Protocol Synthesis in the Dynamic Gossip Problem

Leo Poulson

March 2, 2019

1 Dynamic Epistemic Logic

1.1 Epistemic Logic

Epistemic Logic is a logic for the study of knowledge for a set of agents I who want to reason about a set of propositions Λ. We recursively define the language of epistemic logic over this set of propositions, $\mathcal{L}(\Lambda)$, as follows;

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid K_i\phi \tag{1}$$

where $p \in \Lambda$ and $i \in I$. We can give $\mathcal{L}(\Lambda)$ a semantics using Kripke models as defined below.

A *frame* for a set of agents I is a tuple $\mathcal{M} = (U, R)$, where U is a finite set of possible worlds and R is a set of binary relations over W indexed by an agent: $R_i \subseteq U \times U$ for $i \in I$.

A *Kripke model* over a set of agents I and a set of propositions Λ is a tuple $\mathcal{M} = (U, R, \pi)$, where (U, R) is a frame over I and $\pi : U \to \mathcal{P}(\Lambda)$ is a valuation function; $\pi(w) \subseteq \Lambda)$ for every $w \in U$.

A *pointed Kripke model* is just a pair $(\mathcal{M}, w)$, where w is a world of $\mathcal{M}$.

We can now give a semantics for $\mathcal{L}(\Lambda)$ on pointed Kripke models with the following inductive definition.

- $(\mathcal{M}, w) \models \top$ always.
- $(\mathcal{M}, w) \models p$ iff $p \in \pi(w)$
- $(\mathcal{M}, w) \models \neg\phi$ iff not $(\mathcal{M}, w) \models \phi$
- $(\mathcal{M}, w) \models \phi \wedge \psi$ iff $(\mathcal{M}, w) \models \phi$ and $(\mathcal{M}, w) \models \psi$
- $(\mathcal{M}, w) \models K_i\phi$ iff $(\mathcal{M}, v) \models \phi$ for all v such that $(w, v) \in R_i$.

1.2 Dynamic Epistemic Logic

In order to make our epistemic logic *dynamic*, we need to have some formal way to reason about the dynamics of information, similarly to how Kripke models formalise static information.

We can do this using *action models*. Over a set of propositions Λ, an *action model* is a tuple $\mathcal{E} = \left(E, R^{\mathcal{E}}, \mathsf{pre}, \mathsf{post}\right)$, where E is a finite set of events, $R^{\mathcal{E}}$ is a set of relations $R_i \subseteq (E \times E)$ for each $i \in I$, $\mathsf{pre} : E \to \mathcal{L}(\Lambda)$ is a function assigning each event a formula called a *precondition* and $\mathsf{post} : E \times \Lambda \to \mathcal{L}(\Lambda)$ is a function at each event assigns to each proposition a boolean formula called the *postcondition*.

Given a Kripke model $\mathcal{M}$ and an action model $\mathcal{E}$, we define the *update product* $\mathcal{M} \otimes \mathcal{E} := (U^{\otimes}, R^{\otimes}, \pi^{\otimes})$ to be as follows:

- $W^{\otimes} := \{(w, a) \in U \times A \mid (\mathcal{M}, w) \models \mathsf{pre}\,(a)\}$
- $R_i^{\otimes} := \{((w, a), (v, b)) \mid R_i^{\mathcal{M}}wv$ and $R_i^{\mathcal{A}}ab\}$
- $\pi^{\otimes}((w, a)) := \{p \in \Lambda \mid (\mathcal{M}, w) \models \mathsf{post}\,((a, p))\}$

We define an *action* in a similar way to a pointed Kripke model; they are simply a pair $(\mathcal{A}, a)$ where $a \in A$. To update a pointed Kripke model with an action model, we define $(\mathcal{M}, w) \otimes (\mathcal{E}, a) := (\mathcal{M} \otimes \mathcal{E}, (w, a))$.

We now have everything we need to define the language of Dynamic Epistemic Logic (hereafter DEL). Given a vocabulary Λ, the language $\mathcal{L}_D(\Lambda)$ extends $\mathcal{L}\,(\Lambda)$ and is defined as:

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid K_i\phi \mid [\mathcal{A}, a]\phi \tag{2}$$

where $p \in \Lambda$, $i \in I$ and $(\mathcal{A}, a)$ is an action.

We give identical semantics to $\mathcal{L}_D(\Lambda)$ as for $\mathcal{L}(\Lambda)$, except that we add in an operation for actions as follows.

- $(\mathcal{M}, w) \models [\mathcal{A}, a]\phi$ iff $\mathcal{M}, w \models \mathsf{pre}(a)$ implies $\mathcal{M} \times \mathcal{A}, (w, a) \models \phi$.

2 The Gossip Problem

We now want a use-case for DEL. The *gossip problem* is an old problem about information sharing; we have a group of n gossipers, each of whom know some piece of gossip that the others don't know. They communicate by telephone, and when one calls another, they tell each other everything they each know. Each gossiper may have the phone number of any number of other gossiper; they may only call another gossiper if they have their phone number.

Naturally, this model isn't limited to people chatting on the phone; it could be generalised to distributed copies of a document that can be edited separately but need to be synchronised. However, given that it's simpler to reason about people talking than documents communicating, we'll stick to the slightly more frivolous example of gossiping.

There are a variety of different rules for the gossip problem; that calls must be conference calls, that an agent may choose not to tell the person they're calling all the secrets they know, or that calls can occur in parallel. Here, we will study the simplest set of rules; calls are one-to-one, happen sequentially, and agents tell each other everything they know when they call each other. That is, if a call between a and b is followed by a call from b to c, b will tell c a's secret. We call a gossiper an *expert* if they know the secret of every other gossiper.

The classical gossip problem was studied extensively in the 70s and assumed a total network; that is, one where everyone can calle everyone. It was quickly found that for $n \geqslant 4$ (where n is the number of agents) the minimum number of calls to make every agent an expert was $2n - 4$. This, however, simply stated that such sequences exist; not how to achieve them. These solutions assume some central scheduler controlling the whole network, which finds some optimal sequence of calls and sends this to the agents. Given that the gossip problem is usually used to model distributed agents communicating - e.g. in the example of documents I gave earlier - we would much rather have a distributed algorithm to solve this, in which each agent has a set of rules it uses to decide who to call next. This set of rules is called a *protocol*. An example of a protocol is LNS, standing for Learn New Secrets; in this protocol, an agent may only call another agent if they do not know the other's secret.

2.1 Gossip Graphs & Calls

Given a finite set of agents A, a *gossip graph* $\mathcal{G}$ is a triple (A, N, S). where N and S are binary relations such that $S \subseteq N$. For all agents $a \in A$ we write $N_a = (b \in A \mid N_a b)$ and $S_a = (b \in A \mid S_a b)$.

