A Dynamic Epistemic Logic with Finite Iteration and Parallel Composition

Andreas Herzig\textsuperscript{1}, Frédéric Maris\textsuperscript{2}, Elise Perrotin\textsuperscript{2}
\textsuperscript{1}IRIT, CNRS, Toulouse  
\textsuperscript{2}IRIT, Univ. Toulouse III  

Abstract

Existing dynamic epistemic logics combine standard epistemic logic with a restricted version of dynamic logic. Instead, we here combine a restricted epistemic logic with a rich version of dynamic logic. The epistemic logic is based on ‘knowing-whether’ operators and basically disallows disjunctions and conjunctions in their scope; it moreover captures ‘knowing-what’. The dynamic logic has not only all the standard program operators of Propositional Dynamic Logic, but also parallel composition as well as an operator of inclusive nondeterministic composition: its atomic programs are assignments of propositional variables. We show that the resulting dynamic epistemic logic is powerful enough to capture several kinds of sequential and parallel planning, both in the unbounded and in the finite horizon versions.

1 Introduction

Dynamic epistemic logics (DELs) combine epistemic logic and dynamic logic. Most approaches in the literature combine full-fledged epistemic logic with restricted versions of dynamic logic. The latter typically lack the program operators of propositional dynamic logic (PDL), in particular the operator of finite iteration (‘Kleene star’). The reason is that the latter causes undecidability of satisfiability (Miller and Moss 2005) and even of model checking. The latter is directly related to the general undecidability of DEL-based planning (Bolander and Andersen 2011; Aucher and Bolander 2013; Bolander et al. 2020). In previous work we have proposed a different route whose starting point is a restricted version of standard epistemic logic: boolean combinations of ‘knowing-whether’ operators followed by a propositional variable (Cooper et al. 2016; Cooper et al. 2020a). In that lightweight epistemic logic, the existence of sequential and parallel epistemic plans can be decided in PSPACE (Cooper et al. 2020b). A dynamic extension, DEL-PAO, was proposed in (Herzig, Lorini, and Maffre 2015), and a modelling of planning tasks with no common knowledge and no parallel plans was given in (Cooper et al. 2016). We here give a fuller and more succinct dynamic logic, adding in particular an operator of parallel composition as well as an operator of inclusive nondeterministic composition. Both are imported from Dynamic Logic of Parallel Propositional Assignments DL-PPA (Herzig, Maris, and Vianey 2019), which is conceptually and mathematically simpler than several other proposals for adding parallel composition to dynamic logic such as (Benevides and Schechter 2014; Balbiani and Boudou 2018; Boudou, Herzig, and Troquard 2021). In the resulting DEL the solvability of various planning tasks can be captured: the existence of sequential and parallel plans, both in the unbounded and in the finite horizon versions. The operator of inclusive nondeterministic composition turns out to be instrumental for the succinct modelling of parallel planning.

We generalize the epistemic ‘knowing-whether’ operator to a ‘knowing-what’ (or ‘knowing the value’) operator, introduced and argued for in (Plaza 2007; Wang and Fan 2013; Wang 2018) as a useful and natural addition to logics of knowledge, particularly in the field of AI: agents may not only need to know whether or not a given proposition is true, but also what another agent’s telephone number is or the code to open a door. We show that this notion of ‘knowing what’ can be added to EL-\textsuperscript{O} without much difficulty, leading to an enlarged field of possibilities regarding the planning tasks that can be modelled and worked with. We illustrate our logic by means of the gossip problem where each agent \(i \in \text{Agt}\) knows a secret \(s_i\) that none of the other agents knows and where the goal is to find a sequence of phone calls after which every agent knows every secret (Landau 1954; Knödel 1975; van Ditmarsch, van der Hoek, and Kuijer 2020). Several interesting variants of that problem can be designed, in particular where the goal is to obtain higher-order knowledge (Cooper et al. 2019; Cooper et al. 2020a; van Ditmarsch, Gattinger, and Ramezanian 2020).

We present the static epistemic logic EL-\textsuperscript{OC} in Section 2 and extend it to the dynamic logic DEL-PPAO\textsuperscript{C} in Section 3. In Section 4 we apply DEL-PPAOC to planning. Proofs of the results can be found in (Perrotin forthcoming, Ch. 6).

