Model Checking on Distributed Temporal Logic

Fernando Subtil

October 22, 2019

DTL (distributed temporal logic) is a logic that allows us to study distributed systems from both a global and local perspective when it comes to propositional, temporal and synchronization properties. In this thesis we developed a model checking algorithm based on distributed non-deterministic Büchi automata that returns *true* when a DTL formula is satisfied by a distributed transition system, or returns a path that falsifies such formula. We proved its correction and tested it in a few examples, once again verifying it is correct.

1 Distributed Temporal Logic

1.1 Syntax

Definition 1.1.
We define $\Sigma = \langle Id, \{Prop_i\}_{i \in ID} \rangle$ as a distributed signature, where Id is a finite non-empty set of agent identifiers, and, for each of those agents, $Prop_i$ is its respective set of propositional symbols. We assume that $Prop_i \bigcap Prop_j = \emptyset$ when $i \neq j$.

Having chosen a distributed signature Σ, we define the *local language* $\mathcal{L}_i$ for agent i as:

$$\mathcal{L}_i ::= Prop_i | \neg\mathcal{L}_i | \mathcal{L}_i \Rightarrow \mathcal{L}_i | \mathsf{X}\mathcal{L}_i | \mathsf{G}\mathcal{L}_i | ©_j[\mathcal{L}_j]$$

where $i \neq j \in Id$. The operators $\neg$ and $\Rightarrow$ have the usual meaning of *not* and *implication*, X refers to the *next* timepoint, G means *always from this point*, and finally $©_j[\varphi]$ is used to describe *communication* (or synchronization) between agents, in this case, agent i communicated (or synchronized) with agent j, for which formula $\varphi \in \mathcal{L}_j$ held.

DTL does not encompass only local languages for each agent, but also defines a *global language* $\mathcal{L}$:

$$\mathcal{L} ::= @_i[\mathcal{L}_i] | \neg\mathcal{L} | \mathcal{L} \Rightarrow \mathcal{L} | \mathsf{G}\mathcal{L}$$

The @ operator is used to move on from the global formula into local formulas, that is, when we have $@_i[\varphi]$ it means that at that point φ holds in agent i. From now on, we will use φ and ψ when referring to local formulas and α and β for global formulas.

Next we move on to some auxiliary concepts. $Lit_i = Prop_i \bigcup \{\neg p : p \in Prop_i\}$ is the set of *i-literals*. $\mathcal{V}_i$ is the set of all *i-valuations*, which are sets of i-literals that contain either p or $\neg p$ (never both) for all $p \in Prop_i$.

Let $\varphi_1, \varphi_2 \in \mathcal{L}_i$. Then:

- φ_1 is an i-subformula of φ_1;
- φ_1 is an i-subformula of $\neg\varphi_1$;
- φ_1 and φ_2 are i-subformulas of $\varphi_1 \Rightarrow \varphi_2$;
- φ_1 is an i-subformula of $\mathsf{X}\varphi_1$;
- φ_1 is an i-subformula of $\mathsf{G}\varphi_1$;
- The i-subformula relation is transitive.

From agent j's standpoint, in the formula $©_i[\varphi] \in \mathcal{L}_j$ φ is no longer in j's "domain", and as such φ is not a j-subformula of $©_i[\varphi]$. However, lets say that $©_j[\varphi']$ is an i-subformula of φ. In this case φ' is a j-subformula of $©_i[\varphi]$. Intuitively speaking we can see the i-subformulas of a formula φ as the set of all "parts" of φ that are in $\mathcal{L}_i$. The set of all *i-subformulas* of φ is called $subf_i(\varphi)$.

We also define subformulas at a global level. Let $\alpha_1, \alpha_2 \in \mathcal{L}, \varphi \in \mathcal{L}_i$:

- α_1 is a subformula of α_1;
- α_1 is a subformula of $\neg\alpha_1$;
- α_1 and α_2 are subformulas of $\alpha_1 \Rightarrow \alpha_2$;
- α_1 is a subformula of $\mathsf{G}\alpha_1$;
- φ is a subformula of $@_i[\varphi]$.

Essentially the subformulas of a DTL formula α are all the "parts" that compose α, regardless of being at a global or local level. All i-subformulas of α are also subformulas of α. The set of all subformulas of α is called $subf(\alpha)$ which contains $subf_i(\alpha)$ for all agents $i \in Id$

We define the *i-closure* of $\varphi \in \mathcal{L}_i$ as $clo_i(\varphi) = subf_i(\varphi) \bigcup \{\neg\psi : \psi \in subf_i(\varphi)\}$, where $\neg\neg\psi \equiv \psi$. Analogously $clo(\alpha) = subf(\alpha) \bigcup \{\neg\beta : \beta \in subf(\alpha)\}$. Let $B \subseteq clo(\alpha)$. Then $B \downarrow_i$ is the subset of B from which are removed all local formulas of agents other than i.