The relations N and S model the *knowledge* of the agents. We say that a knows the number of b if $N_a b$; likewise, a knows the secret of b if $S_a b$.

A *call* is an ordered pair $(a, b) \in A \times A$; usually, we write ab for (a, b). In a gossip graph $\mathcal{G}$, a call is possible if and only if $N_a b$. Given a graph $\mathcal{G}$ and a possible call ab, $\mathcal{G}^{ab}$ is the graph (A', N', S') such that:

- $A' ::= A$
- $N'_a ::= N'_b ::= N_a \cup N_b$
- $S'_a ::= S'_b ::= S_a \cup S_b$
- $N'_c ::= N_c$, $S'_c ::= S_c$ if $c \neq a, b$

We can see from this definition that we can read $\mathcal{G}^{ab}$ as $\mathcal{G}$ once updated by call ab. We write a sequence of calls as σ, and write $\mathcal{G}^{\sigma}$ to denote $\mathcal{G}$ after the sequence of calls σ have been applied in turn to it. N^{σ} and S^{σ} are defined identically. We also write $\sigma_a \subseteq \sigma$ to denote the set of calls that involve a, and σ_ϵ for the empty call sequence.

Note that we will leave out any pairs (a, a) for $a \in A$ in any expansions of sets N and S.

2.2 The Problem

The classical gossip problem is defined as follows: given an initial graph $\mathcal{G}$ and a goal formula $\phi \in \mathcal{L}$, is there some call sequence σ such that $(\mathcal{G}, \sigma) \models \phi$?

In our case, we want to know if there's some call sequence such that every agent is an expert. This is what we define as a *successful* call sequence; everyone knows all of the gossip, and can go on to talking about more important things.

3 Protocols

In our gossip problem, an agent u follows the following algorithm:

- Repeat forever:
- Pick an agent $u \in A$ such that $\rho(u, v)$ is satisfied

- Perform call uv

Where ρ is a condition for an agent to call another. Some simple examples of protocol conditions are:

- ANY: $\rho(u, v) ::= \top$
  This will always evaluate to true, as any agent is allowed to make any call.
- LNS: $\rho(u, v) ::= \neg S_u v$
  Recall from earlier that under LNS an agent was only allowed to make a call if they did not know the secret of the person they're calling.
- CMO: $\rho(u, v) ::= \neg C_{ab} a \wedge \neg C_{ab} b$
  Agent a may only call agent b if they have not already called each other in the past.

TODO: Maybe move the definition of the language of protocol conditions up to above the basic protocols part?

Note that, given that an agent must know the phone number of another agent in order to call them, we have that $N_u v$ is *implicit* in every protocol $\rho(u, v)$. It's omitted to improve readability of the protocols and make it easier to see the underlying logic.

Given that we only want to know about when each agent will become an expert, we need to halt the algorithm somehow once this is the case. **TODO: Consider a way to just keep the halting condition to the agents & not have it be controlled elsewhere?**

We can define the language of protocol conditions, $\mathcal{L}_\rho$ as follows:

$$\phi ::= \top \mid N_a b \mid S_a b \mid C_{ab} c \mid \neg\phi \mid \phi \wedge \phi \mid K_a \phi \tag{3}$$

Note the new operator $C_{ab}c$. This can be read as *ab is a call in the list of calls involving c*. So clearly if $c \notin a, b$, $C_{ab}c$ is always true, and that formulas $C_{ab}a$ and $C_{ab}b$ are true in $(\mathcal{G}, \sigma)$ if a call ab has already been made; that is, $ab \in \sigma$. We also have that $C_{ab}a \iff C_{ab}b$.

We also define $\bar{K}_a\phi$ to be $\neg K_a \neg\phi$. This can be read as "a considers it possible that ϕ".

We define a gossip state as the pair $(\mathcal{G}, \sigma)$, where σ is the complete history of calls leading up to the gossip state described, and $\mathcal{G}$ is the initial graph.

We define the evaluation of a formula $\phi \in \mathcal{L}_\rho$ on a gossip state $(\mathcal{G}, \sigma)$ where $\mathcal{G}^\sigma = (A, N^\sigma, S^\sigma)$ as follows:

- $(\mathcal{G}, \sigma) \models \top$ always.
- $(\mathcal{G}, \sigma) \models N_a b$ iff $N^\sigma_a b$.
- $(\mathcal{G}, \sigma) \models S_a b$ iff $S^\sigma_a b$.
- $(\mathcal{G}, \sigma) \models C_{ab} c$ iff $ab \in \sigma_c$
- $(\mathcal{G}, \sigma) \models \neg\phi$ iff not $\mathcal{G}, \sigma \models \phi$.
- $(\mathcal{G}, \sigma) \models \phi \wedge \psi$ iff $\mathcal{G}, \sigma \models \phi$ and $\mathcal{G}, \sigma \models \psi$.
- $(\mathcal{G}, \sigma) \models K_a\phi$ iff for all $(\mathcal{G}', \sigma')$ such that $(\mathcal{G}, \sigma) \preccurlyeq_a (\mathcal{G}', \sigma')$, then $(\mathcal{G}', \sigma') \models \phi$.
- $(\mathcal{G}, \sigma) \models \bar{K}_a\phi$ iff there exists some $(\mathcal{G}', \sigma')$ such that $(\mathcal{G}, \sigma) \preccurlyeq_a (\mathcal{G}', \sigma')$ and $(\mathcal{G}', \sigma') \models \phi$

Note: The e.p.r is changed from the Gattinger semantics to be non-commutative (because that just makes more sense)

Notice the new operator $\preccurlyeq_a$! (**TODO: Find a nicer symbol & name**). This is the *epistemic possibility relation*, and intuitively it means that from the left-hand state, it is epistemically possible that the right hand state is true. If we think about this in the context of the above semantics it makes sense; $\mathcal{G}, \sigma \models K_a\phi$ if for all states that agent a considers episetmically possible from $\mathcal{G}, \sigma$, ϕ holds.

To define this operator concretely, we first need to define what it means to for a call to be *ρ-permitted*. This is a thankfully short definition; we say a call ab is ρ-permitted at a gossip state $\mathcal{G}, \sigma$ just if $\rho(a, b)$ is true at state $\mathcal{G}, \sigma$. **I've just realised there's not really a way for the definition of ρ to take into account the things inside a gossip state; this needs to change!**.