2 The Static Logic: EL-\textsuperscript{OC}

We introduce the Epistemic Logic of Observation with Constants, abbreviated to EL-\textsuperscript{OC}.

2.1 Atoms, Intrusive Atoms, and Atomic Consequence

Let \(\text{Prop}\) be a countable set of propositional variables, let \(\text{Cst}\) be a countable set of constants, and let \(\text{Agt}\) be a finite set
of agents. The set of observability operators is

\[ \text{OBS} = \{ S_i : i \in \text{Agt} \} \cup \{ \emptyset \}, \]

where \( S_i \) stands for individual observability of agent \( i \) and \( \emptyset \) stands for joint observability of all agents. The set of all sequences of observability operators is noted \( \text{OBS}^* \) and the set of all non-empty sequences is noted \( \text{OBS}^+ \). We use \( \sigma, \sigma', \) etc. to denote elements of \( \text{OBS}^* \).

Observability atoms, or atoms for short, are finite sequences of observability operators followed by a propositional variable. The set of all atoms is

\[ \text{ATM} = \{ \sigma \ p : \sigma \in \text{OBS}^+, \ p \in \text{Prop} \} \cup \{ \sigma \ c : \sigma \in \text{OBS}^+, \ c \in \text{Cst} \}. \]

We use \( \alpha, \alpha', \beta, \ldots \) to denote atoms, unless specified as members of \( \text{ATM} \cup \text{Cst} \). Here are some examples: \( S_j \ p \) reads “\( i \) sees the truth value of \( p \)”, i.e., \( i \) knows whether \( p \) is true or false. \( S_j \ c \) reads “\( i \) sees the value of \( c \)”. \( \emptyset \ S_j \ c \) reads “all agents jointly see whether agent \( j \) sees the value of \( c \)”, i.e., there is joint attention in the group of all agents concerning \( 2 \)'s observation of \( c \): agent \( 2 \) may or may not see the value of \( c \), and in both cases this is jointly observed. An atom with an empty sequence of observability operators is nothing but a propositional variable; constants are always preceded by at least one observability operator as they have no truth value.

We follow the principles of EL-O and call an atom introspective if it contains two consecutive \( S_j \), or a \( \emptyset \) that is preceded by a non-empty sequence of observability operators. The set of all introspective atoms is

\[ \text{I-ATM} = \{ \sigma S_j \alpha : \sigma \in \text{OBS}^+ \text{ and } \alpha \in \text{ATM} \cup \text{Cst} \} \cup \{ \sigma \emptyset \ c : \sigma \in \text{OBS}^+ \text{ and } \alpha \in \text{ATM} \cup \text{Cst} \}. \]

We finally define a relation of atomic consequence between observability atoms as follows:

\[ \alpha \implies \beta \iff \alpha = \beta \text{ or there are } \alpha' \in \text{ATM} \cup \text{Cst}, \sigma \in \text{OBS}^+ \text{ such that } \alpha = \emptyset \alpha' \text{ and } \beta = \sigma \alpha'. \]

When \( \alpha \implies \beta \) we say that \( \alpha \) is a cause of \( \beta \) and that \( \beta \) is a consequence of \( \alpha \). We denote by \( \alpha^\circ \) the set of causes of \( \alpha \), and by \( \alpha^\bullet \) the set of its consequences. We generalize this to sets of atoms \( \mathcal{V} \subseteq \text{ATM} \): \( \mathcal{V}^\circ = \bigcup_{\alpha \in \mathcal{V}} \alpha^\circ \).

### 2.2 Language and Semantics

The language of EL-OC is defined by the grammar

\[ \varphi ::= \alpha \mid \neg \varphi \mid (\varphi \land \varphi) \]

where \( \alpha \) ranges over \( \text{ATM} \). The boolean operators \( \top, \bot, \lor, \land \) and \( \rightarrow \) are defined in the standard way.

A state is a subset of the set of states \( \text{ATM} \). We denote states by \( V, U, W \), etc. The set of all states is \( 2^{\text{ATM}} \).

