Strategic Reasoning about General Game Playing under Imperfect Information

Technical Report

Yihui Ilona Zhu and Francesco Belardinelli

September 15, 2019

1 Preliminaries

In this technical report, we mainly focus on the translation from Game Description Language with Imperfect Information (GDL-II) to a Concurrent Game Structure with imperfect information (iCGS) using Alternating-time Temporal Logic (ATL), as well as the relationship between the two. This section provides the background information on the syntax and semantics of GDL-II, and the syntax of ATL and its semantics over iCGS.

1.1 GDL-II

Game Description Language (GDL) has been well-known for its use in General Game Playing Competition to define the general rule of a specific game for a general game player to play. As GDL describes deterministic actions and states in a game, the type of game that can be defined by GDL is restricted. Therefore, Game Description Language with Incomplete/Imperfect Information (GDL-II) is developed based on the general GDL, adding two special keywords, `sees` and `random` [5]:

- `sees` specifies the ability of a specific player/agent to receive information of specific ground term (the ability to "perceive" the term);
- `random` represents the occurrence of randomness and chances in some games, which then represent the non-deterministic nature of certain game.

This extension allows more games to be represented by GDL-II, such as card games where the players receive random cards and they do not have information about other players' cards. An example of game descriptions in GDL-II is given in Figure 1 on the next page.

As GDL and GDL-II are based on a logic programming language Datalog, before we can formally define the syntax and semantics of GDL-II, we begin by acquiring a basic understanding of Datalog.

1.1.1 Datalog Program

Datalog is a query and rule language designed for use as a database language, which syntatically is a subset of Prolog [2]. We now introduce the syntax and semantics for Datalog.

This section refers to the same syntax and semantics of Datalog program presented in [4].

Definition 1.1 (Datalog Syntax - Language, Rules and Programs)**.**
The Datalog Language builds up on a set of atomic propositions $\Pi = \{p, q, \ldots\}$. We denote the set of literals over Π by $l(\Pi) : l(\Pi) = \Pi \cup \{\neg p \mid p \in \Pi\}$.

```
01  (⇐ (role xplayer))
02  (⇐ (role yplayer))
03  (⇐ (role random))
    ...
11  (⇐ (card ace))
12  (⇐ (succ 7 6))
    ...
19  (⇐ (succ ace king))
20  (⇐ (init dealingRound))
21  (⇐ (legal random deal(C,D))
       (true dealingRound)(card C)(card D)
       (distinct C D))
    ...
25  (⇐ (legal R fold)
       (true bettingRound)(role R)
25  (⇐ (legal R fold)
       (distinct R random))
26  (⇐ (sees xplayer yourCard(C))
       (does random deal(C,D)))
    ...
```

```
31 (⇐ (sees yplayer xplayersCard(C))
      (does xplayer allIn)(does yplayer allIn)
      (true hasCard(xplayer,C))
32 (⇐ (next hasCard(xplayer,C))
      (does random deal(C,D)))
...
36 (⇐ (next bettingRound)(true dealingRound))
37 (⇐ (terminal)(not(true dealingRound)))
38 (⇐ (goal R 100)
      (true bet(R,C,allIn))(true bet(S,D,allIn))
      (beats C D))
...
39 (⇐ (goal R 0)
      (true bet(R,C,allIn))(true bet(S,C,allIn))
      (beats D C))
40 (⇐ (beats C D)(succ C D))
41 (⇐ (beats C D)(succ C X)(beats X D))
```

Figure 1: A fragment of a game description in GDL-II of a simple card game, adapted from [5].

A DATALOG *rule* r is of the form $(\Leftarrow (p)(l_1)\dots(l_n))$ where $p \in \Pi$ and $l_i \in l(\Pi)(i \leq n)$. The head of r is p in this case: $p = hd(r)$; and the body of r, $bd(r)$, is defined as the set $\{l_1, \dots, l_n\}$, which can be empty in some cases.

A DATALOG *program* is a set of DATALOG rules.

In order to understand the semantics of DATALOG programs, we define a model for DATALOG programs as a set of atomic propositions which are the atoms that are true. The precise definition is below.

Definition 1.2 (Models for DATALOG Programs)**.**
Given a DATALOG program Δ, a set of atoms $\Sigma \subseteq \Pi$ is a model for Δ iff it satisfies:

- If $(\Leftarrow (p)) \in \Delta$ then $p \in \Sigma$;
- If $(\Leftarrow (p)\, bd) \in \Delta$ and $pos(bd) \subseteq \Sigma$ and $neg(bd) \cap \Sigma = \emptyset$ then $p \in \Sigma$, where $pos(bd)$ is the set of positive literals in bd and $neg(bd)$ is the set of negative ones.

Following this definition, we can roughly interpret a rule $(\Leftarrow (p)(l_1)\dots(l_n))$ as an implication $(l_1 \wedge \dots \wedge l_n) \rightarrow p$.

In this paper, we focus on stratified DATALOG programs, as it is easier to give an interpretation of the semantics of these programs and build their models.

Definition 1.3 (Stratified DATALOG Programs)**.**
A DATALOG program Δ is *stratified* if there does not exist circular reasoning in the program. An atom p is in stratum $i \in \mathbb{N}$ if the maximum number of rules r with p as $hd(r)$ and contains the negation of any other atomic proposition in $bd(r)$ is i. A rule $r \in \Delta$ is of stratum i if $hd(r)$ is in stratum i.

Definition 1.4 (DATALOG Semantics for Stratified DATALOG Programs)**.**
Given a stratified DATALOG program Δ, we can construct a model for it $s = DatlogPMod(\Delta)$. The main idea behind this iterative process is that we first find the heads (atomic propositions) of singleton rules (rules with empty bodies), and then keep going through all the rules and find all the heads of the rules that can be satisfied by the atomic propositions we already have, until nothing can be found.

The procedure is as follows. Firstly, put $t_0 = \{p \mid (\Leftarrow p) \in \Delta\}$. Now assuming t_i is defined, we initialise s_i to t_i, and for all the rules $(\Leftarrow (p)(l_1)\dots(l_n))$ in stratum i such that $s_i \models l_1 \wedge \dots \wedge l_n$, add p to s_i. Then, let $t_{i+1} = s_i$. Suppose the maximum stratum of Δ is k, then let $s = t_{k+1}$. This *DatlogPMod*(Δ) is the DATALOG semantics of Δ.

1.1.2 GDL-II Syntax

We now introduce the formal definition of the syntax of GDL-II game descriptions. This section mainly refers to the GDL syntax discussed in [4] with extension to GDL-II rules from [5].

Definition 1.5 (GDL-II Syntax)**.**
Given a set of agents Ag, a special agent `random`, a set of actions Ac, a set of strings S, a primitive set of propositional atoms $Prim = \{p, q, \dots\}$, a set of integers $[0 \dots 100]$ (an arbitrary range) and a set of terms T, we define the set of atomic propositions of GDL-II $At_{\text{GDL-II}}$ as the smallest set that satisfied the following:

- $Prim \subseteq At_{\text{GDL-II}}$;
- special atom `terminal` $\in At_{\text{GDL-II}}$;
- for all $\mathtt{s_1}, \mathtt{s_2} \in S$, (`distinct` $\mathtt{s_1}$ $\mathtt{s_2}$) $\in At_{\text{GDL-II}}$;
- for each agent `i` $\in Ag \cup \{$`random`$\}$ and each action `a` $\in Ac$, (`legal i a`) $\in At_{\text{GDL-II}}$;
- for each agent `i` $\in Ag$, for all integer `v` in [0,...100], (`goal i v`) $\in At_{\text{GDL-II}}$.

We also define $AtExpr_{\text{GDL-II}}$ - the set of atomic expressions of GDL-II - as the smallest set that satisfies the following:

- for each `p` $\in At_{\text{GDL-II}}$, $\{$(`p`), (`init p`), (`next p`), (`true p`)$\} \subseteq AtExpr_{\text{GDL-II}}$;
- for each agent `i` $\in Ag \cup \{$`random`$\}$ and each action `a`, $\{$(`role i`), (`does i a`)$\} \subseteq AtExpr_{\text{GDL-II}}$;
- for each agent `i` $\in Ag$ and each term `t` $\in T$, (`sees i t`) $\in AtExpr_{\text{GDL-II}}$.

$LitAt_{\text{GDL-II}}$ is defined as $\{\mathtt{p}, (\mathtt{true\ p}), (\mathtt{not\ p}), (\mathtt{not}\,(\mathtt{true\ p})) \mid \mathtt{p} \in At_{\text{GDL-II}}\}$. Finally, we define $LitExpr_{\text{GDL-II}}$ as $AtExpr_{\text{GDL-II}} \cup LitAt_{\text{GDL-II}}$.

We also enforce some restrictions on the use of the keywords in GDL-II:

- `role` only appears in the head of rules;
- `init` only appears as head of rules and does not depend on any of the other keywords;
- `true` only appears in the body of rules;
- `does` only appears in the body of rules and does not depend on `legal`, `terminal` or `goal`;
- `next` and `sees` only appear as head of rules.

