



UvA-DARE (Digital Academic Repository)

Epistemic modelling and protocol dynamics

Wang, Y.

Publication date
2010

[Link to publication](#)

Citation for published version (APA):

Wang, Y. (2010). *Epistemic modelling and protocol dynamics*. [Thesis, fully internal, Universiteit van Amsterdam]. Institute for Logic, Language and Computation.

General rights

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

This chapter introduces a few very basic concepts and notations which are frequently used throughout the thesis. In the follow-up chapters, we will refer to the definitions in this chapter when needed.

2.1 Finite Automata and Regular Expressions

2.1.1. DEFINITION. (Finite Automata on Finite Words) A (non-deterministic) finite automaton is a tuple $A = (Q, \Sigma, q_0, \succrightarrow, F)$ where:

- Q is a finite non-empty set of states, with $q_0 \in Q$ being the *start state*;
- Σ is an alphabet;
- $\succrightarrow \subseteq Q \times \Sigma \times Q$ is the set of labelled transitions over Q ;
- $F \subseteq Q$ is the set of *accept states*.

□

Notation For any $a \in \Sigma$, we write \xrightarrow{a} for $\{(q, q') \mid (q, a, q') \in \succrightarrow\}$. Let Σ^* be the set of finite (possibly empty) strings of labels in Σ , for any $w = (a_0, a_1, \dots, a_n) \in \Sigma^*$, we write $q \xrightarrow{w} q'$ if there is a path $q \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots \xrightarrow{a_n} q'$ in A . Given an unspecified finite automaton A we use $Q_A, \Sigma_A, q_A, \succrightarrow_A$ and F_A for the corresponding components in the definition of the automaton.

Given \succrightarrow , we let the induced transition function $\delta : Q \times \Sigma \mapsto 2^Q$ be defined as $\delta(q, a) = \{q' \mid q \xrightarrow{a} q'\}$. Note that $\delta(q, a)$ may be \emptyset for some q and a . A finite automaton on finite words A is said to be *deterministic*, if for any $q \in Q_A$ and $a \in \Sigma$: $\delta(q, a)$ is a singleton. We can extend the transition function δ to $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$ such that $\delta^*(q, w) = \{q' \mid q \xrightarrow{w} q'\}$. It is clear that deterministic finite automata (DFA) have the property that for any word $w \in \Sigma^*$, $\delta^*(q, w)$ is a singleton.

Given a finite automaton $A = (Q, \Sigma, q_0, \succrightarrow, F)$ and a word $w = (a_1, \dots, a_n) \in \Sigma^*$, we call a sequence $r = (q_0, q_1, \dots, q_n)$ a *run* of A over w if for $0 \leq i \leq n$: $q_i \xrightarrow{a_{i+1}} q_{i+1}$.

A run $r = (q_0, \dots, q_n)$ is said to be *accepting* if $q_n \in F$. We say \mathbf{A} *accepts* w if there exists an accepting run of \mathbf{A} over w . The *language* of a finite automaton \mathbf{A} is the set $\mathcal{L}(\mathbf{A}) = \{w \in \Sigma^* \mid \mathbf{A} \text{ accepts } w\}$. We say \mathbf{A} and \mathbf{A}' are *language equivalent* ($\mathbf{A} =_{\mathcal{L}} \mathbf{A}'$) if $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{A}')$.

Given an alphabet Σ , regular expressions over Σ are of the form:

$$\pi ::= \mathbf{0} \mid \mathbf{1} \mid a \mid \pi + \pi \mid \pi \cdot \pi \mid \pi^*$$

where $a \in \Sigma$ and $\mathbf{0}, \mathbf{1}$ are constants for the *empty language* and the *empty string* respectively. We let Reg_{Σ} be the set of all the regular expressions over Σ .

Given $L, L' \subseteq \Sigma^*$, we define $L \circ L'$ to be the set $\{wv \mid w \in L, v \in L'\}$. For $n \geq 0$ we define $L^0 = \{\epsilon\}$ and $L^{n+1} = L \circ L^n$ where ϵ is the empty string. We write L^* for $\bigcup_{n \geq 0} L^n$.