1.2 Semantics

Now that we have defined DTL, it is time to move on to defining suitable interpretation structures to which DTL can be applied.

Definition 1.2.
Let $i \in Id$. A local life-cycle of i is a discrete, countable, infinite and well-founded total order $\lambda_i = \langle Ev_i, \leq_i \rangle$. Ev_i is a set of local events and $\leq_i$ is the local order of causality.

We can also define a successor relation between events, where we say that $e \rightarrow_i e'$ when $e <_i e'$ and there is no e'' such that $e <_i e'' <_i e'$.

To the family $\lambda = \{\lambda_i\}_{i \in Id}$ of local life-cycles we call it *distributed life-cycle*, and we can define a partial order of global causality $\leq$ for the set of events $Ev = \bigcup\limits_{i \in Id} Ev_i$ by making $e < e'$, where $e \in Ev_i$ and $e' \in Ev_j$, if there is $e'' \in Ev_i \bigcap Ev_j$ such that $e <_i e''$ and $e'' \leq_j e'$.

In a distributed life-cycle events can be shared by several agents as a means to represent communication/synchronization between them. The reason $\leq$ is only a partial order is to accommodate for the fact that the local order needs to be respected by the global order, while allowing some events to be unordered between themselves.

A *local state* of an agent i is $\xi_i \subseteq Ev_i$ which contains all the local events of i that have happened until that point. As such, inclusion defines a total order on the set Ξ_i of local states, having $\emptyset$ as its minimal element, and letting the kth state (ξ_i^k) be $\xi_i^{k-1} \bigcup last(\xi_i^k)$, where $last(\xi_i^k)$ is the last event that happened, that caused the transition from the last state to the next. Also, if $e \in Ev_i$, we say that $e \downarrow_i = \{e' \in Ev_i : e' \leq_i e\}$, which is going to be the local state of i after the occurrence of e.

Likewise, a *global state* ξ is the set of all the events that occurred so far. Notice that in this case there can be different sequences of events, since there can be an event $e_i \in Ev_i$ and $e_j \in Ev_j$ that can happen in different order and as such, the set of all global states Ξ is defined as a lattice. Furthermore we will use $\xi|_i$ to denote the subset of ξ that corresponds to the respective local state of agent i, that is, $\xi|_i = \xi \bigcap Ev_i$.

Definition 1.3. *Let λ be a distributed life-cycle and ϑ a family of labeling functions $\vartheta_i : \Xi_i \to \mathcal{P}(Prop_i)$ that assigns a set of propositional symbols to each local-state of agent i. Then $\mu = \langle \lambda, \vartheta \rangle$ is said to be an interpretation structure. We use μ_i to denote $\langle \lambda_i, \vartheta_i \rangle$.*

Bear in mind that $\mathcal{P}(Prop_i)$ and $\mathcal{V}_i$ are essentially the same sets, and from now on when talking about valuations both definitions can be intertwined. For example, when we have a valuation $\vartheta = \{p_1, \neg p_2, p_3\} \in \mathcal{V}_i$, we can use it in the context of interpretation structures as the set $\{p_1, p_3\} \in \mathcal{P}(Prop_i)$. Also, for simplicity, all interpretation structures are assumed to be built around $Ev = \{e_1, e_2, e_3, e_4 ...\}$ where $e_k \not< e_{k'}$ for $k' \geq k$, since structurally speaking the naming of the events is irrelevant and as such we decide to always name the events based on the order they occur.

Finally, we can now use interpretation structures and states to define *satisfaction relations.* Let's start with local satisfaction:

- $\mu_i, \xi_i \Vdash_i p$ if $p \in \vartheta_i(\xi_i)$, for $p \in Prop_i$;
- $\mu_i, \xi_i \Vdash_i \neg\varphi$ if $\mu_i, \xi_i \nVdash_i \varphi$;
- $\mu_i, \xi_i \Vdash_i \varphi_1 \Rightarrow \varphi_2$ if $\mu_i, \xi_i \Vdash_i \varphi_2$ or $\mu_i, \xi_i \nVdash_i \varphi_1$;
- $\mu_i, \xi_i \Vdash_i \mathsf{X}\varphi$ if $\exists_{e \in Ev_i \setminus \xi_i}$ such that $\xi_i \bigcup \{e\} \in \Xi_i$ and $\mu_i, \xi_i \bigcup \{e\} \Vdash_i \varphi$;
- $\mu_i, \xi_i \Vdash_i \mathsf{G}\varphi$ if $\forall_{\xi_i' \in \Xi_i}$ such that $\xi_i \subseteq \xi_i'$, $\mu_i, \xi_i' \Vdash_i \varphi$;
- $\mu_i, \xi_i \Vdash_i ©_j[\varphi]$ if $last(\xi_i) \in Ev_j$, $\xi_i \neq \emptyset$, and $\mu_j, last(\xi_i) \downarrow_j \Vdash_j \varphi$.

