*, whose satisfaction is related to a specific computation. We present here a subset of ATL* formulas in positive normal form. The subset, which subsumes CTL* but is not closed under negation, is called ATL$^\star_P$. An ATL$^\star_P$ state formula is one of the following:

(S1) p or $\neg p$, for propositions $p \in \Pi$.
(S2) $\varphi_1 \vee \varphi_2$ or $\varphi_1 \wedge \varphi_2$, where φ_1 and φ_2 are ATL$^\star_P$ state formulas.
(S3) $\langle\!\langle A \rangle\!\rangle \psi$ where $A \subseteq \Omega$ is a set of agents and ψ is an ATL$^\star_P$ path formula.

The operator $\langle\!\langle\ \rangle\!\rangle$ is a *path quantifier*. For a singleton set $A = \{a\}$ of agents, we write $\langle\!\langle a \rangle\!\rangle$ instead of $\langle\!\langle \{a\} \rangle\!\rangle$. An ATL$^\star_P$ path formula is one of the following:

(P1) An ATL$^\star_P$ state formula.
(P2) $\psi_1 \vee \psi_2$ or $\psi_1 \wedge \psi_2$, where ψ_1 and ψ_2 are ATL$^\star_P$ path formulas.
(P3) $\bigcirc \psi_1$, $\Box \psi_1$, or $\psi_1 \mathcal{U} \psi_2$, where ψ_1 and ψ_2 are ATL$^\star_P$ path formulas.

The logic ATL$^\star_P$ consists of the set of state formulas generated by the above rules. The logic ATL$_P$, which is an alternating-time extension of CTL, is the fragment of ATL$^\star_P$ that consists of all formulas in which every temporal operator is immediately preceded by a path quantifier.

We interpret the ATL$^\star_P$ formulas over ATS (with the same sets of propositions and agents). Consider a fixed ATS $\mathcal{S} = \langle \Pi, \Omega, Q, q_{in}, \pi, \delta \rangle$. We write $q \models \varphi$ to indicate that the state formula φ holds at state q, and $\eta \models \psi$ to indicate that the path formula ψ holds in computation η. The satisfaction relation $\models$ is defined as for the branching-time logic CTL*, with the path quantifier $\langle\!\langle\ \rangle\!\rangle$ interpreted as follows:

> $q \models \langle\!\langle A \rangle\!\rangle \psi$ iff there exists a set F_A of strategies, one for each agent in A, such that for all computations $\eta \in out_{\mathcal{S}}(q, F_A)$, we have $\eta \models \psi$.

For example, the ATL$^\star_P$ formula $\langle\!\langle a \rangle\!\rangle((\Diamond\Box req) \vee (\Box\Diamond grant))$ asserts that agent a has a strategy to enforce a computation in which either only finitely many requests are sent, or infinitely many grants are given. The ATS $\mathcal{S}$ satisfies the ATL$^\star_P$ formula φ iff $q_{in} \models \varphi$.

Recall that a labeled transition system is an ATS with the single agent *sys*. In this case, there are only two path quantifiers: $\langle\!\langle \emptyset \rangle\!\rangle$ and $\langle\!\langle sys \rangle\!\rangle$, which are equal, respectively, to the universal and existential path quantifiers $\exists$ and $\forall$ of branching-time logics. In other words, over labeled transition systems, ATL$^\star_P$ is identical to CTL*, and ATL$_P$ is identical to CTL.

Logical characterization of alternating simulation

Simulation in labeled transition systems guarantees correct implementation with respect to properties specified in the universal fragment of a branching-time logic. For a set A of agents, we define the fragment A-ATL* of ATL$^\star_P$ as the set of formulas in which all path quantifiers are parameterized by A. In particular, over labeled transition systems, $\emptyset$-ATL* and $\{sys\}$-ATL* coincide with the universal and existential fragments of CTL*. Over ATS, the A-ATL* formulas describe behaviors that the agents in A can enforce no matter what the agents in $\Omega \setminus A$ do.

