This is the supplemental material for the paper "Model Checking Pushdown Epistemic Game Structures", containing omitted semantics of AEMC$_\sigma$ and proofs due to space restriction.

A Semantics of AEMC$_\sigma$

The semantics of AEMC$_\sigma$ is defined over PEGSs. A *valuation* $\xi : \mathcal{Z} \to 2^{C_\mathcal{P}}$ is a function assigning to each proposition variable a set of configurations. We use $\xi[Z \mapsto C]$ to denote the valuation which is equal to ξ except for $\xi[Z \mapsto C](Z) = C$. Given a PEGS $\mathcal{P} = (\mathsf{Ag}, \mathsf{Ac}, P, \Gamma, \Delta, \lambda, \{\sim_i | \ i \in \mathsf{Ag}\})$ and a valuation $\xi : \mathcal{Z} \to 2^{C_\mathcal{P}}$, the denotation function $\| \circ \|^\sigma_{\mathcal{P},\xi}$ that maps AEMC$_\sigma$ formulae to sets of configurations is inductively defined as follows:

- $\|q\|^\sigma_{\mathcal{P},\xi} = \{c \mid q \in \lambda(c)\}$;
- $\|\neg q\|^\sigma_{\mathcal{P},\xi} = C_\mathcal{P} \setminus \|q\|^\sigma_{\mathcal{P},\xi}$;
- $\|Z\|^\sigma_{\mathcal{P},\xi} = \xi(Z)$;
- $\|\phi_1 \wedge \phi_2\|^\sigma_{\mathcal{P},\xi} = \|\phi_1\|^\sigma_{\mathcal{P},\xi} \cap \|\phi_1\|^\sigma_{\mathcal{P},\xi}$;
- $\|\phi_1 \vee \phi_2\|^\sigma_{\mathcal{P},\xi} = \|\phi_1\|^\sigma_{\mathcal{P},\xi} \cup \|\phi_1\|^\sigma_{\mathcal{P},\xi}$;
- $\|\langle A\rangle \mathbf{X}\phi\|^\sigma_{\mathcal{P},\xi} = \{c \in C_\mathcal{P} \mid \exists \upsilon_A : A \to \Theta^\sigma \ s.t. \ \forall \pi \in \mathsf{out}^\sigma(c, \upsilon_A), \ \pi_1 \in \|\phi\|^\sigma_{\mathcal{P},\xi}\}$;
- $\|[A]\mathbf{X}\phi\|^\sigma_{\mathcal{P},\xi} = \{c \in C_\mathcal{P} \mid \forall \upsilon_A : A \to \Theta^\sigma, \ \exists \pi \in \mathsf{out}^\sigma(c, \upsilon_A), \ \pi_1 \in \|\phi\|^\sigma_{\mathcal{P},\xi}\}$;
- $\|\mu Z.\phi\|^\sigma_{\mathcal{P},\xi} = \bigcup\{C \subseteq C_\mathcal{P} \mid \|\phi\|^\sigma_{\mathcal{P},\xi[Z\mapsto C]} \supseteq C\}$;
- $\|\nu Z.\phi\|^\sigma_{\mathcal{P},\xi} = \bigcap\{C \subseteq C_\mathcal{P} \mid \|\phi\|^\sigma_{\mathcal{P},\xi[Z\mapsto C]} \subseteq C\}$;
- $\|\mathbf{K}_i\phi\|^\sigma_{\mathcal{P},\xi} = \{c \in C_\mathcal{P} \mid \forall c' \in C_\mathcal{P}, c \sim_i c' \Longrightarrow c' \in \|\phi\|^\sigma_{\mathcal{P},\xi}\}$;
- $\|\mathbf{E}_A\phi\|^\sigma_{\mathcal{P},\xi} = \{c \in C_\mathcal{P} \mid \forall c' \in C_\mathcal{P}, c \sim^E_A c' \Longrightarrow c' \in \|\phi\|^\sigma_{\mathcal{P},\xi}\}$;
- $\|\mathbf{C}_A\phi\|^\sigma_{\mathcal{P},\xi} = \{c \in C_\mathcal{P} \mid \forall c' \in C_\mathcal{P}, c \sim^C_A c' \Longrightarrow c' \in \|\phi\|^\sigma_{\mathcal{P},\xi}\}$;
- $\|\overline{\mathbf{K}}_i\phi\|^\sigma_{\mathcal{P},\xi} = \{c \in C_\mathcal{P} \mid \exists c' \in C_\mathcal{P}, c \sim_i c' \wedge c' \in \|\phi\|^\sigma_{\mathcal{P},\xi}\}$;
- $\|\overline{\mathbf{E}}_A\phi\|^\sigma_{\mathcal{P},\xi} = \{c \in C_\mathcal{P} \mid \exists c' \in C_\mathcal{P}, c \sim^E_A c' \wedge c' \in \|\phi\|^\sigma_{\mathcal{P},\xi}\}$;
- $\|\overline{\mathbf{C}}_A\phi\|^\sigma_{\mathcal{P},\xi} = \{c \in C_\mathcal{P} \mid \exists c' \in C_\mathcal{P}, c \sim^C_A c' \wedge c' \in \|\phi\|^\sigma_{\mathcal{P},\xi}\}$.

