

Strategic and Epistemic Reasoning for the Game Description Language GDL-II: The Technical Report

Ji Ruan Michael Thielscher
University of New South Wales, Australia
{jiruan,mit}@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-201213
May 2012

THE UNIVERSITY OF
NEW SOUTH WALES



School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

The game description language GDL has been developed as a logic-based formalism for representing the rules of arbitrary games in general game playing. A recent language extension called GDL-II allows to describe nondeterministic games with any number of players who may have incomplete, asymmetric information. In this paper, we show how the well-known Alternating-time Temporal Epistemic Logic (ATEL) can be adapted for strategic and epistemic reasoning about general games described in GDL-II. We provide a *semantic* characterisation of GDL-II descriptions in terms of ATEL models. We also provide a *syntactic* translation of GDL-II descriptions into ATEL formulas, and we prove that these two characterisations are equivalent. We show that model checking in this setting is decidable by giving an algorithm, and we demonstrate how our results can be used to verify strategic and epistemic properties of games described in GDL-II.¹

¹A shorter version of this paper is published at the 20th European Conference on Artificial Intelligence (ECAI 2012).

1 Introduction

The general game description language GDL, which has been established as input language for general game-playing systems [7, 12], has recently been extended to GDL-II to incorporate games with nondeterministic actions and where players have incomplete/imperfect information [18]. We have previously analysed the epistemic logic behind GDL-II and in particular shown that the situation at any stage of a game can be characterised by a multi-agent epistemic (i.e., S5-) model [14]. However, this result merely provides a static characterisation of what players know (and don't know) at a certain stage. As such it cannot be used to reason about how players' knowledge evolves as the game progresses, nor does it allow to reason about the strategic ability of players to reach a desired state (possibly in cooperation with other players), etc. All these aspects presuppose the use of an underlying logic that goes beyond standard epistemic logic in that it combines both strategic and epistemic reasoning. Alternating-time Temporal Epistemic Logic (ATEL) [19], an extension to Alternating-time Temporal Logic (ATL) [1] with incomplete information, is such a formalism. For strategic reasoning alone, it has been shown that ATL can be applied to reason about complete-information games described in the original GDL using model checking methods [15]. Also, model checking for GDL is known to be EXPTIME-complete [15]. But unfortunately, the addition of incomplete information (and perfect recall) in GDL-II renders the model checking problem of ATL/ATEL undecidable [4].

In this paper we provide an adaption of ATEL for strategic and epistemic reasoning about general games described in GDL-II. Our main results are a characterisation of GDL-II descriptions in ATEL and a decidability result for the model checking problem. Specifically, we provide a *semantic* characterisation of GDL-II descriptions in terms of ATEL models and a *syntactic* translation of GDL-II descriptions into ATEL formulas, and we prove that these two characterisations are equivalent. We show that model checking in this setting is decidable and demonstrate how our results allow to verify strategic and epistemic properties of games described in GDL-II.

The paper is organised as follows. Section 2 gives preliminaries on GDL-II and ATEL. Section 3 presents a semantic characterisation of GDL-II using ATEL models and a syntactic mapping from GDL-II to ATEL formulas, along with the main equivalence result. Section 4 shows the decidability of model checking problem by giving an algorithm and discusses the strategic and epistemic properties that can be used for reasoning about GDL-II games. We conclude with a discussion on related work.

2 Preliminaries

2.1 Game Description Language GDL-II

A complete game description consists of the names of (one or more) players, a specification of the initial position, the legal moves and how they affect the position, and the terminating and winning criteria. The emphasis of game description languages is on high-level, declarative game rules that are easy to understand and maintain. At the same time, GDL and its successor GDL-II have a precise semantics and are fully machine-processable. Moreover, background knowledge is not required—a set of rules is all a player needs to know to be able to play a hitherto unknown game. The description language GDL-II uses these *keywords*:

GDL (without `sees` and `random`) is suitable for describing finite, synchronous,

role(?r)	?r is a player
init(?f)	?f holds in the initial position
true(?f)	?f holds in the current position
legal(?r, ?m)	?r can do move ?m
does(?r, ?m)	player ?r does move ?m
next(?f)	?f holds in the next position
terminal	the current position is terminal
goal(?r, ?v)	goal value for role ?r is ?v
sees(?r, ?p)	?r perceives ?p in the next position
random	the random player

and deterministic n -player games with complete information about the game state [12]. The extended game description language GDL-II allows to specify games with randomness and imperfect/incomplete information [18].

In the following, we assume the reader to be familiar with basic notions and notations of logic programming, as can be found in e.g. [10]. The interested reader may take a peek at Figure 2.1 at this point to see an example of a GDL-II specification.

Definition 1. A valid GDL-II specification is a finite set of clauses G where

- role only appears in the head of facts;
- init only appears as head of clauses and does not depend on any of true, legal, does, next, terminal, or goal;
- true only appears in clause bodies;
- does only appears in clause bodies, and none of legal, terminal, or goal depends on does;
- next and sees only appear as head of clauses;
- distinct only appears in clause bodies;¹
- there are no cycles involving a negative edge in the dependency graph² for G ; that is, G must be stratified [3, 5].
- each variable in a clause occurs in at least one positive atom in the body; that is, in the jargon of logic programming, G must be allowed [11].
- If p and q occur in a cycle in the dependency graph and G contains a clause $p(s_1, \dots, s_m) \leftarrow q_1(\vec{t}_1), \dots, q(v_1, \dots, v_k), \dots, q_n(\vec{t}_n)$, then for every $v_i \in \{v_1, \dots, v_k\}$,
 - v_i is ground, or
 - $v_i \in \{s_1, \dots, s_m\}$, or
 - v_i is an element of some \vec{t}_j ($1 \leq j \leq n$) such that q_j does not appear in a cycle with p .

¹The meaning of this predicate is given by assuming the unary clause $\text{distinct}(s, t)$, for every pair s, t of syntactically different ground (i.e., variable-free) terms.

²The nodes of the dependency graph for G are the relation constants in the vocabulary. There is an edge from r_2 to r_1 whenever there is a rule with r_1 in the head and r_2 in the body. That edge is labeled with the negation symbol \neg whenever r_2 is in a negative literal.

This last condition imposes a restriction on the combination of function symbols and recursion to ensure decidability of all relevant derivations [12].

These syntactic restrictions are imposed in order to ensure that a set of GDL-II rules can be effectively and unambiguously interpreted by a state transition system as a formal game model, as follows.

A unique game model can be obtained from a valid GDL-II game description by using the notion of the *stable models* of logic programs with negation [6].

Definition 2. *Given a set of clauses G and an interpretation I (i.e., a set of ground atoms), let G^I be the set of negation-free implications $h \Leftarrow b_1 \wedge \dots \wedge b_k$ obtained by taking all ground instances of clauses in G and*

- *deleting all clauses with a negative body literal $\neg b_i$ such that $b_i \in I$,*
- *deleting all negative body literals from the remaining clauses.*

Then I is a stable model for G if and only if I is the least model for G^I .

A useful property of stable models is that they provide a unique model whenever the underlying set of clauses is stratified [6], as is always the case in GDL-II. In the following, by $G \vdash p$ we denote that ground atom p is contained in this unique standard model for a stratified set of clauses G . The syntactic restrictions in GDL-II ensure that all logic programs we consider have a *unique* and *finite* stable model [12]. Hence, for the following game model underlying GDL-II we assume a finite set of players, finite states, and finitely many legal moves in each state.

Specifically, then, the derivable instances of `role(?r)` from a given game description define the players. The initial state is composed of the derivable instances of `init(?f)`. In order to determine the legal moves in any given state, this state has to be encoded first, using the keyword `true`. Let, to this end, $S = \{f_1, \dots, f_n\}$ be a state (i.e., a finite set of ground terms), then the game rules G are augmented by the n facts

$$S^{\text{true}} \stackrel{\text{def}}{=} \{\text{true}(f_1). \dots \text{true}(f_n).\}$$

Those instances of `legal(?r, ?m)` that are derivable from $G \cup S^{\text{true}}$ define all legal moves M for player R in position S . In the same way, the clauses for `terminal` and `goal(?r, ?n)` define termination and goal values *relative* to the encoding of a given position.

Determining a position update and the percepts of the players requires the encoding of both the current position and a *joint move*. Suppose joint move M is such that players r_1, \dots, r_k make moves m_1, \dots, m_k , then let

$$M^{\text{does}} \stackrel{\text{def}}{=} \{\text{does}(r_1, m_1). \dots \text{does}(r_k, m_k).\}$$

The instances of `next(?f)` derivable from $G \cup M^{\text{does}} \cup S^{\text{true}}$ compose the updated position; likewise, the derivable instances of `sees(?r, ?p)` describe what a player perceives when the given joint move is made in the given position.

