Reasoning about protocol change and knowledge

Yanjing Wang

Department of Philosophy, Peking University, Beijing 100871, China

Abstract. In social interactions, protocols govern our behaviour and assign meaning to actions. In this paper, we investigate the dynamics of protocols and their epistemic effects. We develop two logics, inspired by Propositional Dynamic Logic (PDL) and Public Announcement Logic (PAL), for reasoning about protocol change and knowledge updates. We show that these two logics can be translated back to the standard PDL and PAL respectively.

1 Introduction

Protocols are the rules that govern the actions of humans or machines. They have two major functions in our everyday life. First of all, protocols regulate behaviours and thus let us (or machines) know what to do or what not to do. For example, when you are driving a car you are also driving according to various traffic protocols; in case an accident happens legal protocols are called into play; while you are sending emails or SMS messages to a friend to complain about the bad luck, communication protocols on computers are running to make sure the messages are delivered. Second, protocols assign meaning to actions. For example, we are educated to be polite by following social protocols such as *shaking hands*. The handshaking action itself does not mean anything, it is the conventional protocol: "**if** you want to say hello formally and politely **then** shake hands" which lets this simple action carry some extra information. We can also create a new meaning for an action by setting a new protocol. For example, by popularizing the slogan "Love her, take her to Häagen-Dazs!" in China, the ice cream company Häagen-Dazs managed to let many young Chinese couples believe that buying an ice cream shows their love, no matter what love actually means. Because of the existence of such protocols we save our civilization from chaos and make it more meaningful everyday. Without doubt, protocols rule the world.

Already from the Häagen-Dazs example, we can see that it is important to understand how we can "install" new protocols to people. More generally, we are interested in the changes of protocols and their epistemic effects. Here we give some more examples of dynamics of protocols in social interactions. Imagine that you were told to close the door and on your way to do it you are told again not to close it. Now what to do? As another example, a well-trained spokesman may respond to a yes-no question (viewed as a protocol that says: answer 'Yes!' **or** answer 'No!') by inserting yet another protocol: "before answering your question, tell me what you meant by ϕ." Here is a more interesting example that involves meaning changing: The Chinese are *non-confrontational* in the sense that they will not overtly say “No.”, instead they say “I will think about it.” or “We will see.”. For a western businessman, “We will see.”, according to the standard interpretation, means it is still possible. However, if he is updated with the Chinese protocol: say “We will see.” **if** you want to say “No.”, then he should understand this is just another way of saying “No.”. Clearly the difference in protocols is sometimes the reason behind many conflicts and misunderstandings. Such phenomena invite a formal investigation on the question “*how* to change protocols and *update* your knowledge?”. This paper reports an attempt in addressing this very question.

Our approach is inspired by two well-known logics: Propositional Dynamic Logic (PDL [7]) and Public Announcement Logic (PAL [20, 9]). First of all, as we have seen above, the protocols usually have program-like structures, which suggests a formalization of protocols by regular expressions as in PDL. In the pioneering work of [10] and [6], where protocols involving knowledge were studied, protocols were indeed treated as simple programs in the form of $?\phi \cdot a$ (**if** ϕ holds **then** do a). Here we also need such tests $?\phi$ together with regular expressions to encode how protocols assign meaning to actions, e.g., $?p_{love} \cdot a_{buy}$ for our Häagen-Dazs example. On the other hand, the simplest protocol changing action might be a *public protocol announcement* and it is useful, e.g., a public announcement of what the Chinese means by using “We will see.” could solve many problems in advance. Differing from the public announcements $!\phi$ in PAL, an announcement of a protocol may not have truth values, instead it changes the set of possible (sequences of) actions in addition to the inherent restrictions of actions according to the model. Putting the protocol announcements $[!\pi]$ (π is a regular expression) together with program modalities $[\pi]$ as in PDL, we may express that “Although b is possible according to the current protocol, after the announcement of the new protocol $a \cdot b$, we can not execute b as the first event any more” by the formula $\langle b\rangle\top \wedge [!(a \cdot b)]\neg\langle b\rangle\top$[1]. The more interesting case is when knowledge (expressed by $K_i\phi$) comes in. For instance, in our we-will-see story we would like $[a_{\text{will-see}}]K_i p_{no}$ to be not valid while $[!(?p_{no} \cdot a_{\text{will-see}})][a_{\text{will-see}}]K_i p_{no}$ to be valid. We will define the formal semantics for such an enriched epistemic language in this paper.

The contributions of this paper can be summarized as follows:

- We introduce various protocol announcement operators to PDL-based logics.
- The new logics can be used to reason about: 1. the executable actions according to the current protocol, and 2. the information that the actions carry, thus formally capturing the two functions of protocols.
- Epistemic reasoning in presence of unplanned protocol changes is facilitated.
- New protocol changing operators do not drive the logics beyond PDL or PAL.

Related work Besides the ones we already talked about earlier, some more related work should be mentioned here. Process logic [21, 11] extends PDL in adding

[1] Here we assume that if a protocol is announced then it is followed by its executor.

modalities to specify *progressive* behaviours like "during the execution of program π, ϕ will be true at some point." In this paper, we not only reason about properties in the middle of an execution of a protocol but also handle the protocol changes during the execution. The later feature also distinguishes our work from the work using regular expressions as protocols [2, 17, 26]. Moreover, the semantics of our logics will be defined on the states in the models, instead of on paths as in [21, 11]. Aucher [1] also proposed an extended Dynamic Epistemic Logic (DEL), where the reasoning of the ongoing events is facilitated, however, in a setting without protocols. Unlike the work of switching strategies in the context of games [19], the change of our protocols can be made at any time unplanned and also we incorporate knowledge in the discussions.

The existing work on protocols in DEL enriches the epistemic models with explicit protocols (sets of sequences of DEL events) such that the possible behaviours of agents are not only restricted by the inherent preconditions of epistemic events but also by protocol information [13, 22, 12]. This is similar to the treatment of protocols in Epistemic Temporal Logic (ETL) [10, 18], where the temporal development of a system is generated from an initial situation by a commonly known protocol[2]. In our work, the semantics of our languages with protocol announcements will be defined on *standard* Kripke models. The extra protocol information is only introduced by protocol announcements while evaluating a formula. Such an approach makes it possible to not only model the "installation" of the initial protocol explicitly but also to handle protocol changes during the execution of the current one.

Our treatment for the events that carry meaning is largely inspired by [18], in which the authors give a very general and elegant semantics for messages (events) according to the underlying protocol in the setting of ETL. Here we can explicitly express the protocols and their changes *in* the logical language. Note that in the standard PAL, the interpretations of announcements are fixed and implicitly assumed to be common knowledge, e.g., in PAL an announcement $!\phi$ is assumed to have an inherent meaning: ϕ is true. This is because the semantic objects (event models) are explicitly included in the syntax as in the general DEL framework. However, the same utterance (syntax) may carry different meanings (semantics) as we have seen in the we-will-see example. A closer look at public announcements should separate the utterances and their meanings, as we will demonstrate later in this work.

