No big deal: introducing roles to reduce the size of ATL models

Sjur Dyrkolbotn[1], Piotr Kaźmierczak[2,3]*, Erik Parmann[1], and Truls Pedersen[3]

[1] Department of Informatics, University of Bergen
[2] Department of Computing, Mathematics and Physics, Bergen University College
[3] Department of Information Science and Media Studies, University of Bergen sjur.dyrkolbotn@uib.no, phk@hib.no, erik.parmann@uib.no, truls.pedersen@uib.no

Abstract. In the following paper we present a new semantics for the well-known strategic logic ATL. It is based on adding *roles* to concurrent game structures, that is at every state, each agent belongs to exactly one role, and the role specifies what actions are available to him at that state. We show advantages of the new semantics, analyze model checking complexity and prove equivalence between standard ATL semantics and our new approach.

1 Introduction

One of the most intensively studied [5, 6, 8] areas of research in the field of multi-agent systems are *strategic* or *cooperation* logics – formalisms that allow for reasoning about agents' strategies and behavior in a multi-agent setting. Perhaps the most known such logic is Alur, Henzinger and Kupferman's Alternating-time Temporal Logic (ATL) [2].

The popularity of ATL is in no small part due to its relatively high expressive power, but also due to low complexity of model checking problem for this logic. Model checking of ATL can be solved in polynomial time in the size of the model and the length of the formula [2], which is a very good result. However, as investigated by Jamroga and Dix [4], in both cases the number of agents must be *fixed*. If it is not then model checking of ATL models represented as *alternating transition systems* is NP-complete, and if the models are represented as *concurrent game structures* (CGS) it becomes Σ_2^{P}-complete. Also, van der Hoek, Lomuscio and Wooldridge show [7] that complexity of model checking for ATL is sensitive to model representation. It is polynomial only if an *explicit* enumeration of *all* components of the model is assumed. For models represented in a *(simplified) reactive modules language* (RML) [1] complexity of model checking for ATL becomes as hard as the satisfiability problem for this logic, namely EXPTIME [7].

We present an alternative semantics that interprets formulas of ordinary ATL over concurrent game structures with *roles*. As we describe in Section 2.1, such structures introduce an extra element – a set R of roles. Agents belonging to the same role are considered *homogeneous* in the sense that all consequences of their actions are captured by considering only the number of *votes* an action gets (one vote per agent). We give an example that motivates our approach and prove equivalence with ATL based on concurrent game structures. We then discuss model checking, showing it to be of polynomial complexity in the size of models. This seems significant, since as long as the number of roles remain fixed, the size of our models does *not* grow exponentially in the number of players.

2 Role-based semantics for ATL

The language of ordinary ATL is the following, as presented in [2]:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle A\rangle\rangle \bigcirc \phi \;\mid\; \langle\langle A\rangle\rangle \Box\phi \;\mid \langle\langle A\rangle\rangle \phi \mathcal{U} \phi$$

where p is propositional letter, and A is a coalition of agents. We follow standard abbreviations (e.g. $\langle\!\langle\ \rangle\!\rangle$ for $\langle\langle\emptyset\rangle\rangle$) and skip connectives that are derivable.

* Piotr Kaźmierczak's research was supported by the Research Council of Norway project 194521 (FORMGRID).

2.1 Concurrent Game Structures with Roles

In this section we will introduce *concurrent game structures with roles* (RCGS) and consider an example. We will be using the notation $[n] = \{1, \ldots, n\}$, and we will let A^B denote the set of functions from B to A. We will often work with tuples $v = \langle v_1, \ldots, v_n \rangle$ and we will often view v as a function with domain $[n]$ and write $v(i)$ for v_i. We will do addition and subtraction on tuples of the same arity component by component, e.g. for $v = \langle v_1, \ldots, v_n \rangle, v' = \langle v'_1, \ldots, v'_n \rangle$, $v - v' = \langle v_1 - v'_1, \ldots, v_n - v'_n \rangle$. Given a function $f : A \times B \to C$ and $a \in A$, we will use f_a to denote the function $B \to C$ defined by $f_a(b) = f(a, b)$ for all $b \in B$.

Definition 2.1. *An* RCGS *is a tuple* $H = \langle \mathcal{A}, R, \mathcal{R}, Q, \Pi, \pi, \mathbb{A}, \delta \rangle$ *where:*

- $\mathcal{A}$ *is a non-empty set of players. In this text we assume* $\mathcal{A} = [n]$ *for some* $n \in \mathbb{N}$*, and we will reserve* n *to mean the number of agents.*
- Q *is the non-empty set of states.*
- R *is a non-empty set of roles. In this text we assume* $R = [i]$ *for some* $i \in \mathbb{N}$*.*
- $\mathcal{R} : Q \times R \to \wp(\mathcal{A})$*, such that for every* $q \in Q$ *we have*
  - *For all* $r, r' \in R$*, if* $r \neq r'$ *then* $\mathcal{R}(q, r) \cap \mathcal{R}(q, r') = \emptyset$
  - $\bigcup_{r \in R} \mathcal{R}(q, r) = \mathcal{A}$

  For a coalition $A \subseteq \mathcal{A}$ *we write* $A_{r,q}$ *for the agents in* A *which belong to role* r *at* q*, i.e.* $A_{r,q} = \mathcal{R}(q, r) \cap A$*.*
- Π *is a set of propositional letters and* $\pi : Q \to \wp(\Pi)$ *maps states to the propositions true at that state.*
- $\mathbb{A} : Q \times R \to \mathbb{N}^+$ *is the number of available actions in a given state for a given role.*
- *For* $A = [n] = \{1, \ldots n\}$*, we say that the set of* complete *votes for a role* r *in a state* q *is* $V_r(q) = \{v_{r,q} \in [n]^{[\mathbb{A}(q,r)]} \mid \sum_{1 \leq a \leq \mathbb{A}(q,r)} v_{r,q}(a) = |\mathcal{R}(q, r)|\}$*. The set of* complete *profiles at* q *is* $P(q) = \prod_{r \in R} V_r(q)$*. For each* $q \in Q$ *we have a transition function at* q*,* $\delta_q : P(q) \to Q$ *defining a partial function* $\delta : Q \times \bigcup_{q \in Q} P(q) \to Q$ *such that for all* $q \in Q$*,* $P \in P(q)$*,* $\delta(q, P) = \delta_q(P)$

To illustrate how RCGS differs from an ordinary concurrent game structure, we provide an example.