2.1.2. DEFINITION. (Language of Regular Expressions) The language of a regular expression π (denoted as $\mathcal{L}(\pi)$) is a set of finite strings over Σ defined as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{0}) &= \emptyset & \mathcal{L}(\mathbf{1}) &= \{\epsilon\} & \mathcal{L}(a) &= \{a\} \\ \mathcal{L}(\pi \cdot \pi') &= \mathcal{L}(\pi) \circ \mathcal{L}(\pi') \\ \mathcal{L}(\pi + \pi') &= \mathcal{L}(\pi) \cup \mathcal{L}(\pi') \\ \mathcal{L}(\pi^*) &= (\mathcal{L}(\pi))^* \end{aligned}$$

□

The following result is well-known:

2.1.3. THEOREM (Kleene's Theorem). *For any regular expression π , there exists a finite (deterministic) automaton \mathbf{A} such that $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\pi)$. For any finite (deterministic) automaton \mathbf{A} there is a regular expression π such that $\mathcal{L}(\pi) = \mathcal{L}(\mathbf{A})$.*

2.2 Kripke Models and Bisimulation

2.2.1. DEFINITION. (Kripke Model) A *Kripke model (KM)* is a tuple:

$$\mathcal{M} = (S, \mathbf{P}, \Sigma, \rightarrow, V)$$

where:

- S is a non-empty set of states (or *possible worlds*);
- \mathbf{P} is a set of proposition letters;
- Σ is a non-empty set of labels;
- $\rightarrow \subseteq S \times \Sigma \times S$ is the set of labelled relations over S ;
- $V : S \rightarrow 2^{\mathbf{P}}$ is the valuation function.

We call \mathbf{P} the *vocabulary* of \mathcal{M} and Σ the *set of labels* of \mathcal{M} . (\mathbf{P}, Σ) is called the *signature* of \mathcal{M} . A pointed Kripke model (\mathcal{M}, s) is a KM with a designated point in the set of states. Following the tradition in modal logic, we shall call $\mathcal{F} = (S, \mathbf{P}, \Sigma, \rightarrow)$ a *Kripke frame*. \mathfrak{M}

As \xrightarrow{w} in the case of finite automata, we adapt the notion $s \xrightarrow{a} t$ for $a \in \Sigma$ and $w \in \Sigma^*$ in the context of Kripke models, similarly for $S_{\mathcal{M}}, \mathbf{P}_{\mathcal{M}}, \Sigma_{\mathcal{M}}, \rightarrow_{\mathcal{M}}$ and $V_{\mathcal{M}}$.

A Kripke model \mathcal{M} is said to be *finite*, if $S_{\mathcal{M}}, \Sigma_{\mathcal{M}}$ and $\mathbf{P}_{\mathcal{M}}$ are all finite. A Kripke model is *image-finite* or *finitely branching* if for every state and every label $a \in \Sigma$, there are only at most finitely many a -successors; it is ω -*branching* if for every state and every label $a \in \Sigma$, there are only at most countably many a -successors.

An *S5 Kripke model* \mathcal{M} is a KM whose labelled relations are *equivalence* relations, i.e., for all $a \in \Sigma_{\mathcal{M}}$: \xrightarrow{a} is *reflexive* ($\forall s : s \xrightarrow{a} s$), *symmetric* ($\forall s, t : s \xrightarrow{a} t \iff t \xrightarrow{a} s$), and *transitive* ($\forall s, t, r : (s \xrightarrow{a} t \wedge t \xrightarrow{a} r) \implies s \xrightarrow{a} r$). Therefore, in the case of S5 models, we also use \sim to denote the set of relations. S5 models are standard models for *epistemic logic* where the set of labels are interpreted as the set of agents. In such a context we may use \mathbf{I} instead of Σ when defining an S5 model and use \sim_i instead of \xrightarrow{i} for $i \in \mathbf{I}$, following the standard notations in epistemic logic.

Note that in computer science a Kripke frame is usually called a *Labelled Transition System (LTS)* and Kripke models are sometimes called *Kripke Labelled Transition Systems (KLTS)*.