Structure of the paper In this paper we develop two logics featuring protocol changing operators. As an appetizer, we start in Section 2 with the first logic $\mathtt{PDL}^!$, a version of test-free PDL equipped with protocol announcements $[!\pi]$. The semantics is given in a non-standard style by using *modes* of satisfaction relations [8, 24]. It is shown that $[!\pi]$ and many other similar protocol announcement operators do not increase the expressive power of the logic. Section 3 extends the language $\mathtt{PDL}^!$ with knowledge operators and Boolean tests to handle the cases

[2] However, the framework in [22] can also handle the protocols which are not common known.

like the above we-will-see example where knowing a protocol gives meanings to actions. We show that this new logic, when interpreted on S5 models, is equally expressive as `PAL`. We conclude in Section 4 and point out future work.

2 Protocol Announcement Logic `PDL`$^!$

The formulas of `PDL`$^!$ are built from a set of basic proposition letters $\mathbf{P}$ and a finite set of atomic action symbols $\mathbf{\Sigma}$ as follows:

$$\begin{array}{lcl}\phi & ::= & \top \mid p \mid \neg\phi \mid \phi\wedge\phi \mid [\pi]\phi \mid [!\pi]\phi \\ \pi & ::= & \mathbf{1} \mid \mathbf{0} \mid a \mid \pi\cdot\pi \mid \pi+\pi \mid \pi^*\end{array}$$

where $p \in \mathbf{P}$ and $a \in \mathbf{\Sigma}$. Note that π are actually *regular expressions* based on $\mathbf{\Sigma}$ (we denote the set of such regular expressions as $Reg_{\mathbf{\Sigma}}$). The intended meaning of the formulas is mostly as in `PDL`, but "in context" of the protocol constraints: $[\pi]\phi$ now says that "after any run of the program π which is allowed by the *current protocol*, ϕ holds". The new formula $[!\pi]\phi$ expresses "after the announcement of the new protocol π, ϕ holds".

To give the semantics to `PDL`$^!$, we first recall some basic facts about regular expressions.

The language of a regular expression π is defined as follows:

$$\begin{array}{l}\mathcal{L}(\mathbf{0})=\emptyset \qquad \mathcal{L}(\mathbf{1})=\{\epsilon\} \qquad \mathcal{L}(a)=\{a\}\\ \mathcal{L}(\pi\cdot\pi')=\{wv \mid w\in\mathcal{L}(\pi), v\in\mathcal{L}(\pi')\}\\ \mathcal{L}(\pi+\pi')=\mathcal{L}(\pi)\cup\mathcal{L}(\pi')\\ \mathcal{L}(\pi^*)=\{\epsilon\}\cup\bigcup_{n>0}(\mathcal{L}(\pi^n))\end{array}$$

where ϵ is the 'skip' protocol (empty sequence) and $\pi^n = \underbrace{\pi\cdots\pi}_{n}$.

The language of the *input derivative* $\pi\backslash a$ of a regular expression $\pi \in Reg_{\mathbf{\Sigma}}$ is defined as $\mathcal{L}(\pi\backslash a) = \{v \mid av \in \mathcal{L}(\pi)\}$. With the output function $o : Reg_{\mathbf{\Sigma}} \to \{\mathbf{0},\mathbf{1}\}$ we can axiomatize the operation $\backslash a$ (cf. [4, 5]):

$$\begin{array}{ll}\pi = o(\pi)+\sum_{a\in\mathbf{\Sigma}}(a\cdot\pi\backslash a) & \\ \mathbf{1}\backslash a=\mathbf{0}\backslash a=b\backslash a=\mathbf{0} \qquad (a\neq b) & a\backslash a=\mathbf{1}\\ (\pi\cdot\pi')\backslash a=(\pi\backslash a)\cdot\pi'+o(\pi)\cdot(\pi'\backslash a) & (\pi+\pi')\backslash a=\pi\backslash a+\pi'\backslash a\\ (\pi)^*\backslash a=\pi\backslash a\cdot(\pi)^* & o(\pi\cdot\pi)=o(\pi)\cdot o(\pi')\\ o(\pi^*)=\mathbf{1} & o(\mathbf{1})=\mathbf{1}\\ o(\mathbf{0})=o(a)=\mathbf{0} & o(\pi+\pi')=o(\pi)+o(\pi')\end{array}$$

Given $w = a_0a_1\cdots a_n \in \mathbf{\Sigma}^*$, let $\pi\backslash w = (\pi\backslash a_0)\backslash a_1\cdots\backslash a_n$. It is clear that $\pi\backslash w = \{v \mid wv \in \mathcal{L}(\pi)\}$[3]. Together with the axioms of Kleene algebra [14] we can syntactically derive $\pi\backslash w$ which is intuitively the *remaining* protocol of π after executing a run w. For example:

$$(a+(b\cdot c))^*\backslash b=(a\backslash b+(b\cdot c)\backslash b)\cdot(a+b\cdot c)^*=(\mathbf{0}+(\mathbf{1}\cdot c))\cdot(a+b\cdot c)^*=c\cdot(a+(b\cdot c))^*$$

[3] $\pi\backslash w$ is also a regular language cf. [5].

More generally, we can define $\mathcal{L}(\pi\backslash\pi') = \{v \mid \exists w \in \mathcal{L}(\pi') \text{ such that } wv \in \mathcal{L}(\pi)\}$ (cf. [5]). We say $w \in \mathbf{\Sigma}^*$ is *compliant with* π (notation: $w \propto \pi$) if $\pi\backslash w \neq \mathbf{0}$, namely, executing w is allowed by the protocol π.

Intuitively, to evaluate $[\pi]\phi$ we need to memorize the current protocol in some way. Here we employ a trick similar to the ones used in the semantics developed in [8, 24, 3]: we define the satisfaction relation w.r.t. a *mode* π (notation: $\vDash_\pi$), which is used to record the current protocol. Given the current protocol π, the allowed runs in a program π' w.r.t. π are those $w \in \mathbf{\Sigma}^*$ such that $w \in \mathcal{L}(\pi')$ and $w \propto \pi$. Note that if the current protocol is π, then after executing a run w we have to update π by the remaining protocol $\pi\backslash w$.