All that is left to do is define the global satisfaction:

- $\mu, \xi \Vdash \neg\alpha$ if $\mu, \xi \nVdash \varphi$;
- $\mu, \xi \Vdash \alpha_1 \Rightarrow \alpha_2$ if $\mu, \xi \Vdash \alpha_2$ or $\mu, \xi \nVdash \alpha_1$;
- $\mu, \xi \Vdash \mathsf{G}\alpha$ if $\forall_{\xi' \in \Xi}$ such that $\xi \subseteq \xi'$, $\mu, \xi' \Vdash \alpha$
- $\mu, \xi \Vdash @_i[\varphi]$ if $\mu_i, \xi|_i \Vdash_i \varphi$.

An interpretation structure μ is said to be a model of α if $\mu, \emptyset \Vdash \alpha$, also written as $\mu \Vdash \alpha$, and α is satisfiable whenever $Mod(\alpha) = \{\mu : \mu \Vdash \alpha\} \neq \emptyset$.

Definition 1.4.
Let $\mu = \langle \lambda, \vartheta \rangle$ be an interpretation structure. Then $\mu' = \langle \lambda, \vartheta, < \rangle$ such that $<$ is a total order over Ev is a linear interpretation structure induced by μ.

Essentially, all interpretation structures can be linearized by specific sequence of events, resulting in a linear set of global states instead of a lattice, while all the sets of local states Ξ_i remain unchanged. We define $Mod_l(\alpha)$ as the set of all linear models of α. When dealing with DTL it is useful to linearize interpretation structures ([BCRV10, CGRV]), since linear interpretation structures contain the same information as non-linear ones:

Proposition 1.5.
Let $\mu_l = \{\mu^1, \mu^2, \mu^3, ...\}$ be all the possible linear interpretation structures induced by an interpretation structure μ. Then, $\mu_l \subseteq Mod_l(\alpha)$ iff $\mu \in Mod(\alpha)$.

Just like with non-linear interpretation structures, we will always consider the events of a linear interpretation structure to be $e_1, e_2, ...$, and we go even further as to stating that $e_k < e_{k+1}$ for every k. This can always be assumed due to the fact that the names of the events can be swapped without losing its meaning.

Lemma 1.6.
$\mu_i, \xi_i \Vdash_i G\varphi$ iff $\mu_i, \xi_i \Vdash_i \varphi$ and $\mu_i, \xi_i' \Vdash_i G\varphi$, where $last(\xi_i) \rightarrow_i last(\xi_i')$.

2 Distributed Transition Systems

The usual approach when working with distributed systems consists on taking a distributed transition system and transforming it into a linear transition system. The main goal with this project is to be able to preserve the original structure of the distributed system, and as such we will not apply that method and will always work with distributed transition systems:

Definition 2.1.
A distributed transition system induced by a family of transition systems $\mathcal{T} = \{T_i\}$, where the sets of actions A_i may intersect and P_i are disjoint, is a tuple $T^{\mathcal{T}} = \langle S, A, \rightarrow, I, P, L \rangle$:

- $S = \bigotimes S_i$;
- $A = \{A_i\}$;
- $\rightarrow : S \times A \rightarrow S$ *is such that* $\langle s\downarrow_1, ..., s\downarrow_n \rangle \xrightarrow{a} \langle s'\downarrow_1, ..., s'\downarrow_n \rangle$ *when:*
  1. *If* $a \notin A_i$ *then* $s\downarrow_i = s'\downarrow_i$;
  2. *If* $a \in A_i$ *then* $s\downarrow_i \xrightarrow{a}_i s'\downarrow_i$;
- $I = \bigotimes I_i$;
- $P = \{P_i\}$;
- $L : S \rightarrow \bigotimes \mathcal{P}(P_i)$ *such that* $L(\langle s\downarrow_1, ..., s\downarrow_n \rangle) = \langle L_1(s\downarrow_1), ..., L_n(s\downarrow_n) \rangle$;

In this definition we use $s\downarrow_i$ is denote the i-th projection of the tuple s.

Let T be a distributed transition system. While the path of a non-distributed transition system consisted on the sequence of traversed states, on a DTS we also want to keep information regarding synchronization, and in order to do so we say that a path of T is a sequence $\pi = (s_0, a_0)(s_1, a_1)...$ such that $s_0 \in I$ and $s_k \xrightarrow{a_k} s_{k+1}$. This way even if two sequences of states are exactly the same, they are considered different paths if the actions that were used are different.