For closed AEMC$_\sigma$ formula ϕ, $\|\phi\|^\sigma_{\mathcal{P},\xi}$ is independent of ξ. Therefore, the superscript ξ will be dropped from $\|\phi\|^\sigma_{\mathcal{P},\xi}$, for closed AEMC$_\sigma$ formula ϕ. In addition, the subscript $\mathcal{P}$ is also dropped from $\|\phi\|^\sigma_{\mathcal{P},\xi}$ and $\|\phi\|^\sigma_\mathcal{P}$ when it is clear.

B A Motivating Example

In this section, we show how to use PEGSs to model infinite-state multi-agent systems by the *pushdown scheduler system* introduced in (Murano and Perelli 2015).

Suppose the system consists of two processes a and b, and a scheduler s. The processes a and b can respectively access to a common resource by doing the request r_a and r_b. The requests from the processes a and b are stored in a stack and can be granted by the scheduler s in *last-in-first-out* order. It was shown in (Murano and Perelli 2015) that this pushdown scheduler system in which agents have perfect information can be modeled as a PGS, but cannot be modeled as a finite-state CGS (Alur, Henzinger, and Kupferman 2002). Formally, the PGS model is defined as $\mathcal{P}_{\mathsf{sched}} = (\mathsf{Ag}, \mathsf{Ac}, P, \Gamma, \Delta, \lambda)$, where

- $\mathsf{Ag} = \{1, 2, 3\}$, in which the agent 1 (resp. 2 and 3) denotes the process a (resp. process b and scheduler s);
- $\mathsf{Ac} = \{0, 1, do, a, b\}$, in which the action a (resp. b) denotes that the process a (resp. b) can make the request r_a (resp. r_b), the action 1 denotes that the process a (resp. b) makes the request r_a (resp. r_b) to access the common resource, the action 0 denotes that the process a or b skips the chance to make a request, the action do denotes that the scheduler s can grant requests stored in the stack;
- $P = \{l_a, l_b, l_0, l_{do}\}$, in which the scheduler s controls the control states l_0 and l_{do}, the process a (resp. b) controls the control state l_a (resp. l_b);
- $\Gamma = \{r_a, r_b, \bot\}$, in which r_a (resp. r_b) is the request to be granted by the scheduler and $\bot$ is the bottom of the stack;
- Δ is described in Figure 1 in which,
  - the scheduler s at the control state l_0 can make the action a (resp. b and do) leading to the control state l_a (resp. l_b and l_{do}) without changing the stack (ϵ in Figure 1), whatever the processes a and b do ($* \in \mathsf{Ac}$ in Figure 1),
  - the scheduler s at the control state l_{do} grants the requests stored in the stack in *last-in-first-out* order by popping the top of stack until no more request needs to be granted (i.e., the top of the stack is $\bot$),
  - the process a (resp. b) at the control state l_a (resp. l_b) can either make a request by the action 1 or skip by making the action 0 whatever b and s do, the request r_x for $x \in \{a, b\}$ is pushed onto the stack (push(r_x) in Figure 1) once the request is made,
- $\mathbf{AP} = \{r_a, r_b, g_a, g_b, a, b, e\}$;
- λ is defined as follows: for every $\omega \in \Gamma^*$ and $x \in \{a, b\}$, $\lambda(\langle l_0, r_x\omega\rangle) = r_x$, $\lambda(\langle l_x, \omega\rangle) = x$, $\lambda(\langle l_{do}, r_x\omega\rangle) = g_x$ and $\lambda(\langle l_{do}, \bot\rangle) = e$. Obviously, λ is a regular valuation.