As in the standard `PDL`, we interpret $\texttt{PDL}^!$ formulas (w.r.t. $\mathbf{P}$,$\mathbf{\Sigma}$) on Kripke models $\mathcal{M} = (S, \to, V)$ where S is a non-empty set of states, $\to \subseteq S \times \mathbf{\Sigma} \times S$ is a set of labelled transitions, and the valuation function $V : S \to 2^{\mathbf{P}}$ assigns to each state a set of basic propositions. Now we are ready to give the semantics as follows:

$$\boxed{\begin{array}{l}
\mathcal{M}, s \vDash \phi \Leftrightarrow \mathcal{M}, s \vDash_{\mathbf{\Sigma}^*} \phi \\
\mathcal{M}, s \vDash_\pi p \Leftrightarrow p \in V(s) \\
\mathcal{M}, s \vDash_\pi \neg\phi \Leftrightarrow \mathcal{M}, s \nvDash_\pi \phi \\
\mathcal{M}, s \vDash_\pi \phi \wedge \psi \Leftrightarrow \mathcal{M}, s \vDash_\pi \phi \text{ and } \mathcal{M}, s \vDash_\pi \psi \\
\mathcal{M}, s \vDash_\pi [\pi']\phi \Leftrightarrow \forall(w, s') : w \in \mathcal{L}(\pi'), w \propto \pi, \text{ and } s \overset{w}{\to} s' \implies \mathcal{M}, s' \vDash_{\pi\backslash w} \phi \\
\mathcal{M}, s \vDash_\pi [!\pi']\phi \Leftrightarrow \mathcal{M}, s \vDash \langle\pi'\rangle\top \implies \mathcal{M}, s \vDash_{\pi'} \phi
\end{array}}$$

where the mode $\mathbf{\Sigma}^*$ stands for the *universal protocol* $(a_0 + a_1 + \cdots + a_n)^*$ if the set of atomic actions $\mathbf{\Sigma}$ is $\{a_0, a_1, \ldots, a_n\}$. The first clause says that initially everything is allowed and the last one says that the newly announced protocol overrides the current one. $[\pi']\phi$ is true w.r.t. the current protocol π iff on each s' that is reachable from s by some run w of π' which is allowed by the current protocol π: ϕ holds w.r.t. the remaining protocol $\pi\backslash w$. Note that it is important to remember w which denotes *how you get to* s' as the following example shows:

Example 1. Consider the following model $\mathcal{M}$:

$$\begin{tikzcd}
s \arrow[r, bend left, "a"] & \bullet \arrow[r, bend left, "c"] \arrow[r, bend right, "d"'] & \bullet
\end{tikzcd}$$

It can be verified that:

$$\mathcal{M}, s \vDash [!(a \cdot c + b \cdot d)][a + b](\neg\langle d\rangle\top \wedge \langle c\rangle\top \wedge [!(c + d)]\langle d\rangle\top)$$

The intuition behind is as follows. After announcing the protocol $a \cdot c + b \cdot d$, the program $a + b$ can be executed according to this protocol, but actually only a can be executed on the model. Thus after executing $a + b$ only c is possible according to the remaining protocol $(a \cdot c + b \cdot d)\backslash a = c$. However, if we then announce a new protocol $(c + d)$ then d also becomes available. ♣

Recall the standard `PDL` semantics, it is not hard to see that the following proposition holds.

Proposition 1. *For any test-free* PDL *formula ϕ and any pointed Kripke model $(\mathcal{M}, s)$:*

$$\mathcal{M}, s \vDash_{\texttt{PDL}^!} \phi \iff \mathcal{M}, s \vDash_{\texttt{PDL}} \phi$$

A natural question to ask is that whether $\texttt{PDL}^!$ is more expressive than test-free PDL. To answer the question, we now have a closer look at the strings w in the semantics of $[\pi']\phi$. Given π, let $C_{\mathcal{L}(\pi)}$ be the set of all the *pre-sequences* of π: $\{w \mid w \propto \pi\}$.

We first show that we can partition $C_{\mathcal{L}(\pi)}$ into finitely many regular expressions satisfying certain properties.

Lemma 1. *For any regular expression π there is a minimal natural number k such that $C_{\mathcal{L}(\pi)}$ can be finitely partitioned into $\pi_0, \dots, \pi_k$ and for any $w, v \in \mathcal{L}(\pi_i) : \pi \backslash w = \pi \backslash v$.*

Proof. By Kleene's theorem we can construct a deterministic finite automaton (DFA) recognizing the language of π. It is well known that DFA can be minimized, thus we obtain a smallest DFA that recognizes $\mathcal{L}(\pi)$:

$$\mathsf{A}_\pi = (\{q_0, \dots, q_k\}, \mathbf{\Sigma}, q_0, \rightarrowtail, F)$$

where $\{q_0, \dots, q_k\}$ is a set of states with q_0 being the start state and a subset F being the set of accept states. For each $i \leq k$ such that q_i can reach a state in F: we let π_i be the regular expression corresponding to the automaton $(\{q_0, \dots, q_k\}, \mathbf{\Sigma}, q_0, \rightarrowtail, \{q_i\})$. Since A_π is deterministic, it is not hard to see that these π_i form a partition that we want. □

In the sequel, we call the above unique partition $\pi_0, \dots, \pi_k$ the *pre-derivatives* of π. For example, the minimal deterministic automaton[4] of $a^* \cdot d + b \cdot (c + d)$ is:

$$\begin{tikzcd}
\bullet \arrow[dr, bend right, "c"'] \arrow[dr, "d"] & \mathbf{q_0} \arrow[l, "b"'] \arrow[r, "a"] \arrow[d, "d"] & \bullet \arrow[loop above, "a"] \arrow[dl, "d"] \\
 & \bullet\downarrow &
\end{tikzcd}$$

thus the pre-derivatives of $a^* \cdot d + b \cdot (c + d)$ are $\mathbf{1}$, $a \cdot a^*, b, a^* \cdot d + b \cdot (c + d)$.

Now we define the following translation from $\texttt{PDL}^!$ to the test-free PDL:

$$\begin{array}{rcc}
t(\phi) & = & t_{\mathbf{\Sigma}^*}(\phi) \\
t_\pi(p) & = & p \\
t_\pi(\neg\phi) & = & \neg t_\pi(\phi) \\
t_\pi(\phi_1 \wedge \phi_2) & = & t_\pi(\phi_1) \wedge t_\pi(\phi_2) \\
t_\pi([\pi']\phi) & = & \bigwedge_{i=0}^{k}([\theta_i] t_{\pi \backslash \pi_i}(\phi)) \\
t_\pi([!\pi']\phi) & = & \langle \pi' \rangle \top \to t_{\pi'}(\phi)
\end{array}$$

[4] We omit the transitions to the *"trash"* state which can not reach any accept state.

where $\pi_0, \dots, \pi_k$ are the pre-derivatives of π, θ_i is a regular expression corresponding to $\mathcal{L}(\pi') \cap \mathcal{L}(\pi_i)$, and $\pi\backslash\pi_i = \pi\backslash w$ for any $w \in \mathcal{L}(\pi_i)$ due to Lemma 1.

By this translation we can eliminate the $[!\pi]$ operator in $\texttt{PDL}^!$ and thus showing that $\texttt{PDL}^!$ and the test-free $\texttt{PDL}$ are equally expressive.