Example 2.1. We construct an example similar to the well-known train-controller scenario [2], but in contrast to the original, in our scenario there are n_t trains. Consider a turn-based synchronous game structure with roles $S_{train} = \langle \mathcal{A},\ R, \mathcal{R}, Q, \Pi, \pi, \mathbb{A}, \delta \rangle$ where:

- $\mathcal{A} = \{1, \ldots, n_t, n_t + 1\}$. There are n_t trains and one controller.
- $R = \{train, ctr\}$. There are two roles: one for trains and one for the controller.
- $Q = \{q_0, q_1, q_2, q_3\}$.
- $\mathcal{R}(q_i, train) = n_t$, and $\mathcal{R}(q_i, ctr)$, for all $q_i \in Q$.
- $\Pi = \{out\_of\_gate, in\_gate, request, grant\}$
- $\pi(q_0) = \{out\_of\_gate\}$, $\pi(q_1) = \{out\_of\_gate, request\}$, $\pi(q_2) = \{out\_of\_gate, grant\}$, $\pi(q_3) = \{in\_gate\}$.
- $\mathbb{A}(q_0, train) = 2, \quad \mathbb{A}(q_0, ctr) = 1, \quad \mathbb{A}(q_1, train) = 1, \quad \mathbb{A}(q_1, ctr) = 3,$ $\mathbb{A}(q_2, train) = 2, \quad \mathbb{A}(q_2, ctr) = 1, \quad \mathbb{A}(q_3, train) = 1, \quad \mathbb{A}(q_3, ctr) = 2.$
- and finally

$$\begin{aligned}
&\delta(q_0, \langle (0, n_t), 1 \rangle) = \delta(q_1, \langle n_t, (1, 0, 0) \rangle) = \delta(q_2, \langle (0, n_t), 1 \rangle) \\
&\qquad = \delta(q_3, \langle (a, n_t - a), 1 \rangle) = q_0 && \text{where } 1 \leq a \leq n \\
&\delta(q_0, \langle (a, n_t - a), 1 \rangle) = \delta(q_1, \langle n_t, ((0, 1, 0)) \rangle) = q_1 && \text{where } 1 \leq a \leq n \\
&\delta(q_1, \langle n_t, (0, 0, 1) \rangle) = \delta(q_2, \langle (a, n_t - a), 1 \rangle) = q_2 && \text{where } 2 \leq a \leq n \\
&\delta(q_2, \langle (1, n_t - 1), 1 \rangle) = \delta(q_3, \langle (0, n_t), 1 \rangle) = q_3
\end{aligned}$$

Figure 1 presents the example in a visual way. The model can be seen as a generalization of the classical train-controller example. In q_0 we stay in q_0 unless at least one train issues a request. In q_1 the controller behaves as before; it can postpone making a decision (staying in q_1), reject all requests (going to q_0), or accept the requests (going to q_2). In q_2 the trains can choose to enter the tunnel, but only one of them may do so; if nobody attempts to enter the grant is revoked (or relinquished), if more than one train attempts to enter we stay in q_2, and finally if (the trains reach an agreement and) only one train enters we go to q_3. In q_3 *any* train may decide that the train in the tunnel has to leave (returning to q_0), and the train in the tunnel *must* comply. This reflects the homogeneity among players in the trains role. The action of deciding to leave the tunnel is shared among all trains, and the train actually in the tunnel remains unidentified.

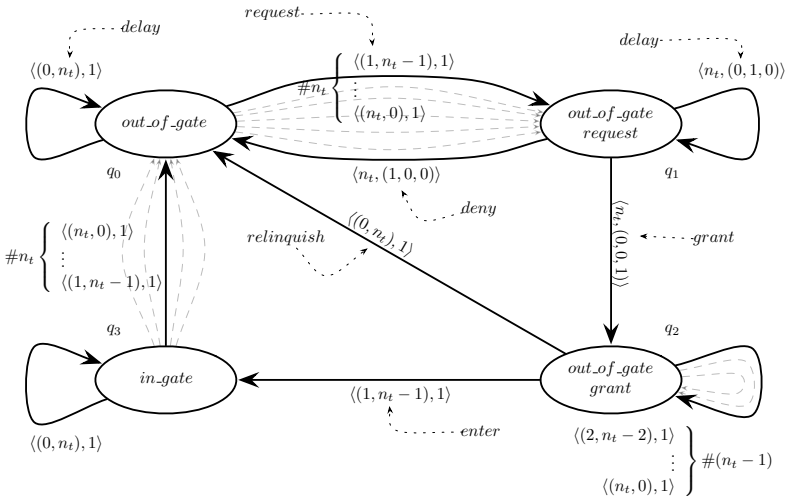

Fig. 1. Train controller model for n_t trains (similar to the one presented in [2]).