Now, because `true p` and `p` have the same meaning, the keyword `true` does not really extend the expressive power of GDL or GDL-II. Hence, in this report, we omit `true` from the rules and proofs of results for GDL-II, i.e. in the following sections of this report, we only include `p` and the corresponding proofs for `true p` are the same.

A *game description* specifies all the atoms from $At_{\text{GDL-II}}$ that are true, in one of the following situations: true in the initial state, true as a result of global constraints, or true as the effect of some joint actions being performed in a specific state. The semantics of atomic expressions will be given using the definition below and game model constructed later.

Definition 1.6 (Game Descriptions)**.**
A GDL-II game description Γ is a set of DATALOG rules r of the form $(\Leftarrow(\mathtt{h})(\mathtt{e_1})\dots(\mathtt{e_m}))$, where the head of the rule $\mathtt{h} = hd(r)$ is from $AtExpr_{\text{GDL-II}}$ and each $\mathtt{e_i}(i \in [1\dots m])$ in $bd(r)$ is from $LitExpr_{\text{GDL-II}}$. We say r has an *empty body* if $m = 0$. For our convenience, we categorise every game description Γ into five different types of rules where:

- $\Gamma_{\mathtt{init}}$ contains all the rules that represent constraints of the initial state of the game, i.e. rules of the form $(\Leftarrow (\mathtt{init\ p}))$. These rules have a head only and an empty body;
- $\Gamma_{\mathtt{role}}$ contains all the rules that specify the agents of the game, i.e. rules of the form $(\Leftarrow (\mathtt{role\ i}))$;
- $\Gamma_{\mathtt{glob}}$ contains all the global constraints, which are the rules of the form $(\Leftarrow (\mathtt{p})(\mathtt{e_1})\dots(\mathtt{e_m}))$, where $\mathtt{p} \in At_{\text{GDL-II}}$ and each $\mathtt{e_i}(i \in [1\dots m])$ is from $LitAt_{\text{GDL-II}}$;
- $\Gamma_{\mathtt{next}}$ contains all rules with a $(\mathtt{next\ p})$ in the head: $(\Leftarrow (\mathtt{next\ p})(\mathtt{e_1})\dots(\mathtt{e_m}))$, where $\mathtt{p} \in At_{\text{GDL-II}}$ and each $\mathtt{e_i}(i \in [1\dots m])$ is from $LitAt_{\text{GDL-II}}$ or of the form $(\mathtt{does\ i\ a})$;
- $\Gamma_{\mathtt{sees}}$ contains all rules with a $(\mathtt{sees\ i\ t})$ in the head: $(\Leftarrow (\mathtt{sees\ i\ t})(\mathtt{e_1})\dots(\mathtt{e_m}))$, where each $\mathtt{e_i}$ $(i \in [1\dots m])$ is from $LitAt_{\text{GDL-II}}$ or of the form $(\mathtt{does\ i\ a})$.

We also assume that any given game description Γ will be stratified according to Def. 1.3.

1.1.3 GDL-II Semantics: Game Model

Although the developer of GDL-II already provided a game model for GDL-II in [5], that game model is not clear enough. Furthermore, as the agent does not have perfect information about the game, situations where an agent cannot distinguish between two states might occur, and the game model should be able to describe these situations. Therefore, for the description of a different game model G, we use the same approach as in [4] but with some slight changes. Again, this section refers to the game model constructed for GDL in [4] with modifications to extend it to GDL-II.

Definition 1.7 (GDL-II Game Model)**.**
Given the set of atomic propositions $At_{\text{GDL-II}}$, a GDL-II Game Model is a tuple $G = \langle S, s_0, Ag, \{Ac_i\}_{i\in Ag}, \{\sim_i\}_{i\in Ag}, \tau, \pi\rangle$ where

- S is a finite non-empty set of states, with s_0 the initial state;
- Ag is the a finite non-empty set of agents in the game, including the `random` agent;
- each Ac_i is a finite non-empty set of possible actions for a specific agent i;
- each indistinguishability relations $\sim_i$ specify the situations where an agent i cannot distinguish between two states in a game.
- $\tau : Ac_1 \times \dots \times Ac_n \times S \to S$ is an update function that takes a state and one action taken by each agent, and then returns the successor state;
- $\pi : S \to 2^{At_{\text{GDL}}}$ is an interpretation function that takes a state and returns a set of atomic propositions that are satisfied in the state.

For our game models, they are pretty much the same as those of [4]. However, as the game descriptions now involve imperfect information, we define certain components differently in order to incorporate the `sees` expressions.

Now we construct all the components of the game model G for a game description Γ. The set of agents Ag follows from the game description Γ: $Ag = \{i \,|\, (\Leftarrow (\mathtt{role\ i})) \in \Gamma_{\mathtt{role}}\} \cup \{\mathtt{random}\}$. The set

of actions for each agent i - Ac_i - also follows from Γ: $Ac_i = \{\mathtt{a} | (\mathtt{legal\ i\ a})$ occurs in $\Gamma\}$. The main idea behind constructing all the states in the game is that we think of every state $s \in S$ as associated with the unique model under the semantics of some stratified DATALOG program Δ derived from Γ. Particularly, we let $\delta(\Gamma_{\mathtt{glob}})$ and $\delta(\Gamma_{\mathtt{init}})$ be derived from Γ, where $\delta(\Gamma_{\mathtt{glob}})$ is the set of global rules where we treat all the `true p` atoms as `p` atoms and $\delta(\Gamma_{\mathtt{init}})$ is the set $\{\Leftarrow \mathtt{p} \,|\, (\mathtt{init\ p}) \in \delta(\Gamma_{\mathtt{init}})\}$.

The steps below outline the procedure of constructing S, τ, and π used in G.

- *Firstly*, we define the initial state s_0. Put

$$\pi(s_0) = \mathit{DatlogPMod}(\delta(\Gamma_{\mathtt{init}}) \cup \delta(\Gamma_{\mathtt{glob}}))$$

  Since $\delta(\Gamma_{\mathtt{glob}})$ is stratified whenever $\Gamma_{\mathtt{glob}}$ is, we obtain that the given game description Γ is stratified, and we ensure we are applying *DatlogPMod* to a stratified program. Additionally, `sees` atoms only appear as the head of a rule, and the body only consists of global constraints and `does` expressions. We can therefore deduce that no `sees` atoms will appear in the initial state as no actions have been taken at this point, so we only consider $\delta(\Gamma_{\mathtt{init}})$ and $\delta(\Gamma_{\mathtt{glob}})$ here.

- *Next*, we generate the finite non-empty set of states S. We begin by supposing that a state $s \in S$ has been defined already. This is in fact true as we begin this step after having defined the initial state s_0, so that we would always have some states already defined in S. We also know from the way the game description is developed that if this state s is not a terminal state ($\mathtt{terminal} \notin \pi(s)$), every agent has at least one legal action available in the state. The way we classify these legal actions is by checking that for action a_i ($\mathtt{legal\ i\ a_i}) \in \pi(s)$. If we know that $\mathtt{terminal} \notin \pi(s)$, there exists a successor state u of s after a profile of legal actions $\langle a_1, \ldots, a_n\rangle$ is performed, i.e., each agent performs one legal action out of all available legal actions for that agent. For each possible profile of legal actions $\langle a_1, \ldots, a_n\rangle$, we define u by first computing all atoms that should be satisfied (true) in the next state according to $\Gamma_{\mathtt{next}}$, using a modified version of the function F_Γ from [4] that considers the `sees` atoms arising from imperfect information.

$$\begin{aligned} F_\Gamma(\langle a_1, \ldots, a_n\rangle, s) \quad = \quad & \{\Leftarrow \mathtt{p} \,|\, \text{there exists some } (\Leftarrow (\mathtt{next\ p})(\mathtt{e_1}) \ldots (\mathtt{e_k})) \in \Gamma_{\mathtt{next}} \text{ and} \\ & \pi(s) \cup \{\neg q \,|\, q \notin \pi(s)\} \cup \{(\mathtt{does\ i\ a_i}) \,|\, i \in [1 \ldots n]\} \models_{cl} \mathtt{e_1} \wedge \ldots \wedge \mathtt{e_k}\} \\ & \cup \{\Leftarrow (\mathtt{sees\ i\ t}) \,|\, \text{there exists some} \\ & (\Leftarrow (\mathtt{sees\ i\ t})(\mathtt{e_1}) \ldots (\mathtt{e_k})) \in \Gamma_{\mathtt{sees}} \text{ and} \\ & \pi(s) \cup \{\neg q \,|\, q \notin \pi(s)\} \cup \{(\mathtt{does\ i\ a_i}) \,|\, i \in [1 \ldots n]\} \models_{cl} \mathtt{e_1} \wedge \ldots \wedge \mathtt{e_k}\} \end{aligned}$$

  Now that we have a set of atoms that need to be satisfied in the next state (when each agent i takes an action a_i), we can add a new state to the game model and put

$$u = \tau(\langle a_1, \ldots, a_n\rangle, s)$$
$$\text{and } \pi(u) = \mathit{DatlogPMod}(F_\Gamma(\langle a_1, \ldots, a_n\rangle, s) \cup \delta(\Gamma_{\mathtt{glob}}))$$

- *Iteration:* repeat the step above to all successor states of the initial state, until reaching all terminal states of the game.

