Deciding Dynamic Deontic Linear-Time Temporal Logic

Christoph Reichenbach <jameson@darmstadt.gmd.de>

Mainhausen, February 19, 2001

Abstract

Dynamic Deontic Linear-Time Temporal Logic (DDLTLB) is known to have several properties that make it interesting with regard to modelling real-life business transactions, contracts, and deadlines. However, it is undecideable and rather complex in practical reasoning. This document discusses restrictions that have to be placed on the DDLTLB in order to make it decideable, and how those restrictions change its practical usefulness.

1 Introduction

Modern businesses are, neccessarily, subject to a vast amount of contracts, obligations, and deadlines. Meeting those while still generating a profit has proven to be a non-trivial matter in the general case, especially since most of the planning and contract checking is done manually. Therefore, it would appear to be desirable to automatize as much of this as possible, in order to save time and other resources (perhaps excluding electricity). This, in turn, implies that a formalism needs to be invented to specify these obligations, contract agreements, and deadlines as precisely as possible.

1.1 The modal logic approach

Dynamic Deontic Linear-Time Temporal Logic (henceforth referred to as "DDLTLB"), first introduced in [1], is a composite modal logic. Basing itself on dynamic logic, it assigns a temporal property to each action, thus describing linear-time temporal logic, and adds the deontic operators with the restriction that it must be possible to fulfill the obligations they represent.

This logic will be summarized and examined in section 2. Section 3 will provide a proposal for embedding it in first-order logic, while Section 4 will restrict the language to make it decideable and discuss certain extensions and how they affect decideability.

2 Dynamic Deontic Linear-Time Temporal Logic

This section is a short summary of the DDLTLB concept proposed by Dignum and Kuiper in [1]. For further details, refer to this document and the preceeding [2].

2.1 Dynamic logic

Dynamic logic[3], well-established in computer science, served as a basis for the DDLTLB. The main property of dynamic logic is that it differentiates between separate states (or *worlds*), each of which may give different semantics to a set of propositions (or predicates if based on a higher-order logic). Transition between states is performed by executing actions, which are defined as relations between states, resulting in a non-deterministic traversal through state space (except for this non-determinism, this concept is similar in many ways to the situation calculus introduced in [5]).

Definition 1: Dynamic logic can be described by means of a *Kripke structure*[1] $M = (\Pi, \Sigma, \pi, R_{\mathcal{A}})$ with the following definitions:

- Σ: The set of all states
- Φ: The set of all propositions
- Π: The set of actions
- $\pi\colon \Sigma \to (\Phi \to \{tt,\ ff\})$: State dependant valuations for all propositions
- $R_{\mathcal{A}}\colon \Pi \to (\Sigma \times \Sigma)$: The state transition relation for all actions

Dynamic logic also defines a set of operations over actions $\alpha, \beta \in \Pi$ and propositions $\phi \in \Phi$:

1. $[\alpha]\phi$: true iff ϕ holds after α is executed
2. $\alpha; \beta$: The action "α, then β"
3. $\alpha \cup \beta$: Non-deterministic action "α or β"

We also define the following operations not found in [1]:

4. $\phi?\alpha\beta$: If ϕ, then α, else β
5. α^*: The action of executing α zero or more times

We will refer to actions not containing any of the operators defined in points 2–5 as *atomic actions.*

2.1.1 Emulating operators on actions

Lemma 1: Operator 4 can be emulated as follows:

$$[\phi?\alpha\beta]\psi \equiv ([\alpha]\phi' \wedge [\beta]\neg\phi') \wedge [\alpha \cup \beta]\psi$$

for a suitable existing or fresh variable ϕ' describing the value of ϕ in the previous state into each of our states. If the propositional value of ϕ was different for two sets of states $\mathcal{P}_1$, where it was true, and $\mathcal{P}_2$, where it was false (implying that we had to introduce a new fresh variable), we split up the current state σ into two separate states σ_1 and σ_2 with the same successors and propositional valuations, except that $\pi(\sigma_1)(\phi') = tt$ and $\pi(\sigma_2)(\phi') = ff$. We choose the successors identically, i.e. by letting

$$\forall \alpha \in \Pi \forall \sigma_{next} \in \Sigma . R_{\mathcal{A}}(\alpha)(\sigma, \sigma_{next}) \iff R_{\mathcal{A}}(\alpha)(\sigma_1, \sigma_{next}) \iff R_{\mathcal{A}}(\alpha)(\sigma_2, \sigma_{next})$$