All of the above is summarised in the following definition.

Definition 3 (GDL-II Game Semantics). *Let G be a valid GDL-II specification. The semantics of G is the state transition system $(R, s_0, t, l, u, \mathcal{I}, g)$ given by*

- *roles $R = \{r \mid \text{role}(r) \in \text{SM}(G)\}$;*

- *initial position* $s_0 = \text{SM}(G \cup \{\text{true}(f) \mid \text{init}(f) \in \text{SM}(G)\})$;
- *terminal positions* $t = \{s \mid \text{terminal} \in s\}$;
- *legal moves* $l = \{(r, m, s) \mid \text{legal}(r, m) \in s\}$;
- *state update function* $u(M, s) = \text{SM}(G \cup \{\text{true}(f) \mid \text{next}(f) \in \text{SM}(G \cup s \cup M^{\text{does}})\})$,
for all joint legal moves M (i.e., where each role in R takes one legal move) and states s ,³
- *information relation* $\mathcal{I} = \{(r, M, s, p) \mid \text{sees}(r, p) \in \text{SM}(G \cup s \cup M^{\text{does}})\}$;
- *goal relation* $g = \{(r, n, s) \mid \text{goal}(r, n) \in s\}$.

Note that a state s contains all ground atoms that are true in the state, which includes the “fluent atoms” $\text{true}(f)$ in, respectively, $\{\text{true}(f) \mid \text{init}(f) \in \text{SM}(G)\}$ (for the initial state) and $\{\text{true}(f) \mid \text{next}(f) \in \text{SM}(G \cup s \cup M^{\text{does}})\}$ (for the successor state of s and M), and all other atoms that can be derived from G and these fluent atoms.

Different runs of a game can be described by *developments*, which are sequences of states and moves by each player up to a certain round. A player *cannot distinguish* two developments if he makes the same moves and perceptions in the two.

Definition 4. [18] Let $\langle R, s_0, t, l, u, \mathcal{I}, g \rangle$ be the semantics of a GDL-II description G , then a development δ is a finite sequence

$$\langle s_0, M_1, s_1, \dots, s_{d-1}, M_d, s_d \rangle$$

such that for all $i \in \{1, \dots, d\}$ ($d \geq 0$), M_i is a joint move and $s_i = u(M_i, s_{i-1})$.

The length of a development δ , denoted as $\text{len}(\delta)$, is the number of states in δ . By $M(j)$ we denote agent j 's move in the joint move M . Let $\delta|_i$ be the prefix of δ up to length $i \leq \text{len}(\delta)$.

A player $j \in R \setminus \{\text{random}\}$ cannot distinguish two developments $\delta = \langle s_0, M_1, s_1, \dots \rangle$ and $\delta' = \langle s_0, M'_1, s'_1, \dots \rangle$ (written as $\delta \sim_j \delta'$) iff $\text{len}(\delta) = \text{len}(\delta')$ and for $i = \text{len}(\delta) - 1$:

- $\{p \mid (j, M_i, s_{i-1}, p) \in \mathcal{I}\} = \{p \mid (j, M'_i, s'_{i-1}, p) \in \mathcal{I}\}$,
- $M_i(j) = M'_i(j)$, and $\delta|_i \sim_j \delta'|_i$.

As an example, Figure 2.1 provides a GDL-II description of a card trading game adapted from [8]. In this game, player Bob (abbreviated as b) plays against the *random* player (line 1). The deck consists of Ace, King and Queen (A, K, Q); it is assumed that A beats K , K beats Q , but Q beats A (lines 2–4). The game starts at round 0 (line 5). Lines 7–11 describe the possible moves for the players in different rounds: First, *random* deals one card to b and one card to itself. Then b decides whether to trade his card for the one in the deck or to keep the current one. The player with the better card wins the game. Lines 13–20 define what will be true in the next state. For example, if *random* does $\text{deal}(\text{?X}, \text{?Y})$, then b will hold ?X in the next state (line 15). Lines 22–24 specify what b can see in the next state: if *random* does $\text{deal}(\text{?X}, \text{?Y})$, then b will see that it holds card ?X in the next state (line 22), and if *random* holds ?X in

³For a joint move M where players r_1, \dots, r_k take moves m_1, \dots, m_k , we define $M^{\text{does}} \stackrel{\text{def}}{=} \{\text{does}(r_1, m_1), \dots, \text{does}(r_k, m_k)\}$.

```

1 role(b). role(random).
2 isDeal(A,K). isDeal(A,Q). isDeal(K,A). isDeal(K,Q).
3 isDeal(Q,A). isDeal(Q,K).
4 beats(A, K). beats(K, Q). beats(Q, A).
5 init(round(0)).
6
7 legal(random,deal(?X,?Y)) <= true (round(0)),isDeal(?X,?Y).
8 legal(b, noop) <= true (round(0)).
9 legal(random, noop) <= true (round(1)).
10 legal(b, trade) <= true (round(1)).
11 legal(b, keep) <= true (round(1)).
12
13 next(round(1)) <= true(round(0)).
14 next(round(2)) <= true(round(1)).
15 next(holds(b, ?X)) <= does(random, deal(?X,?Y)).
16 next(holds(random, ?Y)) <= does(random, deal(?X,?Y)).
17 next(holds(b, ?X)) <= does(b, keep), true(holds(b, ?X)).
18 next(holds(b, ?Z)) <= does(b, trade), true(holds(b, ?X)),
19   true(holds(random, ?Y)), isDeal(?Z, ?X), isDeal(?Z, ?Y).
20 next(holds(random, ?X)) <= true(holds(random, ?X)).
21
22 sees(b, holds(b,?X)) <= does(random, deal(?X,?Y)).
23 sees(b, holds(random, ?X)) <= true(holds(random, ?X)),
24   true(round(1)).
25
26 terminal <= true(round(2)).
27 goal(b, 100) <= true(holds(b, ?X)),
28   true(holds(random, ?Y)), beats(?X, ?Y).
29 goal(b, 0) <= true(holds(b, ?X)),
30   true(holds(random, ?Y)), beats(?Y, ?X).
31 win <= goal(b, 100)

```

Figure 2.1: A card trading game in GDL-II

round 1, then b sees this in the next state (i.e., in round 2) (lines 23–24). Line 26 specifies when the game is terminal, and lines 27–31 give the goal values of player b (where $\text{goal}(b, 100)$ implies a *win* for b). The corresponding game model is sketched in Figure 2.2. Due to limited space, the *noop* actions are not shown in the joint actions; e.g., for $\langle \text{noop}, \text{deal}(A, K) \rangle$ we only show $\text{deal}(A, K)$. States that player b cannot distinguish are connected by a dashed line.

2.2 ATEL with Finite Computations

Definition 5 (Language of ATEL, [19]). *The language of ATEL (with respect to a set of agents Ag and a set of atomic propositions Φ), is given by the following grammar:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \psi \mid \langle\langle X \rangle\rangle\bigcirc\varphi \mid \langle\langle X \rangle\rangle\Box\varphi \mid \langle\langle X \rangle\rangle\varphi\mathcal{U}\psi \mid K_i\varphi \mid C_X\varphi$$

where $p \in \Phi$ is an atomic proposition and $X \subseteq Ag$ is a set of agents. Other logic constants and connectives $\top, \perp, \vee, \rightarrow$ are defined as usual and $\langle\langle X \rangle\rangle\Diamond\varphi$ is defined as $\langle\langle X \rangle\rangle\top\mathcal{U}\varphi$.

Intuitively, $\langle\langle X \rangle\rangle\bigcirc\varphi$ means a coalition of agents X can ensure that φ will hold in the next state; $\langle\langle X \rangle\rangle\Box\varphi$ means a coalition of agents X can ensure that φ will always hold; $\langle\langle X \rangle\rangle\varphi\mathcal{U}\psi$ means a coalition of agents X can ensure that φ will hold until ψ holds; $K_i\varphi$ means agent i knows φ ; and $C_X\varphi$ means that φ is common knowledge among the agents in X . For example, “agent i knows that agent j knows p ” can be expressed as K_iK_jp .

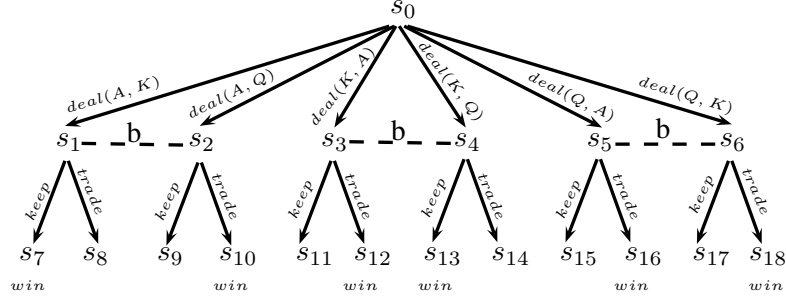


Figure 2.2: The game model for the card trading game.