2.2.2. DEFINITION. (Bisimulation) A binary relation R between the domains of two KMs $\mathcal{M} = (S, \mathbf{P}, \Sigma, \rightarrow, V)$ and $\mathcal{N} = (T, \mathbf{P}', \Sigma', \rightarrow', V')$ is called a *bisimulation* iff $(s, t) \in R$ implies that the following conditions hold:

Invariance For any $p \in \mathbf{P} : p \in V(s) \iff p \in V'(t)$.

Zig if $s \xrightarrow{a} s'$ in \mathcal{M} then there exists a t' in \mathcal{N} such that $t \xrightarrow{a} t'$ and $s'Rt'$.

Zag if $t \xrightarrow{a} t'$ in \mathcal{N} then there exists an s' in \mathcal{M} such that $s \xrightarrow{a} s'$ and $s'Rt'$.

Two pointed Kripke models (\mathcal{M}, s) and (\mathcal{N}, t) are said to be *bisimilar* $(\mathcal{M}, s \Leftrightarrow \mathcal{N}, t)$ if there is a bisimulation R between them such that $(s, t) \in R$. We say a bisimulation R is *total*, if every world in one model is linked by R to some world in the other model. We write $\mathcal{M} \Leftrightarrow \mathcal{N}$ if there is a total bisimulation between \mathcal{M} and \mathcal{N} . \mathfrak{M}

Note that the above standard bisimulation is defined between models with the same signature. In this thesis we will also work with models with different vocabularies. We say two pointed models (\mathcal{M}, s) and (\mathcal{N}, t) are *restricted bisimilar* w.r.t $\mathbf{P}' \subseteq \mathbf{P}_{\mathcal{M}} \cap \mathbf{P}_{\mathcal{N}}$ (notation: $\mathcal{M}, s \Leftrightarrow_{\mathbf{P}'} \mathcal{N}, t$), if \mathcal{M}, s and \mathcal{N}, t are bisimilar with the original invariance condition replaced by a *restricted invariance* condition:

[P'-Invariance] for any $p \in \mathbf{P}' : p \in V_{\mathcal{M}}(s) \iff p \in V_{\mathcal{N}}(t)$.

Similarly we can define total restricted bisimulation w.r.t \mathbf{P}' ($\mathcal{M} \stackrel{\mathbf{P}'}{\simeq} \mathcal{N}$) in the straightforward way.

Note that an *autobisimulation* of a model is an equivalence relation on the state space of a model. Thus we can have a quotient model w.r.t to the maximal autobisimulation on a model.

2.2.3. DEFINITION. (Bisimulation Contraction) Given a Kripke model $\mathcal{M} = (S, \mathbf{P}, \Sigma, \rightarrow, V)$, let $\equiv_b \subseteq S \times S$ be the *autobisimulation*: $\{(s, t) \mid \mathcal{M}, s \stackrel{\mathbf{P}}{\simeq} \mathcal{M}, t\}$. The *bisimulation contraction* of \mathcal{M} is the quotient model

$$\mathcal{M}_{/\equiv_b} = (S', \mathbf{P}, \Sigma, \rightarrow', V')$$

where:

- $S' = \{[s] \mid s \in S\}$ where $[s]$ is the equivalence class containing s w.r.t \equiv_b ;
- $([s], a, [t]) \in \rightarrow'$ iff $(s, a, t) \in \rightarrow$;
- $V'([s]) = V(s)$.

□

We can adapt the definition of bisimulation for finite automata by replacing the invariance condition with the following:

$$[\text{Accept Invariance}] : s \in F \iff t \in F'$$

where F and F' are the sets of accept states in two automata. We say automata \mathbf{A} and \mathbf{B} are bisimilar if there is a bisimulation R between $Q_{\mathbf{A}}$ and $Q_{\mathbf{B}}$ with the accept invariance condition such that $(q_{\mathbf{A}}, q_{\mathbf{B}}) \in R$. It is easy to see that $\mathbf{A} \stackrel{\mathbf{P}}{\simeq} \mathbf{A}' \implies \mathbf{A} =_{\mathcal{L}} \mathbf{A}'$, but the converse does not hold.