As in EL-O semantics, we interpret formulas such that introspection is simulated. The only non-standard case is:

\[ ||\alpha|| = \{ V : \alpha \in V^\circ \cup \text{I-ATM} \}. \]

Hence \( \alpha \) is true in state \( V \) if and only if \( \alpha \) is introspective or \( \beta \implies \alpha \) for some \( \beta \in V \).

### 3 Adding Dynamic Logic

We now describe the full dynamic logic DEL-PPAOC.

#### 3.1 Language of DEL-PPAOC

The language of DEL-PPAOC extends that of EL-OC with the dynamic operator \( \pi \), where \( \pi \) is a program. Programs \( \pi \) and formulas \( \varphi \) are defined by the grammar

\[ \varphi ::= \alpha \mid \neg \varphi \mid \varphi \land \varphi \mid \langle \pi \rangle \varphi, \]

\[ \pi ::= \alpha \rightarrow \varphi \mid \pi \land \pi \mid \pi \lor \pi \mid \pi \land \pi \mid \pi^*, \]

where \( \alpha \) ranges over the set of atomic formulas \( \text{ATM} \). The formula \( \langle \pi \rangle \varphi \) reads “there is a possible execution of \( \pi \) such that \( \varphi \) is true afterwards”. The program \( \alpha \rightarrow \varphi \) assigns the truth value of \( \varphi \) to \( \alpha \). \( \alpha \rightarrow \varphi \) tests whether \( \varphi \) is true (and fails when \( \varphi \) is false). \( \pi_1 ; \pi_2 \) executes \( \pi_1 \) and \( \pi_2 \) in sequence. \( \pi_1 \cup \pi_2 \) nondeterministically chooses between executing either \( \pi_1 \) or \( \pi_2 \); and \( \pi_1 \lor \pi_2 \) nondeterministically chooses between executing either \( \pi_1 \), or \( \pi_2 \), or both. \( \pi_1 \land \pi_2 \) is the parallel composition of \( \pi_1 \) and \( \pi_2 \). The set of all formulas is \( \text{Fin}_\pi \text{DEL-PPAOC} \).

The formula \( [\pi] \varphi \) abbreviates \( \neg \langle \pi \rangle \neg \varphi \) and therefore has to be read “\( \varphi \) is true after every possible execution of \( \pi \)”. We define \( n \)-times iteration of \( \pi \) by induction on \( n: \pi^0 = \top \) and \( \pi^{n+1} = \pi \land \pi^n \). Finally, we define \( \pi^* \) as \( \bigcup_{\pi_i \in \text{Fin}_\pi} \pi_i \). \( \pi^* \) is the unbounded iteration of \( \pi \).

#### 3.2 Semantics of DEL-PPAOC

The interpretation of a formula \( \varphi \) is a set of valuations \( ||\varphi|| \subseteq 2^{\text{ATM}} \), just as in EL-OC. The interpretation of a program \( \pi \) is a ternary relation on the set of valuations: \( ||\pi|| \subseteq 2^{\text{ATM}} \times 2^{\text{ATM}} \times 2^{\text{ATM}} \). When \( (V, U, W) \in ||\pi|| \) then there is an execution of \( \pi \) from state \( V \) to state \( U \) assigning the variables at \( W \). The interpretation function is defined by mutual recursion. The main clauses are given in Table 2; the others are standard.

The interpretation of the assignment \( \alpha \rightarrow \varphi \) is that, either (1) \( \varphi \) is true, \( \alpha \) gets the value true and the set of assigned

\[
\begin{align*}
S_i S_j \alpha & (\text{Vis}_4) \quad \text{JS} \alpha \rightarrow S_i \alpha \quad (\text{Vis}_4) \\
\text{JS} S_i \alpha & (\text{Vis}_5) \quad \text{JS} S_j \alpha \quad (\text{Vis}_5) \\
\text{JS} S_i S_j \alpha & (\text{Vis}_5)
\end{align*}
\]

Table 1: Axioms for introspection, where \( \alpha \in \text{ATM} \cup \text{Cst} \)

Example 1. In the initial state \( V_0 \text{G} \) of the gossip problem every agent only knows her own secret. Therefore \( V_0 \text{G} = \{ S_j, S_i : i \in \text{Agt} \} \) where all \( S_i \) are constants. (This differs from modellings of the gossip problem without ‘knowing-what’ where secrets are assumed to be either true or false.) Then \( V_0 \text{G} \in ||S_j \land \bigwedge_i S_j, S_i|| \) for every \( i \in \text{Agt} \).