We will interpret ATEL formulas on finite tree-like structures (game trees) derived from GDL-II games (such as shown in Figure 2.2). A game is well-formed if it terminates after finite steps and if all players have at least one legal move in non-terminal states [12].⁴ In addition, the setup of general game playing competitions is such that all agents are aware of the time progressing (*Synchronicity*) and remember what they have seen and done in the past (*Perfect Recall*) [18]. For this reason, we adapt the ATEL models from [19, 8] by adding an explicit set of terminal states. These terminal states correspond to those games states in which `terminal` holds.

Definition 6 (Model of ATEL). *An Action-based Alternating Epistemic Transition System (AAETS) is a tuple*

$$\langle Q, s_0, T, Ag, \{Ac_i \mid i \in Ag\}, \rho, \tau, \{\sim_i : i \in Ag\}, \Phi, V \rangle$$

where

- Q is a finite, non-empty set of possible states;
- $s_0 \in Q$ is the initial state;
- $T \subseteq Q$ is a non-empty set of terminal states;
- Ag is a finite, non-empty set of n agents;
- Ac_i is a finite, non-empty set of actions for each $i \in Ag$;
- $\rho : Ac_{Ag} \rightarrow 2^Q$ is an action precondition function, which for each action $a \in Ac_{Ag} (= \bigcup_{i \in Ag} Ac_i)$ defines the set of states $\rho(a)$ from which a may be executed (for any action a it is assumed that $T \cap \rho(a) = \emptyset$, which means that no actions are possible in terminal states);
- $\tau : Ac_1 \times \dots \times Ac_n \times Q \rightarrow Q$ is a partial system transition function, which defines the state $\tau(M, s)$ that would result from agents' actions $M (= a_1, \dots, a_n)$ on state s , given that $s \in \rho(M(i))$ for all agent $i \in Ag$ (i.e., agent i 's action $M(i) = a_i$ can be executed at state s);
- Φ is a finite, non-empty set of atomic propositions;

⁴In general, termination is not guaranteed as GDL-II can describe games that run forever, but all games considered in general game playing competitions are required to be well-formed [7, 12]. Some games (such as TicTacToe) terminate naturally, and in other games (such as Chess) a step counter can be added to enforce termination after finitely many moves.

- each $\sim_i \subseteq Q \times Q$ is an equivalence relation (called the accessibility relation) for agent i ;
- $V : Q \mapsto 2^\Phi$ is a valuation function that assigns each state a set of atomic propositions (said to be true in that state).

A *computation* of an AAETS is a finite sequence of states $\lambda = s_0 s_1 \dots s_k \in Q^+$ such that for each $0 < u \leq k$, there is a joint action M' such that $s_u = \tau(M', s_{u-1})$. A computation λ starting in state s is referred to as an *s-computation*. If $0 \leq u < |\lambda|$ (the size of λ), then we denote by $\lambda[u]$ the u -th state in λ , by $\lambda[0, u]$ the finite prefix $s_0 \dots s_u$ of λ , and by $last(\lambda)$ the last state of λ .

In addition to finiteness, we stipulate the following three properties for our AAETS in accordance with the general game playing setting as discussed above.

Definition 7 (Tree, Synchronicity and Perfect Recall). *An AAETS \mathcal{A} has tree property iff any state s is reached from initial state s_0 via a unique computation. We denote such a computation by $\lambda(s_0, s)$.*

An AAETS \mathcal{A} has synchronicity iff for all $s, t \in \mathcal{A}$ and agents i , $s \sim_i t$ implies that the computations from the initial state s_0 to s (i.e., $\lambda(s_0, s)$) and from s_0 to t (i.e., $\lambda(s_0, t)$) have the same length, i.e., $|\lambda(s_0, s)| = |\lambda(s_0, t)|$.

An AAETS \mathcal{A} has perfect recall iff for all finite computations $\lambda, \lambda' \in Q^+$ and agents i , $\lambda \sim_i \lambda'$ implies that $last(\lambda) \sim_i last(\lambda')$ and $\lambda[0, |\lambda| - 2] \sim_i \lambda'[0, |\lambda'| - 2]$.

Given an agent $i \in Ag$ and a state $s \in Q$, we denote the *options* available to i in s —the actions that i may perform in s —by $options(i, s) = \{m \mid m \in Ac_i \text{ and } s \in \rho(m)\}$. We then define

A *perfect recall strategy* for an agent $i \in Ag$ is a function $\sigma_i : Q^+ \rightarrow Ac_i$

that must satisfy

- the *legality* constraint that $\sigma_i(\lambda) \in options(i, last(\lambda))$ for all finite computations $\lambda \in Q^+$, and
- the *uniformity* constraint that for any two finite computations $\lambda_1, \lambda_2 \in Q^+$, if $\lambda_1 \sim_i \lambda_2$ then $\sigma_i(\lambda_1) = \sigma_i(\lambda_2)$.

A *perfect recall strategy* for a coalition $X = \{i_1, \dots, i_k\} \subseteq Ag$ is a tuple of strategies $\langle \sigma_1, \dots, \sigma_k \rangle$, one for each agent $i \in X$. We denote i 's component of σ_X by σ_X^i . The *outcome* of applying a strategy for coalition X on a finite computation λ is defined as $out(\sigma_X, \lambda) = \{s \mid \exists M \text{ such that } M(i) = \sigma_X^i(\lambda) \text{ for } i \in X, \text{ and } s = \tau(M, last(\lambda))\}$.

Given a perfect recall strategy σ_X for some coalition X , and a state $s \in Q$, we define $comp(\sigma_X, s)$ to be the set of all finite computations that may occur if every agent $i \in X$ follows the corresponding strategy σ_i , starting when the system is in state $s \in Q$ and ending with a terminal state in T : $comp(\sigma_X, s) = \{\lambda \mid \lambda[0] = s, last(\lambda) \in T \text{ and } \forall 0 \leq u < |\lambda| - 1 : \lambda[u + 1] \in out(\sigma_X, \lambda[0, u])\}$.

Note that herein lies the major difference between our models and the models defined in other ATL/A TEL papers [1, 19, 8]. In our case, all the computations in $comp(\sigma_X, s)$ are finite in accordance with well-formed GDL-II games, whereas they are infinite in other papers. This results in the following modified semantics.

Definition 8 (Semantics of ATEL). *For a finite AAETS \mathcal{A} and a state s , let $\lambda(s_0, s)$ denote the finite computation starting from s_0 and ending with s . The truth of ATEL formulas is inductively defined as follows:*

$\mathcal{A}, s \models p$ iff $p \in V(s)$ (where $p \in \Phi$);

$\mathcal{A}, s \models \neg\varphi$ iff $\mathcal{A}, s \not\models \varphi$;

$\mathcal{A}, s \models \varphi \vee \psi$ iff $\mathcal{A}, s \models \varphi$ or $\mathcal{A}, s \models \psi$;

$\mathcal{A}, s \models \langle\langle X \rangle\rangle \bigcirc \varphi$ iff $\exists \sigma_X$, such that $\forall \lambda \in \text{comp}(\sigma_X, s)$ we have $|\lambda| > 1$ and $\mathcal{A}, \lambda[1] \models \varphi$;

$\mathcal{A}, s \models \langle\langle X \rangle\rangle \square \varphi$ iff $\exists \sigma_X$, such that $\forall \lambda \in \text{comp}(\sigma_X, s)$ we have $\mathcal{A}, \lambda[u] \models \varphi$ for all $0 \leq u < |\lambda|$;

$\mathcal{A}, s \models \langle\langle X \rangle\rangle \varphi \mathcal{U} \psi$ iff $\exists \sigma_X$, such that $\forall \lambda \in \text{comp}(\sigma_X, s)$ there exists some $u < |\lambda|$ such that $\mathcal{A}, \lambda[u] \models \psi$, and for all $0 \leq v < u$ we have $\mathcal{A}, \lambda[v] \models \varphi$;

$\mathcal{A}, s \models K_i \varphi$ iff $\forall s'$ such that $\lambda(s_0, s) \sim_i \lambda(s_0, s')$, $\mathcal{A}, s' \models \varphi$;

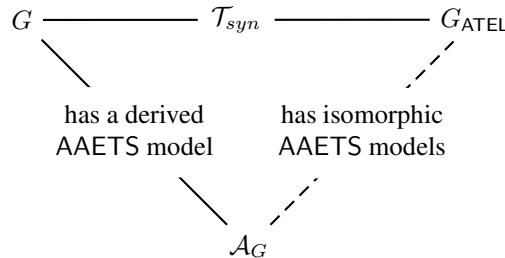
$\mathcal{A}, s \models C_X \varphi$ iff $\forall s'$ such that $\lambda(s_0, s) \sim_X^* \lambda(s_0, s')$, $\mathcal{A}, s' \models \varphi$ where \sim_X^* is the transitive and reflexive closure of $\cup_{i \in X} \sim_i$.