apply, and set the predecessors as

$$\forall \alpha \in \Pi \forall \sigma_{prev} \in \Sigma . R_{\mathcal{A}}(\alpha)(\sigma_{prev}, \sigma) \wedge \pi(\sigma)(\phi) = tt \Rightarrow R_{\mathcal{A}}(\alpha)(\sigma_{prev}, \sigma_1)$$

and

$$\forall \alpha \in \Pi \forall \sigma_{prev} \in \Sigma . R_{\mathcal{A}}(\alpha)(\sigma_{prev}, \sigma) \wedge \pi(\sigma)(\phi) = ff \Rightarrow R_{\mathcal{A}}(\alpha)(\sigma_{prev}, \sigma_2)$$

[1] A non-deterministic finite state machine.

For general usefulness, we would have to apply this to all states $\sigma \in \Sigma$.

We can also emulate operation 3 by realizing that

$$[\gamma_1 \cup \gamma_2]\phi \equiv [\gamma_1]\phi \wedge [\gamma_2]\phi$$

Unfortunately, we cannot emulate it under closure with the Kleene star operator (5) yet, but temporal logic will give us the ability to do so later[2].

2.2 Deontic logic

While dynamic logic is a rather powerful model, Dignum, Weigand and Verharen found it insufficient to express obligations. Specifically, it does not distinguish between statements and obligations for statements. For example, when pressing the handle of a door and pulling, the door should open. In propositional logic, we can write this as

$$p \to o$$

where p stands for "press handle and pull" and o means "door is open" (of course we could write this even more nicely in dynamic logic). However, the door being open afterwards is, in practice, in no way an unavoidable consequence. For instance, the door might be locked, blocked by a barrage of debris on the other side, or just painted on the wall. Even in dynamic logic, it is not easy to express this. If we just stated $\neg o$, our current model would be inconsistant because $\{p, p \to o\}$ hold in it as well.

Also, simply stating that, after attempting to open the door, it is either open or not open does not appear to be particularly useful in a practical context. Three ways to model this situation come to mind:

1. Model every possible situation precisely
2. Model relevant situations, but summarize "miscellaneous" cases
3. Distinguish between facts and things that "ought to be"

Solution (1) is, obviously, not practical. Solution (2) is commonly used in bayesian belief networks[6]. We could attempt to apply this here; for example, we could add a preposition e which would mean "an eventuality is preventing the door from being opened", thus allowing us to re-formulate our statement above to

$$p \wedge \neg e \to o$$

The downside of this approach is that, if we use it in our dynamic model, each state explicitly specifies the truth or falsehood of all propositional values, which include the "eventuality" factor just introduced. Therefore, we must describe the value of e in all cases, thereby reducing it to (1).

So we will have to look into the fourth alternative.

Distinguishing between facts and obligations is the main property of deontic logic, therefore, we will attempt to add the deontic operators:

1. $O(\phi)$: It is obligatory that ϕ is true
2. $F(\phi) = O(\neg\phi)$: It is forbidden that ϕ is true
3. $P(\phi) = \neg O(\neg\phi)$: It is permitted that ϕ is true
4. $I(\phi)$: "Indifference": No statement is given about the deontic nature of ϕ

[2] Temporal logic will also allow us to emulate the '?' operator much more easily.

As we can see, the operators F and P are defined by means of the O operator, so it is sufficient to describe O to model them. The I operator does not appear to add any significant semantic value to our model, so we will omit it in this paper.
Definition 2: The model proposed to combine deontic and dynamic logic is a Kripke structure extended by further relations:

$$\mathcal{K} = (\Pi, \Sigma, \pi, R_{\mathcal{A}}, \leq, R_{\mathcal{O}})$$

The new relations are defined as follows:

- $\leq: \Sigma \times \Sigma$: Reflexive and transitive preferrance relation over states
- $R_{\mathcal{O}}: \Sigma \times \Sigma$: Deontic ideality relation

The meaning of $\leq$ is relatively straightforward: It allows us to define a partial order over states. It is not directly relevant to the adding of deontic logic to our model, but it allows us to define a state-dependant binary preference relation over Π.
On the other hand, $R_{\mathcal{O}}: \Sigma \times \Sigma$, where

$$\forall \sigma \in \Sigma . \exists \sigma' \in \Sigma . R_{\mathcal{O}}(\sigma, \sigma')$$

(i.e. for each state σ there exists at least one "deontically ideal" state σ') allows us to generally define the semantics of a formula $O(\phi)$ as follows:
Definition 3:

$$(\mathcal{K}, \sigma) \vDash O(\phi) \iff \forall \sigma' \in \Sigma . R_{\mathcal{O}}(\sigma, \sigma') \rightarrow (\mathcal{K}, \sigma') \vDash \phi$$

This means that the semantics of the deontic operators are defined solely by $R_{\mathcal{O}}$. We will also require that $R_{\mathcal{O}}(\sigma, \sigma')$ implies that σ' can be reached from σ.