A formula \( \varphi \) is EL-OC valid if \( V \in ||\varphi|| \) for every \( V \subseteq \text{ATM} \); it is satisfiable if \( V \in ||\varphi|| \) for some \( V \subseteq \text{ATM} \). Clearly, atom \( \alpha \) is valid if and only if it is introspective. It is easily shown that all properties of EL-O given in (Cooper et al. 2020a), in particular NP-completeness of the satisfiability problem, still hold in EL-OC. The EL-OC validities are axiomatised in Table 1; the completeness proof closely follows that of (Cooper et al. 2020a).
variables is the singleton \{a\}, or (2) \varphi is false, all causes of
\alpha get the value false and the set of assigned variables is the
set \alpha^\omega of all causes of \alpha.

The interpretation of parallel composition \pi_1 \parallel \pi_2 is that
each subprogram \pi_j is executed locally; then it is checked that the
modifications (in terms of assigned variables) are compatible, in the
sense that variables that are assigned by both subprograms (i.e., those
at \pi_1 \cap \pi_2) get the same truth value. If this is not the case then the
parallel composition fails; otherwise the resulting valuation \varpi is computed
by putting together (1) the unchanged part of V (i.e., \varpi \backslash \pi_j),
(2) the updates of \pi_1 (i.e., \pi_1 \cap \pi_j), (3) the updates of \pi_2
(i.e., \pi_2 \cap \pi_j). Moreover, the set of variables \omega assigned
by a parallel composition is the union of the sets of variables
assigned by the subprograms.

The interpretation of inclusive nondeterministic composition
\pi_1 \parallel \pi_2 is the exclusive nondeterministic composition of
the three programs \pi_1, \pi_2 and \pi_1 \parallel \pi_2.

### 4 Epistemic Planning

In this section we show how \DEL-\PPAO captures simple
episodic planning tasks where actions change not only the
world, but also the agents’ knowledge. We assume deterministic
actions with conditional effects described by add- and
delete-lists. Such conditional effects are crucial: when an
agent performs an action then the effects on another agent’s
epistemic state typically depend on whether that agent sees
the variables that are modified by the action (Andersen,
Bolander, and Jensen 2012; Cooper et al. 2020a).

#### 4.1 Action Descriptions

An action description is a pair a = (pre(a), eff(a)) where
pre(a) \in \Fml_{\DEL-\PPAO} (the precondition of a) and
eff(a) \subseteq \Fml_{\DEL-\PPAO} \times \omega^{\omega} \times \omega^{\omega}
are the conditional effects of a. For a triple
\langle ce, \text{cond}(ce), \text{eff}^+(ce), \text{eff}^-(ce) \rangle
in \text{eff}(a), \text{cond}(ce) is the condition of ce, \text{eff}^+(ce) are the
added atoms, and \text{eff}^-(ce) are the deleted atoms. We
require consistency of effects: if \langle ce_1, ce_2 \in \text{eff}(a) \rangle
and \| \text{pre}(a) \land \text{cond}(ce_1) \land \text{cond}(ce_2) \| \neq 0
then \text{eff}^-(ce_1) \cap \text{eff}^+(ce_2) = \emptyset.

#### Example 2

In the gossip problem a call has two conditional
effects per secret: if agent \( a \) knows \( s_j \), then \( s_j \) becomes known to \( j \),
and vice versa. Hence \text{pre}('call') = \top and
\text{eff}('call') = \{(\{s_k, s_j\}, \emptyset) : 1 \leq k \leq n\} \cup \{(\{s_j, s_k\}, \emptyset) : 1 \leq k \leq n\}.