Now, we can define the indistinguishability relation $\sim_i$, which describes an agent i cannot distinguish between two states q and q', and is defined as $q \sim_i q'$ iff for all terms t:

- $(\mathtt{sees\ i\ t}) \in q$ iff $(\mathtt{sees\ i\ t}) \in q'$, meaning that the agent i receives the exact same information in states q and q';

- if (`sees i t`) is true then $val(\mathtt{t}, q) = val(\mathtt{t}, q')$, where $val(t, q)$ returns the valuation of term `t` in state q. This means that `t` needs to be of the same value in both states if the agent i receives information about the term;
- (`legal a i`) $\in \pi(q)$ iff (`legal a i`) $\in \pi(q')$, meaning that the available actions for agent i are the same in states q and q', ensuring that all strategies are uniform.

Lemma 1.1 *The indistinguishability relation defined above is an equivalence relation.*

Proof. We will prove the lemma by showing the indistinguishability relation $\sim_i$ is reflexive, symmetric and transitive. Suppose we have an arbitrary agent $i \in Ag$.

Firstly, suppose we have an arbitrary state q. It is then very trivial to see that if we have (`sees i t`) $\in q$, then (`sees i t`) $\in q$ is true too, and vice versa. Also, it is very obvious that whenever (`sees i t`) is true then $val(t, q) = val(t, q)$. Similarly, (`legal a i`) $\in \pi(q)$ iff (`legal a i`) $\in \pi(q)$ holds. Therefore, we have for all $q \in S, q \sim_i q$, showing that $\sim_i$ is reflexive.

Secondly, suppose we have two arbitrary states q and q' and assume we have $q \sim_i q'$. We need to show that $q' \sim_i q$. From the assumption, we have (`sees i t`) $\in q$ iff (`sees i t`) $\in q'$, if (`sees i t`) is true then $val(t, q) = val(t, q')$ and (`legal a i`) $\in \pi(q)$ iff (`legal a i`) $\in \pi(q')$. It is then easy to see that (`sees i t`) $\in q'$ iff (`sees i t`) $\in q$, (`legal a i`) $\in \pi(q')$ iff (`legal a i`) $\in \pi(q)$, and if (`sees i t`) is true then $val(t, q') = val(t, q)$ all follow directly from our assumption, by the symmetric property of 'iff' ($\leftrightarrow$) and the symmetric property of '='. We can now deduce that for all $q, q' \in S$, if $q \sim_i q'$ then $q' \sim_i q$, so $\sim_i$ is symmetric.

Finally, suppose we have three arbitrary states q, q' and q'', and assume we have $q \sim_i q'$ and $q' \sim_i q''$. We need to show that $q \sim_i q''$. For the first condition, we have (`sees i t`) $\in q$ iff (`sees i t`) $\in q'$ and (`sees i t`) $\in q'$ iff (`sees i t`) $\in q''$ from our assumption. Therefore, by transitivity of 'iff' ($\leftrightarrow$), we have (`sees i t`) $\in q$ iff (`sees i t`) $\in q''$. Similarly, the second condition follows directly from the assumption by transitivity of equals ('='), so we have if (`sees i t`) is true then $val(t, q) = val(t, q'')$. The last condition works exactly as the first one - (`legal a i`) $\in \pi(q)$ iff (`legal a i`) $\in \pi(q'')$ follow from transitivity of 'iff' ($\leftrightarrow$). We can then conclude that $\sim_i$ is transitive by for all $q, q', q'' \in S$, if we have $q \sim_i q$ and $q' \sim_i q''$, then $q \sim_i q''$. □

After we construct the model G, we can easily verify that it is indeed a game model for Γ with the following definitions.

Definition 1.8 (GDL-II Semantics)**.**
Given that we have constructed a game model $G = \langle S, s_0, Ag, Ac_1, .., Ac_n, \{\sim_i\}_{i \in Ag}, \tau, \pi \rangle$ with a family of indistinguishability relations, let $Ag' = \{i_1, \ldots, i_k\}$ be a set of agents $\subseteq Ag$, each agent with an action $a_x (x \leq k)$. Then, we define t to be an $i_1 : a_1, \ldots, i_k : a_k$ successor of s if there exists a choice for any agent $j \in Ag \backslash Ag'$ for an action b_j from Ac_j such that $\tau(\langle c_1, \ldots, c_n \rangle) = t$, where $c_y = a_x$ if $y = i_x \in Ag$ and $c_y = b_j$ if $y = j \in Ag \backslash Ag'$. Given a game model G, a state s, and $\mathtt{p} \in At_{\text{GDL-II}}$, we define:

$G, s \models_{\text{GDL-II}} \mathtt{p}$ iff $\mathtt{p} \in \pi(s)$;
$G, s \models_{\text{GDL-II}} \mathtt{not\ p}$ iff $G, s \not\models_{\text{GDL-II}} \mathtt{p}$;
$G \models_{\text{GDL-II}} (\Leftarrow (\mathtt{init\ p}))$ iff $G, s_0 \models_{\text{GDL-II}} \mathtt{p}$;
$G \models_{\text{GDL-II}} (\Leftarrow (\mathtt{p})(\mathtt{e_1}) \ldots (\mathtt{e_m}))$
iff for all s : (if for each $i \in [1 \ldots m]$. $G, s \models_{\text{GDL-II}} \mathtt{e_i}$, then $G, s \models_{\text{GDL-II}} \mathtt{p}$);
$G \models_{\text{GDL-II}} (\Leftarrow (\mathtt{next\ p})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ i_k\ a_k}))$
iff for all s, t : ((if for each $i \in [1 \ldots m]$. $G, s \models_{\text{GDL-II}} \mathtt{e_i}$ and
t is an $i_1 : a_1, \ldots, i_k : a_k$ successor of s), then $G, t \models_{\text{GDL-II}} \mathtt{p}$);
$G \models_{\text{GDL-II}} (\Leftarrow (\mathtt{sees\ x\ t})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ i_k\ a_k}))$
iff for all s, t : ((if for each $i \in [1 \ldots m]$. $G, s \models_{\text{GDL-II}} \mathtt{e_i}$ and
t is an $i_1 : a_1, \ldots, i_k : a_k$ successor of s), then $G, t \models_{\text{GDL-II}} (\mathtt{sees\ x\ t})$).

With this definition, we ensure that the constructed game model $G \models_{\text{GDL-II}} \Gamma$.

1.2 ATL

Alternating-time Temporal Logic, also known as ATL, is a branching-time logic that extends CTL to multiple agents, where it describes many possible paths for the future, one of which will be the actual path [3]. The key element of ATL is the *cooperation modalities* $\langle\langle A\rangle\rangle$, where $A \subseteq \mathbb{A}\text{gt}$ is a set of agents. Another key formula of ATL is $\langle\langle A\rangle\rangle\gamma$, where γ is a temporal formula. This formula expresses that the coalition (set of agents) A has a *collective strategy* such that γ is true. A temporal formula is built using unary operators - $\bigcirc$ meaning 'in the next state', $\Box$ meaning 'always', $\Diamond$ meaning 'eventually' - and a binary operator $\mathcal{U}$ meaning 'until' [4]. In this paper, we focus on the language of ATL only, meaning that each temporal operator must be directly following a cooperation modality.

Definition 1.9 (Language $\mathscr{L}_{\text{ATL}}$ [1])**.**
Let Ag be a set of agents and Φ be a set of atomic propositions, the language $\mathscr{L}_{\text{ATL}}$ is given by all formulae generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A\rangle\rangle \bigcirc \varphi \mid \langle\langle A\rangle\rangle \Box\varphi \mid \langle\langle A\rangle\rangle \varphi \,\mathcal{U} \varphi$$

where $p \in \Phi$ is a atomic proposition and $A \subseteq Ag$ is a set of agents.

1.2.1 ATL Semantics: iCGS

In this paper, the semantics of ATL Language $\mathscr{L}_{\text{ATL}}$ is defined over a variant of transition systems - *imperfect information concurrent game structure* (iCGS), which links each transition with a set of actions performed, one by each agent. This section refers to iCGS presented in [1] and the ATL semantics refers to the ATL semantics in [4] adapted to the definition of *strategies* in [1].