The trace of a path $\pi = (s_0, a_0)(s_1, a_1)...$ is the sequence of propositional symbols present in each one of the agents that synchronized for each action, i.e. $trace(\pi) = \{L_i(s_0\downarrow_i) : i \in Id\}\{L_i(s_1\downarrow_i) : a_0 \in A_i\}\{L_i(s_2\downarrow_i) : a_1 \in A_i\}....$ Given $\sigma = \sigma_0\sigma_1... \in Traces(T)$ we define $\sigma_k\downarrow_i$ as $L_i(s_k\downarrow_i)$ when $a_{k-1} \in A_i$, and undefined otherwise. Similarly $\sigma\downarrow_i$ is the i-subtrace of σ, that is, the sequence $\sigma_0\downarrow_i \sigma_1\downarrow_i ...$, where we ignore all the gaps caused by the undefined $\sigma_k\downarrow_i$. This subtrace will be a trace for the nondistributed transition system T_i. We say that $k \to_i k'$ if $a_k, a_{k'} \in A_i$ and no $a_{k''}$ in between is in A_i.

Just like interpretation structures, DTS traces can also satisfy DTL formulas. Let $\sigma = \sigma_0\sigma_1... \in Traces(T)$. We define local satisfaction as follows:

- $\sigma_k\downarrow_i \Vdash_i p$ if $p \in \sigma_k\downarrow_i$, for $p \in P_i$;
- $\sigma_k\downarrow_i \Vdash_i \neg\varphi$ if $\sigma_k\downarrow_i \nVdash_i \varphi$;
- $\sigma_k\downarrow_i \Vdash_i \varphi_1 \Rightarrow \varphi_2$ if $\sigma_k\downarrow_i \Vdash_i \varphi_2$ or $\sigma_k\downarrow_i \nVdash_i \varphi_1$;
- $\sigma_k\downarrow_i \Vdash_i \mathsf{X}\varphi$ if $\sigma_{k'}\downarrow_i \Vdash_i \varphi$, for $k \to_i k'$;
- $\sigma_k\downarrow_i \Vdash_i \mathsf{G}\varphi$ if $\forall_{k'\geq k}$ such that $a_{k'-1} \in A_i$ $\sigma_{k'}\downarrow_i \Vdash_i \varphi$ (for $k = 0$ we need $\sigma_0\downarrow_i \Vdash_i \varphi$ as well);
- $\sigma_k\downarrow_i \Vdash_i \copyright_j[\varphi]$ if $k > 0$, $a_{k-1} \in A_j$ and $\sigma_k\downarrow_j \Vdash_i \varphi$.

Likewise, we define the global satisfaction:

- $\sigma_k \Vdash \neg\alpha$ if $\sigma_k \nVdash \alpha$;
- $\sigma_k \Vdash \alpha_1 \Rightarrow \alpha_2$ if $\sigma_k \Vdash \alpha_2$ or $\sigma_k \nVdash \alpha_1$;
- $\sigma_k \Vdash \mathsf{G}\alpha$ if $\sigma_{k'} \Vdash \alpha, \forall_{k'\geq k}$;
- $\sigma_k \Vdash @_i[\varphi]$ if $\sigma_{k'}\downarrow_i \Vdash_i \varphi$, where $k' \leq k$ is the last index for which $\sigma_{k'}\downarrow_i$ is defined.

We say that σ satisfies a formula α ($\sigma \Vdash \alpha$) when $\sigma_0 \Vdash \alpha$, and T satisfies α ($T \vDash \alpha$) when $\sigma \Vdash \alpha$ for all $\sigma \in Traces(T)$. To the set of all traces that satisfy α $Pr_\alpha = \{\sigma \in (2^{\bigcup \mathcal{P}(P_i)})^\omega : \sigma \Vdash \alpha\}$ we give the name of property α. So this means that $T \vDash \alpha$ iff $Traces(T) \subseteq Pr_\alpha$.

Considering the structure of a trace, it is always possible to transform it into a DTL interpretation structure, so that we can associate DTS satisfaction rules with the interpretation structure ones:

Definition 2.2. *Consider a path $\pi = (s_0, a_0)(s_1, a_1)...$ of a DTS T and its respective trace σ. Then the linear interpretation structure induced by σ, denoted as μ^σ, is defined as:*

- $Ev_i^\sigma = \{e_k : a_{k-1} \in A_i\}$ *where* $e_k <_i e_{k'}$ *if* $k < k'$;
- $e_k < e_{k'}$ *for* $k < k'$;
- *Let* $\xi_i \in \Xi_i$ *such that* $last(\xi_i) = e_k$. *Then* $\vartheta_i(\xi_i) = L_i(s_k\downarrow_i)$. *If* $\xi_i = \emptyset$ *then* $\vartheta_i(\xi_i) = L_i(s_0\downarrow_i)$.