2.3 Propositional linear time temporal logic

Propositional linear time temporal logic, or PLTLB[3] for short, was then added to allow some form of temporal descriptions beyond the scope of dynamic logic. PTLTB assumes a discrete understanding of time and defines the following operators:

1. $\bigcirc\phi$: In the next step, the formula ϕ holds
2. $\ominus\phi$: The formula ϕ held in the previous step
3. $\phi\mathcal{U}\psi$: The formula ϕ holds until ψ holds
4. $\phi\mathcal{S}\psi$: ϕ holds since ψ held

Note that PLTLB actually defines a few more operators, all of which can be derived from the ones mentioned above.
The discrete understanding of time mentioned above is implied by the first two operators, which talk about the concept of "next" and "previous" steps, respectively. Embedding PLTLB into the dynamic deontic logic model constructed above turns out to be non-trivial, though.

[3]The B is intended to point out that the logic defines operators for both future and past tense.

2.3.1 Discrete step operators

The $\bigcirc$ and $\ominus$ operators can be designed in such a way that all actions found in our set of actions Π take an integer multiple of this atomic time unit[4] as the time they need for their execution. In the most simple of cases of each action taking exactly one time unit, a formula $\ominus\psi$ would hold if ψ held right before the last action, while $\bigcirc\phi$ would hold if ϕ held after the the next action was executed.

2.3.2 Until and Since

Definition 4 We define the semantics of the $\mathcal{U}$ and $\mathcal{S}$ operators as follows:

$$(\mathcal{K}, \sigma) \vDash \phi\mathcal{S}\psi \iff (\mathcal{K}, \sigma) \vDash \psi \vee (\phi \wedge \ominus(\phi\mathcal{S}\psi))$$

and

$$(\mathcal{K}, \sigma) \vDash \phi\mathcal{U}\psi \iff (\mathcal{K}, \sigma) \vDash \psi \vee (\phi \wedge \bigcirc(\phi\mathcal{U}\psi))$$

In [1], an alternative definition is provided: $\mathcal{U}$ and $\mathcal{S}$ are described by means of a *Path* structure, which contains the "history" of the current state. Such a path could be described as an element of $(\Sigma \times \Pi)^*$ with additional restrictions, like

$$PATH = \{\langle(\sigma_0, \alpha_0), (\sigma_1, \alpha_1), ..., (\sigma_n, \alpha_n)\rangle \in (\Sigma \times \Pi)^* | \forall 0 \leq i < n.R_{\mathcal{A}}(\alpha_i)(\sigma_i, \sigma_{i+1})\}$$

We could now describe the semantics of the $\mathcal{S}$ and $\mathcal{U}$ operators wrt a model defined by a Kripke structure $\mathcal{K}$, a path $\rho \in PATH$ and a state $\sigma \in \Sigma$. While this solution appears to be equally powerful and describes the complexity and memory structure more appropriately, it does not offer any functional advantages over the cleaner description from [4] at this point. However, we will refer back to it in the next section.

2.3.3 Discrete operators over actions

As defined above, $\bigcirc$ and $\ominus$ are only applicable to propositional formulas. In order to enforce execution of an action α as the next step, we would need a fresh helper proposision ζ, which would have to be added to the set of propositions for this purpose:

$$\neg\zeta \wedge \bigcirc\zeta \wedge [\alpha]\zeta$$

provided that[5]

$$\forall\beta \in \Pi.([\beta]\zeta) \to \beta = \alpha$$

This is abbreviated in [1] as $DO(\alpha)$. Similarly, we can define $DONE(\alpha)$ for the past operator as

$$\zeta \wedge \ominus(\neg\zeta \wedge [\alpha]\zeta)$$

if it is guaranteed that

$$\forall\beta \in \Pi.(\ominus([\beta]\zeta) \to \beta = \alpha)$$

As pointed out in [1], assigning differing time intervals to actions introduces a problem: If α and β require a different time to execute, how long does $\alpha \cup \beta$ take? In order to avoid this problem, it was proposed to split up the actions into several

[4]While theoretically possible, it would appear not to be particularly useful to let $\bigcirc$ and $\ominus$ describe time intervals of different length. Even with a reason to do so, it is always possible to emulate this behaviour by normalizing their common time interval length to the greatest common denominator of their former lengths and using an appropriate sequence of operators in the formula instead.