Definition 1.10 (iCGS)**.**
An iCGS is a tuple $\mathfrak{M} = \langle Ag, St, \Pi, \pi, Act, d, o, \{\sim_i\}_{i \in Ag}\rangle$ where

- Ag is a finite non-empty set of all agents involved in the system;
- St is a finite non-empty set of states;
- Π is a finite non-empty set of atomic propositions;
- π is a valuation function $\pi : \Pi \to \mathscr{P}(St)$ that returns a set of states in which a specific atomic proposition is satisfied;
- Act is a finite non-empty set of actions;
- $d : Ag \times St \to \mathscr{P}(Act)$ is a function that returns a finite non-empty set of available actions for a specific agent in a specific state. We also use $d_a(q)$ for short for $d(a, q)$;
- o is the state transition function that defines the outcome state ($q' = o(q, a_1, \ldots, a_k)$) to a state when a joint move (one action per agent) is carried out;
- each indistinguishability relations $\sim_i \subseteq St \times St$ represent the uncertainty for the specific agent: $q_1 \sim_i q_2$ means that agent i cannot distinguish between state q_1 and state q_2.

The indistinguishability relations are very important as it restricts the semantics of the system - the agent should have the exact same choices in two states that are indistinguishable, which would result in a certain strategy returning the same outcome as the agent has the same information and is bound to play the same action.

Remark. A concurrent game structure (CGS) under perfect information can be thought of as an iCGS where the indistinguishability relation for each agent is the identity relations that takes two states and returns true only when the two input states are identical.

When it comes to describe the strategy for agents in iCGS, an *imperfect information strategy*, or *uniform strategy*, of an agent is a sequence of actions but with the agent's epistemic limitations [1]. This means that the choices for the agent in indistinguishable states must be the same, in order for a strategy to be executable. For our iCGS, we assume all the choices that the agents make are based on the current state only (not including *histories* - previous states). Therefore, we restrict the possible strategies by the following:

- An *imperfect information memoryless strategy* (*ir*-strategy) for an agent i is an uniform Ir-strategy, which is given by a function $s_i : St \to Act$, where $s_i(q) \in d_i(q)$, and satisfies the following constraints: if $q \sim_i q'$ then $s_i(q) = s_i(q')$.
- A *collective ir-strategy* is a combination of individual *ir*-strategies. $\sum_i^{ir}$ denotes the set of the agent i's *ir*-strategies. The set of A's collective imperfect information memoryless strategies is denoted by $\sum_A^{ir} = \prod_{i \in A} \sum_i^{ir}$. $\sum^{ir} = \sum_{Ag}^{ir}$ gives the set of all strategy profiles of the system.

Function $out(q, s_A)$ returns the set of all future paths that may occur as a result of the agents A execute strategy s_A from the state q and onward. This set is given by:

$$\begin{aligned} out(q, s_A) \quad = \quad & \{\lambda = q_0 q_1 q_2 \ldots \mid q_0 = q \text{ and for all } i = 1, 2, \ldots \text{ there exists a tuple of agents'} \\ & \text{decisions } \langle \alpha_{a_1}^{i-1}, \ldots, \alpha_{a_k}^{i-1} \rangle \text{ such that for all } a \in \mathbb{A}\text{gt. } \alpha_a^{i-1} \in d_a(q_{i-1}), \text{ and} \\ & \text{for all } a \in A.\ \alpha_a^{i-1} = s_A \mid_a (q_{i-1}), \text{ and } o(q_{i-1}, \alpha_{a_1}^{i-1}, \ldots, \alpha_{a_k}^{i-1}) = q_i\} \end{aligned}$$

and let $\lambda[i]$ denote the ith position on path λ, starting from $i = 0$ ($\lambda[i] = q_i$).

We also define the notion of image, where $img(q, \rho) = \{q' \mid \rho(q, q')\}$ is the image of a state q wrt a binary relation ρ.

We can now define the satisfaction relation '$\models_{\text{ATL}}$' for ATL between pairs of iCGS $\mathfrak{M}$, state q, and formulae of ATL.

Definition 1.11 (ATL Semantics)**.**
Given an iCGS $\mathfrak{M}$, a state q, the semantics of ATL is defined as follows, where $\sim_A := \cup_{a \in A} \sim_a$:

$\mathfrak{M}, q \models_{\text{ATL}} p$ iff $q \in \pi(p)$, where $p \in \Pi$;
$\mathfrak{M}, q \models_{\text{ATL}} \neg\varphi$ iff $\mathfrak{M}, q \not\models_{\text{ATL}} \varphi$;
$\mathfrak{M}, q \models_{\text{ATL}} \varphi \vee \psi$ iff $\mathfrak{M}, q \models_{\text{ATL}} \varphi$ or $\mathfrak{M}, q \models_{\text{ATL}} \psi$;
$\mathfrak{M}, q \models_{\text{ATL}} \langle\langle A \rangle\rangle \bigcirc \varphi$
iff there is an ir-strategy $s_A \in \sum_A^{ir}$ such that for each $q' \in img(q, \sim_A)$ and every $\lambda \in out(s_A, q')$, we have $\mathfrak{M}, \lambda[1] \models_{\text{ATL}} \varphi$;
$\mathfrak{M}, q \models_{\text{ATL}} \langle\langle A \rangle\rangle \Box\varphi$
iff there is an ir-strategy $s_A \in \sum_A^{ir}$ such that for each $q' \in img(q, \sim_A)$ and every $\lambda \in out(s_A, q')$, we have $\mathfrak{M}, \lambda[u] \models_{\text{ATL}} \varphi$ for all $u \in \mathbb{N}$;
$\mathfrak{M}, q \models_{\text{ATL}} \langle\langle A \rangle\rangle \varphi \,\mathcal{U} \psi$
iff there is an ir-strategy $s_A \in \sum_A^{ir}$ such that for each $q' \in img(q, \sim_A)$ and every $\lambda \in out(s_A, q')$, there exists some $u \in \mathbb{N}$ such that $\mathfrak{M}, \lambda[u] \models_{\text{ATL}} \psi$, and for all $0 \leq v < u$, we have $\mathfrak{M}, \lambda[v] \models_{\text{ATL}} \varphi$.

The remaining classical logic connectives $\wedge$, $\rightarrow$, $\leftrightarrow$ are defined as ..., and $\langle\langle A \rangle\rangle \Diamond \varphi$ is defined as $\langle\langle A \rangle\rangle \top \,\mathcal{U} \varphi$. For better readability, we only include the agents in the coalition $\langle\langle \rangle\rangle$, not the set brackets.

2 Linking GDL-II and iCGS

In this section, we will see how GDL-II description can be linked to iCGS. There are two links that can be built between GDL-II and iCGS. Suppose we have any game G that involves chances

(randomness) and imperfect information for the players, with its GDL-II game description Γ. On the semantics level, we associate the game model of Γ, denoted by G_Γ, to an ATL model - an imperfect information concurrent game structure (iCGS) $\mathcal{A}_\Gamma = \mathcal{T}_{sem}(G_\Gamma)$. On the syntactic level, we associate the GDL-II game description Γ, denoted by $\Gamma_{\text{GDL-II}}$, to an ATL theory Γ_{ATL}. This entire section refers to the two links built in [4] with changes to adapt to the link between GDL-II from [5] and iCGS from [1].

2.1 From GDL-II Game Models to ATL iCGS: $\mathcal{T}_{sem}$

Suppose we have a game model G for a GDL-II description Γ, if we want to interpret ATL formulae over this model, we need to transform it into an iCGS on which we are able to interpret ATL formulae. Therefore, given a GDL-II game model $G = \langle S, s_0, Ag, \{Ac_i\}_{i\in Ag}, \{\sim_i\}_{i\in Ag}, \tau, \pi\rangle$ for a GDL-II game description Γ and a set of atomic propositions $At_{\text{GDL-II}}$, we define an associated iCGS $\mathcal{T}_{sem}(G) = \mathcal{A}_\Gamma = \langle Ag, St, \Pi, \pi', Act, d, o, \{\sim'_i\}_{i\in Ag}\rangle$ with the same sets of agents (Ag). The set of atomic propositions Π is constructed as follows. Def. 2.1 follows directly from Def. 4.1 in [4].

Definition 2.1 (Translation t and t_{old})**.**
We define a translation $t : At_{\text{GDL-II}} \to At_{\text{ATL}}$, where each atom from $At_{\text{GDL-II}}$ is associated with an atom from At_{ATL}.

$$\begin{array}{lll} t(\texttt{p}) & = p\,(p \in Prim) & t(\texttt{goal i v}) = goal(i, v) \\ t(\texttt{legal i a}) & = legal(i, a) & t(\texttt{terminal}) = terminal \\ t(\texttt{distinct s}_1\ \texttt{s}_2) & = distinct(s_1, s_2) & \end{array}$$

Let t_{old} be: $t_{old}(\texttt{p}) = t(\texttt{p})_{old} = p_{old}$, for any $\texttt{p} \in At_{\text{GDL-II}}$.