Theorem 6. *Let $\mathcal{S}$ and $\mathcal{S}'$ be two ATS. Then, for every set A of agents, $\mathcal{S} \leq_A \mathcal{S}'$ iff every A-ATL* formula that is satisfied in $\mathcal{S}$ is also satisfied in $\mathcal{S}'$.*

Proof. We first prove that if $\mathcal{S} \leq_A \mathcal{S}'$, then every A-ATL* formula that is satisfied in $\mathcal{S}$ is also satisfied in $\mathcal{S}'$. For that, we prove a stronger claim. We prove that for all ATS $\mathcal{S}$ and $\mathcal{S}'$, if H is an A-simulation from $\mathcal{S}$ to $\mathcal{S}'$, then the following conditions hold:

- For all states q and q' with $H(q, q')$, every A-ATL* state formula that holds at q holds also at q'.
- For all computations η and η' with $H(\eta, \eta')$, every A-ATL* path formula that holds in η holds also in η'.

The proof proceeds by induction on the structure of formulas. The interesting case is that of formulas of the form $\langle\!\langle A\rangle\!\rangle\psi$. Assume that $H(q,q')$ and $q \models \psi$. Then, there exists a set F_A of strategies in $\mathcal{S}$ for the agents in A such that for all computations $\eta \in out_{\mathcal{S}}(q, F_A)$, we have $\eta \models \psi$. Consider a set F'_A of strategies in $\mathcal{S}'$ for the agents in A such that for every computation $\eta' \in out_{\mathcal{S}}(q', F'_A)$, there exists a computation $\eta \in out_{\mathcal{S}}(q, F_A)$ so that $H(\eta, \eta')$. By Lemma 1, such a set F'_A exists. Since ψ holds in all computations $\eta \in out_{\mathcal{S}}(q, F_A)$, by the induction hypothesis, we are done.

Second, assume that $\mathcal{S} \not\leq_A \mathcal{S}'$. Consider the A-simulation game between the antagonist and the protagonist. Since $\mathcal{S} \not\leq_A \mathcal{S}'$, the antagonist has a winning strategy in the game. Thus, there exists a finite number i such that every strategy of the protagonist fails to match the antagonist's choice in the i-th round of the game or before, when the antagonist follows the winning strategy. Similar to the case of Milner's simulation [Mil90], it is then possible to construct a formula with i nested $\langle\!\langle A\rangle\!\rangle\bigcirc$ operators that is satisfied in $\mathcal{S}$ but not in $\mathcal{S}'$. □

Logical characterization of alternating trace containment

Trace containment in labeled transition systems guarantees correct implementation with respect to properties specified in linear-time logics. For a set A of agents, we define the fragment A-LTL of ATL$^\star_P$ as the set of formulas of the form $\langle\!\langle A\rangle\!\rangle\psi$, where ψ is an ATL$^\star_P$ path formula without path quantifiers; that is, ψ is an LTL formula. In particular, over labeled transition systems, $\emptyset$-LTL coincides with LTL.

Theorem 7. *Let $\mathcal{S}$ and $\mathcal{S}'$ be two ATS. Then, for every set A of agents, $\mathcal{S} \subseteq_A \mathcal{S}'$ iff every A-LTL formula that is satisfied in $\mathcal{S}$ is also satisfied in $\mathcal{S}'$.*

Proof. Assume first that $\mathcal{S} \subseteq_A \mathcal{S}'$. Let $\langle\!\langle A\rangle\!\rangle\psi$ be an A-LTL formula that is satisfied in $\mathcal{S}$. Thus, there exists a set F_A of strategies in $\mathcal{S}$ for the agents in A such that for all computations $\eta \in out_{\mathcal{S}}(q_{in}, F_A)$, we have $\eta \models \psi$. Since $\mathcal{S} \subseteq_A \mathcal{S}'$, there exists a set F'_A such that $trace_{\mathcal{S}'}(F'_A) \subseteq trace_{\mathcal{S}}(F_A)$. Hence, for all computations $\eta' \in out_{\mathcal{S}'}(q'_{in}, F'_A)$, we have $\eta \models \psi$. Assume now that $\mathcal{S} \not\subseteq_A \mathcal{S}'$. Then, as in the case of labeled transition systems, it is possible to construct, using $\bigcirc$ operators, an A-LTL formula that is satisfied in $\mathcal{S}$ but not in $\mathcal{S}'$. □

Alternating bisimilarity

In analogy to the nonalternating case, we say that a symmetric alternating simulation is an alternating bisimulation. Consider two ATS $\mathcal{S} = \langle \Pi, \Omega, Q, q_{in}, \pi, \delta\rangle$ and $\mathcal{S}' = \langle \Pi, \Omega, Q', q'_{in}, \pi', \delta'\rangle$. For a set A of agents, a relation $H \subseteq Q \times Q'$ is an A-*bisimulation* iff for all states q and q' with $H(q,q')$ the following conditions hold:

(1) $\pi(q) = \pi'(q')$.