Once we have this definition, we can define the epistemic possibility relation $\preccurlyeq_a$ inductively as follows:

- $(\mathcal{G}, \sigma_\epsilon) \preccurlyeq_a (\mathcal{G}, \sigma_\epsilon)$
- if $(\mathcal{G}, \sigma) \preccurlyeq_a (\mathcal{G}, \tau)$, $N_b^\sigma = N_b^\tau$, $S_b^\sigma = S_b^\tau$ and call ab is ρ-permitted at $(\mathcal{G}, \sigma)$, then $(\mathcal{G}, \sigma; ab) \preccurlyeq_a (\mathcal{G}, \tau; ab)$
- if $(\mathcal{G}, \sigma) \preccurlyeq_a (\mathcal{G}, \tau)$ then for all $c, d, e, f \neq a$ where cd and ef are ρ-permitted at $(\mathcal{G}, \sigma)$ and $(\mathcal{G}, \tau)$ respectively, then $(\mathcal{G}, \sigma; cd) \preccurlyeq_a (\mathcal{G}, \tau; ef)$

3.0.1 Modal Protocols

Given that we have this modal knowledge operator $K_a\phi$ in our language of protocol conditions, we can design protocols based on these. We can split these conditions into two different sets, dependent on the kind of modality used.

Possible Learning Protocols

These protocol conditions $\rho(u, v)$ are of the form $\bar{K}_u\phi$. In these conditions, an agent can make a call if she considers it possible that herself or the agent being called can learn a new secret.

- PIG: $\rho(u, v) ::= \bar{K}_u \bigvee_{w \in A} (S_u w \iff \neg S_v w)$
  Possible Information Growth: u can call v if u knows v's number and u considers it possible that there's a secret known by one of the two, but not both.
- TSS: $\rho(u, v) ::= \bar{K}_u \bigvee_{w \in A} (S_v w \wedge \neg S_u w)$
  Tell me Some Secret: u can call v if u knows v's number and u considers it possible to learn a new secret from v.
- HSS: $\rho(u, v) ::= \bar{K}_u \bigvee_{w \in A} (\neg S_v w \wedge S_u w)$
  Hear Some Secret: u can call v if u knows v's number and u considers it possible that they know a secret v does not.
- HMS: $\rho(u, v) ::= \bar{K}_u \neg S_v u$
  Hear My Secret: u can call v if u knows v's number and u considers it possible that v does not know their secret.

Definite Learning Protocols

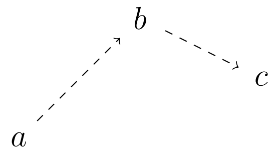
These protocol conditions $\rho(u, v)$ are of the form $K_u\phi$. We'll give *Knowledge Information Growth* as an example. Here, an agent can make a call if she knows that herself or the agent being called will learn a new secret.

We have two subtle variations of this protocol; the *de re* version and the *de dicto* version. (**TODO: remember what the words mean**) The difference is well known in philosophical logic; take for example the statement "Every person has a friend". The *de re* reading of this is that there is a single person that every person is friends with, whereas the *de dicto* reading of this is that every person has a friend (typically a different friend for different people).

- KIGd : $\rho(u, v) ::= K_u \bigvee_{w \in A} (S_u w \iff \neg S_v w)$
  Known Information Growth de dicto: u can call v if u knows v's number and u knows that a secret will be learnt by either u or v, but u does not know whose secret it is that will be learned or which of the pair will learn it.
- KIGr : $\rho(u, v) ::= \bigvee_{w \in A} K_u (S_u w \iff \neg S_v w)$
  Known Information Growth de re: u can call v if u knows v's number and there is an agent w such that u knows that its secret will be learnt by either u or v.

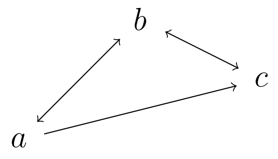
Note that the *de dicto / de re* distinction is ignored for the possible learning protocols, as $\bar{K}_u \bigvee_{w \in A} \phi$ is equal to $\bigvee_{w \in A} \bar{K}_u \phi$.

Naturally, these modal protocols give us a lot more expressive power than the simple propositional protocols like LNS. This is because they let the agent reason about what they consider possible. Take the following example;



We can express this graph as $\mathcal{G} = (\{a, b, c\}, \{(a, b), (b, c)\}, \{\})$, remembering that a gossip graph $\mathcal{G}$ is a triple (A, N, S) where A is the set of agents, N is a set of pairs of agents (x, y) such that x knows y's phone number, and S is a set of pairs of agents (x, y) such that x knows y's secret. As a gossip state, this is just the pair $(\mathcal{G}, \sigma_\epsilon)$.

This graph denotes that a has the phone number of b and b has the phone number of c. After applying the call sequence $bc; ab$, under the protocol LNS, we get the following graph.



This is read as a and b know each other's secrets and numbers, as do b and c. a knows the number and secret of c, but c knows neither the number nor the secret of a. As a gossip state, this is $(\mathcal{G}, ab; bc)$, where $\mathcal{G}$ has been updated by call sequence $ab; bc$. So $\mathcal{G}$ has become $(\{a, b, c\}, \{(a, b), (a, c), (b, a), (b, c), (c, b)\}, \{(a, b), (a, c), (b, a), (b, c), (c, b)\})$.

Under LNS, we are stuck; a will not call c as a knows the secret of c, and c cannot call a as c doesn't know the phone number of a. Explicitly, we can evaluate the semantics of LNS to see why this is the case. Recall that under LNS, $\rho(u, v) ::= N_u v \wedge \neg S_u v$. So we want to find a pair (u, v) such that $(\mathcal{G}, \sigma) \models \rho(u, v)$ and $u \neq v$.

- $(\mathcal{G}, \sigma) \models \rho(a, b) \models N_a b \wedge \neg S_a b \models \neg S_a b \models \neg S^\sigma_a b \models \bot$
- $(\mathcal{G}, \sigma) \not\models \rho(b, a)$ as above.
- $(\mathcal{G}, \sigma) \not\models \rho(a, c)$ as above.
- $(\mathcal{G}, \sigma) \not\models \rho(b, c)$ as above.
- $(\mathcal{G}, \sigma) \not\models \rho(c, b)$ as above.
- $(\mathcal{G}, \sigma) \models \rho(c, a) \models N_a b \wedge \neg S_a b \models N_a b \models \bot$.

Hence we have no call ab such that $(\mathcal{G}, \sigma) \models \rho(a, b)$, so we are stuck!

However under PIG, the call ac is permitted, as a considers it possible that there's a secret known by one of the two but not both. In this case, it is that a considers it possible that a knows a's secret and c doesn't.