Now, we add the following types of atomic propositions to Π:

1. Atoms that represent the current state of the game: for each p in $At_{\text{GDL-II}}$, add $t(\texttt{p})$ to Π.
2. Atoms that represent the previous state of the game: for each p in $At_{\text{GDL-II}}$, add $t(\texttt{p})_{old}$ to Π.
3. Atoms that represent the actions performed by agents during the transition from the previous state to the current state: add atom $done(i, a)$ to Π for each (does i a) and add atom $sees(x, t)$ to Π for each (sees x t).
4. Atoms that determine the initial and end states of the game: add *init* to represent initial state and a special atom $\texttt{s}_\perp$. This atom $\texttt{s}_\perp$ represents what we call as 'sink states' - we add these to $\mathcal{A}_\Gamma$ to make it a proper iCGS. The role of these states is to represent the only successor of a terminal state and of the sink state itself, whereas in a game model G, a terminal state does not have any successors.

The other elements of $\mathcal{A}_\Gamma$ are:

- $St = St_1 \cup St_2$, where $St_1 = S$ and $St_2 = \{\texttt{s}_q \mid q \in St_1\}$ which contains sink states. Here, for $\texttt{s}_q \in St_2$, the atom $\texttt{s}_\perp$ is true and the atoms $done(i, fin_i)$ are true, but for the other atoms, $\texttt{s}_q$ is exactly the same as q;
- $\pi' : \Pi \to \mathscr{P}(St)$ is a valuation function, which for the inputting atomic proposition $\texttt{p} \in \Pi$, returns the set $\{q \mid G, q \models_{\text{GDL-II}} \texttt{p}, q \in St_1\}$. In order to match the semantics of the initial state, we stipulate $\pi'(init) = \{s_0\}$;
- $Act = Ac_1 \cup \{fin_1\} \cup \ldots \cup Ac_n \cup \{fin_n\}$, where $Ac_i \in \{Ac_i\}_{i\in Ag}$;

- $d : Ag \times St \to \mathscr{P}(Act)$ takes a specific agent i and a specific state q, and returns the set $\{\mathtt{a} \mid G, q \models_{\text{GDL-II}} (\mathtt{legal\ i\ a}), \mathtt{a} \in Ac_i \cup \{fin_i\}\}$ as output. Note that two indistinguishable states for an agent i have the same set of legal actions: $d_i(q) = d_i(q')$, for all $q' \sim_i q$. Also note that according to our definition, the only action allowed for any agent i in `terminal` and $\mathtt{s}_\perp$ is fin_i, i.e. $d_i(\mathtt{terminal}) = d_i(\mathtt{s}_\perp) = \{fin_i\}$ for any agent $i \in Ag$;

- o is based on τ. We keep all the mappings in τ and add: $o(q, fin_1, \ldots, fin_n) = \mathtt{s}_q$, for all $q \in St_1$ such that $G, q \models_{\text{GDL-II}} \mathtt{terminal}$;

- the family of indistinguishability relations $\{\sim'_i\}_{i \in Ag}$ is based on $\{\sim_i\}_{i \in Ag}$. We keep the existing definition (i.e. the three conditions listed in Section 1.1.3) for existing states (i.e. St_1) and add the following for each agent $i \in Ag$:

  – for any $\mathtt{s}_q$, $\mathtt{s}_{q'} \in St_2$, $\mathtt{s}_q \sim'_i \mathtt{s}_{q'}$ iff $q \sim_i q'$.

  The reason for this is that the other atoms - $\mathtt{s}_\perp$ and $done(i, fin_i)$ - are always true in any $\mathtt{s}_q$, and all other atoms remain the same as in q, so the indistinguishability relation between $\mathtt{s}_q$ and $\mathtt{s}_{q'}$ is determined by q and q'.

Now, with a given GDL-II game description Γ, we can construct two game models G and $\mathcal{A}_\Gamma$. In order to show that they satisfy all the game rules defined by Γ, we need to first define a translation from GDL-II rules to ATL formulae.

Definition 2.2 (Translation from GDL-II rules to ATL formulae)**.**
Let Γ be a GDL-II game description. We define a translation from any GDL-II rules in $\Gamma_{\mathtt{init}} \cup \Gamma_{\mathtt{glob}} \cup \Gamma_{\mathtt{next}} \cup \Gamma_{\mathtt{sees}}$ to ATL formulae $\mathcal{R} : \text{GDL-II} \to \text{ATL}$ as follows:

- $\mathcal{R}(\Leftarrow (\mathtt{init\ p})) = init \to t(\mathtt{p})$

- $\mathcal{R}(\Leftarrow (\mathtt{p})(\mathtt{e_1}) \ldots (\mathtt{e_m})) = \langle\langle\rangle\rangle\Box(\neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \to t(\mathtt{p}))$

- $\mathcal{R}(\Leftarrow (\mathtt{next\ p})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ i_k\ a_k})) = \langle\langle\rangle\rangle\Box(\neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \to \langle\langle\{i_1, \ldots, i_k\}\rangle\rangle \bigcirc (t(\mathtt{p}) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k)))$ with the special case that:

- $\mathcal{R}(\Leftarrow (\mathtt{next\ p})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ random\ a_r}) \ldots (\mathtt{does\ i_k\ a_k})) = \langle\langle\rangle\rangle\Box(\neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \to \langle\langle\{i_1, \ldots, i_k, \mathtt{random}\}\rangle\rangle \bigcirc (t(\mathtt{p}) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k) \wedge done(random, a_r)))$

- $\mathcal{R}(\Leftarrow (\mathtt{sees\ x\ t})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ i_k\ a_k})) = \langle\langle\rangle\rangle\Box(\neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \to \langle\langle\{i_1, \ldots, i_k\}\rangle\rangle \bigcirc (sees(x, t) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k)))$ also with the special case that:

- $\mathcal{R}(\Leftarrow (\mathtt{sees\ x\ t})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ random\ a_r}) \ldots (\mathtt{does\ i_k\ a_k})) = \langle\langle\rangle\rangle\Box(\neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \to \langle\langle\{i_1, \ldots, i_k, \mathtt{random}\}\rangle\rangle \bigcirc (sees(x, t) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k) \wedge done(random, a_r)))$

where $t : At_{\text{GDL-II}} \to At_{\text{ATL}}$ is defined as in Def. 2.1, and for any GDL-II expression $\mathtt{e_i}$, we specify: $\mathcal{R}(\mathtt{e_i}) = t(\mathtt{p})$ if $\mathtt{e_i} = \mathtt{p}$ or $\mathtt{true\ p}$, and $\mathcal{R}(\mathtt{e_i}) = \neg t(\mathtt{p})$ if $\mathtt{e_i} = \mathtt{not\ p}$ or $\mathtt{not(true\ p)}$. For the special keyword `sees`, we stipulate: $\mathcal{R}(\mathtt{sees\ x\ t}) = sees(x, t)$.

Theorem 2.1
Given a GDL-II game description Γ, we can construct its game model G_Γ and the associated iCGS $\mathcal{A}_\Gamma$. For each rule $r \in \Gamma_{\mathtt{init}} \cup \Gamma_{\mathtt{glob}} \cup \Gamma_{\mathtt{next}} \cup \Gamma_{\mathtt{sees}}$, each $s \in S$ and $\mathtt{e} \in AtExpr_{\text{GDL-II}}$, we have:

$$G_\Gamma, s \models_{\text{GDL-II}} \mathtt{e} \text{ iff } \mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathcal{R}(\mathtt{e}) \text{ and } G_\Gamma \models_{\text{GDL-II}} r \text{ iff } \mathcal{A}_\Gamma \models_{\text{ATL}} \mathcal{R}(r)$$

Proof. The proof of the theorem is done by induction on GDL-II rules.

We first prove the base case for $\mathtt{e} \in AtExpr_{\text{GDL-II}}$.

$G_\Gamma, s \models_{\text{GDL-II}} \mathtt{p}$ iff $\mathtt{p} \in \pi(s)$ (by Definition 1.8)
iff $s \in \pi'(\mathtt{p})$
iff $\mathcal{A}_\Gamma, s \models_{\text{ATL}} t(\mathtt{p})$ (by Definition 1.11)
iff $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathcal{R}(\mathtt{p})$ (by Definition 2.2)

$G_\Gamma, s \models_{\text{GDL-II}} \mathtt{not\ p}$ iff $G_\Gamma, s \not\models_{\text{GDL-II}} \mathtt{p}$ (by Definition 1.8)
iff $\mathtt{p} \notin \pi(s)$ (by Definition 1.8)
iff $s \notin \pi'(\mathtt{p})$
iff $\mathcal{A}_\Gamma, s \not\models_{\text{ATL}} t(\mathtt{p})$ (by Definition 1.11)
iff $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \neg t(\mathtt{p})$ (by Definition 1.11)
iff $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathcal{R}(\mathtt{p})$ (by Definition 2.2)

The cases for the other possibilities of $\mathtt{e}$ can be think of as a special form of GDL-II rules with empty body, and we will prove the equivalence between the two models over their corresponding semantics regarding those rules next.