Notice that in the single-train case ($n_t = 1$), the train can not wait before entering the tunnel after being granted permission (*and* retain the permission). This could of course easily be avoided by adding another action. More importantly, in the case of several trains, the controller can not distinguish between the different trains, so permission must be granted to all or none. This is a consequence of the strict homogeneity in the model: not only are the agents homogeneous in terms of the actions available to them, we can not reasonably distinguish between them as long as they remain in the same role. Notice that this feature allow us to add any number of trains to the scenario without incurring more than a linear increase in the size of the model (total number of profiles). This would not be possible if we did not have roles. If the model above was to be rendered as a concurrent game structure, the number of possible ways in which trains could act would be exponential in all states where trains have to make a choice of what action to perform. This would be the case even if, as in the scenario above, almost all possible combinations of choices should be treated in the same way by the system.

Before we move on we introduce some more notation. Given a role $r \in R$, a state q and a coalition A, the set of A-votes for r at q is $V_r(q, A)$, defined as follows: $V_r(q, A) = \left\{ v \in [|A_{r,q}|]^{[\mathbb{A}(q,r)]} \;\middle|\; \sum_{a \in [\mathbb{A}(q,r)]} v(a) = |A_{r,q}| \right\}$ The A-votes for r at q gives the possible ways agents in A that are in role r at q can vote. Given a state q and a coalition A, we define the set of A-profiles at q: $P(q, A) = \{\langle v_1, \ldots, v_{|R|}\rangle \mid 1 \leq i \leq |R| : v_i \in V_r(q, A)\}$ When we say that a function $v : [\mathbb{A}(q, r)] \to [n]$ is a complete vote (for r at q), we mean that $v \in V_r(q, \mathcal{A})$. For any $v \in V_r(q, A)$ and $w \in V_r(q, B)$ we write $v \leq w$ iff for all $i \in [\mathbb{A}(q, r)]$ we have $v(i) \leq w(i)$. If $v \leq w$, we say that w *extends* v. If $F = \langle v_1, \ldots, v_R\rangle \in P(q, A)$ and $F' = \langle v'_1, \ldots, v'_R\rangle \in P(q, B)$ with $v_i \leq v'_i$ for every $1 \leq i \leq |R|$, we say that $F \leq F'$ and that F extends F'.

An A profile $F \in P(q, A)$ is a *complete* profile iff the sum of its components equal $|\mathcal{A}|$, i.e. $F \in P(q)$ iff $\left(\sum_{r \leq |R|} \sum_{a \in \mathbb{A}(q,r)} v(a)\right) = |\mathcal{A}|$ iff $A = \mathcal{A}$. Given a (partial) profile F' at a state q we write $ext(q, F)$ for the set of all complete profiles that extend F'.

Given two states $q, q' \in Q$, we say that q' is a *successor* of q if there is some $F \in P(q)$ such that $\delta(q, F) = q'$. A *computation* is an infinite sequence $\lambda = q_0 q_1 \ldots$ of states such that for all positions $i \geq 0$, q_{i+1} is a successor of q_i. We follow the standard abbreviations, hence q-computation denotes a computation starting at q, and $\lambda[i]$, $\lambda[0, i]$ and $\lambda[i, \infty]$ denote the i-th state, the finite prefix $q_0 q_1 \ldots q_i$ and the infinite suffix $q_i q_{i+1} \ldots$ of λ for any computation λ and its position $i \geq 0$. An *A-strategy* for $A \subseteq \mathcal{A}$ is a function $s_A : Q \to \bigcup_{q \in Q} P(q, A)$ such that $s_A(q) \in P(q, A)$ for all $q \in Q$. That is, s_A maps states to A-profiles at that state. The set of all A-strategies is denoted by $strat(A)$. If s is an $\mathcal{A}$-strategy and we apply δ_q to $s(q)$, we obtain a unique new state $q' = \delta_q(s(q))$. Iterating, we get the *induced* computation $\lambda_{s,q} = q_0 q_1 \ldots$ such that $q = q_0$ and $\forall i \geq 0 : \delta_{q_i}(s(q_i)) = q_{i+1}$. Given two strategies s and s', we say that $s \leq s'$ iff $\forall q \in Q : s(q) \leq s'(q)$. Given an A-strategy s_A and a state q we get an associated *set* of computations $out(s_A, q)$. This is the set of all computations that can result when at any state, the players in A are voting in the way specified by s_A:
$out(s_A, q) = \{\lambda_{s,q} \mid s \text{ is an } \mathcal{A}\text{-strategy and } s \geq s_A\}$ It will also be useful to have access to the set of states that can result in the next step when $A \subseteq \mathcal{A}$ follows strategy s_A at state q, $succ(q, s_A) = \{q' \in Q \mid \exists F \in ext(q, s_A) : \delta(q, F) = q'\}$. Clearly, $q' \in succ(q, s_A)$ iff there is some $\lambda \in out(q, s_A)$ such that $q' = \lambda[0]$.

2.2 New Semantics for ATL

Definition 2.2. *Given a* RCGS S *and a state* q *in* S*, we define the satisfaction relation* $\models$ *inductively:*

$S, q \models p$ *iff* $p \in \pi(q)$

$S, q \models \phi \wedge \phi'$ *iff* $S, q \models \phi$ *and* $S, q \models \phi'$

$S, q \models \neg\phi$ *iff not* $S, q \models \phi$

$S, q \models \langle\langle A \rangle\rangle \bigcirc \phi$ *iff there is* $s_A \in strat(A)$ *such that for all* $\lambda \in out(s_A, q)$*, we have* $S, \lambda[1] \models \phi$

$S, q \models \langle\langle A \rangle\rangle \Box \phi$ *iff there is* $s_A \in strat(A)$ *such that for all* $\lambda \in out(s_A, q)$ *we have* $S, \lambda[i] \models \phi$ *for all* $i \geq 0$

$S, q \models \langle\langle A \rangle\rangle \phi \mathcal{U} \phi'$ *iff there is* $s_A \in strat(A)$ *such that for all* $\lambda \in out(s_A, q)$ *we have* $S, \lambda[i] \models \phi'$ *and* $S, \lambda[j] \models \phi$ *for some* $i \geq 0$ *and for all* $0 \leq j < i$

3 Equivalence between RCGS and CGS

In this section we show that definition 2.2 provides an equivalent semantics for ATL. We do this by first giving a surjective function f that takes an RCGS and returns a CGS. Then we show that S and $f(S)$ satisfy the same ATL formulas.