2.2.4. DEFINITION. (n -round Bisimulation Game) An n -round bisimulation game $\mathcal{G}_n((\mathcal{M}, s), (\mathcal{N}, t))$ between two pointed KMs (\mathcal{M}, s) and (\mathcal{N}, t) with the same signature is a two player game based on the configurations in $S_{\mathcal{M}} \times S_{\mathcal{N}}$. The initial configuration is (s, t) and the players, Spoiler and Verifier, play in rounds. Each round consists of two moves: first by Spoiler and then by Verifier. At each configuration (s', t') , there are two options:

- Spoiler selects an $a \in \Sigma$ and a state s'' in \mathcal{M} such that $s' \xrightarrow{a}_{\mathcal{M}} s''$ and then Verifier needs to come up with a state in \mathcal{N} such that $t' \xrightarrow{a}_{\mathcal{N}} t''$ and $V(s'') = V(t'')$. The configuration is then changed to (s'', t'') .
- Spoiler selects an $a \in \Sigma$ and a state t'' in \mathcal{N} such that $t' \xrightarrow{a}_{\mathcal{N}} t''$ and then Verifier needs to respond with a state in \mathcal{M} such that $s' \xrightarrow{a}_{\mathcal{M}} s''$ and $V(s'') = V(t'')$. The configuration is then changed to (s'', t'') .

Spoiler wins the game if within $n - 1$ rounds some configuration (s', t') is reached such that Spoiler can make a legal move but Verifier does not have a legal move to respond. Verifier wins the game otherwise. \mathfrak{M}

We say (\mathcal{M}, s) and (\mathcal{N}, t) are *modally equivalent* ($\mathcal{M}, s \equiv_{\text{ML}} \mathcal{N}, t$) if \mathcal{M}, s and \mathcal{N}, t satisfy exactly the same basic modal logic (ML) formulas¹. The following facts are well known (cf., e.g., [BdRV02]).

2.2.5. FACT. For image-finite pointed Kripke models (\mathcal{M}, s) and (\mathcal{N}, t) , the following are equivalent:

- $\mathcal{M}, s \Leftrightarrow \mathcal{N}, t$.
- $\mathcal{M}, s \equiv_{\text{ML}} \mathcal{N}, t$.
- for all $n \in \mathbb{N}$: Verifier has a winning strategy in the game $\mathcal{G}_n((\mathcal{M}, s), (\mathcal{N}, t))$.

\mathfrak{M}

2.3 Three Logics

2.3.1 Propositional Dynamic Logic

Propositional Dynamic Logic (PDL), introduced by Fischer and Ladner [FL79] (following the idea of [Pra76]), is a branching-time logic of programs (represented by regular expressions):

$$\phi ::= \top \mid p \mid \phi \wedge \psi \mid \neg\phi \mid \langle \pi \rangle \phi$$

where p ranges over a set of propositions \mathbf{P} and π is a regular expression over some alphabet Σ with *tests* in terms of PDL formulas:

$$\pi ::= \mathbf{0} \mid \mathbf{1} \mid a \mid ?\phi \mid \pi + \pi \mid \pi \cdot \pi \mid \pi^*$$

where $a \in \Sigma$. When Σ is not fixed, we use PDL_Σ to denote the PDL language based on Σ . As usual, we define \perp , $\phi \vee \psi$, $\phi \rightarrow \psi$ and $[\pi]\phi$ as the abbreviations of $\neg\top$, $\neg(\neg\phi \wedge \neg\psi)$, $\neg\phi \vee \psi$ and $\neg\langle \pi \rangle \neg\phi$ respectively.

Intuitively, $\langle \pi \rangle \phi$ says that there is an execution of program π such that after the execution ϕ holds.