3 Mapping GDL-II into ATEL

GDL-II serves as a language to describe games, while ATEL is for reasoning about such games. We build two links between GDL-II and ATEL:

- On the *semantic* level, every GDL-II description G induces an AAETS *model* \mathcal{A}_G .
- On the *syntactic* level, every GDL-II description G can be translated into an ATEL *theory* G_{ATEL} (defined as $\mathcal{T}_{\text{syn}}(G)$ below).

Thus, we are able to interpret ATEL formulas φ over G either via ATEL semantics, i.e., define $G \models \varphi$ as $\mathcal{A}_G \models_{\text{ATEL}} \varphi$. Or, we can use the syntactic characterisation, i.e., define $G \models \varphi$ as $\models_{\text{ATEL}} \mathcal{T}_{\text{syn}}(G) \rightarrow \varphi$. As our main result, we will prove that these two characterisations are equivalent. The following diagram depicts the main idea, and we are now going to present it in detail.



Since ATEL does not support first-order predicates, we follow [16, 15] in applying a pre-processing step to GDL-II descriptions by replacing all predicates with variables, such as $\text{isDeal}(\ ?X, \ ?Y)$, by all relevant instances (also called ground atoms), such as $\text{asDeal}(\ A, \ K)$. This maps an arbitrary GDL-II description into an equivalent variable-free specification. We then translate such ground atoms to atomic propositions in ATEL.

Definition 9 (Translation t and t_{pre}). Let At_{GDL-II} be the set of ground atoms in GDL-II, and At_{ATEL} be the set of atomic propositions in ATEL. The translation t maps every GDL-II formulas to an ATEL formula as follows.

Base case:

$$t(\mathbf{p}) = p \text{ for all } \mathbf{p} \in At_{GDL-II}$$

where $p \in At_{ATEL}$.

Extended cases:

$$\begin{aligned} t(\text{not } \mathbf{p}) &= \neg t(\mathbf{p}); \\ t(\{\mathbf{p}_1, \dots, \mathbf{p}_k\}) &= \{t(\mathbf{p}_1), \dots, t(\mathbf{p}_k)\} \text{ for all literals } \mathbf{p}_i. \end{aligned}$$

Note that a literal is either \mathbf{p} or $\text{not } \mathbf{p}$ for all $\mathbf{p} \in At_{GDL-II}$. Furthermore, let t_{pre} be defined by $t_{pre}(\mathbf{p}) = t(\mathbf{p})_{pre}$ for the base case (and similar to t for the extended cases). An atom p_{pre} will represent the value of atom p in the previous state (in ATEL). For convenience, we abbreviate $does(i, m)_{pre}$ by $done(i, m)$.

We define how to induce an AAETS from a GDL-II description.

Definition 10 (Semantic interpretation of GDL-II in ATEL). Given a GDL-II description G with semantics $\langle R, s'_0, t, l, u, \mathcal{I}, g \rangle$, an AAETS for G (denoted as \mathcal{A}_G) is a tuple

$$\langle Q, s_0, T, Ag, \{Ac_i \mid i \in Ag\}, \rho, \tau, \{\sim_i \mid i \in Ag\}, \Phi, V \rangle$$

where

- Q is the set of states of G ;
- $s_0 \in Q$ is the initial game state s'_0 ;
- T is the set of terminal states as t ;
- Ag is the set of roles $R \setminus \{\text{random}\}$ (assume n agents);
- $Ac_i = \{m \mid (i, m, s) \in l, s \in Q\}$ is the set of moves of agent i ;
- $\tau : Ac_1 \times \dots \times Ac_n \times Q \mapsto Q$ is a partial function that maps a set of action and a state to another state such that $\tau(M, s) = u(M, s)$;
- $\sim_i \subseteq Q \times Q$ is the accessibility relation for agent $i \in Ag$ given by $(s, s') \in \sim_i$ (also written as $s \sim_i s'$) iff role i cannot distinguish any two developments δ and δ' such that $\delta = \langle s_0 \dots s \rangle$ and $\delta' = \langle s_0 \dots s' \rangle$ (cf. Definition 4);
- $\Phi \subseteq At_{ATEL}$ is a set of atomic propositions translated by t and t_{pre} from the atoms in At_{GDL-II} ;
- $V : Q \rightarrow 2^\Phi$ is an interpretation function which associates with each state s the set of atoms that satisfies the following requirements: if $\mathbf{p} \in s$ then $t(\mathbf{p}) \in V(s)$; if $s = u(M, s')$ then for any agent i we have $done(i, M(i)) \in V(s)$, and for any $\mathbf{p} \in s'$ we have $t_{pre}(\mathbf{p}) \in V(s)$; and $p_{init} \in V(s_0)$.

The interpretation function requires that if there is a transition from s' to s , then the moves that were made on s' are recorded in $V(s)$ and the atoms true in s' are also recorded in s using the corresponding atoms labelled with *pre*. The accessibility relation of states is given according to the developments that end with such states. This ensures that \mathcal{A}_G always has synchronicity and perfect recall.

Next, we give the full syntactic translation of GDL-II descriptions into ATEL formulas.

Definition 11 (Syntactic Translation \mathcal{T}_{syn}). *Given a game description G , we define its ATEL theory $G_{ATEL} = \mathcal{T}_{syn}(G)$ as a conjunction:*

$$\Gamma_{ini} \wedge \Gamma_{norm} \wedge \Gamma_{leg} \wedge \Gamma_{act} \wedge \Gamma_{mem} \wedge \Gamma_{next} \wedge \Gamma_{sees}.$$

These conjuncts are described in detail as follows.

- Γ_{ini} . The initial state is captured by a conjunction of: all the fluent atoms that are initially true plus the extra atomic proposition p_{init} (which is only true in s_0). Formally,

$$\Gamma_{ini} = \bigwedge_{\text{init}(f) \in G} \text{true}(f) \wedge (p_{init} \wedge \langle\langle\rangle\rangle \circ \langle\langle\rangle\rangle \square \neg p_{init}).$$

- Γ_{norm} . For normal rules (i.e., rules without any of the keywords `role`, `init`, `next` or `sees` in the head), we group those by heads:

$$\begin{array}{lcl} r_1 : & \mathbf{p} & \Leftarrow bd_1 \\ \dots & \dots & \dots \\ r_k : & \mathbf{p} & \Leftarrow bd_k \end{array}$$

Such rules decide whether \mathbf{p} is true in the current state. Let $R_{\mathbf{p}} = \{r_1, \dots, r_k\}$, H be the heads of all such rules, and AH be the set of atoms that do not appear in the heads of any rules and are not “does” atoms, then:

$$\Gamma_{norm} = \langle\langle\rangle\rangle \square (\bigwedge_{\mathbf{p} \in H} (\text{CP}(R_{\mathbf{p}}) \wedge \text{LF}(R_{\mathbf{p}})) \wedge \bigwedge_{\mathbf{p} \in AH} \neg t(\mathbf{p}))$$

where $\text{CP}(R_{\mathbf{p}}) = t(\mathbf{p}) \leftrightarrow (\bigvee_{j \in [1..k]} (\bigwedge t(bd_j)))$, and $\text{LF}(R_{\mathbf{p}})$ is a *loop* formula (see below). Notice that bd_i (a set of literals) is the body of rule i . Specially, if bd_i is empty, then $\bigwedge t(bd_j)$ is \top , and if \mathbf{p} does not appear in the head of any rules and is not a “does” atom, then it must be false, which is captured by $\neg t(\mathbf{p})$.

Formula $\text{CP}(R_{\mathbf{p}})$ applies Clark’s completion [2] to a given set of GDL-II rules $R_{\mathbf{p}}$. An example from the card trading game is: $\text{CP}(R_{win}) = win \leftrightarrow goal(b, 100)$. But in general the semantics of the completion of a (stratified) logic program is too weak to fully characterise the standard model in the presence of redundant rules like $\mathbf{p} \Leftarrow \mathbf{p}$. The standard model remains the same when such “superfluous” clauses are added, but Clark’s completion is weakened by them [9]. This issue is solved by a propositional formula denoted as $\text{LF}(R_{\mathbf{p}})$ (which is also called a loop formula); we refer to [9] for a detailed algorithm to compute such a formula.