In the model $\mathcal{P}_{\mathsf{sched}}$, all the information is viable for the scheduler s and processes a and b. However, in practice, the requests stored in the stack are only visible to the scheduler s rather than the processes a and b who made the requests. In addition, the process a (resp. b) can only see its controlled control state l_a (resp. l_b), while the scheduler s can see all the control states. This characterization can be expressed using epistemic accessibility relations $\sim_i$ for $i \in \{\mathsf{Ag}\}$, which make the model more precise. Indeed, we can model the pushdown scheduler system with stack only visible to the scheduler as the PEGS $\mathcal{P}'_{\mathsf{sched}} = (\mathsf{Ag}, \mathsf{Ac}, P, \Gamma, \Delta, \lambda, \{\sim_i | \ i \in \mathsf{Ag}\})$, where Ag, Ac, P, Γ, Δ and λ are defined as in $\mathcal{P}_{\mathsf{sched}}$, epistemic accessibility relations $\sim_i$ for $i \in \mathsf{Ag}$ are the least relations satisfying the following conditions:

- for all $\omega, \omega' \in \Gamma^*$ and all $p, p' \in \{l_0, l_b, l_{do}\}$,

$$\langle l_a, \omega\rangle \sim_1 \langle l_a, \omega'\rangle \text{ and } \langle p, \omega\rangle \sim_1 \langle p', \omega'\rangle;$$

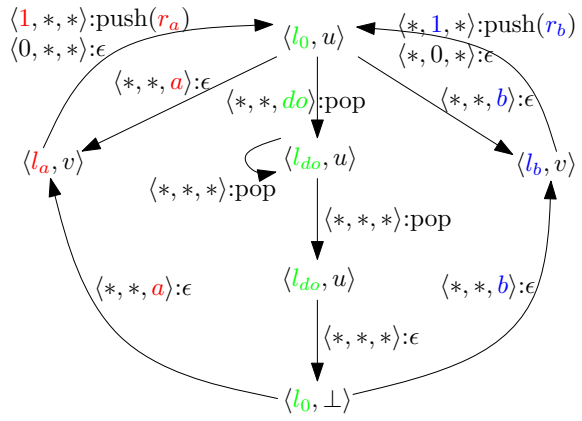

Figure 1: A pushdown scheduler system. The label of each edge consists of two parts: the decision and the stack operation. The

- for all $\omega, \omega' \in \Gamma^*$ and all $p, p' \in \{l_0, l_a, l_{do}\}$,

$$\langle l_b, \omega\rangle \sim_2 \langle l_b, \omega'\rangle \text{ and } \langle p, \omega\rangle \sim_2 \langle p', \omega'\rangle;$$

- for all $p \in P$ and $\omega \in \Gamma^*$, $\langle p, \omega\rangle \sim_3 \langle p, \omega\rangle$.

It is easy to observe that the epistemic accessibility relations $\sim_i$ for $i \in \mathsf{Ag}$ are simple and can be expressed as $\{\approx_i | \ i \in \mathsf{Ag}\}$, where

- $l_a \approx_1 l_a$, $l_0 \approx_1 l_b \approx_1 l_{do}$;
- $l_b \approx_2 l_b$, $l_0 \approx_2 l_a \approx_2 l_{do}$;
- $l_0 \approx_3 l_a \approx_3 l_b \approx_3 l_{do}$.