Explicitly, first recall that under protocol PIG, $\rho(u, v)$ is defined as $N_u v \wedge \bar{K}_u \bigvee_{w \in A} (S_u w \iff \neg S_v w)$. We can see that $(\mathcal{G}, ab; bc) \models \rho(a, c)$ as follows:

- $(\mathcal{G}, ab; bc) \models N_a c$ as when we inspect $\mathcal{G}^{ab;bc}$ we see that $(a, c) \in N^{ab;bc}$.
- To prove $(\mathcal{G}, ab; bc) \models \bar{K}_a \bigvee_{w \in A} (S_a w \iff \neg S_b w)$, we need to pick some $(\mathcal{G}', \sigma')$ such that $(\mathcal{G}, ab; bc) \preccurlyeq_a (\mathcal{G}', \sigma')$ and $(\mathcal{G}', \sigma') \models \phi$. We don't have to look very far for this, and we can just use $(\mathcal{G}, ab; bc)$.
  Now that we have our $(\mathcal{G}', \sigma')$, we want to show that $(\mathcal{G}, ab; bc) \models \bigvee_{w \in A} (S_u w \iff \neg S_v w)$. To this end, we need to pick $w \in A$ such that $(S_a w \iff \neg S_c w)$. We can pick a for this, remembering that because b called c before b knew a's secret, b didn't tell c a's secret.
  Now all that's left is to show that $(\mathcal{G}, ab; bc) \models (S_a w \iff \neg S_c w)$. If we update $\mathcal{G}$ with call sequence $ab; bc$, we have that $S^{ab;bc} = \{(a, b), (a, c), (b, a), (b, c), (c, b)\}$. Clearly then (remembering that for any $a \in A$, $(a, a) \in S$) $S_a a$ and $\neg S_c a$, thus $(S_a w \iff \neg S_c w)$ for $w = a$ and we're done.
- Given the two above results, we have that $(\mathcal{G}, ab; bc) \models N_u v \wedge \bar{K}_u \bigvee_{w \in A} (S_u w \iff \neg S_v w)$, and so call ac is permitted.

3.1 Protocol Classification

We say that a protocol P is *successful* on an initial graph G if all agents are experts after executing it. We can further categorise them by saying a protocol is *strongly successful* if after all possible executions of P on G, every agent is an expert. We say that a protocol is *weakly successful* if after some possible executions of P on G, every agent is an expert.

In van Ditmarsch et al. 2016, the protocols mentioned earlier are further categorised into classes of graphs which they are strongly or weakly successful upon.

3.2 The Planning Problem

Often we would like to know a sequence of calls that can be made from an initial gossip state such that after all of them have been made everyone is an expert. This is an instance of a wider field of study known as automated planning, where we want to find a finite sequence of actions that takes a given system from its initial state to a goal state.

This has recently been extended **add citations** to epistemic planning, where given initial epistemic states of the agents (e.g. agent i knows ϕ), a set of available events and an epistemic objective (e.g. agent j knows ψ), we want to find a sequence of events which takes us from the initial state to a state where the epistemic objective is satisfied. For a multi-agent system like the gossip problem this has been proven to be undecidable (Bolander and Andersen 2012), but in the case of restricting the events to be propositional (i.e, their pre- and post-conditions are just propositional) we have decidability (Aucher, Maubert, and Pinchinat 2014). The propositional case is known as the *propositional planning problem*.

We can see that our gossip problem lends itself perfectly to the propositional planning problem. Our event models here are just *calls*; we see that in section 2.1 the update that happens as a result of a phone call is just the union of phone number and secret knowledge of the two callees. It's important to notice that in our case this is not an epistemic event, as our pre- and post-conditions are not of the form $K_i \phi$; rather $N_u v$ and $S_u v$. Given that the set of agents is finite, the set of possible values $N_u v$ and $S_u v$ is also finite, so we can just see these as a finite set of normal propositions that just happen to be given names containing elements of the set of agents A.

We can formalise the propositional epistemic planning problem as follows. Given a pointed epistemic model $(\mathcal{M}, w)$, an event model $\mathcal{E}$, a set of events $\mathsf{E} \subseteq \mathcal{E}$ and a goal formula $\phi \in \mathcal{L}(\Lambda)$, decide if there exists a finite sequence of events $e_1, e_2, \ldots, e_n$ in E such that $(\mathcal{M}, w) \otimes (\mathcal{E}, e_1) \otimes \ldots \otimes (\mathcal{E}, e_n)$.

4 Construction of $\mathcal{ME}^*$

In Aucher, Maubert, and Pinchinat 2014, it is proved that the propositional planning problem is decidable through construction of a finite word automaton $\mathcal{P}$ such that $\mathcal{L}(\mathcal{P})$ is exactly the set of all solution plans; that is, call sequences that take us from our initial state to a successful one.

4.1 Background

To make the next section easier, we will recall some basic definitions on finite state automata and transducers.

A deterministic word automaton is a tuple $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ where Σ is an alphabet, Q a finite set of states, $\delta : Q \times \Sigma \to Q$ is a partial transition function and F is a set of accepting states. The language accepted by a word automaton $\mathcal{A}$ is the set of words accepted by $\mathcal{A}$, and is written as $\mathcal{L}(\mathcal{A})$. In the setting of the gossip problem, our alphabet will be the set of possible calls, and the language will be strings of calls.

A finite state transducer is a finite word automaton that reads from two tapes at each transition. It is a tuple $\mathcal{T} = (\Sigma, Q, \Delta, q_0, F)$, where everything is defined the same as for a finite word automaton except for the transition relation $\Delta \subseteq Q \times \Sigma \times \Sigma \times Q$. The relation that a transducer $\mathcal{T}$ recognises is denoted by $\mathcal{L}_{\mathcal{T}}(\mathcal{T}) \subseteq \Sigma^* \times \Sigma^*$. We say that a relation is either *regular* or *rational* if there exists a transducer that recognises it.

A *relational structure* is a tuple $\mathcal{I} = (D, \{\rightsquigarrow_i\}_{i \in A}, V)$ where D is the domain of $\mathcal{I}$, for each $i \in A$, $\rightsquigarrow_i \subseteq D \times D$ is a binary relation and $V : \Lambda \to \mathcal{P}(D)$ is a valuation function.

Then a relational structure $\mathcal{I} = (D, \{\rightsquigarrow_i\}_{i \in A}, V)$ is a *regular structure* over a finite alphabet Σ if its domain $D \subseteq \Sigma^*$ is a regular language over Σ, for each $i \in A$, $\rightsquigarrow_i \subseteq \Sigma^* \times \Sigma^*$ is a regular relation (i.e., it can be accepted by a finite state transducer) and for each $p \in \Lambda, V(p) \subseteq D$ is a regular language. Then given deterministic word automaton $\mathcal{A}_{\mathcal{I}}$ and $\mathcal{A}_p$ for each $p \in \Lambda$, as well as transducers $\mathcal{T}_i$ for $i \in A$, we say that $(\mathcal{A}_{\mathcal{I}}, \{\mathcal{T}_{i \in A}\}, \{\mathcal{A}_p\}_{p \in \Lambda})$ is a *representation* of $\mathcal{I}$ if $\mathcal{L}(\mathcal{A}_{\mathcal{I}}) = D$, for each $i \in A$, $\mathcal{L}_{\mathcal{T}}(\mathcal{T}_i) = \rightsquigarrow_i$ and for each $p \in \Lambda, \mathcal{L}(\mathcal{A}_p) = V(p)$.