Remember that a concurrent game structure is a tuple $\langle \mathcal{A}, Q, \Pi, \pi, d, \delta' \rangle$ where every element is defined as for an RCGS except $d : \mathcal{A} \times Q \to \mathbb{N}^+$ that maps agents and states to actions available at that state, and δ' that is a partial function from states and action tuples to states defined by $\delta'(q, t) = \delta'_q(t)$ where $\delta'_q : \prod_{a \in \mathcal{A}} [d_a(q)] \to Q$ is a transition function at q based on tuples of actions rather than profiles. The satisfaction relation for ATL based on CGSs can be defined exactly as in definition 2.2, the difference concerning only what counts as a strategy.

We refer to elements of $\prod_{a \in \mathcal{A}} [d_a(q)]$ as *complete* action tuples at q. A (memory-less) strategy for $a \in \mathcal{A}$ in a CGS M is a function $s_a : Q \to \mathbb{N}^+$ such that for all $q \in Q$, $s_a(q) \in [d_a(q)]$ while a strategy for $A \subseteq \mathcal{A}$ is a list of strategies for all agents in A, $s_A = \langle s_{a_1}, s_{a_2}, \ldots, s_{a_{|A|}} \rangle$, for $A = \{a_1, a_2, \ldots, a_{|A|}\}$. We denote the set of strategies for $A \subseteq \mathcal{A}$ by $strat(A)$. When needed to distinguish between different structures we write $strat(S, A)$ to indicate that we are talking about the set of strategies for A in S.

We say that a complete action tuple at q, $t = \langle i_{a_1}, \ldots, i_{a_n} \rangle$ extends a strategy $s_A \in strat(A)$ if for all $a_j \in A$ we have $i_{a_j} = s_{a_j}(q)$. We denote the set of all complete action tuples at q extending s_A by $ext(q, s_A)$.

For any state $q \in Q$ we have the set of all computations that comply with s_A:

$$\begin{aligned} out(q, s_A) = &\{\lambda = q_0 q_1 q_2 \dots \\ &\mid q = q_0 \text{ and for all } i \in \mathbb{N} : \exists t \in ext(q_i, s_A),\ \delta(q, t) = q_{i+1}\} \end{aligned}$$

We define the set of s_A-successors at $q \in Q$: $succ(q, s_A) = \{q' \in Q \mid \exists t \in ext(q, s_A),\ \delta(q, t) = q'\}$. When we need to make clear which structure we are talking about, we write $succ(S, q, s_A)$. Observe that $q' \in succ(q, s_A)$ iff $q' = \lambda[1]$ for some $\lambda \in out(q, s_A)$.

The translation function f from RCGS to CGS is defined as follows:

$$f\langle \mathcal{A}, R, \mathcal{R}, Q, \Pi, \pi, \mathbb{A}, \delta \rangle = \langle \mathcal{A}, Q, \Pi, \pi, d, \delta' \rangle, \text{ where:}$$

$$\begin{aligned} d_a(q) &= \mathbb{A}(q, r) && \text{where } a \in \mathcal{R}(q, r) \\ \delta'(q, \alpha_1, \dots, \alpha_n) &= \delta(q, v_1, \dots, v_{|R|}) && \text{where for each role } r \\ v_r &= \langle |\{i \in \mathcal{R}(q, r) \mid \alpha_i = 1\}|, \dots, |\{i \in \mathcal{R}(q, r) \mid \alpha_i = \mathbb{A}(q, r)\}| \rangle \end{aligned}$$

We can see straight away that f is surjective because for any CGS S' with n agents we could define a RCGS S with that many roles where each role contains exactly one agent. A vote for a role r, v_r, at q would then simply be a $d_a(q)$-tuple consisting of a single 1 (representing the agents chosen action) and otherwise zeros. It is easy to verify that $f(S) = S'$.

Given either a CGS or an RCGS S, we define the set of sets of states that a coalition A can *enforce* in the next state of the game: $force(S, q, A) = \{succ(q, s_A) \mid s_A \text{ is a strategy for } A \text{ in } S\}$. The first thing we do towards showing equivalence is to describe a surjective function $m : strat(f(S)) \to strat(S)$ mapping action tuples and strategies of $f(S)$ to profiles and strategies of S respectively. For all $A \subseteq \mathcal{A}$ and any action tuple for A at q, $t_q = \langle \alpha_{a_1}, \alpha_{a_2}, ..., \alpha_{a_{|A|}} \rangle$ with $1 \leq \alpha_{a_i} \leq d_{a_i}(q)$ for all $1 \leq i \leq |A|$, the A-profile $m(t_q)$ is defined in the following way:

$$\begin{aligned} m(t_q) &= \langle v(t_q, 1), \dots, v(t_q, |R|) \rangle \text{ where for all } 1 \leq r \leq |R| \text{ we have} \\ v(t_q, r) &= \langle |\{a \in A_{r,q} \mid \alpha_a = 1\}|, \dots, |\{a \in A_{r,q} \mid \alpha_a = \mathbb{A}(q, r)\}| \rangle \end{aligned}$$

Thus the i-th component of $v(t_q, r)$ will be the number of agents from A in role r at q that perform action i.