Theorem 1. *For any pointed Kripke model $\mathcal{M}, s$:*

$$\mathcal{M}, s \vDash_{\texttt{PDL}^!} \phi \iff \mathcal{M}, s \vDash_{\texttt{PDL}} t(\phi).$$

Proof. By induction on ϕ we can show: $\mathcal{M}, s \vDash_\pi \phi \iff \mathcal{M}, s \vDash_{\texttt{PDL}} t_\pi(\phi)$. The only non-trival case is for $[\pi']\phi$:
$\mathcal{M}, s \vDash_\pi [\pi']\phi$
$\iff \forall(w, s') : w \in \mathcal{L}(\pi'), w \propto \pi,$ and $s \overset{w}{\to} s' \implies \mathcal{M}, s' \vDash_{\pi\backslash w} \phi$
$\iff \forall(w, s') :$ if there is a pre-derivative $\pi_i : w \in \mathcal{L}(\pi'), w \in \mathcal{L}(\pi_i),$ and $s \overset{w}{\to} s'$ then $\mathcal{M}, s' \vDash_{\pi\backslash w} \phi$
$\iff$ for all pre-derivatives π_i, for all s' : if there is a $w \in \mathcal{L}(\pi') \cap \mathcal{L}(\pi_i)$ and $s \overset{w}{\to} s'$ then $\mathcal{M}, s' \vDash_{\pi\backslash w} \phi$
$\iff \mathcal{M}, s \vDash \bigwedge_{i=0}^{k} [\theta_i] t_{\pi\backslash\pi_i}(\phi)$

□

Discussion In this section, we take a rather *liberal* view on the "default" protocol, namely we assume that everything is allowed initially. On the other hand, we can well start with a *conservative* initialization where nothing is allowed unless announced later. It is not hard to see that we can also translate this conservative version of $\texttt{PDL}^!$ to $\texttt{PDL}$ if we let $t(\phi) = t_{\mathbf{1}}(\phi)$ where $\mathbf{1}$ is the constant for empty sequence i.e., the *skip* protocol. For example, $t_{\mathbf{1}}([a]\bot \wedge [!a]\langle a + b\rangle\top) = [\mathbf{0}]\bot \wedge t_a(\langle a + b\rangle\top) = \langle a\rangle\top$.

Moreover, $[!\pi]$ is rather *radical* in the sense that it changes the protocol completely. We may define a more general operation as follows: Let $\pi(x) \in Reg_{\mathbf{\Sigma}\cup\{x\}}$, namely, $\pi(x)$ is a regular expression with a variable x. Now we define:

$$\boxed{\mathcal{M}, s \vDash_\pi [!\pi'(x)]\phi \Leftrightarrow (\mathcal{M}, s \vDash \langle\pi'(\pi)\rangle\top \implies \mathcal{M}, s \vDash_{\pi'(\pi)} \phi)}$$

We can then concatenate, add, insert and repeat protocols by announcing $x \cdot \pi'$, $x+\pi'$, $\pi'+x$, and x^* respectively. It is easy to see that the announcement operator $[!\pi]$ introduced previously is a special case of $[!\pi(x)]$. We can still translate the logic with the generalized protocol announcements to $\texttt{PDL}$ with an easy revision of the translation:

$$t_\pi([!\pi'(x)]\phi) = \langle\pi'(\pi)\rangle\top \to t_{\pi'(\pi)}(\phi)$$

Similarly, without adding the expressive power, we can introduce a *refinement operator* $[!(a/\pi')]$ for each $a \in \mathbf{\Sigma}$ with the following semantics:

$$\boxed{\mathcal{M}, s \vDash_\pi [!(a/\pi')]\phi \Leftrightarrow \mathcal{M}, s \vDash \langle\pi[a/\pi']\rangle\top \implies \mathcal{M}, s \vDash_{\pi[a/\pi']} \phi}$$

where $\pi[a/\pi']$ is the regular expression obtained by substituting each a in π with π'. Intuitively the operator $[!(a/\pi')]$ refines the current protocol by making the atomic step a more complicated.

3 Public Event Logic $\texttt{PDL}^{!?_b}$

In this section, we allow tests in the protocol announcements and study how agents update their knowledge according to the protocols and their observations of the public events. We shall see that by announcing a protocol with tests, we can let actions carry propositional information as motivated in the introduction.

Given a finite set $\mathbf{P}$ of basic propositions, a finite set $\mathbf{\Sigma}$ of atomic actions and a set $\mathbf{I}$ of agents, the language of $\texttt{PDL}^{!?_b}$ is defined as follows:

$$\begin{array}{lcl} \phi & ::= & \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid [\pi']\phi \mid [!\pi]\phi \mid K_i\phi \\ \pi & ::= & ?\phi_b \mid a \mid \pi \cdot \pi \mid \pi + \pi \mid \pi^* \end{array}$$

where $i \in \mathbf{I}$, ϕ_b are Boolean formulas based on $\mathbf{P}$, and π' are a test-free regular expressions. Note that we do not include $\mathbf{1}$ and $\mathbf{0}$ as atomic actions in π since they can be expressed by the Boolean tests $?\top$ and $?\bot$. We call the programs with possibly Boolean tests *guarded regular expressions.*

In this section, we assume that all the $a \in \mathbf{\Sigma}$ are *public events* which can be observed by all the agents, while the tests, unless announced, are not observable to the agents. Here $[\pi']\phi$ is intended to express that "ϕ holds after the agents observe any sequence of public events which is not only allowed in π' but also complaint with the current protocol." Therefore only test-free programs are considered in the modality $[\pi']$, since the tests can not be observed anyway. Now we can express the Häagen-Dazs slogan mentioned in the introduction by the protocol: $\pi_{H\text{-}D} = ?p_{love} \cdot a_{buy}$. A suitable semantics should let $[!\pi_{H\text{-}D}][a_{buy}]K_i p_{love}$ be valid. However, without the announcement $!\pi_{H\text{-}D}$, buying an ice cream does not mean anything: $[a_{buy}]K_i p_{love}$ should not be valid.

To prepare ourselves for the definition of the semantics, we first interpret guarded regular expressions as the languages of *guarded strings* following the definitions in [15]. A *(uniform) guarded string* over finite sets $\mathbf{P}$ and $\mathbf{\Sigma}$ is a sequence $\rho a_1 \rho a_2 \rho \dots \rho a_n \rho$ where $a_i \in \mathbf{\Sigma}$ and $\rho \subseteq \mathbf{P}$ representing the valuations of basic propositions in $\mathbf{P}$ ($p \in \rho$ iff p is true according to ρ as a valuation). For any $\rho \subseteq \mathbf{P}$, let ϕ_ρ be the characteristic formula $\phi_\rho = \bigwedge_{p \in \rho} p \wedge \bigwedge_{p \in \mathbf{P} - \rho} \neg p$. On the other hand, for any Boolean formula ψ, let $X_\psi \subseteq 2^{\mathbf{P}}$ be the corresponding set of valuations, represented by subsets of $\mathbf{P}$, that make ψ true.

Now we can define the language of guarded strings associated with a guarded regular expression over $\mathbf{\Sigma}$ and $\mathbf{P}$:

$$\begin{array}{l} \mathcal{L}_g(a) = \{\rho a \rho \mid \rho \subseteq \mathbf{P}\} \\ \mathcal{L}_g(?\psi) = \{\rho \mid \rho \in X_\psi\} \\ \mathcal{L}_g(\pi_1 \cdot \pi_2) = \{w \diamond v \mid w \in \mathcal{L}_g(\pi_1), v \in \mathcal{L}_g(\pi_2)\} \\ \mathcal{L}_g(\pi_1 + \pi_2) = \mathcal{L}_g(\pi_1) \cup \mathcal{L}_g(\pi_2) \\ \mathcal{L}_g(\pi^*) = \{\epsilon\} \cup \bigcup_{n>0}(\mathcal{L}_g(\pi^n)) \end{array}$$

where $\pi^n = \underbrace{\pi \cdots \pi}_{n}$, and $\diamond$ is the *fusion product*: $w \diamond v = w'\rho v'$ when $w = w'\rho$ and $v = \rho v'$.