4.2 $\mathcal{ME}^*$

We're now ready to define the structure $\mathcal{ME}^*$. This structure enumerates all of the *possible histories* of our gossip graph; that is, it considers all of the possible sequences of events in $\mathcal{E}$ (in our case, calls) arising from an initial state $\mathcal{M}$ (in our case, the initial gossip state).

For an epistemic model $\mathcal{M} = (W, \{R_i\}_{i \in A}, V)$ and an event model $\mathcal{E} = (E, \{R_i^{\mathcal{E}}\}_{i \in A}, \mathsf{pre}, \mathsf{post})$, we define the family of epistemic models $\mathcal{ME}^n{}_{n \geq 0}$ by letting $\mathcal{ME}^0 = \mathcal{M}$ and $\mathcal{ME}^{n+1} = \mathcal{ME}^n \otimes \mathcal{E}$.

Note that we do not pick precisely what the element of $\mathcal{E}$ is here; rather, we read $\mathcal{ME}^n$ as $\mathcal{M}$ with n calls following it; it doesn't particularly matter what the calls are. Indeed, we can see it as a tree rooted in the initial state $\mathcal{M}$ with the branches being $\mathcal{M}$ updated with some call e_i. Then the tree depth would be n.

Then for each n, $\mathcal{ME}^n = (W^n, \{R_i^n\}_{i \in A}, V^n)$, we define the *relational structure* generated by $\mathcal{M}$ and $\mathcal{E}$ as $\mathcal{ME}^* = (D, \{\rightsquigarrow_i\}_{i \in A}, V)$ where:

- $D = \bigcup_{n \geq 0} W^n$,
- $h \rightsquigarrow_i h'$ if there is some n such that $h, h' \in \mathcal{ME}^n$ and $hR_i^n h'$,
- $V(p) = \bigcup_{n \geq 0} V^n(p)$.

4.3 A Regular Structure for $\mathcal{ME}^*$

We now put forward a method to construct a representation of the regular structure $\mathcal{ME}^*$. It is proved in Aucher, Maubert, and Pinchinat 2014 that this represesntation has size $2^{\mathcal{O}(|A|)} . (|\mathcal{M}| + |\mathcal{E}|)^{\mathcal{O}(1)}$

Let $\mathcal{M} = (W, R, V)$ be an epistemic model and let $\mathcal{E} = (E, R^{\mathcal{E}}, \mathsf{pre}, \mathsf{post})$ be a propositional event model, and let $\mathcal{ME}^* = (D, \{\rightsquigarrow_i\}_{i \in A}, V_D)$.

Then define the finite word automaton $\mathcal{A}_D = (\Sigma, Q, \delta, q_0, F)$, where $\Sigma = W \cup E$, $F = \{q_v \mid v \subseteq \Lambda\}$ and $Q = F \cup \{q_0\}$. For a world $w \in W$, we define its valuation as a function $v : W \to \mathcal{P}(\Lambda)$ such that $v(w) = \{p \in \Lambda \mid w \in V(p)\}$. We can now define the partial transition function δ as follows:

$$
\begin{array}{ll}
\forall w \in W, \forall e \in E, & \\
\delta(q_0, w) = q_{v(w)} & \delta(q_0, e) = \bot \\
\delta(q_v, w) = \bot & \delta(q_v, e) = \begin{cases} q_{v'}, \text{where } v' = \{p \mid v \models \mathsf{post}(e,p)\} & \text{if } v \models \mathsf{pre}(e) \\ \bot & \text{otherwise.} \end{cases}
\end{array}
$$

This definition looks very complicated however intuitions quickly simplify it. We recall that this structure $\mathcal{ME}^*$ encodes a sequence of calls $e_n \in E$ from an initial state $w \in W$. So the first transition of our automaton will take a world w from which to start. For this reason, the transition $\delta(q_0, e)$ is undefined; we may also think of it as going to a rejecting state. The same reasoning explains why $\delta(q_v, w)$ is undefined; once we have a starting world w, we don't want to consume any other worlds; we only care about events. With this we can see how the structure $\mathcal{ME}^n$ is built up.

The definition of $\delta(q_v, e)$ states that if the event we are consuming is possible to happen at our current state, then we travel to the state where the set of consequences of our event are true, and are possible given the true formulas of the prior state. **Give a worked example here**.

We can see that $\mathcal{L}(\mathcal{A}_D) = D$, so D is a regular language.

For valuations, take some $p \in \Lambda$. Let $\mathcal{A}_p = (\Sigma, Q, \delta, q_0, F_p)$, where everything is defined as above except for $F_p = \{q_v \mid p \in v\}$. We see that the accepting states of $\mathcal{A}_p$ are the states are which p is made true; hence, $\mathcal{L}(\mathcal{A}_p) = V_d(p)$ and $V_d(p)$ is a regular language.

For our relations, let $i \in A$ and consider the one-state transducer $\mathcal{T}_i = (\Sigma, Q^{\mathcal{T}}, \Delta_i, q_o, F^{\mathcal{T}})$, with $Q^{\mathcal{T}} = \{q\}$, $q_0 = q$, $F^{\mathcal{T}} = \{q\}$ and $\Delta_i = \{(q, w, w', q) \mid wR_i w'\} \cup \{q, e, e', q \mid eR_i^{\mathcal{E}} e'\}$. Remember that valid words accepted by $\mathcal{A}_{\mathcal{D}}$ are of the form $we_0e_1e_2\ldots$ for $w \in W, e_i \in E$. A transducer $\mathcal{T}_i$ will accept words like these, but it will also accept words of the form $w_0w_1w_2w_3\ldots$, $w_0e_0e_1w_1e_2e_3\ldots$ and so on. Hence we see that $\rightsquigarrow_i = \mathcal{L}(\mathcal{T}_i) \cap D \times D$. Given that $\mathcal{L}(\mathcal{T}_i)$ is a regular relation and D is a regular language, we can define a transducer $\mathcal{T}_i' = \mathcal{T}_D \circ \mathcal{T}_i \circ \mathcal{T}_D$, where $\mathcal{T}_D$ is the identity transducer on D.

Then finally, $\mathcal{ME}^*$ is a regular structure that accepts $(\mathcal{A}_D, \{\mathcal{T}_i'\}_{i \in I}, \{\mathcal{A}_p\}_{p \in \Lambda})$.