Given a strategy s_A in $f(S)$ we define the strategy $m(s_A)$ for S by taking $m(s_A)(q) = m(s_A(q))$ for all $q \in Q$.

Surjectivity of m is helpful since it means that for every possible strategy that exists in the RCGS S, there is a corresponding one in $f(S)$. This in turn means that when we quantify over strategies in one of S and $f(S)$ we are implicitly also quantifying over strategies in the other. Showing equivalence, then, can be done by showing that these corresponding strategies have the same strength. Before we proceed, we give a proof of surjectivity of m.

Lemma 3.1. *For any* RCGS S *and any* $A \subseteq \mathcal{A}$*, the function* $m : strat(f(S), A) \to strat(S, A)$ *is surjective*

Using the surjective function m we can prove the following lemma, showing that the "next time" strength of any coalition A is the same in S as it is in $f(S)$.

Lemma 3.2. *For any* RCGS S*, any state* $q \in Q$ *and any coalition* $A \subseteq \mathcal{A}$*, we have* $force(S, A, q) = force(f(S), A, q)$

Given a structure S (with or without roles), and a formula ϕ, we define $true(S, \phi) = \{q \in Q \mid S, q \models \phi\}$. Equivalence of models S and $f(S)$ is now demonstrated by showing that the equivalence in next time strength established in lemma 3.2 suffices to conclude that $true(S, \phi) = true(f(S), \phi)$ for all ϕ.

Theorem 3.1. *For any* RCGS S*, any* ϕ *and any* $q \in Q$*, we have* $S, q \models \phi$ *iff* $f(S), q \models_{CGS} \phi$

4 Model checking and the size of models

We have already seen that using roles can lead to a dramatic decrease in the size of ATL-models. In this section we give a more formal account, first by investigating the size of models in terms of the number of roles, players and actions, then by an analysis of model checking ATL over concurrent game structures with roles.

Given a set of numbers $[a]$ and a number n, it is a well-known combinatorial fact that the number of ways in which to choose n elements from $[a]$, allowing repetitions, is $\frac{(n+(a-1))!}{n!(a-1)!}$. Furthermore, this number satisfies the following two inequalities:[4]

$$\frac{(n+(a-1))!}{n!(a-1)!} \leq a^n \ , \ \frac{(n+(a-1))!}{n!(a-1)!} \leq n^a \tag{1}$$

These two inequalities provides us with an upper bound on the *size* of RCGS models that makes it easy to compare their sizes to that of CGS models. Typically, the size of concurrent game structures is dominated by the size of the domain of the transition function. For an RCGS and a given state $q \in Q$ this is the number of complete profiles at q. To measure it, remember that every complete profile is an $|R|$-tuple of votes v_r, one for each role $r \in R$. It follows that $|P(q)|$ is the set of all possible combinations of votes for each role. Also remember that a vote v_r for $r \in R$ is an $\mathbb{A}(q,r)$-tuple such that the sum of entries is $|\mathcal{R}(q,r)|$. Equivalently, the vote v_r can be seen as the number of ways in which we can make $|\mathcal{R}(q,r)|$ choices, allowing repetitions, from a set of $\mathbb{A}(q,r)$ alternatives. Looking at it this way, we obtain: $|P(q)| = \prod_{r\in R} \frac{(|\mathcal{R}(q,r)|+(\mathbb{A}(q,r)-1))!}{|\mathcal{R}(q,r)|!(\mathbb{A}(q,r)-1))!}$.

We sum over all $q \in Q$ to obtain what we consider to be the size of an RCGS S. In light of equation 1, it follows that the size of S is upper bounded by both of the following expressions.

$$\mathcal{O}(\textstyle\sum_{q\in Q} \prod_{r\in R} |\mathcal{R}(q,r)|^{\mathbb{A}(q,r)}) \ , \ \mathcal{O}(\textstyle\sum_{q\in Q} \prod_{r\in R} \mathbb{A}(q,r)^{|\mathcal{R}(q,r)|}) \tag{2}$$

We observe that growth in the size of models is polynomial in $a = max_{q\in Q, r\in R}\mathbb{A}(r,q)$ if $n = \mathcal{A}$ and $|R|$ is fixed, and polynomial in $p = max_{q\in Q, r\in R}|\mathcal{R}(q,r)|$ if a and $|R|$ are fixed. This identifies a significant potential advantage arising from introducing roles to the semantics of ATL. The size of a CGS M, when measured in the same way, replacing complete profiles at q by complete action tuples at q, grows exponentially in the players whenever $d_a(q) > 1$ for each player a. We stress that we are *not* just counting the number of transitions in our models differently. We do have an additional parameter, the roles, but this is a genuinely new semantic construct that gives rise to genuinely different semantic structures. We show that it is possible to use them to give the semantics of ATL, but this does not mean that there is not more to be said about them. Particularly crucial is the question of model checking over RCGS models.

4.1 Model checking using roles

For strategic logics, checking satisfiability is usually non-tractable, and the question of model checking is often crucial in assessing the usefulness of different logics. For ATL there is a well known "standard" algorithm, see e.g. [2]. It does model checking in time linear in the length of the formula and the size of the model. The algorithm is based on the fixed point equation 3 from the proof of Theorem 3.1, so it will work also when model checking RCGS models. It is not clear, however, how the high level description should be implemented and, crucially, what the complexity will be in terms of the new parameters that arise.