Moving onto the GDL-II rules r, we start with $\Gamma_{\mathtt{init}}$. We have that

$G_\Gamma \models_{\text{GDL-II}} (\Leftarrow (\mathtt{init\ p}))$ iff $G_\Gamma, s_0 \models_{\text{GDL-II}} \mathtt{p}$ (by Definition 1.8)
iff $\mathtt{p} \in \pi(s_0)$ (by Definition 1.8)
iff $s_0 \in \pi'(\mathtt{p})$
iff $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} \mathtt{p}$ (by Definition 1.11)
iff for all $s \in St$, if $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathit{init}$ then $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathtt{p}$
iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \mathit{init} \rightarrow t(\mathtt{p})$
iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \mathcal{R}(\Leftarrow (\mathtt{init\ p}))$ (by Definition 2.2)

For $\Gamma_{\mathtt{glob}}$, we apply induction and our inductive hypothesis is that for all states s and each $\mathtt{e_i} \in \{\mathtt{e_1}, \ldots, \mathtt{e_m}\}$, we have already proven $G_\Gamma, s \models_{\text{GDL-II}} \mathtt{e_i}$ iff $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathcal{R}(\mathtt{e_i})$. We thus have

$G_\Gamma \models_{\text{GDL-II}} (\Leftarrow (\mathtt{p})(\mathtt{e_1}) \ldots (\mathtt{e_m}))$
iff for all $s \in S$: (if for each $i \in [1 \ldots m]$. $G_\Gamma, s \models_{\text{GDL-II}} \mathtt{e_i}$, then $G_\Gamma, s \models_{\text{GDL-II}} \mathtt{p}$) (by Definition 1.8)
iff for all $s \in St_1$: (if for each $i \in [1 \ldots m]$. $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathcal{R}(\mathtt{e_i})$, then $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathcal{R}(\mathtt{p})$) (by Inductive Hypothesis)
iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow \mathcal{R}(\mathtt{p})$ ($\neg \mathtt{s}_\perp$ always true as $\mathtt{s}_\perp \notin St_1$)
iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow t(\mathtt{p})$ (by Definition 2.2)
iff for any arbitrary state s, we have $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow t(\mathtt{p})$ for all agents $i \in Ag$
iff there is an ir-strategy $s_{Ag} \in \sum^{ir}_{Ag}$ such that for each $s' \in img(s, \sim_{Ag})$ and every $\lambda \in out(s_{Ag}, s')$, $\mathcal{A}_\Gamma, \lambda[u] \models_{\text{ATL}} \neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow t(\mathtt{p})$ for all $u \in \mathbb{N}$, where $\sim_{Ag} := \cup_{i \in Ag} \sim_i$ (by arbitrary choice of s to be $\lambda[u]$)
iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \langle\langle\rangle\rangle \Box (\neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow t(\mathtt{p}))$ (by Definition 1.11)
iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \mathcal{R}(\Leftarrow (\mathtt{p})(\mathtt{e_1}) \ldots (\mathtt{e_m}))$ (by Definition 2.2)

Similarly, for $\Gamma_{\mathtt{next}}$ and $\Gamma_{\mathtt{sees}}$, we use induction and the same inductive hypothesis - for all states s and each $\mathtt{e_i} \in \{\mathtt{e_1}, \ldots, \mathtt{e_m}\}$, we have already proven $G_\Gamma, s \models_{\text{GDL-II}} \mathtt{e_i}$ iff $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathcal{R}(\mathtt{e_i})$. In the below, we use A to denote the set of agents $\{i_1, \ldots, i_k\}$ that appear in the body of the rules.

$G_\Gamma \models_{\text{GDL-II}} (\Leftarrow (\mathtt{next\ p})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ i_k\ a_k}))$
iff for all $s, t \in S$: ((if for each $i \in [1 \ldots m]$. $G_\Gamma, s \models_{\text{GDL-II}} \mathtt{e_i}$ and t is an $i_1 : a_1, \ldots, i_k : a_k$ successor of s), then $G_\Gamma, t \models_{\text{GDL-II}} \mathtt{p}$) (by Definition 1.8)
iff for all $s, t \in St_1$: ((if for each $i \in [1 \ldots m]$. $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \mathcal{R}(\mathtt{e_i})$ and $o(s, a_1, \ldots, a_k, \ldots, a_n) = t$, then $\mathcal{A}_\Gamma, t \models_{\text{ATL}} \mathcal{R}(\mathtt{p})$), where $\{a_{k+1}, \ldots, a_n\}$ are actions performed by other agents $i' \in Ag \backslash A$ (by Inductive Hypothesis)
iff for any arbitrary state s and a corresponding state t, we have

$\mathcal{A}_\Gamma \models_{\text{ATL}} \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \wedge o(s, a_1, \ldots, a_k, \ldots, a_n) = t \rightarrow \mathcal{R}(\mathtt{p})$, where $\{a_{k+1}, \ldots, a_n\}$ are actions performed by other agents $i' \in Ag \backslash A$

iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m})$ and there is an ir-strategy $s_A \in \sum_A^{ir}$ such that for each $s' \in img(s, \sim_A)$ and every $\lambda \in out(s_A, s')$, we have $\mathcal{A}_\Gamma, \lambda[1] \models_{\text{ATL}} \mathcal{R}(\mathtt{p}) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k))$, where $\sim_A := \cup_{a \in A} \sim_a$ (by t being $\lambda[1]$ for any arbitrary choice of s and indistinguishable states and uniform strategies mean that the agents are bound to perform the same action)

iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow \langle\langle\{i_1, \ldots, i_k\}\rangle\rangle \bigcirc (\mathcal{R}(\mathtt{p}) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k))$ (by Definition 1.11)

iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow \langle\langle\{i_1, \ldots, i_k\}\rangle\rangle \bigcirc (t(\mathtt{p}) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k))$ (by Definition 2.2)

iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow \langle\langle\{i_1, \ldots, i_k\}\rangle\rangle \bigcirc (t(\mathtt{p}) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k))$ ($\neg \mathtt{s}_\perp$ always true as $\mathtt{s}_\perp \notin St_1$)

iff for any arbitrary state s, we have $\mathcal{A}_\Gamma, s \models_{\text{ATL}} \neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow \langle\langle\{i_1, \ldots, i_k\}\rangle\rangle \bigcirc (t(\mathtt{p}) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k))$, for all agents $i \in Ag$

iff there is an ir-strategy $s_{Ag} \in \sum_{Ag}^{ir}$ such that for each $s' \in img(s, \sim_{Ag})$ and every $\lambda \in out(s_{Ag}, s')$, $\mathcal{A}_\Gamma, \lambda[u] \models_{\text{ATL}} \neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow \langle\langle\{i_1, \ldots, i_k\}\rangle\rangle \bigcirc (t(\mathtt{p}) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k))$ for all $u \in \mathbb{N}$, where $\sim_{Ag} := \cup_{i \in Ag} \sim_i$ (by arbitrary choice of s to be $\lambda[u]$)

iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \langle\langle\rangle\rangle \Box (\neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow \langle\langle\{i_1, \ldots, i_k\}\rangle\rangle \bigcirc (t(\mathtt{p}) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k)))$ (by Definition 1.11)

iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \mathcal{R}(\Leftarrow (\mathtt{next\ p})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ i_k\ a_k}))$ (by Definition 2.2)

$G_\Gamma \models_{\text{GDL-II}} (\Leftarrow (\mathtt{sees\ x\ t})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ i_k\ a_k}))$

iff for all s, t : ((if for each $i \in [1 \ldots m]$. $G_\Gamma, s \models_{\text{GDL-II}} \mathtt{e_i}$ and t is an $i_1 : a_1, \ldots, i_k : a_k$ successor of s), then $G_\Gamma, t \models_{\text{GDL-II}} (\mathtt{sees\ x\ t})$) (by Definition 1.8)

iff ... (same as for $\Gamma_{\mathtt{sees}}$ but with $sees(x, t)$ replaced for $t(\mathtt{p})$ by Definition 2.2)

iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \langle\langle\rangle\rangle \Box (\neg \mathtt{s}_\perp \wedge \mathcal{R}(\mathtt{e_1}) \wedge \ldots \wedge \mathcal{R}(\mathtt{e_m}) \rightarrow \langle\langle\{i_1, \ldots, i_k\}\rangle\rangle \bigcirc (sees(x, t) \wedge done(i_1, a_1) \wedge \ldots \wedge done(i_k, a_k)))$ (by Definition 1.11)

iff $\mathcal{A}_\Gamma \models_{\text{ATL}} \mathcal{R}(\Leftarrow (\mathtt{sees\ x\ t})(\mathtt{e_1}) \ldots (\mathtt{e_m})(\mathtt{does\ i_1\ a_1}) \ldots (\mathtt{does\ i_k\ a_k}))$ (by Definition 2.2)

□

2.2 From GDL-II Game Descriptions to ATL formulae: $\mathcal{T}_{syn}$

Now we move to the syntactic level link between a GDL-II game description $\Gamma_{\text{GDL-II}}$ and an ATL theory $\Gamma_{\text{ATL}} = \mathcal{T}_{syn}(\Gamma_{\text{GDL-II}})$, where the game description and the ATL theory characterise the same game. The syntactic translation in this section is taken form that presented in [4] with modifications so it adapts to GDL-II game descriptions.