(2) For every set $T \in \delta(q, A)$, there exists a set $T' \in \delta'(q', A)$ such that for every $R' \in \delta'(q', \Omega \setminus A)$, there exists $R \in \delta(q, \Omega \setminus A)$ so that $(T \cap R) \times (T' \cap R') \subseteq H$.

(3) For every set $T' \in \delta'(q', A)$, there exists a set $T \in \delta(q, A)$ such that for every $R \in \delta(q, \Omega \setminus A)$, there exists $R' \in \delta(q', \Omega \setminus A)$ so that $(T \cap R) \times (T' \cap R') \subseteq H$.

If there exists an A-bisimulation H from $\mathcal{S}$ to $\mathcal{S}'$ with $H(q_{in}, q'_{in})$, we say that the ATS $\mathcal{S}$ and $\mathcal{S}'$ are *A-bisimilar*, denoted $\mathcal{S} \equiv_A \mathcal{S}'$. Intuitively, $\mathcal{S} \equiv_A \mathcal{S}'$ means that the agents in A can induce the same behaviors in both $\mathcal{S}$ and $\mathcal{S}'$.

It is easy to check that A-bisimilarity is an equivalence relation on ATS. Furthermore, as in the nonalternating case, $\mathcal{S} \equiv_A \mathcal{S}'$ implies both $\mathcal{S} \leq_A \mathcal{S}'$ and $\mathcal{S}' \leq_A \mathcal{S}$, yet the converse is does not hold. Thus, alternating bisimulation is stronger than mutual alternating simulation.

Given two ATS $\mathcal{S}$ and $\mathcal{S}'$ and a set A of agents, the *alternating-bisimilarity problem* is to determine whether $\mathcal{S} \equiv_A \mathcal{S}'$. For two sets A_1 and A_2 of agents, we define the fragment $\langle A_1, A_2 \rangle$-ATL* of ATL$^\star_P$ as the set of formulas in which all path quantifiers are parameterized by either A_1 or A_2. Using techniques similar to alternating simulation, we establish the following two theorems.

Theorem 8. *The alternating-bisimilarity problem is PTIME-complete.*

Theorem 9. *Let $\mathcal{S}$ and $\mathcal{S}'$ be two ATS. Then, for every set A of agents, $\mathcal{S} \equiv_A \mathcal{S}'$ iff $\mathcal{S}$ and $\mathcal{S}'$ agree on all $\langle A, \Omega \setminus A \rangle$-ATL* formulas.*

Acknowledgments. We thank Shaz Qadeer and Sriram Rajamani for helpful discussions.

References

[AHK97] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proc. 38th Symp. on Foundations of Computer Science*, pp. 100–109. IEEE Computer Society, 1997. Full version in *Compositionality–The Significant Difference*. Springer-Verlag Lecture Notes in Computer Science, 1998.

[BGS92] J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4:638–648, 1992.

[HF89] J.Y. Halpern and R. Fagin. Modeling knowledge and action in distributed systems. *Distributed Computing*, 3:159–179, 1989.

[HHK95] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. 36rd Symp. on Foundations of Computer Science*, pp. 453–462. IEEE Computer Society, 1995.

[Imm81] N. Immerman. Number of quantifiers is better than number of tape cells. *J. Computer and System Sciences*, 22:384–406, 1981.

[KV98] O. Kupferman and M.Y. Vardi. Verification of fair transition systems. *Chicago J. Theoretical Computer Science*, 1998(2).

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *Proc. 2nd Int. Joint Conf. on Artificial Intelligence*, pp. 481–489. British Computer Society, 1971.

[Mil90] R. Milner. Operational and algebraic semantics of concurrent processes. In *Handbook of Theoretical Computer Science*, Vol. B, pp. 1201–1242. Elsevier, 1990.

[MS87] D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[MS95] D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata: new results and new proofs of theorems of Rabin, McNaughton, and Safra. *Theoretical Computer Science*, 141:69–107, 1995.

[Sha53] L.S. Shapley. Stochastic games. In *Proc. National Academy of Science*, 39:1095–1100, 1953.

[VW86] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. Computer and System Sciences*, 32:182–221, 1986.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.