[5]This can be enforced by splitting states as done in the proof of Lemma 1.

sub-actions. An alternative approach would be to introduce a finite set of new states, linked only by this specific action.
Let us assume that we have a set $T \subset \mathbb{N}$ of numbers of steps after which an action α might terminate. Choose $n \in T$ so that $\forall j \in T.j \leq n$. If we assume that T is finite, we can now, for every state $\sigma \in \Sigma$ where $\exists \sigma' \in \Sigma.R_{\mathcal{A}}(\alpha)(\sigma, \sigma')$ add in-between states $\sigma_2 ... \sigma_n$. We will also call $\sigma_1 \equiv \sigma$. Now, for each $j \in T$ we add a relational connection $R_{\mathcal{A}}(\alpha)(\sigma_j, \sigma')$ and require the following:

$$\forall j.R_{\mathcal{A}}(\beta)(\sigma_j, \theta) \iff \beta = \alpha \wedge (\theta = \sigma_{j+1} \vee (\theta = \sigma' \wedge j \in T))$$

For states with a possibly infinite duration but periodic termination indices, we can add extra states linking to themselves or to previous states. Infinite non-periodic sets of termination points, such as the set $\{n^2 | n \in \mathbb{N}\}$, cannot be modelled this easily. Fortunately, they are very unlikely to be of any practical relevance.

2.4 The resulting composite logic

We have now taken properties and operators of dynamic, deontic, and propositional linear-time temporal logic as the foundation of our dynamic deontic linear-time temporal logic. As in [1], we also add predicates and arithmetic expressions in the usual way.

3 DDLTLB and first-order logic

There are many ways to embed DDLTLB into first-order logic. Our approach will be based on two finite sets of propositions and predicates; we will call them $PROP$ and $PRED$, respectively. In addition to the elements contained in $PROP$, all formulas $\dot{\neg}\phi$, $\phi\dot{\vee}\psi$, $\phi\dot{\wedge}\psi$ and $\phi\dot{\rightarrow}\psi$ will be considered to be propositional formulas in our first order logic, for ϕ and ψ from the set of propositional formulas.
We will use the notation $[\![\phi]\!]_{\sigma,\rho,t}$ for the semantice of the formula ϕ with respoect to a state $\sigma \in \Sigma$, a history $\rho \in PATH$, and a time index $t \in \mathbb{N}$. σ, rho and t will usually be omitted for brevity unless relevant for the semantics of the formula in quesiton.

3.1 Sorts

First, we must remember the sorts we have in DDLTLB:

1. Σ: States
2. Π: Actions
3. Φ: Propositions
4. $PATH$: Temporal traces

We can define each sort by means of a unary predicate like these:

- $[\![\sigma \in \Sigma]\!]$:= state-p(σ) $\dot{\wedge}\dot{\neg}$action-p(σ) $\dot{\wedge}\dot{\neg}$prop-p(σ)$\dot{\wedge}\dot{\neg}$path-p(σ)
- $[\![\alpha \in \Pi]\!]$:= action-p(α) $\dot{\wedge}\dot{\neg}$state-p(α) $\dot{\wedge}\dot{\neg}$prop-p(α)$\dot{\wedge}\dot{\neg}$path-p(α)
- $[\![\phi \in \Phi]\!]$:= prop-p(ϕ) $\dot{\wedge}\dot{\neg}$state-p(ϕ) $\dot{\wedge}\dot{\neg}$action-p(ϕ)$\dot{\wedge}\dot{\neg}$path-p(ϕ)
- $[\![\rho \in PATH]\!]$:= path-p(ρ) $\dot{\wedge}\dot{\neg}$state-p(ρ) $\dot{\wedge}\dot{\neg}$prop-p(ρ)$\dot{\wedge}\dot{\neg}$action-p(ρ)

This guarantees that our states, propositions and actions are disjunct.