We write $\pi_1 \equiv_g \pi_2$ if $\mathcal{L}_g(\pi_1) = \mathcal{L}_g(\pi_2)$. For example, we have:

$$?p \cdot ?q \cdot a \equiv_g ?(p \wedge q) \cdot a \equiv_g ?(p \wedge p) \cdot ?q \cdot a$$

$$?(p \wedge q) \cdot a + ?(p \wedge \neg q) \cdot a \equiv_g ?p \cdot a \text{ and } ?p \cdot a \cdot a \not\equiv_g ?p \cdot a.$$

We now define the language of input derivative $\pi \backslash w$ for a guarded string w as:

$$\mathcal{L}_g(\pi \backslash w) = \{v \mid w \diamond v \in \mathcal{L}_g(\pi)\}$$

and we say $w \propto_g \pi$ if $\mathcal{L}_g(\pi \backslash w) \neq \emptyset$. As in the previous section, we let $C_{\mathcal{L}_g(\pi)} = \{w \mid w \propto_g \pi\}$ and let $\mathcal{L}_g(\pi_1 \backslash \pi_2) = \{v \mid \exists w \in \mathcal{L}_g(\pi_2) \text{ and } w \diamond v \in \mathcal{L}_g(\pi_1)\}$. For example, if p is the only proposition letter then $\mathcal{L}_g((?p \cdot a \cdot b \cdot b + ?\neg p \cdot a \cdot b \cdot c) \backslash (a \cdot b)) = \{\{p\}b\{p\}, \emptyset c \emptyset\} = \mathcal{L}_g(?p \cdot b + ?\neg p \cdot c)$.

Before defining the semantics of $\mathtt{PDL}^{!?_b}$ formally, let us recall the semantics of PAL (cf. e.g., [20, 9]). Given a set of agents $\mathbf{I}$, the language of PAL extends the propositional logic with the standard knowledge operators K_i for each $i \in \mathbf{I}$ and propositional announcement operators $[!\psi]$ with the following semantics based on S5 Kripke models $(S, \{\sim_i\}_{i \in \mathbf{I}}, V)$[5]:

$$\boxed{\begin{array}{l} \mathcal{M}, s \vDash K_i \phi \Leftrightarrow \text{for all } t, \text{ if } s \sim_i t \text{ then } \mathcal{M}, t \vDash \phi \\ \mathcal{M}, s \vDash [!\psi]\phi \Leftrightarrow \text{ if } \mathcal{M}, s \vDash \psi \text{ then } \mathcal{M}|_\psi, s \vDash \phi \end{array}}$$

where $\mathcal{M}|_\psi = (S', \{\sim'_i\}_{i \in \mathbf{I}}, V')$ with $S' = \{s \in S \mid \mathcal{M}, s \vDash \psi\}$; $\sim'_i = \sim_i \cap (S' \times S')$; and $V' = V|_{S'}$ (i.e. the restriction of $V_\mathcal{M}$ on the domain S'). Intuitively the effect of announcing a formula ψ is to restrict the model to the ψ-worlds. In our setting, observing a sequence of public events is similar to hearing a sequence of public announcements, but the propositional information carried by the public events are given by the previously announced protocols, not by the syntactic forms of the public events. What we need to do in the semantics is to let agents match the protocol knowledge with their observations and find out what tests have been done when executing the protocol so far. According to the information about the tests, the agents can eliminate some possible worlds as in PAL.

Given a sequence v of atomic actions and Boolean tests, let $\mathcal{L}_p(v)$ be the subsequence of v obtained by ignoring all the tests but keeping all the public events $a_0 \ldots a_k$, e.g., $\mathcal{L}_p(?p \cdot a \cdot b) = \mathcal{L}_p(?q \cdot a \cdot ?p \cdot b) = a \cdot b$. Now we interpret $\mathtt{PDL}^{!?_b}$ on the $S5$ models $(S, \{\sim_i\}_{i \in \mathbf{I}}, V)$ as follows:

$$\boxed{\begin{array}{rl} \mathcal{M}, s \vDash \phi \Leftrightarrow & \mathcal{M}, s \vDash_{\mathbf{\Sigma}^*} \phi \\ \mathcal{M}, s \vDash_\pi p \Leftrightarrow & p \in V(s) \\ \mathcal{M}, s \vDash_\pi \neg\phi \Leftrightarrow & \mathcal{M}, s \nvDash_\pi \phi \\ \mathcal{M}, s \vDash_\pi \phi \wedge \phi' \Leftrightarrow & \mathcal{M}, s \vDash_\pi \phi \text{ and } \mathcal{M}, s \vDash_\pi \phi' \\ \mathcal{M}, s \vDash K_i \phi \Leftrightarrow & \text{for all } t\text{: } s \sim_i t \implies \mathcal{M}, t \vDash \phi \\ \mathcal{M}, s \vDash_\pi [\pi']\phi \Leftrightarrow & \text{for all } w \in \mathcal{L}(\pi') : \mathcal{M}, s \vDash \phi^w_\pi \implies \mathcal{M}|_{\phi^w_\pi}, s \vDash_{\pi \backslash w} \phi \\ \mathcal{M}, s \vDash_\pi [!\pi']\phi \Leftrightarrow & (\exists w : w = \rho v \in \mathcal{L}_g(\pi') \text{ and } V(s) = \rho) \implies \mathcal{M}, s \vDash_{\pi'} \phi \end{array}}$$

[5] An S5 model is a Kripke model where the relations $\sim_i$ are equivalence relations.

where:

$$\phi_\pi^w = \bigvee\{\phi_\rho \mid v = \rho a_1 \rho a_2 \rho \cdots \rho a_k \rho, \mathcal{L}_p(v) = w, v \propto_g \pi\}$$

Note that we do not include the transitions labelled by $a \in \mathbf{\Sigma}$ in the models since we assume that each public event is executable at each state unless it is not compliant with the current protocol (e.g., you can talk about anything in public unless constrained by some law or conventions). Since the public events are intended to be announcement-like events, we also assume that executing a protocol of such event does not result in changing the facts on the real state. This explains the uniformity of ρ in guarded strings.

Intuitively ϕ_π^w in the above clause for $[\pi']\phi$ is the propositional information agents can derive from observing $w \in \mathbf{\Sigma}^*$ in the context of protocol π. To see this, consider an observable sequence $w = a_1 a_2 \cdots a_k \in \mathbf{\Sigma}^*$, each $v = \rho a_1 \rho a_2 \rho \cdots \rho a_k \rho$ such that $v \propto_g \pi$ is a possible actual execution of the protocol consistent to the observation w. However, agents can not distinguish $v, v' \propto_g \pi$ if $\mathcal{L}_p(v) = \mathcal{L}_p(v') = w$. Therefore the disjunction ϕ_π^w is then the information which can be derived from the observation of w according to the protocol π. The intuition behind the last clause is that the new protocol is updated only if it is executable at the current pointed model.