To every action \( a \) we associate a partial function \( \tau_a \) on
states as follows: \( \tau_a(V) \) is defined if \( V \in \| \text{pre}(a) \| \); and when
\( \tau_a(V) \) is defined then for every conditional effect ce \in \text{eff}(a),
if its triggering condition \text{cond}(ce) is satisfied ('ce fires')
then the negative effects of ce are removed, as well as their
causes, and the positive effects of ce are added:
\[ \tau_a(V) = V \setminus \left( \bigcup_{ce \in \text{eff}(a)} \neg \text{cond}(ce) \right) \cup \left( \bigcup_{ce \in \text{eff}(a)} \text{eff}^+(ce) \right) . \]

We capture \( \tau_a \) by a program testing the precondition
and processing in parallel the conditional effects:
\[ \text{exeAct}(a) = \text{pre}(a); \bigcap_{ce \in \text{eff}(a)} \left( \text{cond}(ce) \land \left( \begin{array}{l} \text{cond}(ce) \land \neg \top \land (ce \leftarrow \top) \land (ce \leftarrow \bot) \end{array} \right) \right) . \]

Note that thanks to the constraint of consistency of action
effects the big parallel composition is always executable.

#### 4.2 Consistency of a Set of Actions at a State

We give two consistency constraints for parallel execution.
First, \( a_1 \) and \( a_2 \) have no contradictory effects at \( V \) if
for every ce_1 \in \text{eff}(a_1) and ce_2 \in \text{eff}(a_2), if \( V \in \| \text{cond}(ce_1) \land \text{cond}(ce_2) \| \) then \text{eff}^-(ce_1) \cap \text{eff}^+(ce_2) = \emptyset. \)

The case \( a_1 = a_2 \) is already excluded by our requirement on
action descriptions in Section 4.1.
2. $V$ and $\tau_A(V)$ agree on $pre(a_1)$ and on the condition $\text{cond}(c_1)$ of every conditional effect $c_1 \in \text{eff}(a_1)$, where "$V$ and $V'$ agree on $\varphi$" means that both $V \in \|\varphi\|$ and $V' \in \|\varphi\|$, or both $V \notin \|\varphi\|$ and $V' \notin \|\varphi\|$. Hence neither action preconditions nor effect conditions are modified by the effect of another action executed in parallel, guaranteeing that they can be interleaved. For example, the actions of two different agents picking up the same object cross-interact.

A set of actions $A$ is consistent at $V$ if for every $a, a' \in A$ such that $a \neq a'$, (1) $a$ and $a'$ have no contradictory effects at $V$; (2) $a$ and $a'$ have no cross interaction at $V$.

Example 3. A gossiping conference call $\text{call}_i$ is consistent. A way to exclude this is to replace $\text{call}_i$ by $\text{Tcall}_i$, with $pre(\text{Tcall}_i) = \top$ and $\text{eff}(\text{Tcall}_i) = \text{eff}(\text{call}_i) \cup \{(tg_i, 0, \{tg_i\}) \cup (\neg tg_i, \{tg_i\}, 0) \cup (\neg tg_j, \{tg_j\}, 0)\}$.

Then two different calls involving $i$ each toggle the value of $tg_i$, ensuring that they cross-interact in any state.

4.3 Semantics of a Set of Actions

A set of actions $A$ determines a partial function $\tau_A$ on valuations. It is defined at $V$ if $V \in \|\bigwedge_{a \in A} \text{pre}(a)\|$ and $A$ is consistent at $V$. When it is defined at $V$ then:

$$\tau_A(V) = \left\{ V \setminus \left( \bigcup_{a \in A, c \in \text{eff}(a), \forall V \in \text{val}(c)} (\text{eff}(c)) \right) \right\} \cup \left( \bigcup_{a \in A, c \in \text{eff}(a), \forall V \in \text{val}(c)} (\text{eff}(c)) \right).$$

Thanks to consistency of $A$, the order in which negative and positive effects are processed does not matter.

Let us capture this in DEL-PPAO. First, the formula

$$\text{Insensitive}(a, a') = \bigwedge_{c \in \text{eff}(a)} \text{cond}(c) \leftrightarrow (\text{exec}(a') \text{cond}(c))$$

expresses that neither executability nor effects of $a$ are sensitive to the execution of $a'$. Then to every $A$ we associate the DEL-PPAO program

$$\text{exec}(A) = \bigwedge_{a, a' \in A, a \neq a'} \text{Insensitive}(a, a') ?; \bigwedge_{a \in A} \text{exec}(a).$$

Proposition 1. For every finite set of actions $A$:

1. $\tau_A$ is defined at $V$ if and only if $\langle V, U, W \rangle \in \|\text{exec}(A)\|$ for some $U, W$.