3.2 Propositions

We can define the semantics of our propositional valuation function π with a predicate *pi* as follows (for $\chi \in \Phi$ and ϕ, ψ formulas in our DDLTLB):

1. $[\![\chi]\!]_\sigma := pi(\sigma, \chi)$
2. $[\![\neg\phi]\!]_\sigma := \dot{\neg}[\![\phi]\!]_\sigma$
3. $[\![\phi \wedge \psi]\!]_\sigma := [\![\phi]\!]_\sigma \dot{\wedge} [\![\psi]\!]_\sigma$
4. $[\![\phi \vee \psi]\!]_\sigma := [\![\phi]\!]_\sigma \dot{\vee} [\![\psi]\!]_\sigma$
5. $[\![\phi \rightarrow \psi]\!]_\sigma := [\![\phi]\!]_\sigma \dot{\rightarrow} [\![\psi]\!]_\sigma$
6. $[\![P(\phi, ..., \psi)]\!]_\sigma :=$ddltlb$(\sigma, P, [\![\phi]\!]_\sigma, ..., [\![\psi]\!]_\sigma)$

where P as a predicate symbol is present in our set of predicate symbols.

3.3 Actions

The semantics of $R_{\mathcal{A}}$ can be modelled similarly to the way this is done in the situation calculus:

$$[\![R_{\mathcal{A}}(\alpha)(\sigma, \sigma')]\!] := result(\alpha, \sigma, \sigma')$$

where α is an atomic action (Note that this definition does not yet cover the temporal operators, so it will need to be changed in section 3.5).
Of course, we also need to handle the various dynamic operators:

1. $[\![[\alpha]\phi]\!]_\sigma := \forall \sigma'.[\![R_{\mathcal{A}}(\alpha)(\sigma, \sigma')]\!] \dot{\rightarrow} [\![\phi]\!]_{\sigma'}$
2. $[\![R_{\mathcal{A}}(\alpha; \beta)(\sigma, \sigma'')]\!] := [\![R_{\mathcal{A}}(\alpha)(\sigma, \sigma')]\!] \dot{\wedge} [\![R_{\mathcal{A}}(\beta)(\sigma', \sigma'')]\!]$
3. $[\![R_{\mathcal{A}}(\alpha \cup \beta)(\sigma, \sigma')]\!] := [\![R_{\mathcal{A}}(\alpha)(\sigma, \sigma')]\!] \dot{\vee} [\![R_{\mathcal{A}}(\beta)(\sigma, \sigma')]\!]$
4. $[\![R_{\mathcal{A}}(\phi?\alpha\beta)(\sigma, \sigma')]\!] := ([\![\phi]\!]_\sigma \dot{\wedge} [\![R_{\mathcal{A}}(\alpha)(\sigma, \sigma')]\!]) \dot{\vee} (\dot{\neg}[\![\phi]\!]_\sigma \dot{\wedge} [\![R_{\mathcal{A}}(\beta)(\sigma, \sigma')]\!])$
5. $[\![R_{\mathcal{A}}(\alpha^*)(\sigma, \sigma'')]\!] := (\sigma \dot{=} \sigma'') \dot{\vee} (\exists \sigma' [\![R_{\mathcal{A}}(\alpha)(\sigma, \sigma')]\!] \dot{\wedge} [\![R_{\mathcal{A}}(\alpha^*)(\sigma', \sigma'')]\!])$

3.4 Deontic obligation

As per definition, O can be implemented using the deontic ideality relation:

$$[\![R_{\mathcal{O}}(\sigma, \sigma')]\!] := di(\sigma, \sigma')$$

and therefore

$$[\![O(\phi)]\!]_\sigma := (\forall \sigma'.di(\sigma, \sigma') \dot{\rightarrow} \phi) \dot{\wedge} \dot{\exists} \rho = \langle(\sigma, \alpha_0)...(\sigma', \alpha_n)\rangle.\text{path-p}(\rho) \wedge di(\sigma, \sigma')$$

3.5 Temporal operators

In order to model past and future, we have to resort to a trace of our past and future, as suggested in section 2.3.2. To do so, two predicates *next* and *prev* will be introduced, with the following definitions:

$$next(\rho, t, \sigma_t, \alpha_t, \sigma_{t+1}) \iff \rho = \langle...(\sigma_t, \alpha_t)(\sigma_{t+1}, \alpha_{t+1})...\rangle$$

$$prev(\rho, t, \sigma_t, \alpha_t, \sigma_{t-1}) \iff \rho = \langle...(\sigma_{t-1}, \alpha_{t-1})(\sigma_t, \alpha_t)...\rangle$$