- Γ_{leg} . In all non-terminal states, each agent must make one legal move. This means that if $legal(i, m)$ is true in the current state, then agent i can enforce $done(i, m)$ to be true in a next state and on the other hand if $done(i, m)$ is true in the current state, then $legal(i, m)$, must be true in the previous state, i.e., $legal(i, m)_{pre}$ is true in the current state. This is captured by the following:

$$\Gamma_{leg} = \langle\langle\rangle\rangle \square (\neg terminal \rightarrow \bigwedge_{i \in Ag, m \in Ac_i} (legal(i, m) \leftrightarrow \langle\langle i \rangle\rangle \circ done(i, m)) \wedge \bigwedge_{i \in Ag, m \in Ac_i} (done(i, m) \rightarrow legal(i, m)_{pre})).$$

- Γ_{act} . For all non-initial states, each agent should have done exactly one action in the previous state, and agents always know what they did:

$$\Gamma_{act} = \langle\langle\rangle\rangle \square (\neg p_{init} \rightarrow \bigwedge_{i \in Ag} \text{XOR}_{m \in Ac_i} done(i, m)) \wedge \langle\langle\rangle\rangle \square (\bigwedge_{i \in Ag, m \in Ac_i} (done(i, m) \leftrightarrow K_i done(i, m))).$$

where XOR is the *exclusive OR* operator.

- Γ_{mem} . For non-terminal states, we use the atom $t_{pre}(\mathbf{p})$ to record the truth-value of $t(\mathbf{p})$ for it to be used in the next states. Let At be At_{GDL-II} but without “does” atoms, then:

$$\Gamma_{mem} = \langle\langle\rangle\rangle\Box(\neg terminal \rightarrow \bigwedge_{\mathbf{p} \in At} ((t(\mathbf{p}) \leftrightarrow \langle\langle\rangle\rangle \circ t_{pre}(\mathbf{p})) \wedge (\neg t(\mathbf{p}) \leftrightarrow \langle\langle\rangle\rangle \circ \neg t_{pre}(\mathbf{p}))).$$

- Γ_{next} . Suppose these are all the rules with head $\mathbf{next}(\mathbf{f})$:

$$\begin{array}{l} r_1 : \mathbf{next}(\mathbf{f}) \Leftarrow bd_1 \\ \dots \quad \dots \quad \dots \quad \dots \\ r_k : \mathbf{next}(\mathbf{f}) \Leftarrow bd_k \end{array}$$

There are two alternatives for translating these rules into ATEL formulas. From the perspective of a *current state*, the truth of \mathbf{f} in the next state is determined by the truth of the propositions in the bodies of rules r_1, \dots, r_k in the current state and the actions that are chosen by the agents for the transition. From the perspective of a *next state*, the truth of \mathbf{f} in this state is determined by the previous truth of the propositions in the bodies of rules r_1, \dots, r_k and the actions that have just been done. We adopt the second perspective. Let HN be the heads of all such rules, and define

$$\Gamma_{next} = \langle\langle\rangle\rangle\Box(\neg p_{init} \rightarrow \bigwedge_{\mathbf{next}(\mathbf{f}) \in HN} (true(\mathbf{f}) \leftrightarrow (\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j))))).$$

- Γ_{sees} . The rules with “sees” are similar to those with “next”, but instead of defining what will be true they specify what will be seen by the agents next. Suppose these are all the rules with head $\mathbf{sees}(\mathbf{i}, \mathbf{x})$:

$$\begin{array}{l} r_1 : \mathbf{sees}(\mathbf{i}, \mathbf{x}) \Leftarrow bd_1 \\ \dots \quad \dots \quad \dots \quad \dots \\ r_k : \mathbf{sees}(\mathbf{i}, \mathbf{x}) \Leftarrow bd_k. \end{array}$$

Again we adopt the perspective of a next state. Let HS be the heads of all “sees” rules, and define

$$\Gamma_{sees} = \langle\langle\rangle\rangle\Box(\neg p_{init} \rightarrow \bigwedge_{\mathbf{sees}(\mathbf{i}, \mathbf{x}) \in HS} ((K_i \mathbf{sees}(\mathbf{i}, \mathbf{x}))_{pre} \leftrightarrow (\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j)))) \wedge (K_i \neg \mathbf{sees}(\mathbf{i}, \mathbf{x}))_{pre} \leftrightarrow \neg(\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j))))).$$

Note that the size of the ATEL theory is polynomial in the size of a variable-free GDL-II description. Our translation is correct in the sense that the resulting ATEL formula is satisfiable in the ATEL model derived from the same game description.

Proposition 1 (Correctness). *For a GDL-II game description G ,*

$$\mathcal{A}_G \models G_{ATEL}.^1$$

Proof. Let G be a GDL-II game description with semantics $\langle R, s'_0, t, l, u, \mathcal{I}, g \rangle$, and \mathcal{A}_G (the ATEL model derived from G) be a tuple

$$\langle Q, s_0, T, Ag, \{Ac_i \mid i \in Ag\}, \rho, \tau, \{\sim_i \mid i \in Ag\}, \Phi, V \rangle.$$

¹We sometimes also write this as $\mathcal{A}_G, s_0 \models G_{ATEL}$.

We need to verify that

$$\mathcal{A}_G, s_0 \models \Gamma_{ini} \wedge \Gamma_{norm} \wedge \Gamma_{leg} \wedge \Gamma_{act} \wedge \Gamma_{mem} \wedge \Gamma_{next} \wedge \Gamma_{sees}.$$

We check for each conjunct Γ of G_{ATEL} as follows. (The conjuncts were given in Definition 11).

- Γ_{ini}

By definition, for any $true(f) \in \Gamma_{ini}$, it holds that $\text{init}(f) \in G$. Then according to the definition of the initial state s_0 in Definition 3, it holds that $\text{true}(f) \in s_0$. Following Definition 10, we have that $true(f) \in V(s_0)$, which means $\mathcal{A}_G, s_0 \models true(f)$. Also from Definition 10, we have $p_{init} \in V(s_0)$ and $p_{init} \notin V(s)$ for any reachable state s from s_0 . Thus $\mathcal{A}_G, s_0 \models \Gamma_{ini}$.

- Γ_{norm}

Let H be the set of atoms which appear as the heads of rules in G and do not include the keywords `role`, `init`, `next` or `sees`. And let AH be the set of atoms that do not appear in the heads of any rules and are not “does” atoms.

To show $\mathcal{A}_G, s_0 \models \Gamma_{norm}$, we just need to show that for any reachable state s from s_0 , we have $\mathcal{A}_G, s \models \bigwedge_{p \in H} (\text{CP}(R_p) \wedge \text{LF}(R_p)) \wedge \bigwedge_{p \in AH} \neg t(p)$. Note that the formula does not have temporal operator and is only propositional.

We distinguish two cases. If $p \in H$, then let $R_p = \{r_1, \dots, r_k\}$, the set of rules with head p , and

$$\begin{array}{l} r_1 : p \quad \Leftarrow \quad bd_1 \\ \dots \quad \dots \quad \dots \quad \dots \\ r_k : p \quad \Leftarrow \quad bd_k \end{array}$$

Now by applying Clark’s completion [2] and the method in [9], we have $\mathcal{A}_G, s \models \bigwedge_{p \in H} (\text{CP}(R_p) \wedge \text{LF}(R_p))$. If $p \in AH$, then there is no rule to make it true, therefore $p \notin s$ for any state s . This implies that $\mathcal{A}_G, s \models \bigwedge_{p \in AH} \neg t(p)$.

- Γ_{leg}

To show $\mathcal{A}_G, s_0 \models \Gamma_{leg}$, we just need to show that for any reachable state s from s_0 , we have $\mathcal{A}_G, s \models (\neg \text{terminal} \rightarrow \bigwedge_{i \in Ag, m \in Ac_i} (\text{legal}(i, m) \leftrightarrow \langle\langle i \rangle\rangle \circ \text{done}(i, m))) \wedge \bigwedge_{i \in Ag, m \in Ac_i} (\text{done}(i, m) \rightarrow \text{legal}(i, m)_{pre})$.

Suppose that $\mathcal{A}_G, s \models \neg \text{terminal}$. This means that s is not a terminal state. For an arbitrary $\text{legal}(i, m)$, we show $\mathcal{A}_G, s \models (\text{legal}(i, m) \leftrightarrow \langle\langle i \rangle\rangle \circ \text{done}(i, m))$. From left to right, assume $\mathcal{A}_G, s \models \text{legal}(i, m)$. Then in the game model, we have $\text{legal}(i, m) \in s$, which means that player i is allowed to choose move m . Combined with the definition of V in Definition 10, the result of choosing this move makes $\text{done}(i, m)$ true in the next states of s , and therefore $\mathcal{A}_G, s \models \langle\langle i \rangle\rangle \circ \text{done}(i, m)$. From right to left, assume $\mathcal{A}_G, s \models \langle\langle i \rangle\rangle \circ \text{done}(i, m)$. There must be a next state s' of s such that $\mathcal{A}_G, s' \models \text{done}(i, m)$, and again using Definition 10, we know that $\mathcal{A}_G, s' \models \text{legal}(i, m)$.