2. $\tau_A$ is defined at $V$ then $\tau_A(V) = U$ iff $\langle V, U, W \rangle \in \|\text{exec}(A)\|$ for some $W$.

4.4 Simple Epistemic Planning Tasks

A simple epistemic planning task is a triple $\langle \text{Act}, V_0, \text{Goal} \rangle$ where $\text{Act}$ is a finite set of actions, $V_0 \in 2^{\text{ATM}}$ is a finite state (the initial state), and $\text{Goal} \in \text{Finl}_{\text{DEL-PPAO}}$ is a DEL-PPAO formula.

Example 4. The gossip problem can be viewed as the planning task $G = \langle \text{Act}^G, V_0^G, \text{Goal}^G \rangle$ with

$$\text{Act}^G = \{ \text{Tcall}_i^{ij} : i, j \in \text{Act} \text{ and } i \neq j \}$$ (cf. Example 3),

$$V_0^G = \{ S_i : i \in \text{Act} \}$$ (cf. Example 1),

$$\text{Goal}^G = \bigwedge_{i, j \in \text{Act}, i \neq j} \exists S_i, S_j.$$

State $V$ is reachable by a parallel plan from $V_0 \in 2^{\text{ATM}}$ via a set of actions $\text{Act}$ if there is an $m \geq 0$ and sequences $\langle V_0, \ldots, V_m \rangle$ and $\langle A_1, \ldots, A_m \rangle$ such that $V_m = V$ and for $1 \leq k \leq m$, $V_k \in 2^{\text{ATM}}$, $A_k \subseteq \text{Act}$, and $\tau_A(V_{k-1}) = V_k$.

A simple epistemic planning task $\langle \text{Act}, V_0, \text{Goal} \rangle$ is solvable by a parallel plan if there is a state $V$ that is reachable by a parallel plan from $V_0$ via $\text{Act}$ such that $V \in \|\text{Goal}\|$; otherwise it is unsolvable by a parallel plan.

Theorem 1. A planning task $\langle \text{Act}, V_0, \text{Goal} \rangle$ is solvable by a parallel plan with no more than $k$ steps if and only if:

$$V_0 \in \Bigl\| \left( \bigcup_{a \in \text{Act}} \text{exec}(a) \right)^{k} \text{Goal} \Bigr\|,$$

where the $x_a$ are fresh variables and where

$$\pi_{x} = \bigwedge_{a, a' \in \text{Act}, a \neq a'} \left( (x_a \land x_{a'}) \rightarrow \text{Insensitive}(a, a') \right) ;$$

$$\bigwedge_{a \in \text{Act}} \left( \neg x_a \lor (x_a ? : \text{exec}(a)) \right).$$

Example 5. Task $G$ can be solved in $[\log_2 n]$ steps of parallel calls if the number of agents $n$ is even, and in $[\log_2 n] + 1$ steps if $n$ is odd (Bavelas 1950; Landau 1954; Knödel 1975; Cooper et al. 2019). For instance, for $n = 4$ the parallel plan $\langle \text{Tcall}_2^{12}, \text{Tcall}_2^{23}, \text{Tcall}_2^{34} \rangle$ solves $G$ in 2 steps.

Solvability by a sequential plan is the special case where the parallel plan is a sequence of singletons.

Theorem 2. A planning task $\langle \text{Act}, V_0, \text{Goal} \rangle$ is solvable by a sequential plan with no more than $k$ actions if and only if:

$$V_0 \in \Bigl\| \left( \bigcup_{a \in \text{Act}} \text{exec}(a) \right)^{k} \text{Goal} \Bigr\|.$$
Acknowledgements
We would like to thank the KR 2021 reviewers for their comments. Our work is partially supported by the EU ICT-48 2020 project TAILOR (No. 952215) and the ANR project CoPains (No. ANR-18-CE33-0012).

References