Given $\Gamma_{\text{GDL-II}}$, assuming it comes with a finite set of atomic propositions, we define the ATL theory Γ_{ATL} as a conjunction of ATL formulae:

$$\Gamma_{\text{ATL}} = DIST \wedge INIT \wedge MEM \wedge ONE\_DONE \wedge LEGAL \wedge STRAT \wedge SEES \wedge TERM \wedge SINK$$

The above conjuncts each plays a different role. *DIST* describes the `distinct` keyword. *INIT* represents the initial state. Moreover, *MEM* is used to remember the previous state; *ONE_DONE* and *LEGAL* make sure that for each non-terminal state and for each agent, there is one legal action performed in that state. Combined with these three conjuncts, *STRAT* computes the next state

given the previous state and all actions performed by the agents. Lastly, *TERM* and *SINK* guarantee that all terminal states transit to the special sink state.

Now, we define these conjuncts. Firstly, we specify the intended meaning of the atom $distinct(\cdot,\cdot)$ as follows:

$$DIST = \langle\langle\rangle\rangle\Box distinct(t_1, t_2) \text{ for all terms that are syntactically different.}$$

Let $S_0 = DatlogPMod(\delta(\Gamma_{\texttt{init}}) \cup \delta(\Gamma_{\texttt{glob}}))$, which returns the set of atomic propositions that need to be satisfied from all the global rules and all the (`init p`) expressions. We now want an ATL formula that describes the full initial state. Consider:

$$INIT = init \wedge \langle\langle\rangle\rangle \bigcirc \langle\langle\rangle\rangle\Box\neg init \wedge P_{S_0} \wedge \bigwedge_{p_{old} \in At_{\text{ATL}}} \neg p_{old} \wedge N_{done} \wedge \neg \mathtt{s}_{\perp}$$

where

$$P_{S_0} = \bigwedge_{\mathtt{p} \in S_0} p \wedge \bigwedge_{\mathtt{p} \notin S_0} \neg p$$

and

$$N_{done} = \bigwedge_{i \in Ag} \bigwedge_{a \in Ac_i \cup \{fin_i\}} \neg done(i, a).$$

This formula makes sure that the special atom *init* we defined is only true in the initial state (false in all following states), and that the other atoms in the initial state of the game model G have the same truth value here. Also, the formula ensures that all the *old* and *done* propositions are false as there is no previous state to the initial state. It also shows that the initial state is not a sink state.

We record the old - previous - state, including the previous truth values of all terms in a state (p_{old}) by the *MEM* conjunct:

$$MEM = \langle\langle\rangle\rangle\Box \bigwedge_{\mathtt{p} \in At_{\text{ATL}}} ((t(\mathtt{p}) \wedge \neg terminal \rightarrow \langle\langle\rangle\rangle \bigcirc t(\mathtt{p})_{old}) \wedge (\neg t(\mathtt{p}) \wedge \neg terminal \rightarrow \langle\langle\rangle\rangle \bigcirc \neg t(\mathtt{p})_{old}))$$

This formula ensures that for all non-terminal states, the truth value of the propositions in the current state is reflected properly in the next state.

The next conjunct ensures that for all non-initial states, exactly one action is performed by each agent:

$$ONE\_DONE = \langle\langle\rangle\rangle\Box(\neg init \rightarrow \bigwedge_{i \in Ag} XOR_{a \in Ac_i \cup \{fin_i\}} done(i, a))$$

where *XOR* is the *exclusive* OR operator from classical logic - a Boolean operator that returns true if and only if exactly one of the operands is true.

Each GDL-II game description makes the assumption that each agent must only perform legal actions in the game, and this is captured by:

$$LEGAL = \langle\langle\rangle\rangle\Box \bigwedge_{i \in Ag, a_i \in Ac_i} ((legal(i, a_i) \wedge \neg terminal) \leftrightarrow \langle\langle i\rangle\rangle \bigcirc done(i, a_i))$$

This formula enforces that an agent i should have a strategy to guarantee that it can perform an action a_i when the action is legal and the current state is not a terminal state. In particular, for $done(\cdot,\cdot)$ atoms, we have the following equivalence:

$$\models_{\text{ATL}} \langle\langle i\rangle\rangle \bigcirc done(i, a_i) \leftrightarrow \langle\langle Ag\rangle\rangle \bigcirc done(i, a_i)$$

Now, let $bd_1, bd_2, \ldots$ be the variables over possible bodies of rules, i.e. the set of literals not including any (`does i a`) term. Also let $\mathtt{p} \in At_{\text{GDL-II}}$. Suppose that all the rules r in $\Gamma_{\text{GDL-II}}$ with $hd(r) \in \{(\mathtt{p}), (\mathtt{next\ p})\}$ are the following:

$$
\begin{array}{lll}
r_1 : \Leftarrow (\mathtt{p}) & bd_1 & \\
\vdots \quad \vdots \quad \vdots & \vdots & \\
r_h : \Leftarrow (\mathtt{p}) & bd_h & \\
s_1 : \Leftarrow (\mathtt{next\ p}) & bd'_1 & (\mathtt{does\ i}_{1_1}\ \mathtt{a}_{1_1}) \ldots (\mathtt{does\ i}_{1_m}\ \mathtt{a}_{1_m}) \\
\vdots \quad \vdots \quad \vdots & \vdots \quad \vdots \quad \vdots & \\
s_k : \Leftarrow (\mathtt{next\ p}) & bd'_k & (\mathtt{does\ i}_{k_1}\ \mathtt{a}_{k_1}) \ldots (\mathtt{does\ i}_{k_m}\ \mathtt{a}_{k_m})
\end{array}
$$

We map all the rules for $\mathtt{p}$ to an ATL formula $\varphi(\mathtt{p})$. To do this, we first translate the GDL-II symbols to ATL symbols using the functions t and t_{old} defined in Definition 2.1. For better readability, we use $t(bd_i)$ and $t_{old}(bd_i)$ to denote the translation of all propositions in bd_i by t and t_{old} correspondingly. For each atom $\mathtt{p} \in At_{\text{GDL-II}}$, we define an constraint $MIN(\mathtt{p})$ as:

$$
MIN(\mathtt{p}) = t(\mathtt{p}) \leftrightarrow \left(\bigvee_{i \leq h} t(bd_i) \vee \bigvee_{j \leq k} (t_{old}(bd'_j) \wedge done(i_{j_1}, a_{j_1}) \wedge \ldots \wedge done(i_{j_m}, a_{j_m})) \right)
$$

And if $\mathtt{p}$ does not appear in the head of any rule in $\Gamma_{\text{GDL-II}}$, we define $MIN(\mathtt{p}) = \neg p$.

Now, we can represent the semantics of stratified program $\Gamma_{\text{GDL-II}}$ by the $STRAT$ conjunct:

$$
STRAT = \langle\langle\rangle\rangle\Box \bigwedge_{\mathtt{p} \in At_{\text{GDL-II}}} (\neg init \wedge \neg \mathtt{s}_\perp \rightarrow MIN(\mathtt{p}))
$$

Suppose that all the rules r in $\Gamma_{\text{GDL-II}}$ with $hd(r) = (\mathtt{sees\ i\ t})$ are the following:

$$
\begin{array}{lll}
v_1 : \Leftarrow (\mathtt{sees\ i\ t}) & bd''_1 & (\mathtt{does\ i}_{1_1}\ \mathtt{a}_{1_1}) \ldots (\mathtt{does\ i}_{1_m}\ \mathtt{a}_{1_m}) \\
\vdots \quad \vdots \quad \vdots & \vdots \quad \vdots \quad \vdots & \\
v_n : \Leftarrow (\mathtt{sees\ i\ t}) & bd''_n & (\mathtt{does\ i}_{n_1}\ \mathtt{a}_{n_1}) \ldots (\mathtt{does\ i}_{n_m}\ \mathtt{a}_{n_m})
\end{array}
$$

The $SEES$ conjunct characterises when $hd(r)$ is satisfied. Similarly, we need to first translate the GDL-II symbols to ATL symbols using the functions t and t_{old} defined in Definition 2.1. We thus have:

$$
SEES = sees(i, t) \leftrightarrow \bigvee_{l \leq n} (t_{old}(bd''_l) \wedge done(i_{l_1}, a_{l_1}) \wedge \ldots \wedge done(i_{l_m}, a_{l_m}))
$$