We then show $\mathcal{A}_G, s \models \bigwedge_{i \in Ag, m \in Ac_i} (\text{done}(i, m) \rightarrow \text{legal}(i, m)_{pre})$. Suppose $\mathcal{A}_G, s \models \text{done}(i, m)$. Again by Definition 10, it means that there exists a predecessor s' of s such that $\mathcal{A}_G, s' \models \text{legal}(i, m)$, and hence $\mathcal{A}_G, s \models \text{legal}(i, m)_{pre}$.

- Γ_{act}

To show $\mathcal{A}_G, s_0 \models \Gamma_{act}$, we just need to show that for any reachable state s from s_0 , $\mathcal{A}_G, s \models (\neg p_{init} \rightarrow \bigwedge_{i \in Ag} \text{XOR}_{m \in Ac_i} done(i, m))$ and $\mathcal{A}_G, s \models (\bigwedge_{i \in Ag, m \in Ac_i} (done(i, m) \leftrightarrow K_i done(i, m)))$.

To show the first, assume $\mathcal{A}_G, s \models \neg p_{init}$. This means that s is not the initial state. Therefore s must have a predecessor s' . Suppose from s' to s , the joint action is M . For arbitrary agent i , we have $\mathcal{A}_G, s \models done(i, M_i)$, and hence $\mathcal{A}_G, s \models \bigwedge_{i \in Ag} \text{XOR}_{m \in Ac_i} done(i, m)$.

To show the second, first assume (from left to right) $\mathcal{A}_G, s \models done(i, m)$. Then assume i can not distinguish t from s . This means that i can not distinguish the development from s_0 to s , and the development from s_0 to t . Since i has perfect recall, it follows that both in the predecessor of s and that of t , i chose same action m , therefore we have $\mathcal{A}_G, t \models done(i, m)$. Thus $\mathcal{A}_G, s \models K_i done(i, m)$. The other direction follows from the semantics of knowledge operators directly.

- Γ_{mem}

To show $\mathcal{A}_G, s_0 \models \Gamma_{mem}$, we just need to show that for any reachable state s from s_0 , $\mathcal{A}_G, s \models (\neg terminal \rightarrow \bigwedge_{p \in At} ((t(p) \leftrightarrow \langle\langle \rangle \rangle \circ t_{pre}(p)) \wedge (\neg t(p) \leftrightarrow \langle\langle \rangle \rangle \circ \neg t_{pre}(p))))$. Assume $\mathcal{A}_G, s \models \neg terminal$. This means that s is not a terminal state. Let X be the set of successors of s and p be an atom without does keyword. Pick any $x \in X$ and we distinguish two cases. (1) If $\mathcal{A}_G, s \models t(p)$, then it follows from Definition 10 that $\mathcal{A}_G, x \models t_{pre}(p)$. Since x is arbitrary, we have $\mathcal{A}_G, s \models \langle\langle \rangle \rangle \circ t_{pre}(p) \wedge \neg \langle\langle \rangle \rangle \circ \neg t_{pre}(p)$. (2) If $\mathcal{A}_G, s \models \neg t(p)$, then it follows from Definition 10 that $\mathcal{A}_G, x \models \neg t_{pre}(p)$. Since x is arbitrary, we have $\mathcal{A}_G, s \models \langle\langle \rangle \rangle \circ \neg t_{pre}(p) \wedge \neg \langle\langle \rangle \rangle \circ t_{pre}(p)$.

- Γ_{next}

To show $\mathcal{A}_G, s_0 \models \Gamma_{next}$, we just need to show that for any reachable state s from s_0 , $\mathcal{A}_G, s \models \neg p_{init} \rightarrow \bigwedge_{\text{next}(f) \in HN} (true(f) \leftrightarrow (\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j))))$.

Suppose $\mathcal{A}_G, s \models \neg p_{init}$. It means that s is not the initial state. Suppose s' is the predecessor of s and M is a joint action such that $s = u(M, s')$. Pick any $\text{next}(f) \in HN$ and suppose the rules with head $\text{next}(f)$ are:

$$\begin{aligned} r_1 : \text{next}(f) &\Leftarrow bd_1 \\ \dots & \dots \dots \dots \\ r_k : \text{next}(f) &\Leftarrow bd_k \end{aligned}$$

$\mathcal{A}_G, s \models true(f)$ iff, by GDL-II semantics, $\text{next}(f) \in \text{SM}(G \cup s' \cup M^{\text{does}})$ iff there is a rule r_j such that $\text{SM}(G \cup s' \cup M^{\text{does}}) \models bd_j$ iff $\mathcal{A}_G, s \models \bigwedge t_{pre}(bd_j)$. Therefore $\mathcal{A}_G, s \models true(f)$ iff $\mathcal{A}_G, s \models (\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j)))$.

- Γ_{sees}

To show $\mathcal{A}_G, s_0 \models \Gamma_{sees}$, we just need to show that for any reachable state s from s_0 , $\mathcal{A}_G, s \models (\neg p_{init} \rightarrow$

$$\bigwedge_{\text{sees}(i,x) \in HS} ((K_i \text{sees}(i,x)_{pre} \leftrightarrow (\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j)))) \wedge (K_i \neg \text{sees}(i,x)_{pre} \leftrightarrow \neg (\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j)))))$$

Now suppose $\mathcal{A}_G, s \models \neg p_{init}$. It follows that s is not the initial state. Suppose s' is the predecessor of s and M is a joint action such that $s = u(M, s')$. For any $\text{sees}(i, x) \in HS$, we collect all the rules with head $\text{sees}(i, x)$:

$$\begin{aligned} r_1 : \text{sees}(i, x) &\Leftarrow bd_1 \\ \dots &\dots \dots \dots \\ r_k : \text{sees}(i, x) &\Leftarrow bd_k. \end{aligned}$$

Suppose $\mathcal{A}_G, s \models \bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j))$, then there must be a rule r_j such that $\mathcal{A}_G, s \models \bigwedge t_{pre}(bd_j)$. We have that $\mathcal{A}_G, s' \models \text{sees}(i, x)$ and hence $\mathcal{A}_G, s \models \text{sees}(i, x)_{pre}$. It also follows that agent i is informed of perception x in state s . Now suppose for any s'' such that $\lambda(s_0, s'') \sim_i \lambda(s_0, s)$ (i.e., agent i can not distinguish two computations from s_0 to s and s''), it follows that agent i should have the same perception, therefore agent i is informed of perception x in state s'' as well. This means that $\mathcal{A}_G, s'' \models \text{sees}(i, x)_{pre}$. Since s'' is arbitrary, we have $\mathcal{A}_G, s \models K_i \text{sees}(i, x)_{pre}$. This gives us $\mathcal{A}_G, s \models (\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j))) \rightarrow K_i \text{sees}(i, x)_{pre}$. The other direction can be shown using a reverse reasoning.

On the other hand, suppose $\mathcal{A}_G, s \models \neg \bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j))$, then there is no rule to make $\text{sees}(i, x)$ true in the state s' . We have that $\mathcal{A}_G, s' \models \neg \text{sees}(i, x)$ and hence $\mathcal{A}_G, s \models \neg \text{sees}(i, x)_{pre}$. It also follows that agent i is *not* informed of perception x in state s . Now suppose for any s'' such that $\lambda(s_0, s'') \sim_i \lambda(s_0, s)$, agent i is *not* informed of perception x in state s'' as well. So we have $\mathcal{A}_G, s'' \models \neg \text{sees}(i, x)_{pre}$ and hence $\mathcal{A}_G, s \models K_i \neg \text{sees}(i, x)_{pre}$. This gives us $\mathcal{A}_G, s \models \neg (\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j))) \rightarrow K_i \neg \text{sees}(i, x)_{pre}$. The other direction can be shown using a reverse reasoning.

□

To show that \mathcal{T}_{syn} is an adequate syntactic characterisation of GDL-II descriptions, we define an equivalence relation on AAETSs.

Definition 12 (AAETS Isomorphism). *Let $\mathcal{A} = \langle Q, s_0, T, Ag, \{Ac_i \mid i \in Ag\}, \rho, \tau, \{\sim_i \mid i \in Ag\}, \Phi, V \rangle$ and $\mathcal{A}' = \langle Q', s'_0, T', Ag, \{Ac'_i \mid i \in Ag\}, \rho', \tau', \{\sim'_i \mid i \in Ag\}, \Phi, V' \rangle$ be two AAETSs, then they are isomorphic (denoted as $\mathcal{A} \cong \mathcal{A}'$) iff there is a function f such that:*

- f maps every state in Q to a state in Q' and it is a bijection; in particular $f(s_0) = s'_0$, and for all $s \in T$, $f(s) \in T'$;
- f maps every action in Ac_i to an action in Ac'_i and it is a bijection;
- for every state s and action m , $s \in \rho(m)$ iff $f(s) \in \rho'(f(m))$;
- for every state s, s' and joint action M , $s = \tau(M, s')$ iff $f(s) = \tau'(f(M), f(s'))$;
- for every state s and agent i , $s \sim_i s'$ iff $f(s) \sim'_i f(s')$;
- for every proposition p and state $s \in Q$, $p \in V(s)$ iff $p \in V'(f(s))$.