Proposition 2.3.
$\sigma_k \Vdash \alpha$ *iff* $\mu^\sigma, \xi^k \Vdash \alpha$ *and, if* $\sigma_k\downarrow_i$ *is not undefined,* $\sigma_k\downarrow_i \Vdash_i \varphi$ *iff* $\mu_i^\sigma, \xi^k|_i \Vdash_i \varphi$.

3 Distributed Non-determinisitic Büchi Automata

We will now start a construction of a new type of automaton with the purpose of being able to describe all the linear models of a DTL formula. First we will utilize GNBA's to model local subformulas for each agent, and then we will combine them all in a single automata in a way that respects global and local rules.

3.1 Elementary sets

Let α be a global DTL formula, and consider $B \subseteq clo(\alpha)$. B is said to be:

- Consistent (for both local or global formulas):
  - (i) $\psi_1 \Rightarrow \psi_2 \in B$ iff $\neg\psi_1 \in B$ or $\psi_2 \in clo(\alpha)$ for $\psi_1 \Rightarrow \psi_2 \in B$;
  - (ii) if $\psi \in B$ then $\neg\psi \notin B$;
  - (iii) if $\top \in clo(\alpha)$ then $\top \in B$;
- Locally consistent: if $\mathsf{G}\psi \in B$ then $\psi \in B$, for every $\mathsf{G}\psi \in clo(\alpha)$;
- i-consistent: $@_i[\psi] \in B$ iff $\psi \in B$, for every $@_i[\psi] \in clo(\alpha)$;
- maximal: if $\psi \notin B$ then $\neg\psi \in B$, for all $\psi \in clo(\alpha)$;

B is said to be i-elementary when all above properties are verified.

3.2 Local GNBA

Let α be a global DTL formula. Then, for each agent $i \in Id$ we define its local GNBA $\mathcal{A}_i$ as:

1. $Q_i = \{q\downarrow_i : q \text{ is } i\text{-elementary and } q \subseteq clo(\alpha) \bigcup Lit_i\}$;
2. $Q_{0_i} = \{q \in Q_i : \alpha \in q \text{ and } ©_j[\psi] \notin q, \forall_{j\in Id, \psi \in \mathcal{L}_j}\}$;
3. $\Sigma_i = \mathcal{V}_i$;
4. $F_i = \{F_{\mathsf{G}\psi} : \mathsf{G}\psi \in clo(\alpha)\}$, where $F_{\mathsf{G}\psi} = \{q \in Q_i : \mathsf{G}\psi \in q \text{ or } \psi \notin q\}$;
5. $q \xrightarrow{\vartheta}_i q'$ if:
   - (i) $\vartheta = q \bigcap Lit_i$
   - (ii) $\mathsf{X}\psi \in q$ iff $\psi \in q'$, for all $\mathsf{X}\psi \in clo(\alpha)$;
   - (iii) $\mathsf{G}\psi \in q$ iff $\psi \in q$ and $\mathsf{G}\psi \in q'$, for al $\mathsf{G}\psi \in clo(\alpha)$;

This automata captures all sequences of possible i-valuations, respecting the rules defined by the subformulas of α present in each state. The states of $\mathcal{A}_i$ are sets of global subformulas and i-subformulas of α that are supposed to hold on that point. Being i-elementary assures that there are no contradictions on the formulas that can hold at each point, that is, if $\psi \in q$ then $\neg\psi \notin q$ for example. μ is a model of α if $\mu, \emptyset \Vdash \alpha$, and as such we define the initial states as being the ones that contain α. Additionally, recall that $\mu, \emptyset \nVdash ©_j[\psi]$, so we can't have initial states with communication subformulas.

The acceptance sets $F_{\mathsf{G}\psi}$ are used to eliminate runs where ψ is always present from a certain point in time without having $\mathsf{G}\psi$ holding. This is achieved by forcing runs to pass infinitely often by all $F_{\mathsf{G}\psi}$ sets, i.e., when we have a run for which ψ always holds from k onwards, then $\mathsf{G}\psi$ will hold infinitely often, and by condition (iii) it means that $\mathsf{G}\psi$ will always hold as well from k onwards.

The alphabet is the set of valuations of agent i, and each state transitions by the valuation present in that state. As such, the $\mathcal{L}(\mathcal{A}_i)$ will be the set sequences of i-valuations that respect the i-elementary conditions, the temporal conditions indicated by (ii) and (iii), while having α hold on the first state.

Throughout the next sections, we will be using two new functions to separate local and global formulas from NBA states: given a NBA state q, we say that $loc(q)$ and $gl(q)$ are the sets of local and global formulas of q respectively.