4.4 Within Context

It's important that we consider how the use of automata relates to our gossip problem. We will use the following initial gossip state as our example:

$$\begin{tikzcd}
 & & b \arrow[dll, dashed] \\
a \arrow[dr, dashed] & & \\
 & c &
\end{tikzcd}$$

This is our initial state; we give this as an input to the automata $\mathcal{A}_{\mathcal{D}}$. This takes us to the state labelled $\{N_b a, N_a c\}$, remembering that our states are labelled by the things that the agents know. Indeed, we will only ever care about propositions of the form $N_u v$ and $S_u v$, which model knowledge of a phone number and knowledge of a secret. Given that our set of agents is finite, this set of propositions is also finite, and hence can be modelled with a finite automata.

Labelling our states with just the propositions that are true at a given point lets us model an infinite set of possible call sequences with a finite automata; it doesn't matter if a and b call each other hundreds of times in a row, we care only about the change in information between the states. This is classically the merit of automata and it translates very well to our case.

We can then model events as transitions between our states. For example, the call ba on the above gossip state would lead us to the following state:

$$\begin{tikzcd}
 & & b \arrow[dll] \arrow[ddl, dashed] \\
a \arrow[urr] \arrow[dr, dashed] & & \\
 & c &
\end{tikzcd}$$

BELOW IS WRONG - WE NEED TO UPDATE TO REFLECT THE ABOVE GRAPH

In the automata $\mathcal{A}_{\mathcal{D}}$, this is the state indexed $\{N_a b, N_b a, N_a c, S_a b, S_b a\}$.

Recall the definition of an event model $\mathcal{E} = (E, R^{\mathcal{E}}, \mathsf{pre}, \mathsf{post})$. In order to use the $\mathcal{ME}^*$ construction for our current example, we need an event model for it. So call our event model $\mathcal{E}$, and let E be the set $\{ab, ac, ba, bc, ca, cb\}$, where uv denotes a call from u to v. Notice that the set E enumerates all possible calls between all agents; the eagle-eyed reader will notice that given the initial gossip state above, only calls ba and ac are possible. This is where the function pre comes into play; it tells us what must hold of a state before an event may occur. For our event model $\mathcal{E}$, the function pre is defined extremely simply:

$$\begin{array}{ccc}
\mathsf{pre}(ab) = N_a b & \mathsf{pre}(ac) = N_a c & \mathsf{pre}(ba) = N_b a \\
\mathsf{pre}(bc) = N_b c & \mathsf{pre}(ca) = N_c a & \mathsf{pre}(cb) = N_c b
\end{array}$$

We see that for all $u, v \in I$, $\mathsf{pre}(uv) = N_u v$. This makes perfect sense intuitively, as the only precondition for an agent to be able to call another is that the caller knows the phone number of the agent.

Note that we may encode the protocol into the pre function. The above example is for the ANY protocol, however we may very easily adapt the definition to a protocol such as LNS with ease, by setting pre such that for all $u, v \in I$, $\mathsf{pre}(u, v) = N_u v \wedge \neg S_u v$. However, for the time being, let us just work under the protocol ANY.

Recall from earlier that the planning problem with the epistemic operator $K_i \phi$ in our pre- and post-conditions is undecidable. This means we cannot have a pre-condition of the form

$\mathsf{pre}(u, v) = K_u S_u v \wedge N_u v$, which unfortunately means we cannot simply encode epistemic protocols like those in Section 3.0.1. **Investigate whether there's a way to encode these in some other way?**.

We also need to define the function $\mathsf{post} : E \to \Lambda \to \mathcal{L}(\Lambda)$. The definition of post is a bit more nuanced than pre ; we can even see from its type that it's a bit more tricky. The intuition for post is that we give it two arguments - an event and a formula - and it tells us *what needs to have held* at the previous state in order for the given formula to be true. In the definition of the transition function δ in Section 4.3; we travel to a state indexed by the set of propositions $p_1, \ldots, p_n$ that hold as a result of the event e occuring at a world that satisfies $\mathsf{post}(e, p_i)$ for some i.

We can give a few examples in the context of our initial gossip state above.

- $\mathsf{post}(ba, S_a b) = S_a b \vee S_b b$.
  We can read this as saying that for $S_b a$ to be true after the call ab has been executed, then we need for either $S_a b$ or $S_b b$ to have been true before the call.
- $\mathsf{post}(ba, S_a c) = S_a c \vee S_b c$.
  In this case we need that agent a already knew the secret of c, or that b did before the call. In the above example, neither $S_a c$ nor $S_b c$ are true, hence after the call ba has been made $S_a c$ will not be true.

The transition $\delta(q_{N_b a, N_a c}, ba)$ then takes us to the state labelled with $\{N_a b, N_b a, N_a c, S_a b, S_b a\}$. This is firstly because $\mathsf{pre}(ba) = N_b a$, and $v = \{N_b a, N_a c\} \models N_b a$, and as above $\{N_b a, N_a c\} \models \mathsf{post}(ba, S_a b) = S_a b \vee S_b b$, and similarly $\{N_b a, N_a c\} \models \mathsf{post}(ba, S_b a)$. Remember that we do not include propositions of the form $S_u u$, as they are trivially true.

You may wonder what happens to calls we cannot make; for instance, call cb from our initial state. If you inspect the definition for δ in Section 4.3 we see that if $v \not\models \mathsf{pre}(e)$ then the transition is undefined; that is, there is no transition, it is rejected. This means that we can build up a tree of knowledge states, with calls permitted *at the origin state* as transitions between them. We can see an example of this for the gossip state $\{N_b a, N_a c\}$ below.

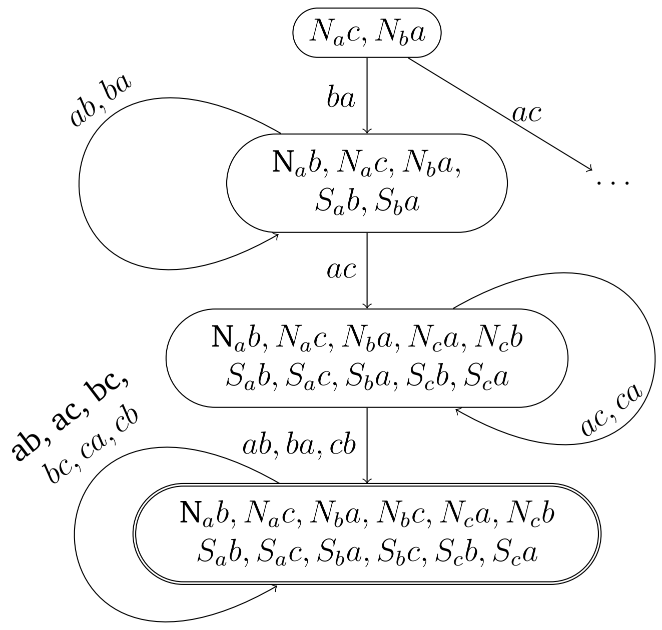
$$\begin{tikzcd} & \boxed{N_a c, N_b a} \arrow[dl, "ac"'] \arrow[dr, "ba"] & \\ \boxed{\mathrm{N}_a c, N_b a, N_c a, S_a c, S_c a} & & \boxed{\mathrm{N}_a b, N_a c, N_b a, S_a b, S_b a} \end{tikzcd}$$

TODO: find out why on earth that N is not normal font

This puts into effect what we have been discussing. From the state $\{N_b a, N_a c\}$, we have exactly two transitions; the call ba and the call ac. This is because they are the only events e in our event model $\mathcal{E}$ where $\{N_b a, N_a c\} \models \mathsf{pre}(e)$. For all other calls the transition is undefined; meaning we may not make the transition.