We define the satisfaction relation \models between a pointed model (\mathcal{M}, s) with the signature (\mathbf{P}, Σ) and a PDL_Σ formula ϕ as follows:

$\mathcal{M}, s \models p$	\Leftrightarrow	$p \in V_{\mathcal{M}}(s)$
$\mathcal{M}, s \models \neg\phi$	\Leftrightarrow	$\mathcal{M}, s \not\models \phi$
$\mathcal{M}, s \models \phi \wedge \psi$	\Leftrightarrow	$\mathcal{M}, s \models \phi$ and $\mathcal{M}, s \models \psi$
$\mathcal{M}, s \models \langle \pi \rangle \phi$	\Leftrightarrow	$\exists s' : s \llbracket \pi \rrbracket s'$ and $\mathcal{M}, s' \models \phi$

where $\llbracket \pi \rrbracket$ is defined as:

¹ML extends propositional logic with modal formulas $\Box\phi$ and their Boolean combinations.

$s \llbracket \mathbf{1} \rrbracket s'$	$\Leftrightarrow s = s'$
$s \llbracket \mathbf{0} \rrbracket s'$	$\Leftrightarrow \text{never}$
$s \llbracket a \rrbracket s'$	$\Leftrightarrow s \xrightarrow{a} s'$
$s \llbracket ?\psi \rrbracket s'$	$\Leftrightarrow s = s' \text{ and } \mathcal{M}, s' \models \psi$
$s \llbracket \pi_1 \cdot \pi_2 \rrbracket s'$	$\Leftrightarrow s \llbracket \pi_1 \rrbracket \circ \llbracket \pi_2 \rrbracket s'$
$s \llbracket \pi_1 + \pi_2 \rrbracket s'$	$\Leftrightarrow s \llbracket \pi_1 \rrbracket \cup \llbracket \pi_2 \rrbracket s'$
$s \llbracket (\pi_1)^* \rrbracket s'$	$\Leftrightarrow s \llbracket \pi_1 \rrbracket^* s'$

where \circ , \cup and $*$ are the usual composition, union and reflexive transitive closure on relations respectively.

We can view π as a regular expression over $\Sigma \cup \{?\phi \mid ?\phi \text{ appears in } \pi\}$, then:

$\mathcal{M}, s \models \langle \pi \rangle \phi \Leftrightarrow$ there exists a path $s = s_0 \llbracket a_1 \rrbracket s_1 \llbracket a_2 \rrbracket \cdots \llbracket a_n \rrbracket s_n$ in \mathcal{M} such that $\mathcal{M}, s_n \models \phi$ and $a_0 a_1 \dots a_n \in \mathcal{L}(\pi)$

PDL can be axiomatized by the following axioms and inference rules [Seg82, Par78]²:

TAUTOLOGY	all the tautologies
K	$[\pi](\phi \rightarrow \phi') \rightarrow ([\pi]\phi \rightarrow [\pi]\phi')$
0	$[\mathbf{0}]\phi \leftrightarrow \top$
1	$[\mathbf{1}]\phi \leftrightarrow \phi$
TEST	$[\psi]\phi \leftrightarrow (\psi \rightarrow \phi)$
SEQ	$\langle \pi_1 \cdot \pi_2 \rangle \phi \leftrightarrow \langle \pi_1 \rangle \langle \pi_2 \rangle \phi$
OR	$\langle \pi_1 + \pi_2 \rangle \phi \leftrightarrow (\langle \pi_1 \rangle \phi \vee \langle \pi_2 \rangle \phi)$
Star1	$\langle \pi^* \rangle \phi \leftrightarrow (\phi \vee \langle \pi \rangle \langle \pi^* \rangle \phi)$
Star2	$[\pi^*](\phi \rightarrow [\pi]\phi) \rightarrow (\phi \rightarrow [\pi^*]\phi)$
Rules	
\square	$\frac{\phi}{[\pi]\phi}$
MP	$\frac{\phi, \phi \rightarrow \psi}{\psi}$

Note that with the presence of tests $?\phi$ we can eliminate basic programs **0** and **1** by defining them as $?\perp$ and $?\top$ respectively. Sometimes we are interested in the *test-free* fragment of PDL in which we do not have $?$ as one of the program constructors but we do have **0** and **1**.