Given a structure with roles, S, and a formula ϕ, the standard model checking algorithm returns the set $true(S,\phi)$, proceeding as detailed in algorithms 1 and 2.

Given a structure S, a coalition A, a state $q \in Q$ and a set of states Q', the method $enforce$ answers true or false depending on whether or not A can enforce Q' from q. That is, it tells us if at q there is $Q'' \in force(q,A)$ such that $Q'' \subseteq Q'$. Given a fixed length formula and a fixed number of states, this step dominates the running time of *mcheck* (algorithm 1). It is also the only part of the standard algorithm that behaves in a different way after addition of roles to the structures. It involves the following steps:

[4] If this is not clear, remember that n^a and a^n are the number of functions $[n]^{[a]}$ and $[a]^{[n]}$ respectively. It should not be hard to see that all ways in which to choose n elements from a induce non-intersecting sets of functions of both types

Algorithm 1 $mcheck(S, \phi)$

if $\phi = p \in \Pi$ **then**
  return $\pi(p)$
if $\phi = \neg\psi$ **then**
  return $Q \setminus mcheck(S, \psi)$
if $\phi = \psi \wedge \psi'$ **then**
  return $mcheck(S, \psi) \cap mcheck(S, \psi')$
if $\phi = \langle\langle A \rangle\rangle \bigcirc \psi$ **then**
  return $\{q \mid enforce(S, q, A, mcheck(S, \psi))\}$
if $\phi = \langle\langle A \rangle\rangle \Box \psi$ **then**
  $Q_1 := Q$, $Q_2 := mcheck(S, \psi)$
  while $Q_1 \not\subseteq Q_2$ **do**
    $Q_1 := Q_2$, $Q_2 := \{q \in Q \mid enforce(S, A, q, Q_2)\} \cap Q_2$
    return Q_1
if $\phi = \langle\langle A \rangle\rangle \psi \mathcal{U} \psi'$ **then**
  $Q_1 := \emptyset$, $Q_2 = mcheck(S, \psi)$, $Q_3 = mcheck(S, \psi')$
  while $Q_3 \not\subseteq Q_1$ **do**
    $Q_1 := Q_1 \cup Q_3$, $Q_3 := \{q \in Q \mid enforce(S, A, q, Q_1)\} \cap Q_2$
    return Q_3

Algorithm 2 $enforce(S, A, q, Q')$

for $F \in P(q, A)$ **do**
  $p = true$
  for $F' \in ext(q, F)$ **do**
    if $\delta(q, F') \notin Q'$ **then**
      $p = false$
  if $p = true$ **then**
    return $true$
return $false$

For all profiles $F \in P(q, A)$ the algorithm runs through all complete profiles $F' \in P(q)$ that extend F. Over CGSs, given a coalition A and two action tuples $t = \langle \alpha_{a_1}, \alpha_{a_2}, \ldots, \alpha_{a_{|A|}} \rangle, t' = \langle \alpha'_{a_1}, \alpha'_{a_2}, \ldots, \alpha'_{a_{|A|}} \rangle$ for A at q, the sets of complete action tuples that extend t and t' respectively do not intersect. It follows that running through all such extensions for all possible action tuples for A at q is at most linear in the total number of complete action tuples at q. This is no longer the case for RCGS models. Given two profiles P, P' for A at q, there can be many shared extensions. In fact, P and P' can share exponentially many in terms of the number of players and actions available.[5] So, in general, running $enforce$ requires us to make several passes through the set of all complete profiles, and the complexity is no longer linear. Still, it is polynomial in the number of complete profiles, since for any coalition A and state q we have $|P(q, A)| \leq |P(q)|$, meaning that the complexity of $enforce$ is upper bounded by $|P(q)|^2$. It follows that model checking of ATL over concurrent game structures with roles is polynomial in the size of the model. We summarize this result.

Proposition 4.1. *Given a* CGS S *and a formula* ϕ, $mcheck(S, \phi)$ *takes time* $\mathcal{O}(le^2)$ *where* l *is the length of* ϕ *and* $e = \sum_{q \in Q} P(q)$ *is the total number of transitions in* S

Since model checking ATL over CGSs takes only linear time, $\mathcal{O}(le)$, adding roles apparently makes model checking harder. On the other hand, the *size* of CGS models can be bigger by an exponential factor, making

[5] To see this, consider $P = \langle v_1, v_2 \ldots, v_{|R|} \rangle$ and $P' = \langle v'_1, v'_2, \ldots, v'_{|R|} \rangle$. Each $v_r, v'_r \in V_A(q, r)$ sums to $\Sigma_{1 \leq j \leq \mathbb{A}(q,r)} v_i(j) = |A_{q,r}|$. Then form a complete profile $P'' = \langle v''_1, v''_2, \ldots, v''_{|R|} \rangle$ at q such that for all $1 \leq r \leq |R|$ and all $1 \leq j \leq \mathbb{A}(q, r)$ we have $v''_r(j) = max(v_r(j), v'_r(j))$. Then, if it exists, choose a coalition A' such that $|A'_{r,q}| = \Sigma_{1 \leq j \leq \mathbb{A}(q,r)} v''_r(j)$. It is clear that the number of complete profiles that extends both v and v' is equal to the number of *all* $\mathcal{A} \setminus A'$-profiles at q.

model checking much easier after adding roles. In light of the bounds we have on the size of models, c.f. equation 2, we find that as long as the roles and the actions remain fixed, complexity of model checking is only polynomial in the number of agents. This is a potentially significant argument in favor of roles.