As an example for post, we can see that for all propositions $p \in \{N_a c, N_b a, N_c a, S_a c, S_c a\}$, $\{N_b a, N_a c\} \models \mathsf{post}(ca, p)$.

Let's continue constructing the $\mathcal{ME}^*$ automata. Continuing from the state $\{N_a b, N_a c, N_b a, S_a b, S_b a\}$, we have the following automata:


N_ac, N_ba
ba
ac
ab, ba
$\mathrm{N}_ab, N_ac, N_ba,$
S_ab, S_ba
$\dots$
ac
$\mathrm{N}_ab, N_ac, N_ba, N_ca, N_cb$
$S_ab, S_ac, S_ba, S_cb, S_ca$
ac, ca
ab, ac, bc,
bc, ca, cb
ab, ba, cb
$\mathrm{N}_ab, N_ac, N_ba, N_bc, N_ca, N_cb$
$S_ab, S_ac, S_ba, S_bc, S_cb, S_ca$

Our goal in the gossip problem is to get to a state where all of the agents know all of the secrets of the other agents. Formally, this is a state in the $\mathcal{ME}^*$ automata, with the set P as the propositions that hold at the state, where $\{S_ij \mid i \in I, j \in J, i \neq j\} \subseteq P$. Clearly such a state exists in the automata above and is labelled as the accepting state. So we have a variety successful call sequences, such as $\{ba, ac, ba\}$, $\{ba, ac, ca, ba\}$, $\{ba, ba, ba, ba, ac, ba\}$ and so on. We have come across another merit of automata for this task; the ability to represent an infinite set in a finite manner.

TODO: Show an example using a different protocol (i.e. LNS)

5 The Powerset Construction

So far we have only considered winning conditions P with $\{S_ij \mid i \in I, j \in J, i \neq j\} \subseteq P$. This is perfectly useful, however we may want a stronger case - for example, every agent knows that every agent is an expert. It's easy enough to think of a motivating case where this might happen. Consider that the various agents are sensors that need to pool their individual observations, in order to obtain a composite observation of a certain scene. Once all of the sensors have collected all of the secrets of the other sensors they may want to perform some kind of computation and then pool this information again once they know all the other sensors have finished their respective computation. This second pooling of information could only occur once all of the agents know that all of the other agents have finished their respective computation, which would only happen when an agent is an expert.

This is not expressible in our propositional fragment of $\mathcal{L}(\Lambda)$ - we need to use our knowledge modality K_i. Using E_i to abbreviate $\{S_ij \mid j \in J, i \neq j\}$, this winning condition looks like $\forall i \in I, K_i \{E_j \mid j \in J\}$.

Verifying that this is true is significantly more difficult than a strictly propositional winning condition **...why?**. Our main intuition is that for a propositional winning condition we can simply look at the propositions that hold at a given state and check if these satisfy the condition

$\forall i \in I, E_i$. This is what we call *positional evaluation*. However, for a modal winning condition we need to also take into account the previous calls. To understand this, again consider our initial gossip state from Section 4.4. After call ab occurs, then agent a knows that b knows their secret and vice versa. After call ac, a knows that c knows their secret and the secret of b and vice versa. Finally after call cb again,

To the end of simplifying this process, Maubert and Pinchinat 2014 puts forward a conversion process from a game arena to the corresponding powerset game arena of the input. A game arena is a structure similar to the finite word automata that we use in this thesis, except for being designed for use by two players. The conversion from the construction put forward in Maubert and Pinchinat 2014 is surprisingly non-trivial and we put forward an equivalent construction for finite word automata here.

So for an automata $\mathcal{ME}^* = (\Sigma, Q, \delta, q_0, F)$ and a transducer $\mathcal{T}_i = (\Sigma, Q_\mathcal{T}, \Delta, q_0, F')$ that recognises $\rightsquigarrow_i$, we construct a powerset automata $\widehat{\mathcal{ME}^*}$ where formulas of the form $K_i\phi$ can be evaluated positionally when ϕ is in the propositional fragment of $\mathcal{L}(\Lambda)$.

To create our powerset automata, we need to simulate stepping through the automata and the transducer simultaneously. This way we can keep track of the states related to our current one, and this ultimately lets us positionally evaluate modal formulas.

To perform this, we extend our states to be of the form $(v, S, Last) : Q \times \mathcal{P}(Q_\mathcal{T}) \times (Q_\mathcal{T}, \mathcal{P}(Q))$. $S \subseteq Q_\mathcal{T}$ is the set of possible current states of the transducer $\mathcal{T}$, and $Last : S \rightarrow \mathcal{P}(Q)$ associates to a state $q \in S$ the set of possible last positions on the output tape of T if the current state is q. Our transitions in this automata follow those in $\mathcal{ME}^*$, but hold with them more details - that is, S and $Last$. These two things encode the information about the state of the transducer.

Naturally, we need to define another initial state for our automata $\widehat{\mathcal{ME}^*}$. This is state $\widehat{q_0} = \left(\widehat{v_0}, \widehat{S_0}, \widehat{Last_0}\right)$. We need to simulate the execution of $\mathcal{T}$ starting from its initial state and reading $\widehat{v_0}$. So we introduce an artificial position $\widehat{v}_{-1}$ that initializes the transducer before reading the first position of a call sequence **go over this**. Then by definition $\widehat{v_1} = (v_{-1}, S_{-1}, Last_{-1})$, with $v_{-1} \notin Q$, $S_{-1} = \{q_0\}$ as before starting the transducer is in its initial state, and $Last_{-1}(q_0) = \emptyset$ as nothing is written on the output tape.