ATEL$^*_\sigma$ and AEMC$_\sigma$ can express several interesting properties. For instance, consider $\phi_a = \mathbf{K}_1(\nu Z_1.(\langle\emptyset\rangle\mathbf{X}\, Z_1 \wedge (r_a \rightarrow \mu Z_2.(g_a \vee \mathbf{E}_{\{3\}}\langle\{3\}\rangle\mathbf{X}\, Z_2))))$, $\phi_{ks} = \nu Z.(\mathbf{K}_3 e \vee \mathbf{E}_{\{3\}}\langle\{3\}\rangle\mathbf{X}Z)$, $\phi_s = \nu Z.(e \vee \mathbf{E}_{\{3\}}\langle\{3\}\rangle\mathbf{X}Z)$ and $\phi_{live} = \mathbf{K}_s\langle\{3\}\rangle(\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}e)$. ϕ_a states that the process a knows that whenever it makes a request, the scheduler has a way to grant the request later. ϕ_{ks} states that the agent 3 (i.e., scheduler s) has a strategy which it knows that all the stored requests will be granted, can be recomputed along the computation, and guarantees that it will know when all the stored requests has been granted. ϕ_s states that the agent 3 has a recomputable strategy that it knows to grant all the stored requests, but it will not necessarily know when to grant (Bulling and Jamroga 2011). ϕ_{live} states that the schedule s knows that it has a strategy to let the propositions a, b and e to occur infinitely often.

C Proofs in Section 4

Theorem 3. The model checking problems for ATL$_{\mathbf{ir}}$ and ATL$^*_{\mathbf{ir}}$ on PEGSs with size-preserving epistemic accessibility relations are undecidable.

Proof. We prove this by a reduction from the model checking problem on CEGSs under **iR** setting. It is known that the model checking problem for ATL on CEGSs under **iR** setting is undecidable (Dima and Tiplea 2011).

Given a CEGS $\mathcal{P} = (\mathsf{Ag}, \mathsf{Ac}, P, \Delta, \lambda, \{\sim_i | \ i \in \mathsf{Ag}\})$, we construct a PEGS $\mathcal{P}' = (\mathsf{Ag}, \mathsf{Ac}, P, \Gamma, \Delta', \lambda', \{\sim'_i | \ i \in \mathsf{Ag}\})$, where

- $\Gamma = P \cup \{\bot\}$,
- Δ' is the least function such that for every $\Delta(p, \mathbf{d}) = (p')$, $\Delta'(p, \gamma, \mathbf{d}) = (p', p\gamma)$ for every $\gamma \in \Gamma$,
- $\sim'_i \subseteq C_{\mathcal{P}'} \times C_{\mathcal{P}'}$ such that for every configurations $c = \langle p, p_1 ... p_m\rangle$ and $c' = \langle p', p'_1 ... p'_m\rangle$, $c \sim'_i c'$ iff $p \sim_i p'$ and $p_j \sim_i p'_j$ for all $j \in [m]$.
- λ' is defined as: for every configuration $\langle p, \gamma_1 ... \gamma_m\rangle \in P \times \Gamma^*$, $\lambda'(\langle p, \gamma_1 ... \gamma_m\rangle) = \lambda(p)$.

For every ATL$_{\mathbf{ir}}$/ATL$^*_{\mathbf{ir}}$ ϕ, it is easy to see that $\langle p\rangle \models_{\mathbf{iR}} \phi$ in the CEGS $\mathcal{P}$ iff $\langle p, \bot\rangle \models_{\mathbf{ir}} \phi$ in the PEGS $\mathcal{P}'$. □

D Proofs in Section 4

Let $\mathcal{P} = (\mathsf{Ag}, \mathsf{Ac}, P, \Gamma, \Delta, \lambda, \{\sim_i | \ i \in \mathsf{Ag}\})$ be a PEGS with a regular epistemic accessibility relation such that, for each $i \in \mathsf{Ag}$, $\sim_i$ is given as the pair of $(\approx_i, \mathcal{A}_i)$, where $\approx_i \subseteq P \times \Gamma$ is an equivalence relation and $\mathcal{A}_i = (S_i, \Gamma, \delta_i, s_{i,0})$ is a DFA.

Let $\vec{\mathcal{A}} = (\vec{S}, \Gamma, \vec{\delta}, \vec{s_0})$ be the product automaton of $\mathcal{A}_i$'s for $i \in \mathsf{Ag}$, such that $\vec{S} = S_1 \times ... \times S_n$, $\vec{s_0} = [s_{1,0}, ..., s_{n,0}]$, and $\vec{\delta}(\vec{s_1}, \gamma) = \vec{s_2}$ if for every $i \in [n]$, $\delta_i(s_{i,1}, \gamma) = s_{i,2}$, where $s_{i,j}$ denotes the state of $\mathcal{A}_i$ in $\vec{s_j}$.