Consider the Häagen-Dazs example, let $\mathcal{M}$ be a two-world model representing that a girl i does not know whether a boy loves her or not (she is not sure between a p_{love}-world s and a $\neg p_{love}$ world t). Let $\pi_0 = ?p_{love} \cdot a_{buy}$. Note that $\mathcal{L}_p(\{p_{love}\}a_{buy}\{p_{love}\}) = \mathcal{L}_p(\emptyset a_{buy} \emptyset) = \{a_{buy}\}$, thus $\phi_{\mathbf{\Sigma}^*}^{a_{buy}} = p \vee \neg p = \top$. On the other hand $\phi_{\pi_0}^{a_{buy}}$ is clearly p_{love}. We now show $\mathcal{M}, s \nvDash [a_{buy}]K_i p_{love}$:

$$\begin{aligned}
& \mathcal{M}, s \vDash [a_{buy}]K_i p_{love} \\
\iff & \mathcal{M}, s \vDash_{\mathbf{\Sigma}^*} [a_{buy}]K_i p_{love} \\
\iff & \text{for all } w \in \mathcal{L}(a_{buy}), \mathcal{M}, s \vDash_{\mathbf{\Sigma}^*} \phi_{\mathbf{\Sigma}^*}^w \implies \mathcal{M}|_{\phi_{\mathbf{\Sigma}^*}^w}, s \vDash_{\mathbf{\Sigma}^* \backslash w} K_i p_{love} \\
\iff & \mathcal{M}, s \vDash \phi_{\mathbf{\Sigma}^*}^{a_{buy}} \implies \mathcal{M}|_{\phi_{\mathbf{\Sigma}^*}^{a_{buy}}}, s \vDash_{\mathbf{\Sigma}^* \backslash a_{buy}} K_i p_{love} \\
\iff & \mathcal{M}, s \vDash_{\mathbf{\Sigma}^*} K_i p_{love}
\end{aligned}$$

Since $s \sim_i t$ and $\mathcal{M}, t \vDash \neg p_{love}$ then $\mathcal{M}, s \nvDash [a_{buy}]K_i p_{love}$. On the other hand:

$$\begin{aligned}
& \mathcal{M}, s \vDash [!\pi_0][a_{buy}]K_i p_{love} \\
\iff & \mathcal{M}, s \vDash_{\pi_0} [a_{buy}]K_i p_{love} \\
\iff & \mathcal{M}, s \vDash_{\pi_0} \phi_{\pi_0}^{a_{buy}} \implies \mathcal{M}|_{\phi_{\pi_0}^{a_{buy}}}, s \vDash_{\pi_0 \backslash a_{buy}} K_i p_{love} \\
\iff & \mathcal{M}|_{p_{love}}, s \vDash_{?p_{love}} K_i p_{love}
\end{aligned}$$

It is clear that $\mathcal{M}, s \vDash [!\pi_0][a_{buy}]K_i p_{love}$.

Similarly, for the we-will-see scenario mentioned in the introduction, if $\mathcal{M}$ is a two-world model representing that a Westerner i does know whether p_{no} (state s) or $\neg p_{no}$ (state t) then we can show that:

$$\mathcal{M}, s \vDash [!(?\top \cdot a_{will-see})]([a_{will-see}]\neg K_i p_{no} \wedge [!(?p_{no} \cdot a_{will-see})][a_{will-see}]K_i p_{no})$$

where $?\top \cdot a_{will-see}$ is the default protocol a Westerner may have as the standard interpretation for the sentence "we will see" which does not carry any useful information. As a more complicated example, the reader can check the model validity of the following formula on a model where agent i is not sure about p:

$$[!(?p \cdot a \cdot b + ?\neg p \cdot a \cdot c)]([a]\neg(K_i p \vee K_i \neg p) \wedge [a \cdot (b + c)](Kp \vee K\neg p)).$$

Note that the semantics of $[\pi']\phi$ is very similar to the one for public announcement, but with protocol updates and a quantification over $w \in \mathcal{L}(\pi')$. It is not hard to see that each PAL formula with Boolean announcements only can be mimicked by a $\mathtt{PDL}^{!?_b}$ formula by setting the meaning of announcements by protocols at the beginning. For example $[!p](K_i p \wedge [!q]q)$ can be reinterpreted in $\mathtt{PDL}^{!?_b}$ as $[!(?p \cdot a + ?q \cdot b)^*][a](K_i p \wedge [b]q)$. In this way we separate an announcement as an action with its meaning.

In the rest of this section we will show that $\mathtt{PDL}^{!?_b}$ can be translated back to PAL. We will follow a similar strategy as in the previous section for the expressivity of $\mathtt{PDL}^{!}$. This time we need to use automata on guarded strings.

Given $\mathbf{P}$ let $\mathcal{B}(\mathbf{P})$ be the set $2^{2^{\mathbf{P}}}$. Intuitively, $X \in \mathcal{B}(\mathbf{P})$ represent Boolean formulas over $\mathbf{P}$. We denote the corresponding formula of $X \in \mathcal{B}(\mathbf{P})$ by ϕ_X. Based on the exposition in [15], we define the automata which recognize uniform guarded strings.

Definition 1. ***(Automata on guarded strings)*** *A finite automaton on (uniform) guarded strings (or simply guarded automaton) over a finite set of actions $\mathbf{\Sigma}$ and a finite set of atomic tests $\mathbf{P}$ is a tuple $\mathsf{A} = (Q, \mathbf{\Sigma}, \mathbf{P}, q_0, \rightarrowtail, F)$ where the transitions are labelled by atomic actions in $\mathbf{\Sigma}$ (action transitions) and sets $X \in \mathcal{B}(\mathbf{P})$ (test transitions). A accepts a finite string w over $\mathbf{\Sigma} \cup \mathcal{B}(\mathbf{P})$ (notation: $w \in \mathcal{L}_{\mathbf{\Sigma}\cup\mathcal{B}(\mathbf{P})}(\mathsf{A})$), if it accepts w as a standard finite automaton over label set $\mathbf{\Sigma} \cup \mathcal{B}(\mathbf{P})$. The acceptance for guarded strings is defined based on the acceptance of normal strings and the following transformation function G which takes a string over $\mathbf{\Sigma} \cup \mathcal{B}(\mathbf{P})$ and outputs a set of uniform guarded strings.*

$$\begin{aligned} G(a) &= \{\rho a \rho \mid \rho, \rho \subseteq \mathbf{P}\} \\ G(X) &= \{\rho \mid \rho \in X\} \\ G(ww') &= \{v\rho v' \mid v\rho \in G(w) \text{ and } \rho v' \in G(w')\} \end{aligned}$$

We say A accepts a finite guarded string $v : \rho a_0 \rho \dots a_{k-1} \rho$ over $\mathbf{\Sigma}$ and $\mathbf{P}$, if $v \in G(w)$ for some string $w \in \mathcal{L}_{\mathbf{\Sigma}\cup\mathcal{B}(\mathbf{P})}(\mathsf{A})$. Let $\mathcal{L}_g(\mathsf{A})$ be the language of guarded strings accepted by A. ♠

We say a guarded automaton is *deterministic* if the following hold (cf. [15]):

- Each state is either a state that only has outgoing action transitions (*action state*) or a state that only has outgoing test transitions (*test state*).
- The outgoing action transitions are deterministic: for each action state q and each $a \in \mathbf{\Sigma}$, q has one and only one a-successor.
- The outgoing test transitions are deterministic: they are labelled by $\{\{\rho\} \mid \rho \subseteq \mathbf{P}\}$ and for each test state q and each ρ, q has one and only one $\{\rho\}$-successor. Clearly these tests ρ at a test state are logically pairwise exclusive and altogether exhaustive (viewing ρ as the Boolean formula ϕ_ρ).