We write $K_i\phi$ (i knows ϕ) and $\widehat{K}_i\phi$ (i thinks ϕ is possible) for $[i]\phi$ and $\langle i \rangle \phi$ respectively, when interpreting PDL_I on S5 models in an epistemic setting. We write $C_G\phi$ (ϕ is common knowledge among the agents in G) as $[(i_1 + \cdots + i_n)^*]\phi$ if $G = i_1, \dots, i_n \subseteq \mathbf{I}$.

²The PDL formulas which are valid (i.e. hold on all the pointed models) are precisely the ones that can be derived from the following proof system.

2.3.2 Epistemic Temporal Logic

Developed independently by [PR85] and [HM90], and later made popular by the seminal book [FHMV95], the *Interpreted Systems* (IS) (or *Epistemic Temporal Logic* (ETL)) framework nicely combines the temporal developments of a system (in runs) with epistemic ones in a distributed setting. Following the exposition in [FHMV95], we give the definition of interpreted systems as follows:

2.3.1. DEFINITION. (Interpreted System) Given a set of agents $\mathbf{I} = \{i_1, \dots, i_n\}$, given $n + 1$ non-empty sets $L_\varepsilon, L_1, \dots, L_n$ of *local states* of (one for the environment ε , and one for each agent in \mathbf{I}), the set of *global states* for an interpreted system is a set $S \subseteq L_\varepsilon \times L_1 \times \dots \times L_n$. An interpreted system \mathcal{I} is a triple (R, P, V) where R is a set of *runs*, i.e. functions $r : \mathbb{N} \rightarrow S$, and $V : S \mapsto 2^{\mathbf{P}}$ is a valuation function. We denote the finite history (m -prefix) of a run r as (r, m) . (r, m) and (r', m') are indistinguishable for agent i (notation: $(r, m) \sim_i (r', m')$) if global states $r(m)$ and $r'(m')$ have the same local state for i . A *pointed IS* is an IS with a designated finite history, e.g., \mathcal{I}, r, n . \blacksquare

Each interpreted system can be viewed as an infinite Kripke model with the set of labels $\mathbf{I} \cup \{\tau\}$ where for each $i \in \mathbf{I} : \sim_i$ is an equivalence relation, and $\xrightarrow{\tau}$ represents the temporal development of the system. In the setting of ETL [PR85], the temporal transitions are labelled with explicit actions e in a set Σ . Various Epistemic Temporal languages can be defined on such models, for example, the simplest language is:

$$\phi ::= \top \mid p \mid \phi \wedge \psi \mid \neg\phi \mid K_i\phi \mid \langle e \rangle\phi$$

with the following semantics on IS:

$\mathcal{I}, r, n \models p$	\Leftrightarrow	$p \in V_{\mathcal{I}}(r(n))$
$\mathcal{I}, r, n \models \neg\phi$	\Leftrightarrow	$\mathcal{I}, r, n \not\models \phi$
$\mathcal{I}, r, n \models \phi \wedge \psi$	\Leftrightarrow	$\mathcal{I}, r, n \models \phi$ and $\mathcal{I}, r, n \models \psi$
$\mathcal{I}, r, n \models K_i\phi$	\Leftrightarrow	for all (r', m) such that $(r, n) \sim_i (r', m) : \mathcal{I}, r', m \models \phi$
$\mathcal{I}, r, n \models \langle e \rangle\phi$	\Leftrightarrow	$(r, n) \xrightarrow{e} (r, n + 1)$ and $\mathcal{I}, r, n + 1 \models \phi$

$\langle e \rangle$ in the above simple language can be replaced by any temporal operator thus obtaining more expressive epistemic temporal logics.

2.3.3 Dynamic Epistemic Logic

A different perspective on the dynamics of multi-agent system is provided by the development of so-called Dynamic Epistemic Logic (DEL) [Pla89, GG97, BMS98]. The focus of DEL is not on the temporal structure of the system but rather on the epistemic impact of the events as the agents perceive them. The following PDL-style DEL language is based on the exposition in [vBvEK06]:

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle \mathcal{A}, e \rangle\phi \mid \langle \pi \rangle\phi$$

where \mathcal{A} is an *event model* defined below with e as its designated event.

