Abstraction of Situation Calculus Concurrent Game Structures – Extended Abstract*

Yves Lespérance[1] , **Giuseppe De Giacomo**[2] , **Maryam Rostamigiv**[3]
Shakil M. Khan[3]
[1]York University, Toronto, ON, Canada
[2]University of Oxford, Oxford, UK
[3]University of Regina, Regina, SK, Canada
lesperan@eecs.yorku.ca, giuseppe.degiacomo@cs.ox.ac.uk, maryam.rostamigiv@uregina.ca, shakil.khan@uregina.ca

Many multi-agent applications can be viewed as *games* where some agents try to ensure that certain objectives hold no matter how the environment and other agents behave. Logics such as Alternating-Time Temporal Logic (ATL) (Alur, Henzinger, and Kupferman 2002) have been defined to specify properties of such systems and verify them through model checking, with semantics based on concurrent game structures. However, as the game/system becomes more complex, it becomes very important to use *abstraction* to explain how the game evolves and do strategic reasoning more effectively. In this paper, we develop a general account of *abstraction for multi-agent synchronous games* based on the agent abstraction framework of (Banihashemi, De Giacomo, and Lespérance 2017) (BDL17) and the situation calculus synchronous game structures (SCSGS) of (De Giacomo, Lespérance, and Pearce 2016) (DLP16). To express strategic properties of abstract and concrete games, we use a first-order variant of alternating-time μ-calculus, μATL-FO. We show that we can exploit abstraction in verifying μATL-FO properties of SCSGSs under the assumption that agents can always execute abstract moves to completion even if not fully controlling their outcomes. Our framework is based on the situation calculus, which provides a first-order representation of the state and allows us to model how plays depend on the data/objects involved.

SCSGS. These are a special kind of situation calculus basic action theory where we have a finite set of n agents, a finite set of move types, the possible moves of the agents, represented by functions that may take object arguments, and one action type $tick(m_1, \ldots, m_n)$, which represents the synchronous execution of a joint move by all the agents. A key component is a characterization of the legal moves available to each agent; specified using special predicate $LegalM$, which is defined by statements of the following form (one for each agent Ag_i and move type M_i):

$$LegalM(Ag_i, M_i(\vec{x}), s) \doteq \Phi_{Ag_i,M_i}(\vec{x}, s)$$

For the action $tick$ we have the following precondition axiom: $Poss(tick(m_1, \ldots, m_n), s) \equiv \bigwedge_{i=1,\ldots,n} LegalM(Ag_i, m_i, s)$. We have *successor state axioms* $\mathcal{D}_{ssa}$, specifying the effects and frame conditions of the joint moves $tick(m_1, \ldots, m_n)$ on the fluents; as usual these are domain specific and characterize the actual game under consideration. Finally, the initial state of the game is axiomatized in the *initial situation description* $\mathcal{D}_{S_0}$ as usual, in a domain specific way.

For example, we might have a high-level SCSGS $\mathcal{D}_h^{rs}$ representing a repair shop that includes the following axioms:

$$LegalM(ag, repair(i), s) \doteq (ag = RR_1 \vee ag = RR_2) \wedge Assigned(i, ag, s) \wedge \neg Repaired(i, s)$$
$$LegalM(ag, ship(i), s) \doteq ag = Sh \wedge Repaired(i, s) \wedge \neg Shipped(i, s)$$

Abstraction of SCSGS. We assume that there is a high-level (HL) (i.e., abstract) specification represented by SCSGS $\mathcal{D}_h$ and a low-level (LL) (i.e., concrete) specification represented by SCSGS $\mathcal{D}_l$. The HL and LL theories are related by a *SCSGS refinement mapping* m which is a triple $\langle m_m, m_a, m_f \rangle$ where m_m associates each HL move type m in $Moves_h$ to a move-determined (MD) move-based Golog program δ_m defined over the LL SCSGS theory that implements the move, i.e., $m_m(m(\vec{x})) = \delta_m(\vec{x})$, m_a maps the unique HL action $tick_h \in A_h$ to a Golog *system program* that executes the mapping of the moves involved synchronously in parallel, i.e., $m_a(tick(m_1, \ldots, m_n)) = sync(m_m(m_1), \ldots, m_m(m_n))$, and (as in (BDL17)) m_f maps each situation-suppressed HL fluent $F(\vec{x})$ in $\mathcal{F}_h$ to a situation-suppressed formula $\phi_F(\vec{x})$ defined over the LL theory that characterizes the concrete conditions under which $F(\vec{x})$ holds in a situation, i.e., $m_f(F(\vec{x})) = \phi_F(\vec{x})$. Move-based Golog programs are Golog programs where atomic actions are replaced by atomic moves.