3.3 Distributed Non-Deterministic Büchi Automata

Lets' consider now that we already have a local NBA $\mathcal{A}_i$ for each of our agents $Id = \{1, ..., n\}$ (which is achieved by transforming the GNBA's). Our goal now is to combine these NBA's in such a way that the resulting automaton captures $Mod_l(\alpha)$ in its language. The following construction is similar to the synchronized product automaton on [Muk12], but adapted as to follow DTL rules.

Definition 3.1.
Given a set of local NBA's $\{\mathcal{A}_i\}_{i\in ID}$ for a DTL formula α, a distributed non-deterministic Büchi automaton is a tuple $\mathcal{D} = \langle Q, \Sigma, \delta, Q_0, \mathcal{F}\rangle$ such that:

- $Q = \{(q\downarrow_0, q\downarrow_1, ..., q\downarrow_n) : (q\downarrow_0 \bigcup q\downarrow_i) \in Q_i, q\downarrow_0 = gl(q\downarrow_0 \bigcup q\downarrow_i)$ *and* $q\downarrow_i = loc(q\downarrow_0 \bigcup q\downarrow_i)\}$;
- $Q_0 = \{q \in Q : (q\downarrow_0 \bigcup q\downarrow_i) \in Q_{0_i}\}$;
- $\Sigma = \{a \subseteq \bigcup \mathcal{V}_i : a \neq \emptyset$ *and* $|a \bigcap \mathcal{V}_i| \leq 1\}$;
- $\mathcal{F} = \{\mathcal{F}_i\}_{i\in ID}$, *where* $\mathcal{F}_i = \{q \in Q : (q\downarrow_0 \bigcup q\downarrow_i) \in F_i\}$;
- $q \xrightarrow{a} q'$ *if:*
  1. *when* $a \bigcap \mathcal{V}_i = \emptyset$ *we have* $q\downarrow_i = q'\downarrow_i$;
  2. *when* $a \bigcap \mathcal{V}_i \neq \emptyset$ *we have* $(q'\downarrow_0 \bigcup q'\downarrow_i) \in \delta_i(q\downarrow_0 \bigcup q\downarrow_i, a \bigcap \mathcal{V}_i)$;
  3. *if* $a \bigcap \mathcal{V}_i \neq \emptyset, ©_j[\psi] \in q'\downarrow_i$ *then* $a \bigcap \mathcal{V}_j \neq \emptyset$ *and* $\psi \in q'\downarrow_j$;
  4. *if* $a \bigcap \mathcal{V}_i \neq \emptyset$, $a \bigcap \mathcal{V}_j \neq \emptyset$, $\psi \in q'\downarrow_j$ *and* $©_j[\psi] \in subf_i(\alpha)$ *then* $©_j[\psi] \in q'\downarrow_i$.

The states of the DNBA are all the possible combinations of states from the NBA's, such that all the global formulas are separated from the local ones, where $q\downarrow_0$ are the global formulas and $q\downarrow_i$ are the local formulas of agent i. This way, all states from the NBA's are combined in a way that makes sure the global formulas present on each agent are the same, avoiding contradictions. As for the initial states, we choose the ones for which all the combined states were respective initial states for their own local NBA. The acceptance sets $\mathcal{F}_i$ correspond to the final sets of each NBA, so that for a run to be accepted globally it needs to be accepted individually by each local automata.

The alphabet of our DNBA consists on sets of valuations, at least one agent and at most one valuation per agent. These will be used to indicate the sequences of present propositional symbols, as well as which agents synchronize at which point(active agents). To better understand it let's take a look at how transitions work. Condition 1 states that the agents without valuations in a can't move which means that the local formulas remain the same. As for the ones with valuations in a the global and local formulas need respect all the local transitions by the respective valuation. Note that $a \bigcap \mathcal{V}_i$ is loosely used as its only element. Finally the third and fourth conditions make sure that the synchronization rules are respected. If agent i is active and there's a communication with j on formula ψ, then j must also be active and must satisfy ψ. Similarly, when agents i and j are both active and ψ is present in j then this means agent i must have communicated with j on ψ, assuming that $©_j[\psi]$ is an i-subformula of α.

The goal of this construction is to have a global automaton that connects all local NBA's, and as such, by noticing that runs in an NBA are infinite, we need to specify that all runs in the resulting DNBA are *fair*, i.e., each particular agent needs to be active infinitely often. This makes it so we can extract valid words $\omega \downarrow_i$ from a global word ω.

Theorem 3.2.
Let $\mathcal{D}$ be the DNBA for a DTL formula α. Then $\mathcal{L}(\mathcal{D})$ captures $Mod_l(\alpha)$, that is, it is possible to establish a bijection between $\mathcal{L}(\mathcal{D})$ and $Mod_l(\alpha)$, meaning that the language accepted by $\mathcal{D}$ is actually $Mod_l(\alpha)$.