This said, given an automata $\mathcal{ME}^* = (\Sigma, Q, \delta, q_0, F)$ and a transducer $\mathcal{T}_i = (\Sigma, Q_\mathcal{T}, \Delta, q_0, F')$, we can now define the powerset automata $\widehat{\mathcal{ME}^*} = \left(\widehat{\Sigma}, \widehat{Q}, \widehat{\delta}, \widehat{q_0}, \widehat{F}\right)$. We slightly stretch our notation and say that $\widehat{\delta} : \widehat{Q} \times \widehat{\Sigma} \rightarrow \widehat{Q}$.

- $\widehat{Q} = Q \times \mathcal{P}(Q_\mathcal{T}) \times (Q_\mathcal{T}, \mathcal{P}(Q))$
- $\widehat{\Sigma} = \Sigma$ **check this one is correct**
- $\widehat{\delta}\left((u, S, Last), \sigma\right) = (v, S', Last')$ if
  - $u = v_{-1}$ and $v = v_0$, or $\delta(u, \sigma) = v$
  - $S' = \{q' \mid \exists q \in S, \exists \lambda' \in \Sigma^*, (\lambda', q') \in \Delta(q, \sigma)\}$
  - $Last'(q') = \{v' \mid \exists q \in S, \exists \lambda' \in \Sigma^*, (\lambda' \cdot v', q') \in \Delta(q, v)$, or $(\epsilon, q') \in \Delta(q, v)$ and $v' \in Last(q)\}$
- $\widehat{q_0}$ is the only $\widehat{q} \in \widehat{Q}$ s.t. $\widehat{\delta}(\widehat{q_{-1}}, —) = \widehat{q}$ **Need to figure out what goes in —**
  Could it be that — is some initial world? Our transducer accepts things of the shape $we_1e_2e_3$, and so we need to give it an initial world to travel to to then start taking events from.
- $\widehat{F} = \{\widehat{q} = (v, S, Last) \mid v \in F\}$

We remember that the set accepted by the transducer is the set of indistinguishable plays. We can read the definition of S as saying that there exists some set of events λ' that are indistinguishable from the event that just occured, hence taking us to a state indistinguishable from our previous one. Given that there may be many possible sets of events λ', we have a set of such states. $Last'$ tells us that if some position v' is at the end of the FST tape after the transducer reads v and reached q', it is either because whilst reading v the last letter it wrote is v', or it wrote nothing and v' was already at the eend of the output tape before we read v.

With these two definitions we can see how $\widehat{\delta}$ simulates us stepping through the automata and the FST simultaneously; we can see from the first sub-point that the transitions of $\widehat{\delta}$ necessarily follow those of δ, except we add in the information from the FST in S and $Last$.

6 Graphs for poster

$$\begin{tikzcd}
 & & b \arrow[dll, dashed] \\
a \arrow[dr, dashed] & & \\
 & c &
\end{tikzcd}$$

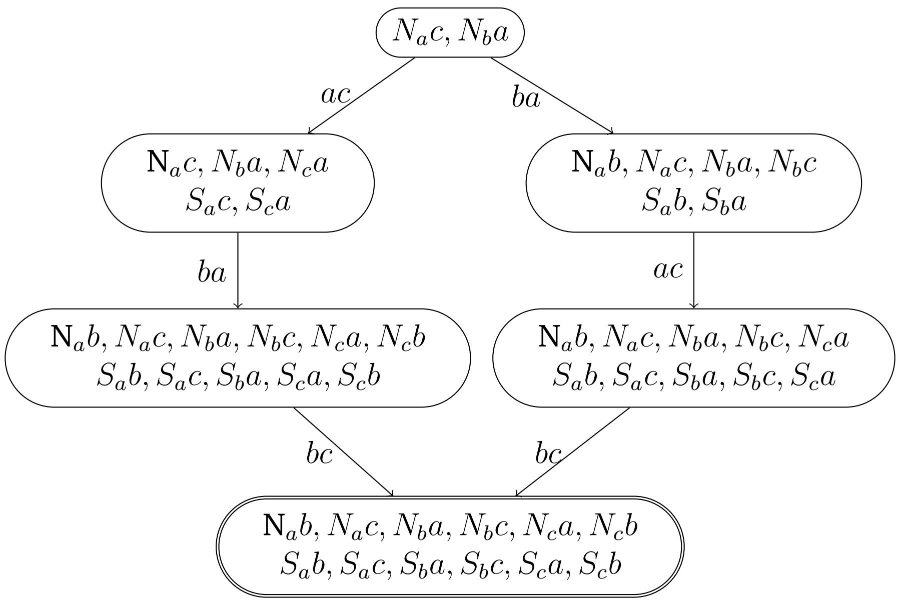
$$\begin{tikzcd}
 & & b \arrow[ddl, dashed] \\
a \arrow[urr, leftrightarrow] \arrow[dr, dashed] & & \\
 & c &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & b \arrow[dll, dashed] \\
a \arrow[dr, leftrightarrow] & & \\
 & c &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & b \arrow[ddl] \\
a \arrow[urr, leftrightarrow] \arrow[dr, leftrightarrow] & & \\
 & c &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & b \arrow[ddl, dashed, bend left] \\
a \arrow[urr, leftrightarrow] \arrow[dr, leftrightarrow] & & \\
 & c \arrow[uur, bend left] &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & b \arrow[ddl, leftrightarrow] \\
a \arrow[urr, leftrightarrow] \arrow[dr, leftrightarrow] & & \\
 & c &
\end{tikzcd}$$


$N_a c, N_b a$
ac
ba
$\mathrm{N}_a c, N_b a, N_c a$
$S_a c, S_c a$
$\mathrm{N}_a b, N_a c, N_b a, N_b c$
$S_a b, S_b a$
ba
ac
$\mathrm{N}_a b, N_a c, N_b a, N_b c, N_c a, N_c b$
$S_a b, S_a c, S_b a, S_c a, S_c b$
$\mathrm{N}_a b, N_a c, N_b a, N_b c, N_c a$
$S_a b, S_a c, S_b a, S_b c, S_c a$
bc
bc
$\mathrm{N}_a b, N_a c, N_b a, N_b c, N_c a, N_c b$
$S_a b, S_a c, S_b a, S_b c, S_c a, S_c b$

Bibliography

Aucher, Guillame, Bastien Maubert, and Sophie Pinchinat (2014). “Automata Techniques for Epistemic Protocol Synthesis”. In: *2nd Workshop on Strategic Reasoning*.

Bolander, Thomas and Mikkel Birkegaard Andersen (2012). “Epistemic planning for single- and multi-agent systems”. In: *Journal of Applied Non-Classical Logics*.

Maubert, Bastien and Sophie Pinchinat (2014). “A General Notion of Uniform Strategies”. In: *International Game Theory Review*.

van Ditmarsch, Hans et al. (2016). “Epistemic Protocols for Dynamic Gossip”. In: *Journal of Applied Logic*.