Let $\mathcal{P}' = (\mathsf{Ag}, \mathsf{Ac}, P, \Gamma', \Delta', \lambda', \{\sim'_i | \ i \in \mathsf{Ag}\})$ be the PEGS, where $\Gamma' = \Gamma \times \vec{S}$, and for each $i \in \mathsf{Ag}$, $\sim'_i$ is specified by an equivalence relation $\approx'_i$ defined as follows: $(p, [\gamma, \vec{s}]) \approx'_i (p', [\gamma', \vec{s'}])$ iff $(p, \gamma) \approx_i (p', \gamma')$ and $\vec{s} = \vec{s'}$, in addition, Δ' is defined as follows: for every state $\vec{s} \in \vec{S}$,

1. for every $\langle p, \gamma\rangle \overset{\mathbf{d}}{\hookrightarrow}_{\mathcal{P}} \langle p', \epsilon\rangle$, we have $\langle p, [\gamma, \vec{s}]\rangle \overset{\mathbf{d}}{\hookrightarrow}_{\mathcal{P}'} \langle p', \epsilon\rangle$,
2. for every $\langle p, \gamma\rangle \overset{\mathbf{d}}{\hookrightarrow}_{\mathcal{P}} \langle p', \gamma_k ... \gamma_1\rangle$ with $k \geq 1$ and $\vec{\delta}(\overrightarrow{s_j}, \gamma_j) = \overrightarrow{s_{j+1}}$ for every $j : \ 1 \leq j \leq k-1$ (where $\overrightarrow{s_1} = \vec{s}$), then $\langle p, [\gamma, \vec{s}]\rangle \overset{\mathbf{d}}{\hookrightarrow}_{\mathcal{P}'} \langle p', [\gamma_k, \overrightarrow{s_k}] ... [\gamma_1, \overrightarrow{s_1}]\rangle$.

Finally, the valuation λ is adjusted accordingly to λ', i.e., for every $\langle p', [\gamma_k, \overrightarrow{s_k}] ... [\gamma_0, \overrightarrow{s_0}]\rangle \in C_{\mathcal{P}'}$, $\lambda'(\langle p', [\gamma_k, \overrightarrow{s_k}] ... [\gamma_0, \overrightarrow{s_0}]\rangle) = \lambda(\langle p', \gamma_k ... \gamma_0\rangle)$.

Theorem 4. The model checking problem of an ATEL$_{\mathbf{ir}}$ (resp. ATEL$^*_{\mathbf{ir}}$) formula ϕ on a PEGS $\mathcal{P}$, with stack alphabet Γ and regular epistemic accessibility relations $\sim_i = (\approx_i, \mathcal{A}_i)$ for $i \in \mathsf{Ag}$, can be reduced to the model checking problem of ϕ on a PEGS $\mathcal{P}'$ with simple epistemic accessibility relations $\sim'_i$, such that the state space of $\mathcal{P}'$ is the same as that of $\mathcal{P}$, and the stack alphabet of $\mathcal{P}'$ is $\Gamma \times \vec{S}$, where $\vec{S}$ is the state space of the product of $\mathcal{A}_i$'s for $i \in \mathsf{Ag}$.

Proof. A configuration $\langle p, [\gamma_k, \vec{s_k}] ... [\gamma_0, \vec{s_0}]\rangle \in C_{\mathcal{P}'}$ is *consistent* iff for every $i : \ 0 \leq i < k$, $\vec{\delta}(\vec{s_i}, \gamma) = \vec{s_{i+1}}$.