4 DTL Model Checking

The main step of our algorithm is already completed. After obtaining a DNBA with a language corresponding to all the linear models of α we will now use that same DNBA and combine it with our DTS. This way the states of the DNBA will keep track of what subformulas should hold in each state of the DTS.

Definition 4.1.
Let $T = \langle S, A, \xrightarrow{\cdot} I, Prop, L\rangle$ be a distributed transition system, and $\mathcal{D} = \langle Q, \Sigma, \delta, Q_0, \mathcal{F}\rangle$ a DNBA such that $\Sigma = \{a \in \bigcup \mathcal{V}_i : a \neq \emptyset$ and $|a \bigcap \mathcal{V}_i| \leq 1\}$, that is, the language of $\mathcal{D}$ consists on non-empty sets of valuations, with at most one valuation for each agent. Then we define the product $T \bigotimes \mathcal{D} = \langle S', A', \rightarrow' , I', Prop', L'\rangle$ as the following (non distributed) transition system:

- $S' = \{\langle s, q\rangle : s \in S, q \in Q$ *and* $L_i(s) = q \bigcap Prop_i\}$;
- $A' = A$;
- $\langle s, q\rangle \xrightarrow{a} \langle s', q'\rangle$ *if:*
  1. $s \xrightarrow{a} s'$;
  2. $q' \in \delta(q, \{L_i(s' \downarrow_i) : a \in A_i\})$
- $I' = \{\langle s, q\rangle \in S' : s \in I$ *and* $q \in Q_0\}$;
- $Prop' = Q$;
- $L'(\langle s, q\rangle) = \{q\}$.

Remember that $L_i(s' \downarrow_i) \in \mathcal{P}(Prop_i)$ and in this case we are actually referring to the corresponding valuation in $\mathcal{V}_i$.

Definition 4.2. *Let $\varphi = \langle \varphi_1, ..., \varphi_n\rangle$ be any propositional formulas over a set of propositions Γ. A distributed persistence property induced by φ is $P^{\varphi}_{dp} = \{\sigma \in (2^{\Gamma})^{\omega} : \exists_i \exists_j \forall_{k>j} \sigma_k \Vdash \varphi_i\}$.*

To clarify, P_{dp}^{φ} is the set of traces over Γ for which at least one of the φ_i formulas always holds from a certain point forward (hence called persistence).

In particular, given a DNBA $\mathcal{D} = \langle Q, \Sigma, \delta, Q_0, \mathcal{F} \rangle$, we define $P_{dp}(\mathcal{D})$ as the distributed persistence property over $\Gamma = Q$ induced by $\langle \bigwedge_{q \in \mathcal{F}_1} \neg q, ..., \bigwedge_{q \in \mathcal{F}_n} \neg q \rangle$. As such, $P_{dp}(\mathcal{D})$ will contain all the traces over Q (infinite sequences of states) for which at least one of the $\bigwedge_{q \in \mathcal{F}_i} \neg q$ persists from certain point forward, that is, $\mathcal{F}_i$ is never visited again.

Lemma 4.3.
Let $s = s_0 \xrightarrow{a_0} s_1 ...$ be a run on a DTS T over a set of agents Id with $\sigma = Trace(s)$, and $\pi = q_0 q_1 ...$ a run on a DNBA $\mathcal{D}$ over the same set of agents, with $\omega = \omega_0 \omega_1 ...$ its respective word. If $q_0 \downarrow_i \bigcap Prop_i = L_i(s_0 \downarrow_i)$, and ω_k's elements are the valuations of $\mathcal{V}_i$ that correspond to the elements of σ_{k+1}, then $\mu^{\sigma} = \mu^{\omega}$.

Similarly, if we know that $\mu^{\sigma} = \mu^{\omega}$ for some $\sigma \in Traces(T)$ and ω a word for a run in $\mathcal{D}$, then σ and ω will be in the conditions we just specified.

To clarify, we want the set of propositions on the states of T and $\mathcal{D}$ to match, such that ω is equivalent to σ starting from σ_1, that is, if $Prop_1 = \{p_1, p_2\}, Prop_2 = \{p'_1, p'_2, p'_3\}, Prop_3 = \{p''_1\}$, and $\sigma_{k+1} = \{\{p_1\}, \{p'_2\}\}$, we want $\omega_k = \{\{p_1, \neg p_2\}, \{\neg p'_1, p'_2, \neg p'_3\}\}$. For simplicity we can say that $\{p1, \neg p2\} = \{p1\}$.

Definition 4.4.
Given a set of traces Tr or a set of words $\mathcal{L}$, we define μ^{Tr} and $\mu^{\mathcal{L}}$ as the sets of their respective induced interpretation structures, that is, $\mu^{Tr} = \{\mu^{\sigma} : \sigma \in Tr\}$ and $\mu^{\mathcal{L}} = \{\mu^{\omega} : \omega \in \mathcal{L}\}$.