In practice, however, finding an optimal RCGS for a given CGS model M might be at least as difficult as model checking on M directly. It involves identifying the structure from $f^-(M)$ that has the minimum number of roles. In general, one cannot expect this task to have sub-linear complexity in the size of M.[6] Roles should be used at the modelling stage, as they give the modeller an opportunity for exploiting homogeneity in the system under consideration. We think that it is reasonable to hypothesize that in practice, most large scale systems that lends themselves well to modelling by ATL do so precisely because they exhibit significant homogeneity. If not, identifying an accurate ATL model of the system, and model checking it, seems unlikely to be tractable at all.

The question arises as to whether or not using an RCGS is *always* the best choice, or if there are situations when the losses incurred in the complexity of model checking outweigh the gains we make in terms of the size of models. A general investigation of this in terms of how fixing or bounding the number of roles affect membership in complexity classes is left for future work. Here, we conclude with the following proposition which states that as long we use the standard algorithm, model checking any CGS M can be done at least as quickly by model checking an *arbitrary* $S \in f^-(M)$.

Proposition 4.2. *Given any* CGS*-model* M *and any formula* ϕ*, let* $c(mcheck(M, \phi))$ *denote the complexity of running* $mcheck(M, \phi)$*. We have, for all* $S \in f^-(M)$*, that complexity of running* $mcheck(S, \phi)$ *is* $\mathcal{O}(c(mcheck(M, \phi))$

5 Conclusions and future work

In this paper we have described a new type of semantics for strategic logic ATL. We have provided motivational examples and argued that although in principle model checking ATL interpreted over concurrent game structures is harder than the standard approach, it is still polynomial and generates exponentially smaller models. We believe this provides conclusive evidence that concurrent game structures with roles are an interesting semantics for ATL, and should be investigated further.

References

1. Rajeev Alur and Thomas A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(11):7–48, July 1999.
2. Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM (JACM)*, 49(5):672–713, 2002.
3. Wojciech Jamroga. Easy yet hard: Model checking strategies of agents. In Michael Fisher, Fariba Sadri, and Michael Thielscher, editors, *Computational Logic in Multi-Agent Systems*, pages 1–12. Springer-Verlag, Berlin, Heidelberg, 2009.
4. Wojciech Jamroga and Jürgen Dix. Do agents make model checking explode (computationally)? In M. Pechoucek, P. Petta, and L. Z. Varga, editors, *Multi-Agent Systems and Applications IV (LNAI Volume 3690)*, 2005.
5. Wojciech Jamroga and Wiebe van der Hoek. Agents that Know How to Play. *Fundamenta Informaticae*, 63(2-3):185–219, 2004.
6. Magdalena Kacprzak and Wojciech Penczek. A SAT-based approach to unbounded model checking for alternating-time temporal epistemic logic. *Synthese*, 142(2):203–227, November 2004.
7. Wiebe van der Hoek, Alessio Lomuscio, and Michael Wooldridge. On the complexity of practical ATL model checking. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 201–208. ACM, 2006.
8. Wiebe van der Hoek and Michael Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1-2):81–119, May 2005.

[6] Although in many practical cases, when models are given in some compressed form, the situation might be such that it is possible. The question of how to efficiently find small RCGS-models will be investigated in future work.

A Proofs

Proof of Lemma 3.1

Proof. Let p_A be some strategy for A in S. We must show there is a strategy s_A in $f(S)$ such that $m(s_A) = p_A$. For all $q \in Q$, we must define $s_A(q)$ appropriately. Consider the profile $p_A(q) = \langle v_1, \ldots, v_{|R|} \rangle$ and note that by definition of a profile, all v_r for $1 \leq r \leq |R|$ are A-votes for r and that by definition of an A-vote, we have $\sum_{1 \leq i \leq \mathbb{A}(q,r)} v_r(i) = |A_{r,q}|$. Also, for all agents $a, a' \in A_{r,q}$ we know, by definition of f, that $d_a(q) = d_{a'}(q) = \mathbb{A}(q,r)$.

From this it follows that there are functions $\alpha : A \to \mathbb{N}^+$ such that for all $a \in A$, $\alpha(a) \in [d_a(q)]$ and $|\{a \in A_{r,q} \mid \alpha(a) = i\}| = v_r(i)$ for all $1 \leq i \leq \mathbb{A}(q,r)$, i.e.

$$v_r = \langle |\{a \in A_{r,q} | \alpha(a) = 1\}|, \ldots, |\{a \in A_{r,q} | \alpha(a) = \mathbb{A}(q,r)\}| \rangle$$

We choose some such α and $s_A = \langle \alpha(a_1), \ldots, \alpha(a_{|A|}) \rangle$. Having defined s_A in this way, it is clear that $m(s_A) = p_A$. □