The existence of an isomorphism between two AAETSs implies that they satisfy the same formulas.

Proposition 2. Given two AAETSs \mathcal{A} and \mathcal{A}' along with an arbitrary ATEL formula φ ,

$$\mathcal{A} \cong \mathcal{A}' \text{ implies } (\mathcal{A} \models \varphi \text{ iff } \mathcal{A}' \models \varphi).$$

Proof. This follows from a standard induction on the structure of φ . \square

Proposition 3. Let G be a game description and φ an ATEL formula, then the following holds

$$\models G_{\text{ATEL}} \rightarrow \varphi \text{ iff } \mathcal{A}_G \models \varphi.$$

Proof. It suffices to show that for any AAETS \mathcal{A} with synchronicity and perfect recall such that $\mathcal{A} \models G_{\text{ATEL}}$, there is an isomorphism between

$$\mathcal{A} = \langle Q_1, s_0, T_1, Ag, \{Ac_i \mid i \in Ag\}, \rho_1, \tau_1, \{\sim_i \mid i \in Ag\}, \Phi, V_1 \rangle$$

and

$$\mathcal{A}_G = \langle Q_2, t_0, T_2, Ag, \{Ac'_i \mid i \in Ag\}, \rho_2, \tau_2, \{\sim'_i \mid i \in Ag\}, \Phi, V_2 \rangle.$$

Since if \mathcal{A} and \mathcal{A}_G are isomorphic, we have that $\mathcal{A} \models \varphi$ iff $\mathcal{A}_G \models \varphi$. Combined with Proposition 1, it then follows that $\models G_{\text{ATEL}} \rightarrow \varphi$ iff $\mathcal{A}_G \models \varphi$.

Assume $\mathcal{A} \models G_{\text{ATEL}}$. We construct an isomorphism f between \mathcal{A} and \mathcal{A}_G by doing an induction on the states.

For the initial state s_0 of \mathcal{A} , it holds that $\mathcal{A}, s_0 \models \Gamma_{\text{ini}} \wedge \Gamma_{\text{norm}}$. It is the case that $p_{\text{init}} \in V_1(s_0)$ and $p_{\text{init}} \in V_2(t_0)$. For an arbitrary proposition $p = t(\mathbf{p})$, we show $p \in V_1(s_0)$ iff $p \in V_2(t_0)$. If $p = \text{true}(f)$ and $p \in V_1(s_0)$, then $\text{init}(\mathbf{f}) \in G$, therefore $\text{true}(f) \in V_2(t_0)$. If p is not $\text{true}(f)$ and $p \in V_1(s_0)$, then by Γ_{norm} , $\mathcal{A}, s_0 \models (\bigvee_{j \in [1..k]} (\bigwedge t(\text{bd}_j)))$. It follows that \mathbf{p} must be the head of a normal rule such that the body is true or empty; thus \mathbf{p} must be in the stable model of the initial state of G and therefore $p \in V_2(t_0)$. The other direction is similar by a reverse reasoning. So valuation $V_1(s_0)$ coincides with $V_2(t_0)$, then we make $t_0 = f(s_0)$.

Now suppose s is an arbitrary non-terminal state in \mathcal{A} , and has a corresponding state $t = f(s)$ in \mathcal{A}_G . Pick up an arbitrary successor state $s' = \tau_1(M, s)$ of s . By Γ_{act} , there is only one $\text{done}(i, m)$ true in s' for each agent i , and by Γ_{leg} , $\text{done}(i, m) \in V_1(s')$ implies that $\text{legal}(i, m) \in V_1(s)$, thus $\text{legal}(i, m) \in V_2(t)$. Then we map the action of i , $M(i)$, to this m , i.e., $m = f(M(i))$. This gives us a joint action M' in \mathcal{A}_G , and hence we have $t' = \tau'_2(M', t)$.

We now show that for any proposition p , $p \in V_2(s')$ iff $p \in V_2(t')$. If $p = \text{true}(f)$ and $p \in V_1(s')$, then by Γ_{next} , $\mathcal{A}, s' \models (\bigvee_{j \in [1..k]} (\bigwedge t_{\text{pre}}(\text{bd}_j)))$, which implies that $\mathcal{A}, s \models (\bigvee_{j \in [1..k]} (\bigwedge t(\text{bd}_j)))$. Since $t = f(s)$, we have $\mathcal{A}, t \models (\bigvee_{j \in [1..k]} (\bigwedge t(\text{bd}_j)))$. Now take all the rules with $\text{next}(\mathbf{f})$:

$$\begin{array}{l} r_1 : \text{next}(\mathbf{f}) \Leftarrow \text{bd}_1 \\ \dots \quad \dots \quad \dots \quad \dots \\ r_k : \text{next}(\mathbf{f}) \Leftarrow \text{bd}_k. \end{array}$$

It follows that one of the bodies of r_1, \dots, r_k is true in state t , therefore $\text{true}(f)$ must be true in state t' . If p is not $\text{true}(f)$ and $p \in V_1(s')$, then similar to the case of s_0 , we again by Γ_{norm} , derive that \mathbf{p} must be the head of a normal rule such that the body is true or empty; and thus $p \in V_2(t')$. The other direction follows from a reverse reasoning. We make $t' = f(s')$.

Formula Γ_{sees} makes the accessibility relations correspond. Suppose there are two states $s', s'' \in \mathcal{A}$ such that agent i can not distinguish them, i.e., $\lambda(s_0, s') \sim_i \lambda(s_0, s'')$. We denote s_x as the predecessor of s' such that $s' = \tau(M', s_x)$, and s_y as the predecessor of s'' such that $s'' = \tau(M'', s_y)$. Since model \mathcal{A} satisfies perfect recall, we

know that agent i makes same actions in M' and M'' , i.e., $M'(i) = M''(i)$, and agent i can not distinguish the predecessors of s' and s'' , i.e., $\lambda(s_0, s_x) \sim_i \lambda(s_0, s_y)$. Our inductive assumption is that $\lambda(t_0, f(s_x)) \sim_i \lambda(t_0, f(s_y))$. Now we need to show $\lambda(t_0, f(s')) \sim_i \lambda(t_0, f(s''))$, i.e., (1) agent i makes same actions in the transition from $f(s_x)$ to $f(s')$ and in the transition from $f(s_y)$ to $f(s'')$, and (2) agent i 's perceptions are the same in state $f(s')$ and $f(s'')$.

It follows from $M'(i) = M''(i)$ that $f(M')(i) = f(M'')(i)$, which implies that (1) holds. To show (2), we pick an arbitrary $\text{sees}(i, x)$ and take all the rules with $\text{sees}(i, x)$:

$$\begin{array}{l} r_1 : \text{sees}(i, x) \Leftarrow bd_1 \\ \dots \quad \dots \quad \dots \quad \dots \\ r_k : \text{sees}(i, x) \Leftarrow bd_k. \end{array}$$

There are two cases: either $\mathcal{A}, s' \models \text{see}(i, x)_{pre}$ or $\mathcal{A}, s' \models \neg \text{see}(i, x)_{pre}$. For the first case, by perfect recall and the semantics of $\text{sees}(i, x)$, we have $\mathcal{A}, s' \models K_i \text{sees}(i, x)_{pre}$, and then by Γ_{sees} , we have $\mathcal{A}, s' \models (\bigvee_{j \in [1..k]} (\bigwedge t_{pre}(bd_j)))$. It follows from Γ_{mem} that $\mathcal{A}, s_x \models \text{sees}(i, x) \wedge (\bigvee_{j \in [1..k]} (\bigwedge t(bd_j)))$. By induction hypothesis, we have $\mathcal{A}_G, f(s_x) \models \text{sees}(i, x) \wedge (\bigvee_{j \in [1..k]} (\bigwedge t(bd_j)))$, and thus $\mathcal{A}_G, f(s') \models \text{sees}(i, x)_{pre}$. Similarly, we have $\mathcal{A}_G, f(s'') \models \text{sees}(i, x)_{pre}$. Together, we have $\mathcal{A}_G, f(s') \models K_i \text{sees}(i, x)_{pre}$. It implies that agent i can see x at both $f(s')$ and $f(s'')$, the successor states of $f(s_x)$ and $f(s_y)$ respectively. Since $\text{sees}(i, x)$ is arbitrary, we can conclude that agent i can see the same at both $f(s')$ and $f(s'')$, i.e., (2) holds. For the second case, the argument is similar. □

This is a main result in this paper. It shows that G_{ATEL} completely characterises G in ATEL in the sense that it entails any formula that is satisfied in the AAETS derived from G directly, and vice versa.