– The start state is a test state and all accept states are action states.
– Each cycle contains at least one action transition.

The Kleene theorem between guarded automata and guarded regular expressions is proved in [15][6].

Theorem 2. *[15, Theorem 3.1, 3.4] For each guarded regular expression π over $\mathbf{P}$ and $\mathbf{\Sigma}$ there is a (deterministic) guarded automaton A over $\mathbf{P}$ and $\mathbf{\Sigma}$ such that $\mathcal{L}_g(\pi) = \mathcal{L}_g(\mathsf{A})$, and vice versa.*

Given a guarded regular expression π, we let $C_p(\pi) = \{\mathcal{L}_p(w) \mid w \in C_{\mathcal{L}_g(\pi)}\}$. Namely, $C_p(\pi)$ is the collection of all the possible observations of public events (no tests) according to π. Following the idea in the previous section, we need to finitely partition it.

Lemma 2. *Given a guarded regular expression π over $\mathbf{\Sigma}$ and $\mathbf{P}$, we can finitely partition $C_p(\pi)$ into test-free regular expressions $\pi_0, \dots, \pi_n$ such that for any $i \leq n : w, v \in \mathcal{L}(\pi_i) \implies \phi^w_\pi = \phi^v_\pi$ and $\pi \backslash w = \pi \backslash v$.*

Proof. (Sketch) The strategy for the proof is as follows: we first partition $C_p(\pi)$ into $\pi_0, \dots, \pi_k$ such that for any $i \leq k$, for any $w, v \in \mathcal{L}(\pi_i) \implies \phi^w_\pi = \phi^v_\pi$, then we further partition each π_i according to the shared derivatives like in the proof of Lemma 1.

From Theorem 2, we can build a deterministic guarded automaton A_π such that $\mathcal{L}_g(\mathsf{A}_\pi) = \mathcal{L}_g(\pi)$. Based on A_π it is easy to build a deterministic automaton A such that $\mathcal{L}_g(\mathsf{A}) = C_{\mathcal{L}_g(\pi)}$ by setting new accept states. From the definition of deterministic guarded automata, the start state in A has only test transitions labelled by $\{\rho\}$ for each $\rho \subseteq \mathbf{P}$. Since we only consider uniform guarded strings in the language, then an accepting path starting with a transition $\{\rho\}$ can never go through any other test transition labelled by $\{\rho'\}$ for any $\rho' \neq \rho$. Then we can prune and massage A into the following shape while keeping the accepting language intact:

$$\begin{tikzcd}
{} & s_0 \arrow[r, "a"] \arrow[dr, "b"] & \bullet \arrow[r, dotted, -] \arrow[d, dotted, -] & {} \\
q_0 \arrow[ur, "\{\rho_0\}"] \arrow[dr, "\{\rho_1\}"] \arrow[d, dotted, "\{\rho_k\}"'] & & \bullet \arrow[r, dotted, -] \arrow[d, dotted, -] & {} \\
{} \arrow[r, dotted, -] \arrow[d, dotted, "c"'] & s_1 \arrow[r, "a'"] \arrow[dr, "b'"] & \bullet \arrow[r, dotted, -] \arrow[d, dotted, -] & {} \\
{} \arrow[rr, dotted, -] & & \bullet \arrow[r, dotted, -] & {}
\end{tikzcd}$$

[6] In [15], the author considered general guarded strings whose guards need not to be uniform throughout the string. Our definitions of the languages of the guarded regular expressions and the languages of guarded automata are essentially restrictions of the corresponding definitions of languages in [15] to uniform guarded strings. Therefore the Kleene theorem proved in [15] also applies to our restricted setting.

where $k = |2^{\mathbf{P}}|$, the start state q_0 is the only test state, and there is no incoming transition at q_0.

Let B_{s_i} be the standard finite automaton over the action set $\mathbf{\Sigma}$: $(Q_{act}, \mathbf{\Sigma}, s_i, \rightarrowtail, F)$ where Q_{act} is the set of action states in Q, F is the set of accept states in A, and $q \overset{a}{\rightarrowtail} q'$ in $\mathsf{B}_{s_i} \iff q \overset{a}{\rightarrowtail} q'$ in A. Given $Z \subseteq \{\rho_0, \dots, \rho_k\}$ (intuitively a Boolean formula), let D_Z be the product automaton $\Pi_{\rho_i \in Z}\mathsf{B}_{s_i} \times \Pi_{\rho_i \notin Z}\overline{\mathsf{B}_{s_i}}$ where $\overline{\mathsf{B}_{s_i}}$ is the complement automaton of B_{s_i}. We can show that D_Z recognizes all the sequences w based on $\mathbf{\Sigma}$ such that $\{\rho \mid w = \mathcal{L}_p(v), v = \rho a_1 \rho \cdots \rho a_k \rho \in \mathcal{L}_g(\mathsf{A})\} = Z$. By Kleene Theorem, we can turn each D_Z into a regular expression.

Thus, we can finitely partitioned $C_p(\pi)$ into $\pi_0, \dots, \pi_n$ such that for any $i \leq n$, for any $w, v \in \mathcal{L}(\pi_i) \implies \phi^w_\pi = \phi^v_\pi$. By the similar techniques as in the proof of Lemma 1, we can further partition each of these regular expressions π_i into finitely many regular expressions $\pi_{i0} \dots \pi_{im}$ such that for any $w, v \in \mathcal{L}(\pi_{ij})$: $\pi \backslash w = \pi \backslash v$. This gives us the desired final partition. $\Box$

Now we define the following translation from $\mathtt{PDL}^{!?_b}$ to $\mathtt{PAL}$:

$$\begin{array}{rcl}
t(\phi) & = & t_{\mathbf{\Sigma}^*}(\phi) \\
t_\pi(p) & = & p \\
t_\pi(\neg\phi) & = & \neg t_\pi(\phi) \\
t_\pi(\phi_1 \wedge \phi_2) & = & t_\pi(\phi_1) \wedge t_\pi(\phi_2) \\
t_\pi(K_i\phi) & = & K_i t_\pi(\phi) \\
t_\pi([\pi']\phi) & = & \bigwedge\{[!\psi_j]t_{\theta_j}(\phi) \mid \mathcal{L}(\pi_j) \cap \mathcal{L}(\pi') \neq \emptyset\} \\
t_\pi([!\pi']\phi) & = & \chi_{\pi'} \rightarrow t_{\pi'}(\phi)
\end{array}$$

where:

- all the π_j form a partition of $C_p(\pi)$ satisfying the desired properties stated in the above lemma,
- $\psi_j = \phi^w_\pi$ for some $w \in \mathcal{L}(\pi_j)$,
- $\theta_j = \pi \backslash w$ for some $w \in \mathcal{L}(\pi_j)$,
- $\chi_{\pi'} = \bigvee\{\phi_\rho \mid \rho v \in \mathcal{L}_g(\pi')$ for some sequnece $v\}$.