E.g., a partial refinement mapping for the repair shop:

$$m^{rs}(ship(i)) = (Diagnosed(i) \wedge Fixed(i) \wedge \neg(Packed(i) \wedge DroppedOff(i)))?; pack(i); dropOff(i)$$
$$m^{rs}(Repaired(i)) = Diagnosed(i) \wedge Fixed(i)$$

To be able to do strategic reasoning at the HL and then refine the resulting strategies into LL ones, we need to ensure that the mapping captures *all the legal behaviors* that agents can display at the LL. To get this, we impose the following constraint:

*The work described in this extended abstract was originally published as (Lespérance et al. 2024). It is partially supported by the National Science and Engineering Research Council of Canada, by the ERC Advanced Grant WhiteMech (No. 834228), by York University, and by the University of Regina.

Constraint 1. *(Proper Refinement Mapping)*
For every high-level system action sequence $\vec{\alpha}$ and every agent $i \in Agents$, we have that:

$$\begin{array}{l}\mathcal{D}_l \cup \mathcal{C} \models \forall s.(Do(m_a(\vec{\alpha}), S_0, s) \supset \\ \quad \forall m_i, s'.(Do(sync((\pi \vec{m.m})^*, m_m(m_i), (\pi \vec{m.m})^*), s, s') \supset \\ \quad\quad \exists m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n. \\ \quad\quad\quad Do(m_a(tick(m_1, \ldots, m_{i-1}, m_i, m_{i+1}, \ldots, m_n)), s, s')))\end{array}$$

Thus if an agent i completes a HL move m_i , there must be HL moves $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n$ by the other agents that capture their LL behavior. We must also ensure that agents can only execute refinements of HL moves and that they begin and end at the same time .

m-Bisimulation. To relate the HL and LL models/theories, we apply the definition of m-bisimulation of (BDL17) to system primitive actions, i.e., joint moves. A relation $B \subseteq \Delta_S^{M_h} \times \Delta_S^{M_l}$ (where Δ_S^M stands for the situation domain of M) is an *m-bisimulation relation between M_h and M_l* if $\langle s_h, s_l \rangle \in B$ implies that: (i) $s_h \simeq_m^{M_h, M_l} s_l$, i.e., s_h and s_l evaluate HL fluents the same; (ii) for every HL primitive action type A in A_h, if there exists s_h' such that $M_h, v[s/s_h, s'/s_h'] \models Poss(A(\vec{x}), s) \wedge s' = do(A(\vec{x}), s)$, then there exists s_l' such that $M_l, v[s/s_l, s'/s_l'] \models Do(m_a(A(\vec{x})), s, s')$ and $\langle s_h', s_l' \rangle \in B$; and (iii) for every HL primitive action type A in A_h, if there exists s_l' such that $M_l, v[s/s_l, s'/s_l'] \models Do(m_a(A(\vec{x})), s, s')$, then there exists s_h' such that $M_h, v[s/s_h, s'/s_h'] \models Poss(A(\vec{x}), s) \wedge s' = do(A(\vec{x}), s)$ and $\langle s_h', s_l' \rangle \in B$. M_h is *m-bisimilar* to M_l, written $M_h \sim_m M_l$, iff there exists an m-bisimulation relation B between M_h and M_l such that $\langle S_0^{M_h}, S_0^{M_l} \rangle \in B$.

Abstraction in Verifying Strategic Properties. To express properties of games, we use μATL-FO (DLP16), a first-order variant of alternating-time μ-calculus, μATL (Alur, Henzinger, and Kupferman 2002). We have the following syntax for μATL-FO formulas:

$$\Psi \leftarrow \varphi \mid Z \mid \neg\Psi \mid \Psi_1 \wedge \Psi_2 \mid \exists x.\Psi \mid \langle\langle G \rangle\rangle \bigcirc \Psi \mid \mu Z.\Psi(Z)$$

In the above, φ is an arbitrary, possibly open, situation-suppressed situation calculus uniform formula and Z is a predicate variable of a given arity. $\langle\langle G \rangle\rangle \bigcirc \Psi$ means that coalition G can force Ψ to hold next, i.e., there is a vector of legal moves for the agents in G such that for all legal moves by the other agents, Ψ holds afterwards. $\mu Z.\Psi(Z)$ is the *least fixpoint* construct from the μ-calculus, which denotes the least fixpoint of the formula $\Psi(Z)$ (we use this notation to emphasize that Z may occur free, i.e., not quantified by μ in Ψ). Similarly $\nu Z.\Psi(Z)$, defined as $\neg\mu Z.\neg\Phi[Z/\neg Z]$ (where we denote with $\Phi[Z/\neg Z]$ the formula obtained from Φ by substituting each occurrence of Z with $\neg Z$), denotes the *greatest fixpoint* of $\Psi(Z)$. We also use the usual abbreviations for first-order logic such as disjunction ($\vee$) and universal quantification $\forall$. Moreover we denote by $[[G]] \bigcirc \Psi$ the dual of $\langle\langle G \rangle\rangle \bigcirc \Psi$, i.e., $[[G]] \bigcirc \Psi \doteq \neg\langle\langle G \rangle\rangle \bigcirc \neg\Psi$.