4 Model Checking Strategic and Epistemic Properties

Our main result in this paper allows us to consider the following model checking problem: *given a game represented by GDL-II, and a property represented by an ATEL formula, decide whether the property is true for the game description*. If the agents have incomplete information and perfect recall, the model checking problem for ATEL in traditional semantics is undecidable (see [4]). Hence, had we used the standard semantics, the above problem would also be undecidable since we can reduce the problem by deriving an ATEL model from a GDL-II description and then perform the ATEL model checking. However, with our new ATEL semantics—over AAETS models with *finite* computations—the model checking problem becomes decidable as we can give an algorithm for it.

We only sketch the algorithm below for the case of $\langle\langle X \rangle\rangle \bigcirc \varphi$. This algorithm terminates because only a finite number of strategies and computations needs to be checked.

```

mcheck(A, s, <<X>> O Phi) {
  found := True;
  foreach sigma(X) do {
    foreach c in comp(sigma(X), A, s) do {
      if |c|=1 or mcheck(A, c[1], Phi) == False
      then found := False;
    }
  }
  if found == True then return True; }
return False; }
```

How complex is the problem? We know for sure that the problem is at least 2EXPTIME-hard due to [13], which shows that the outcome problem of the Private-PEEK game is complete in double exponential time (2EXPTIME-complete). The Private-PEEK game can be rather straightforwardly specified in GDL-II and the outcome problem can be equivalently expressed as deciding whether $\langle\langle 1 \rangle\rangle \diamond win_i$ is true in the initial state of the game. Finding an upper bound for the complexity, however, is left for future work.

In [15], ATL is used to characterise some interesting playability properties for the original GDL games. With ATEL being the language to express properties for GDL-II games, we can now not only express the above properties but also a new class of properties that are not expressible in ATL. We discuss two kinds of such properties.

Coherence Knowledge Properties. There are some properties φ that involve pure knowledge, i.e., where no temporal modalities occur in φ . For such φ , we call $\langle\langle \rangle\rangle \square \varphi$ a *coherence knowledge property*. We know that $\mathcal{A}, s_0 \models \langle\langle \rangle\rangle \square \varphi$ iff φ is true in all reachable states from s_0 .

In GDL-II, agents may not always know their legal moves. In order to check this, we can express the property that “if a move is legal for an agent then the agent knows it” as a formula:

$$\langle\langle \rangle\rangle \square \bigwedge_{i \in Ag, m \in Ac_i} (legal(i, m) \rightarrow K_i legal(i, m)).$$

The following is not necessarily true for a GDL-II game: if the game has terminated, then this is common knowledge,

$$\langle\langle \rangle\rangle \square (terminal \rightarrow C_{Ag} terminal).$$

If we want to ensure that a GDL-II description G has this property, then we can either verify $\mathcal{A}_G, s_0 \models \langle\langle \rangle\rangle \square (terminal \rightarrow C_{Ag} terminal)$, or prove

$$\models G_{ATEL} \rightarrow \langle\langle \rangle\rangle \square (terminal \rightarrow C_{Ag} terminal).$$

Properties with knowledge and strategic power. This class of properties mixes knowledge and coalition modalities, allowing us to talk about the agent’s knowledge and power simultaneously.

The following property says that if i has a winning strategy, then he knows it:

$$\langle\langle i \rangle\rangle \diamond win_i \rightarrow K_i \langle\langle i \rangle\rangle \diamond win_i.$$

Suppose that in the current state $\mathcal{A}, s \models K_i \langle\langle i \rangle\rangle \diamond win_i$. This does not give a winning strategy for agent i explicitly. But agent i can check the following on any state that he cannot distinguish from s (i.e., $s \sim_i s'$):

$$\mathcal{A}, s' \models \langle\langle i \rangle\rangle \circ (done(i, m) \wedge \langle\langle i \rangle\rangle \diamond win_i).$$

If the above holds, then agent i can safely choose $does(i, m)$, and it still guarantees him a winning position in the next state.

We conclude with the following strategic property, which says that if i knows φ , then he can ensure that agent j knows ψ next:

$$K_i \varphi \rightarrow \langle\langle i \rangle\rangle \circ K_j \psi.$$

5 Related Work and Conclusion

There are just a few papers on reasoning about games in GDL and its extension GDL-II. In [15], a method based on ATL is given to verify properties of general games, but this is restricted to original GDL and hence to games where players can maintain complete state information. Our paper extends this approach to GDL-II and uses a version of ATEL for this purpose. Our characterisation formula is inspired by the one given in [15], but we make these improvements: (1) we can deal with imperfect-information games; (2) we show that the models that satisfy this formula are isomorphic to \mathcal{A}_G , rather than a weaker relation as alternating bisimulation given in [15]; (3) we do not require an extra “sink” state as there is no need to make computations infinite with our new semantics.

In [17], it is shown how GDL-II can be formally translated into the Situation Calculus as a first-order axiomatisation that allows players to reason about their percepts and what they know about the legality and effects of moves based on the game description. In [14], the epistemic structure and expressiveness of GDL-II is analysed in terms of epistemic modal logic. It was shown that the operational semantics of GDL-II entails that the situation at any stage of a game can be characterised by a multi-agent epistemic (i.e., S5-) model and GDL-II is sufficiently expressive to model any situation that can be described by a (finite) multi-agent epistemic model. Our work extends the static epistemic model into a dynamic AAETS, and therefore a larger class of strategic and epistemic properties can be addressed by our approach.

Acknowledgements We thank the anonymous reviewers from ECAI 2012 for their helpful comments on the conference version of this paper. This research was supported under Australian Research Council’s (ARC) *Discovery Projects* funding scheme (project DP 120102023). Michael Thielscher is the recipient of an ARC Future Fellowship (project FT 0991348). He is also affiliated with the University of Western Sydney.

Bibliography

- [1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.
- [2] K. Apt. Introduction to logic programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 494–574. Elsevier, 1990.
- [3] Krzysztof Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, chapter 2, pages 89–148. Morgan Kaufmann, 1987.
- [4] Cătălin Dima and Ferucio Laurențiu Țiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. Research Report 1102.4225, arXiv, February 2011.
- [5] Allen Van Gelder. The alternating fixpoint of logic programs with negation. In *Proceedings of the 8th Symposium on Principles of Database Systems*, pages 1–10. ACM SIGACT-SIGMOD, 1989.

- [6] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proceedings of the International Joint Conference and Symposium on Logic Programming (IJCSLP)*, pages 1070–1080, Seattle, OR, 1988. MIT Press.
- [7] Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the AAI competition. *AI Magazine*, 26(2):62–72, 2005.
- [8] Wojciech Jamroga and Wiebe van der Hoek. Agents that know how to play. *Fundam. Inform.*, 63(2-3):185–219, 2004.
- [9] Fangzhen Lin and Yuting Zhao. Assat: computing answer sets of a logic program by sat solvers. *Artif. Intell.*, 157(1-2):115–137, 2004.
- [10] John Lloyd. *Foundations of Logic Programming*. Series Symbolic Computation. Springer, second, extended edition, 1987.
- [11] John W. Lloyd and Rodney W. Topor. A basis for deductive database systems II. *J. Log. Program.*, 3(1):55–67, 1986.
- [12] Nathaniel Love, Timothy Hinrichs, David Haley, Eric Schkufza, and Michael Genesereth. General Game Playing: Game Description Language Specification. Technical Report LG–2006–01, Stanford Logic Group, Computer Science Department, Stanford University, 2006.
- [13] John H. Reif. The complexity of two-player games of incomplete information. *J. Comput. Syst. Sci.*, 29(2):274–301, 1984.
- [14] Ji Ruan and Michael Thielscher. The epistemic logic behind the game description language. In *Proceedings of the Conference on the Advancement of Artificial Intelligence (AAAI)*, pages 840–845, San Francisco, 2011.
- [15] Ji Ruan, Wiebe van der Hoek, and Michael Wooldridge. Verification of games in the game description language. *Journal Logic and Computation*, 19(6):1127–1156, 2009.
- [16] Stephan Schiffel and Michael Thielscher. Fluxplayer: A successful general game player. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI-07)*, pages 1191–1196. AAAI Press, 2007.
- [17] Stephan Schiffel and Michael Thielscher. Reasoning about general games described in GDL-II. In *Proceedings of the Conference on the Advancement of Artificial Intelligence (AAAI)*, pages 846–851, San Francisco, 2011.
- [18] Michael Thielscher. A general game description language for incomplete information games. In *Proceedings of AAAI*, pages 994–999, 2010.
- [19] W. van der Hoek and M. Wooldridge. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.