Suppose the path of $\mathcal{P}'$ reaches the consistent configuration $\langle p, [\gamma_k, \vec{s_k}] ... [\gamma_0, \vec{s_0}]\rangle \in C_{\mathcal{P}'}$, the path of $\mathcal{P}$ is at the configuration $\langle p, \gamma_k ... \gamma_0\rangle \in C_{\mathcal{P}}$ and for every $i \in [n]$, the DFA $\mathcal{A}_i$ is at the state $s_{i,k}$ after reading the word $\gamma_0 ... \gamma_{k-1}$. If there is a transition rule $\langle p, \gamma_k\rangle \overset{\mathbf{d}}{\hookrightarrow}_{\mathcal{P}} \langle p', \epsilon\rangle$, then $\mathcal{P}$

moves from the configuration $\langle p, \gamma_k ... \gamma_0 \rangle$ to the configuration $\langle p', \gamma_{k-1} ... \gamma_0 \rangle$ if the agents cooperatively made the decision $\mathbf{d}$. Meanwhile, the DFA $\mathcal{A}_i$ should go to the state $s_{i,k-1}$ after reading the word $\gamma_0 ... \gamma_{k-2}$, as $\mathcal{A}_i$ is deterministic. These are mimicked by the path of $\mathcal{P}'$ which moves from the configuration $\langle p, [\gamma_k, \mathbf{s}_k] ... [\gamma_0, \mathbf{s}_0] \rangle$ to the configuration $\langle p', [\gamma_{k-1}, \vec{s_{k-1}}] ... [\gamma_0, \vec{s_0}] \rangle$ (c.f. Item 1).

Analogously, if there is a transition rule $\langle p, \gamma_k \rangle \overset{\mathbf{d}}{\hookrightarrow}_{\mathcal{P}} \langle p', \gamma'_t ... \gamma'_k \rangle$ for some $t \geq k$, then $\mathcal{P}$ moves from the configuration $\langle p, \gamma_k ... \gamma_0 \rangle$ to the configuration $\langle p', \gamma'_t ... \gamma'_k \gamma_{k-1} ... \gamma_0 \rangle$ if the agents cooperatively made the decision $\mathbf{d}$. Suppose $\vec{s_j} \xrightarrow{\gamma'_j} \vec{s_{j+1}}$ for every $j : \ k \leq j \leq t-1$. To encode the computation information of DFAs, we add the transition rule $\langle p, [\gamma_k, \vec{s_k}] \rangle \overset{\mathbf{d}}{\hookrightarrow}_{\mathcal{P}'} \langle p', [\gamma'_t, \vec{s_t}] ... [\gamma'_k, \vec{s_k}] \rangle$ which allows the path of $\mathcal{P}'$ moves from the configuration $\langle p, [\gamma_k, \vec{s_k}] ... [\gamma_0, \vec{s_0}] \rangle$ to the configuration $\langle p, [\gamma'_t, \vec{s_t}] ... [\gamma'_k, \vec{s_k}] [\gamma_{k-1}, \vec{s_{k-1}}] ... [\gamma_0, \vec{s_0}] \rangle$.

Therefore, for every configurations $\langle p, \gamma_k ... \gamma_0 \rangle$, $\langle p', \gamma'_{k'} ... \gamma'_0 \rangle \in C_{\mathcal{P}}$ and $i \in \mathsf{Ag}$, $\langle p, \gamma_k ... \gamma_0 \rangle \sim_i \langle p', \gamma'_{k'} ... \gamma'_0 \rangle$ iff there are two consistent configurations $\langle p, [\gamma_k, \vec{s_k}] ... [\gamma_0, \vec{s_0}] \rangle, \langle p', [\gamma'_{k'}, \vec{s'_{k'}}] ... [\gamma'_0, \vec{s'_0}] \rangle \in C_{\mathcal{P}'}$ such that $\langle p, [\gamma_k, \vec{s_k}] ... [\gamma_0, \vec{s_0}] \rangle \sim'_i \langle p', [\gamma'_{k'}, \vec{s'_{k'}}] ... [\gamma'_0, \vec{s'_0}] \rangle$. Since the automata $\mathcal{A}'_i s$ are deterministic, all the reachable configurations starting from consistent configurations are consistent configurations. The result immediately follows.

□

References

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

Bulling, N., and Jamroga, W. 2011. Alternating epistemic mu-calculus. In *Proceedings of the 22rd International Joint Conference on Artificial Intelligence*, 109–114.

Dima, C., and Tiplea, F. L. 2011. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR* abs/1102.4225.

Murano, A., and Perelli, G. 2015. Pushdown multi-agent system verification. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 1090–1097.