In general, implementation of a HL move is a nondeterministic program and strategic reasoning by the agent and cooperation from other agents may be required to ensure its execution terminates. We can require that:

Constraint 2. *(Agents Can Always Execute HL Moves)*
For every high-level system action sequence $\vec{\alpha}$ and every agent $i \in Agents$, we have that:

$$\begin{array}{l}\mathcal{D}_l \cup \mathcal{C} \models \forall s.(Do(m_a(\vec{\alpha}), S_0, s) \supset \\ \quad \forall m_i.(\exists s'.Do(sync((\pi \vec{m.m})^*, m_m(m_i), (\pi \vec{m.m})^*), s, s') \supset \\ \quad\quad CanForce(\{i\}, sync((\pi \vec{m.m})^*, m_m(m_i), (\pi \vec{m.m})^*), s)))\end{array}$$

It then follows that for any HL joint move by a non-empty coalition that is possibly executable at the LL, the coalition has a strategy to execute it to termination no matter how the agents outside the coalition behave. Note that ability to execute HL moves to termination *does not mean that the agent can control the outcome!*

We can extend our mapping to map a HL μATL-FO formula Ψ to a LL one $m_l(\Psi)$. This is straightforward except for $\langle\langle G \rangle\rangle \bigcirc \Psi$: if Ψ at the next instant at the HL, then $m_l(\Psi)$ should hold not at the next LL instant, but after the refinements of the HL moves have completed. We can impose constraints that ensure that LL joint move sequences map back to a unique HL joint move sequence, and that we have a LL state formula $Hlc(s)$ that holds iff a HL joint move sequence has just completed. Then we can define:

$$\begin{array}{l}\langle\langle G \rangle\rangle \bigcirc_h \Psi \doteq \langle\langle G \rangle\rangle \bigcirc (\neg Hlc\, \mathcal{U}\, (Hlc \wedge \Psi)) \\ \quad\quad \doteq \langle\langle G \rangle\rangle \bigcirc \mu Z.((Hlc \wedge \Psi) \vee (\neg Hlc \wedge \langle\langle G \rangle\rangle \bigcirc Z))\end{array}$$

and use this to define the mapping $m(\Psi)$ for arbitrary HL μATL-FO formulas.

Then we can show our main result:

Theorem 3.
For any μATL-FO formula Ψ, if $M_h \sim_m M_l$, the constraints hold, $s_h \sim_m^{M_h, M_l} s_l$, and $s_h \in (\Psi)_{v,V}^{M_h}$, then $s_l \in (m_l(\Psi))_{v,V}^{M_l}$.

i.e., in m-bisimilar models, if a μATL-FO property Ψ holds at the HL, then the mapped version $m_l(\Psi)$ must also hold at LL in m-similar situations. Usually, the HL game structure is much smaller than the LL one, so it is easier to verify a property at the HL.

E.g., consider the first-order ATL^* property stating that the coalition of all repair shop agents C has a strategy to ensure that all items that arrive are eventually shipped $\langle\langle C \rangle\rangle \forall i.\Box(Arrived(i) \supset \Diamond Shipped(i))$, which can be expressed in μATL-FO as $\forall i.\nu X.(Arrived(i) \supset \mu Y.Shipped(i) \vee \langle\langle C \rangle\rangle \bigcirc Y) \wedge \langle\langle C \rangle\rangle \bigcirc X$. We can show that this property, call it Ψ_h^{GaFs}, holds at the HL, i.e., $S_0 \in (\Psi_h^{GaFs})^{M_h^{rs}}$. By Theorem 3, it follows that the mapped property also holds at the LL: $S_0 \in (m_l(\Psi_h^{GaFs}))^{M_l^{rs}}$.

References

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *J. ACM* 49(5):672–713.

Banihashemi, B.; De Giacomo, G.; and Lespérance, Y. 2017. Abstraction in situation calculus action theories. In *AAAI*, 1048–1055.

De Giacomo, G.; Lespérance, Y.; and Pearce, A. R. 2016. Situation calculus game structures and GDL. In *ECAI*, 408–416.

Lespérance, Y.; De Giacomo, G.; Rostamigiv, M.; and Khan, S. M. 2024. Abstraction of situation calculus concurrent game structures. In *AAAI*, 10624–10634. AAAI Press.