2.3.2. DEFINITION. (Event Model) An event model \mathcal{A} is a tuple:

$$\mathcal{A} = (E, \Sigma, \succrightarrow, Pre)$$

where:

- E is a finite non-empty set of events.
- Σ is a set of labels.
- $\succrightarrow \subseteq E \times \Sigma \times E$ is the set of labelled transitions.
- $Pre : E \mapsto Form(\text{DEL})$ where $Form(\text{DEL})$ is the set of DEL formulas. Intuitively, Pre assigns to each action a precondition in the form of a DEL formula that can be constructed in an earlier stage of the inductive definition of the language.

□

Notation In the epistemic setting, the relations \succrightarrow^i in the action model are assumed to be equivalence relations, thus we may use \leftrightarrow_i to denote them. \leftrightarrow_i models agent i 's observational power on events in E (e.g. $e_1 \leftrightarrow_i e_2$ means agent i can not distinguish event e_1 and e_2).

The semantics for PDL formulas is as usual and for $\langle \mathcal{A}, e \rangle \phi$:

$$\boxed{\mathcal{M}, s \vDash \langle \mathcal{A}, e \rangle \phi \iff \mathcal{M}, s \vDash Pre(e) \text{ and } \mathcal{M} \otimes \mathcal{A}, (s, e) \vDash \phi}$$

where \otimes is defined as follows:

2.3.3. DEFINITION. (Product Update \otimes) Given a Kripke model $\mathcal{M} = (S, \Sigma, \rightarrow, V)$ and an event model $\mathcal{A} = (E, \Sigma, \succrightarrow, Pre)$, the product model $(\mathcal{M} \otimes \mathcal{A})$ is a Kripke model $(\mathcal{M} \otimes \mathcal{A}) = (S', \Sigma, \rightarrow', V')$ where:

$$\begin{aligned} S' &= \{(s, e) \mid \mathcal{M}, s \vDash Pre(e)\} \\ \xrightarrow{a'} &= \{((s, e), (s', e')) \mid s \xrightarrow{a} s' \text{ and } e \succrightarrow^a e'\} \\ V'((s, e)) &= V(s) \end{aligned}$$

□

The simplest event model is perhaps the one modelling a *public announcement* of ϕ (notation: $!\phi$) depicted as the following event model $\mathcal{A}_{!\phi}$:



where ϕ is the precondition of this singleton model, and \xrightarrow{I} denotes the reflexive relations for each $i \in I$. Let $\mathcal{M}_{!\phi}$ be the Kripke model $(S, \mathbf{P}, \mathbf{I}, \sim, V)$ where:

- $S = \{s \in S_{\mathcal{M}} \mid \mathcal{M}, s \vDash \phi\}$;
- $\sim = \sim_{\mathcal{M}} \cap (S \times \Sigma \times S)$;

- $V = V_{\mathcal{M}}|_S$ (i.e. the restriction of $V_{\mathcal{M}}$ on the domain S).

It is easy to see that updating $\mathcal{A}_{!}\phi$ on a static model \mathcal{M} amounts to restricting the \mathcal{M} by the states which satisfy ϕ : $\mathcal{M} \otimes \mathcal{A}_{!}\phi \simeq \mathcal{M}|_{\phi}$.

As a simple yet important fragment of DEL, the *Public Announcement Logic* (PAL) [Pla89, GG97] is usually presented as follows:

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid K_i\phi \mid [!\psi]\phi$$

where $K_i\phi$ and $[!\psi]\phi$ are equivalent to $[i]\phi$ and $[A_{!}\psi]\phi$ in DEL respectively.

As for the expressivity of DEL, [vBvEK06] showed that adding product updates to PDL does not increase the expressive power of PDL:

2.3.4. FACT. ([vBvEK06]) For any DEL formula ϕ there is a PDL formula ϕ' such that for all pointed Kripke models \mathcal{M}, s : $\mathcal{M}, s \models_{\text{DEL}} \phi \iff \mathcal{M}, s \models_{\text{PDL}} \phi'$. \heartsuit