Proof of Lemma 3.2

Proof. By definition of $force$ and lemma 3.1 it is sufficient to show that for all $s_A \in strat(f(S), A)$, we have $succ(S, m(s_A), q) = succ(f(S), s_A, q)$. We show $\subseteq$ as follows: Assume that $q' \in force(S, m(s_A), q)$. Then there is some complete profile $P = \langle v_1, \ldots, v_{|R|} \rangle$, extending $m(s_A)(q)$, such that $\delta(q, P) = q'$. Let $m(s_A)(q) = \langle w_1, \ldots, w_{|R|} \rangle$ and form $P' = \langle v'_1, \ldots, v'_{|R|} \rangle$ defined by $v'_i = v_i - w_i$ for all $1 \leq i \leq |R|$. Then each v'_i is an $(\mathcal{A} \setminus A)$-vote for role i, meaning that the sum of entries in the tuple v'_i is $|(\mathcal{A} \setminus A)_{r,q}|$. This means that we can define a function $\alpha : \mathcal{A} \to \mathbb{N}^+$ such that for all $a \in \mathcal{A}$, $\alpha(a) \in [d_a(q)]$ and for all $a \in A$, $\alpha(a) = s_a(q)$ and for every $r \in R$ and every $a \in (\mathcal{A} \setminus A)$, and every $1 \leq j \leq \mathbb{A}(q,r)$, $|\{a \in (\mathcal{A} \setminus A)_{r,q} \mid \alpha(a) = j\}| = v'_r(j)$. Having defined α like this it follows by definition of m that for all $1 \leq j \leq \mathbb{A}(q,r)$, $|\{a \in A_{r,q} \mid \alpha(a) = j\}| = w_r(j)$. Then for all $r \in R$ and all $1 \leq j \leq \mathbb{A}(q,r)$ we have $|\{a \in \mathcal{R}(q,r) \mid \alpha(a) = j\}| = v_r(j)$. By definition of $f(S)$ it follows that $q' = \delta(q, P) = \delta'(q, \alpha)$ so that $q' \in force(f(S), s_A, q)$. We conclude that $force(S, f(s_A), q) \subseteq force(f(S), s_A, q)$. The direction $\supseteq$ follows easily from the definitions of m and f. □

Proof of Theorem 3.1

Proof. We prove the theorem by showing that for all ϕ, we have $true(S, \phi) = true(f(S), \phi)$. We use induction on complexity of ϕ. The base case for atomic formulas and the inductive steps for Boolean connectives are trivial, while the case of $\langle\langle A \rangle\rangle \bigcirc \phi$ is a straightforward application of lemma 3.2. For the cases of $\langle\langle A \rangle\rangle \Box \phi$ and $\langle\langle A \rangle\rangle \phi \mathcal{U} \psi$ we rely on the following fixed point characterizations, which are well-known to hold for ATL, see for instance [3], and are also easily verified against definition 2.2:

$$\begin{aligned} \langle\langle A \rangle\rangle \Box \phi &\leftrightarrow \phi \wedge \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \Box \phi \\ \langle\langle A \rangle\rangle \phi_1 \mathcal{U} \phi_2 &\leftrightarrow \phi_2 \vee (\phi_1 \wedge \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \phi_1 \mathcal{U} \phi_2 \end{aligned} \tag{3}$$

We show the induction step for $\langle\langle A \rangle\rangle \Box \phi$, taking as induction hypothesis $true(S, \phi) = true(f(S), \phi)$. The first equivalence above identifies $Q' = true(S, \langle\langle A \rangle\rangle \Box \phi)$ as the maximal subset of Q such that ϕ is true at every state in Q' and such that A can enforce a state in Q' from every state in Q', i.e. such that $\forall q \in Q' : \exists Q'' \in force(q, A) : Q'' \subseteq Q'$. Notice that a unique such set always exists. This is clear since the union of two sets satisfying the two requirements will itself satisfy them (possibly the empty set). The first requirement, namely that ϕ is true at all states in Q', holds for S iff if holds for $f(S)$ by induction hypothesis. Lemma 3.2 states $force(S, q, A) = force(f(S), q, A)$, and this implies that also the second requirement holds in S iff it holds in $f(S)$. From this we conclude $true(S, \langle\langle A \rangle\rangle \Box \phi) = true(f(S), \langle\langle A \rangle\rangle \Box \phi)$ as desired. The case for $\langle\langle A \rangle\rangle \phi \mathcal{U} \psi$ is similar, using the second equivalence. □

Proof of Proposition 4.2

Proof. It is clear that for any $S \in f^{-}(M)$, running $mcheck(S, \phi)$ and $mcheck(M, \phi)$, a difference in overall complexity can arise only from a difference in the complexity of $enforce$. So we compare the complexity of $enforce(S, A, q, Q'')$ and $enforce(M, A, q, Q'')$ for some arbitrary $q \in Q$, $Q'' \subseteq Q$. The complexity in both cases involves passing through all complete extensions of all strategies for A at q. The sizes of these sets are can be compared as follows, the first inequality is an instance of equation 1 and the equalities follow from definition of f and the fact that $M = f(S)$.

$$\begin{aligned}
\prod_{r \in R} \left(\frac{(|A_{r,q}| + (\mathbb{A}(r,q) - 1))!}{|A_{r,q}|!(\mathbb{A}(r,q) - 1)!} \right) \times \prod_{r \in R} \left(\frac{((|\mathcal{R}(r,q)| - |A_{r,q}|) + (\mathbb{A}(r,q) - 1))!}{(|\mathcal{R}(r,q)| - |A_{r,q}|)!(\mathbb{A}(r,q) - 1)!} \right) \\
\leq \left(\prod_{r \in R} \mathbb{A}(r,q)^{|A_{r,q}|} \times \prod_{r \in R} \mathbb{A}(r,q)^{|\mathcal{R}(r,q)| - |A_{r,q}|} \right) \\
= \prod_{r \in R} \left(\prod_{a \in A_{r,q}} \mathbb{A}(r,q) \right) \times \prod_{r \in R} \left(\prod_{a \in \mathcal{R}(a,r) \setminus A_{r,q}} \mathbb{A}(r,q) \right) \\
= \left(\prod_{a \in A} d_a(q) \times \prod_{a \in \mathcal{A} \setminus A} d_a(q) \right) = \prod_{a \in \mathcal{A}} d_a(q)
\end{aligned}$$

We started with the number of profiles (transitions) we need to inspect when running $enforce$ on S at q, and ended with the number of action tuples (transitions) we need to inspect when running $enforce$ on $M = f(S)$. Since we showed the first to be smaller or equal to the latter and the execution of all other elements of $mcheck$ are identical between S and M, the claim follows. □