Notice how $\mu^{\mathcal{L}(\mathcal{D})} = Mod_l(\alpha)$ if $\mathcal{D}$ is the DNBA constructed for α.

Theorem 4.5.
Let $T = \langle S, A, \dashrightarrow I, Prop, L \rangle$ be a DTS and $\mathcal{D} = \langle Q, \Sigma, \delta, Q_0, \mathcal{F} \rangle$ the DNBA for which $\mathcal{L}(\mathcal{D}) = Mod_l(\neg \alpha)$. Then the following are equivalent:

1. $T \vDash \alpha$
2. $\mu^{Traces(T)} \bigcap \mu^{\mathcal{L}(\mathcal{D})} = \emptyset$
3. $T \bigotimes \mathcal{D} \vDash P_{dp}(\mathcal{D})$

We can finally complete our algorithm for DTL model checking on distributed transition systems. Let's say we have a DTS T for which we want to check α.

1. Build the DNBA $\mathcal{D}$ for $\neg \alpha$;
2. Construct the product $T \bigotimes \mathcal{D}$;
3. Reduce $T \bigotimes \mathcal{D}$ to its reachable states;
4. Search for a path $\langle \tau, \pi \rangle$ of $T \bigotimes \mathcal{D}$ such that $\pi \notin P_{dp}(\mathcal{D})$:
   (a) If no trace is found then $T \vDash \alpha$;
   (b) Otherwise we return τ as an example of a path of T that proves that $T \nvDash \alpha$.

To search for $\langle \tau, \pi \rangle$ we do as follows:

1. Consider $F_i = \{\langle s, q\rangle : q \in \mathcal{F}_i\}$;
2. From each F_i remove the states that can't be in a cycle, that is, the states for which there is no path that connects them to themselves;
3. Now consider $C = \bigotimes_i F_i$, which is the set of cycle candidates, that is, the elements of C are tuples $\langle c_1, c_2, ..., c_n\rangle$ for which we want to know whether there is a cycle that goes through all of them;
4. For each $\langle c_1, c_2, ..., c_n\rangle \in C$ check if c_{k+1} is reachable from c_k and if c_1 is reachable from c_n;

If we find a tuple $\langle c_1, ..., c_n\rangle \in C$ that verifies the condition on 4, then we can conclude there is a cycle that goes through $c_1, ..., c_n$. Remember that $c_i = \langle s_i, q_i\rangle$ where $q_i \in \mathcal{F}_i$, so , if there is a cycle that goes through all c_i, it means that every $\mathcal{F}_i$ is visited infinitely often and as such it means that a path that follows that cycle can't be in $P_{dp}(\mathcal{D})$, since it would be required for at least one $\mathcal{F}_i$ to never be visited again. As such, the corresponding path in T does not verify α, as seen on theorem 5.1.5. On the other hand, if no possible cycles are found in C, then we know that $T \vDash \alpha$, since $T \bigotimes \mathcal{D} \vDash P_{dp}(\mathcal{D})$.

5 Concluding Remarks

The goal of this project was to find a way of applying model checking algorithms to distributed systems without the need to project said systems onto a transformed linear model. As intended, we were able to preserve the nature of the systems, keeping them distributed, and developed a Distributed Temporal Logic model checking algorithm based on Distributed Non-Deterministic Büchi Automata that either confirms that a formula is true or, when it is false, gives us an example of a path on the distributed system that does not satisfy the intended formula.

References

[BCRV10] David Basin, Carlos Caleiro, Jaime Ramos, and Luca Viganò. *Distributed Temporal Logic for the Analysis of Security Protocol Models.* 2010.

[BK08] Christel Baier and Joost P. Katoen. *Principles of Model Checking.* The MIT Press, May 2008.

[CGRV] Carlos Caleiro, Paula Gouveia, Jaime Ramos, and Luca Viganò. *A Tableux-based decision procedure for distributed temporal logic.*

[CVB05] Carlos Caleiro, Luca Viganò, and David A. Basin. *Metareasoning about Security Protocols using Distributed Temporal Logic.*, volume 125. 2005.

[EC80] E. Allen Emerson and Edmund M. Clarke. *Characterizing Correctness Properties of Parallel Programs Using Fixpoints.*, volume 85 of *Lecture Notes in Computer Science.* Springer, 1980.

[EC00] Hans-Dieter Ehrich and Carlos Caleiro. *Specifying Communication in Distributed Information Systems.*, volume 36. 2000.

[Muk12] M. Mukund. *Automata on Distributed Alphabets, Modern Application of Automata Theory.* 2012.

[Pnu77] Amir Pnueli. *The Temporal Logic of Programs.* IEEE Computer Society, 1977.