Note that by the properties of the partition, ψ_j and θ_j are well-defined. Intuitively, $\chi_{\pi'}$ is the precondition of π' to be executed. Based on the above translation t, it is not hard to prove:

Theorem 3. *For any pointed $S5$ Kripke model $\mathcal{M} = (S, \{\sim_i\}_{i \in \mathbf{I}}, V, s)$:*

$$\mathcal{M}, s \vDash_{\mathtt{PDL}^{!?_b}} \phi \iff \mathcal{M}, s \vDash_{\mathtt{PAL}} t(\phi).$$

Since $\mathtt{PAL}$ is equally expressive as the standard epistemic logic ($\mathtt{EL}$) without any announcement operators (cf. e.g., [9]) and $\mathtt{PDL}^{!?_b}$ is clearly at least as expressive as $\mathtt{EL}$ then $\mathtt{PAL}$ and $\mathtt{PDL}^{!?_b}$ are equally expressive.

4 Conclusions and Future Work

In this paper we proposed two PDL-style logics with simple and natural operators for reasoning about protocol changes and knowledge updates: Logic $\mathtt{PDL}^{!}$ handles protocol changes in a context without knowledge; $\mathtt{PDL}^{!?_b}$ includes tests in the protocols and knowledge operators to deal with the situations where events carry information for agents according to the knowledge of the protocols. We showed that $\mathtt{PDL}^{!}$ is equally expressive as the test-free PDL and $\mathtt{PDL}^{!?_b}$ is equally expressive as PAL. By using the new languages, what we gain is the explicitness and convenience in modelling scenarios with protocol changes and knowledge updates, as we demonstrated by various examples. In [25] we also investigated another closely related logic $\mathtt{PDL}^{\boxtimes}$, which extends the DEL framework with more general product update operations taking general guarded automata as update models. For interested readers who want to see more applications of the protocol changing operations, we refer to [27] where we integrated a similar protocol changing operator in a specific setting of communications over channels.

It is shown in [16] that the public announcement logic, though equally expressive as epistemic logic, is exponentially more succinct than the pure epistemic logic in expressing certain properties on K models. Here we conjecture that similar results apply to our new logics as well. However, we leave out the succinctness and complexity analysis for future work.

Also note that the logic $\mathtt{PDL}^{!?_b}$ is interpreted on S5 epistemic models which do not have action transitions, since we implicitly assume all the public events are like public communications by words which are always executable unless constrained by protocols. We can well consider more general models with action transitions or consider actions that can change the facts in models as discussed in [23]. Another restriction in $\mathtt{PDL}^{!?_b}$ is that we only consider Boolean tests for simplicity. To make the logic more interesting we would like to include epistemic tests in the future. Such extensions may essentially increase the expressive power of the logic.

Acknowledgements The author would like to thank Ram Ramanujam and Hans van Ditmarsch for their insightful remarks on various versions of this work, and the anonymous referees for their useful comments.

References

1. G. Aucher. BMS revisited. In *Proceedings of TARK '09*, pages 24–33, 2009.
2. Mario Benevides and L. Schechter. A propositional dynamic logic for CCS programs. *Logic, Language, Information and Computation*, pages 83–97, 2008.
3. D. Bonnay and P. Égré. Inexact knowledge with introspection. *Journal of Philosophical Logic*, 38(2):179–227, 2009.
4. J. A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.
5. J. H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, 1971.

6. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. Knowledge-based programs. *Distributed Computing*, 10(4):199–225, 1997.
7. M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.
8. D. M. Gabbay. A theory of hypermodal logics: Mode shifting in modal logic. *Journal of Philosophical Logic*, 31(3):211–243, 2002.
9. J. Gerbrandy and W. Groeneveld. Reasoning about information change. *Journal of Logic, Language and Information*, 6(2):147–169, 1997.
10. J. Y. Halpern and R. Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3(4):159–177, 1989.
11. D. Harel, D. Kozen, and R. Parikh. Process logic: Expressiveness, decidability, completeness. *Journal of Computer and System Sciences*, 25(2):144–170, 1982.
12. T. Hoshi. *Epistemic Dynamics and Protocol Information.* PhD thesis, Stanford, 2009.
13. T. Hoshi and A. Yap. Dynamic epistemic logic with branching temporal structures. *Synthese*, 169(2):259–281, 2009.
14. D. Kozen. A completeness theorem for kleene algebras and the algebra of regular events. In *Proceedings of LiCS '91*, pages 214–225, 1991.
15. D. Kozen. Automata on guarded strings and applications. *Matématica Contemporânea*, 24:117–139, 2003.
16. Carsten Lutz. Complexity and succinctness of public announcement logic. In *Proceedings of AAMAS '06*, pages 137–143, New York, NY, USA, 2006. ACM.
17. J. J. Meyer. A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136, 1987.
18. R. Parikh and R. Ramanujam. A knowledge based semantics of messages. *Journal of Logic, Language and Information*, 12(4), 2003.
19. S. Paul, R. Ramanujam, and S. Simon. Stability under strategy switching. In Klaus Ambos-Spies, Benedikt Löwe, and Wolfgang Merkle, editors, *Mathematical Theory and Computational Practice*, volume 5635, chapter 40, pages 389–398. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
20. J. A. Plaza. Logics of public communications. In M. L. Emrich, M. S. Pfeifer, M. Hadzikadic, and Z. W. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989.
21. V. R. Pratt. Process logic. In *Proceedings of POPL '79*, pages 93–100, New York, NY, USA, 1979. ACM.
22. J. van Benthem, J. Gerbrandy, T. Hoshi, and E Pacuit. Merging frameworks for interaction. *Journal of Philosophical Logic*, 38(5):491–526, 2009.
23. J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.
24. Y. Wang. Indexed semantics and its application in modelling interactive unawareness. Master's thesis, University of Amsterdam, 2006.
25. Y. Wang. *Epistemic Modelling and Protocol Dynamics.* PhD thesis, University of Amsterdam, 2010.
26. Y. Wang, L. Kuppusamy, and J. van Eijck. Verifying epistemic protocols under common knowledge. In *Proceedings of TARK '09*, pages 257–266, New York, NY, USA, 2009. ACM.
27. Y. Wang, F. Sietsma, and J. van Eijck. Logic of information flow on communication channels (extended abstract). In van der Hoek, Kaminka, Lespérance, Luck, and Sen, editors, *Proceedings of AAMAS '10)*, 2010.