From our semantic link between GDL-II game model and iCGS, we know that when reaching a terminal state, no further actual actions are played by agents. Instead, all agents can only and they always play the fin_i actions:

$$
TERM = \langle\langle\rangle\rangle\Box \left((terminal \vee \mathtt{s}_\perp) \leftrightarrow \langle\langle\rangle\rangle \bigcirc (\mathtt{s}_\perp \wedge \bigwedge_{i \in Ag} done(i, fin_i)) \right)
$$

When a sink state is reached, everything should reamin the same:

$$
SINK = \langle\langle\rangle\rangle\Box \bigwedge_{p \in At_{\text{ATL}}} (((terminal \vee \mathtt{s}_\perp) \wedge p) \leftrightarrow \langle\langle\rangle\rangle \bigcirc (\mathtt{s}_\perp \wedge p))
$$

In Section 2.1, we saw that we can transform a GDL-II game model into an iCGS. We now prove the soundness result for our $\mathcal{T}_{sem}$ translation. Suppose we have a GDL-II game description Γ and its corresponding game model G with initial state s_0; $\mathcal{A}_\Gamma$ being the iCGS constructed, we have the following:

$$
\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} \Gamma_{\text{ATL}}
$$

Proof. As Γ_{ATL} is defined as a conjunction of ATL formulae, we prove the above result by showing $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}}$ each conjunct.

Starting with *DIST*, for all possible path starting from the initial state, we have $distinct(t_1, t_2)$ from the definition of the keyword `distinct` and its corresponding translation from t. We thus have $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} \langle\langle\rangle\rangle\Box distinct(t_1, t_2)$, which is equivalent to $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} DIST$.

For *INIT*, we have, from our definition of *init*, that *init* is only satisfied in the initial state, meaning that for all possible paths from the initial state, *init* will no longer be satisfied. Also, based on our definition of the initial state being the first state of the game (i.e. no previous state), and $DatlogPMod(\delta(\Gamma_{\texttt{init}}) \cup \delta(\Gamma_{\texttt{glob}}))$ returning the set of atomic propositions satisfied in the initial state, the other two conjuncts of *INIT* follows from our definition. We thus have $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} INIT$.

Next, *MEM* follows from our definition of the translation t and t_{old}. We have that for all possible paths from the initial state, for any non-terminal state and for any atomic proposition `p` (i.e. $t(\texttt{p})$ in our iCGS), its truth value will be reflected correctly in the next state. We achieve this by adding $t(\texttt{p})$ to Π when we constructed Π, which means that if $t(\texttt{p})$ is satisfied in the current state, $t(\texttt{p})_{old}$ will be satisfied in the next state as the current state becomes the previous state and all atoms from the previous state would be added to Π; the same thing applies to $\neg t(\texttt{p})$. Therefore, we have that $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} MEM$.

Moving onto *ONE_DONE*, this follows from the type of game we restrict our GDL-II game descriptions to. We assumed that all the games described by the GDL-II game description Γ would be multi-agent and for each rule r in Γ with (`does` $\texttt{i}_\texttt{m}$ $\texttt{a}_\texttt{m}$) atoms in the body, each i_m refers to a different agent $i_m \in Ag$ in our definition of the syntax and semantics of GDL-II. This means that only one action can be performed by each agent in any state that is not the initial state, so we have $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} ONE\_DONE$.

For *LEGAL*, this can be proven using our definition of the `legal` keyword. We defined (`legal i a`) to describe that the agent i can perform the action a in the current state, which means that the agent i must have a strategy to ensure that it can play this action (so in the next state i would have done the action). By this definition, we can see that in any state in any path from the initial state, we have that $legal(i, a)$ iff i has an strategy such that in the next state $done(i, a)$ becomes true, and thus $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} LEGAL$.

Now, in order to show that $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} STRAT$, we need to first ensure that for any state in any path from the initial state that is not the initial state or a sink state, and for any atomic proposition, we have $MIN(\texttt{p})$ satisfied. $MIN(\texttt{p})$ follows from the assumption that the only two types or rule with $hd(r) \in \{(\texttt{p}), (\texttt{next p})\}$ are of the form: $(\Leftarrow (\texttt{p})\ bd_1)$ and $(\Leftarrow (\texttt{next p})\ bd'_1\ (\texttt{does}\ \texttt{i}_\texttt{n}\ \texttt{a}_\texttt{n}) \ldots (\texttt{does}\ \texttt{i}_\texttt{n}\ \texttt{a}_\texttt{n}))$, and from Definition 1.2 that we can interpret these rules as implications. Therefore, we have that for any state in all possible paths from the initial state, and is not a sink state, for any atomic proposition `p`, $MIN(\texttt{p})$ should hold. We thus have $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} STRAT$.

Similarly, *SEES* can be deduced from the Datalog semantics introduced in Definition 1.2 and the assumption that all the rules with $hd(r) = (\texttt{sees i t})$ are of the form: $(\Leftarrow (\texttt{sees i t})\ bd''_1\ (\texttt{does}\ \texttt{i}_\texttt{n}\ \texttt{a}_\texttt{n}) \ldots (\texttt{does}\ \texttt{i}_\texttt{n}\ \texttt{a}_\texttt{n}))$. As the rule can be interpreted as the `sees` atom will be satisfied in the next state iff the body of the rule is true in the current state, meaning that the agents each performs an action (i.e. $done(i_n, a_n)$ is true in the next state), we can prove the body of *SEES* by the assumption and t_{old}, so $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} SEES$.

In addition, *TERM* follows from the definition of sink states and the fin_i actions. Since we defined fin_i to be the only action that any agent can play in `terminal` or sink states, we can deduce that for any state in any path from the initial state that is `terminal` or sink states, each agent will perform the corresponding fin_i action, meaning that $done(i, fin_i)$ will become true in the next state. We thus have $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} TERM$.

Finally, *SINK* also follows from our definition of sink states $\texttt{s}_\perp$: we defined $\texttt{s}_\perp$ to be a special state that is the successor of all `terminal` states and of itself without changing anything else, in order to make our iCGS a proper one. This means that we have for all atomic propositions that are satisfied in `terminal` or $\texttt{s}_\perp$ remain true and only sink state can be the successor of these two special

kind of states. Therefore, we have $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} \mathit{SINK}$.

From the above, we have shown that $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} \mathit{DIST} \wedge \mathit{INIT} \wedge \mathit{MEM} \wedge \mathit{ONE\_DONE} \wedge \mathit{LEGAL} \wedge \mathit{STRAT} \wedge \mathit{SEES} \wedge \mathit{TERM} \wedge \mathit{SINK}$; we therefore have proven that $\mathcal{A}_\Gamma, s_0 \models_{\text{ATL}} \Gamma_{\text{ATL}}$.

□

3 Conclusion

In recent years, the relationship between logic and games has been one of the popular research area. Particularly, there has been many studies conducted on the use of ATL-like logics for reasoning about multi-agent systems that can represent games. In this article, we have extended the connections between ATL and GDL presented in [4] to similar connections between ATL and GDL-II. Specifically, we presented the semantic and syntactic links between a GDL-II game description and an iCGS over ATL. On the semantics level, we showed that we can transform each GDL-II game model into an iCGS; on the syntactic level, we showed that the GDL-II game description itself can be associated with an ATL theory. We have thus demonstrated that GDL-II can be interpreted as a specification language for ATL models and we can verify ATL formulae over GDL-II using the ATL models.

For future research, we should continue exploring the link between ATL and GDL-II, such as checking whether the iCGS we construct from the game model of the GDL-II game description is alternating bisimilar to all other game structures that satisfy the ATL theory we get from the original game description. Additionally, the complexity of actually verifying ATL formulae over iCGS is worth exploring, too.

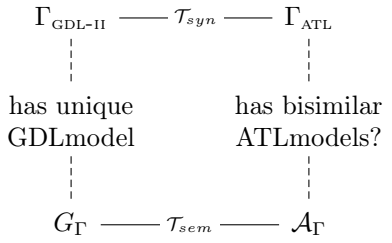

Figure 2: Connections between GDL-II and ATL

References

[1] N. Bulling, J. Dix, and W. Jamroga. Model checking logics of strategic ability: Complexity*. In Hindriks K. Meyer JJ Dastani, M., editor, *Specification and Verification of Multi-agent Systems*, pages 125–159. Springer, Boston, MA, 2010.

[2] W. F. Clocksin and C. S. Mellish. *Programming in Prolog*. Springer, 1981.

[3] M. Huth and M. Ryan. Verification by model checking. In *Logic in Computer Science: Modelling and Reasoning about Systems*, pages 207–216. Cambridge University Press, 2004.

[4] J. Ruan, W. van der Hoek, and M. Wooldridge. Verification of games in the game description language. *Journal of Logic and Computation*, 2009.

[5] M. Thielscher. A general game description language for incomplete information games. *AAAI-10*, pages 994–999, 2010.