The semantics can then be defined as follows:

$$[\![\bigcirc(\phi)]\!]_{\sigma,\rho,t} := next(\rho, t, \sigma, \alpha, \sigma_n) \rightarrow [\![\phi]\!]_{\sigma_n,\rho,t+1}$$

$$[\![\ominus(\phi)]\!]_{\sigma,\rho,t} := prev(\rho,t,\sigma,\alpha,\sigma_p) \rightarrow [\![\phi]\!]_{\sigma_p,\rho,t-1}$$

We can now insert the descriptions of the $\mathcal{S}$ and $\mathcal{U}$ operators from section 2.3.2:

$$[\![\phi\mathcal{S}\psi]\!]_{\sigma,\rho,t} := [\![\psi]\!]_{\sigma,\rho,t}\dot{\vee}([\![\phi]\!]_{\sigma,\rho,t}\dot{\wedge}[\![\ominus(\phi\mathcal{S}\psi)]\!]_{\sigma,\rho,t})$$

$$[\![\phi\mathcal{U}\psi]\!]_{\sigma,\rho,t} := [\![\psi]\!]_{\sigma,\rho,t}\dot{\vee}([\![\phi]\!]_{\sigma,\rho,t}\dot{\wedge}[\![\bigcirc(\phi\mathcal{U}\psi)]\!]_{\sigma,\rho,t})$$

In order for the paths ρ to have any actual effect, we need to modify the semantics of all previous operators to carry both the state and the path. However, in the case of the first semantic definition in section 3.3, another change is required:

$$[\![R_{\mathcal{A}}(\alpha)(\sigma,\sigma')]\!]_{\sigma,\rho,t} := next(\rho,t,\sigma,\alpha,\sigma')\dot{\wedge}result(\alpha,\sigma,\sigma')$$

4 Deciding DDLTLB

As we have seen, it is relatively easy to reduce DDLTLB to first-order logic. However, this logic has the distinct disadvantage of not being decideable, so it seems natural to try to restrict the DDLTLB. The first and most obvious way to do so would appear to be ristricting it to propositional logic again.
However, the modal enhancements we have made are not covered– and do not appear to be easy to cover– by the standard resolution algorithm for propositional logic. Therefore, it cannot be assumed that DDLDLB is decideable yet.

4.1 Finiteness

Our first requirement to the language should be the finiteness of states. This implies a finiteness of propositions, since having a finite number of states Σ with an infinite number of propositions Φ implies that all propositions must be renamings of (at worst) $2^{|\Sigma|}$ propositions from Φ. On the other hand, a finite number of propositions does not imply a finite number of states, unless we also require that all states should be reachable by a finite sequence of actions. This might not be intuitively apparent to work in conjunction with obligations, but note that we have required (in section 2.2) that all deontically ideal states must be reachable; therefore they must be reachable by a finite number of steps. □

4.2 Sparse DDLTLB

Proposition 1: *DDLTLB with a finite set of states, without the star operator for actions, and without predicates, quantifiers or arithmetical formulas is decideable.*

Proof: Given an extended Kripke structure $\mathcal{K} = (\Pi, \Sigma, \pi, R_{\mathcal{A}}, \leq, R_{\mathcal{O}})$ and a set Φ of propositions to describe a specific DDLTLB calculus, and a set of formulas $\mathcal{F}$ to decide, we can re-write each formula $F \in \mathcal{F}$ in PTL, which is known to be decideable. For this, we introduce the following:

- A set of propositions ψ_σ with $\sigma \in \Sigma$, where each ψ_σ emulates the state σ
- A set of propositions κ_α with $\alpha \in \Pi$, to simulate the actions executed

Now, we re-write the formulas recursively:

1. $\phi \in \Phi, tt, ff$: Recursion anchor

2. $O(\phi)$: For each obligation ϕ, we have to consider the full range of possibilities. We can replace it by a conjunction of formulas

$$F = \bigwedge_{\sigma \in \Sigma} F_\sigma$$

where each $F_\sigma = \psi_\sigma \to \varphi$, where φ is true iff $\forall \sigma' \in \Sigma . R_{\mathcal{O}}(\sigma, \sigma').(\mathcal{K}, \sigma) \vDash \phi$, which can be determined by recursion on $\phi \wedge \varphi$, where φ is a conjunction of all elements $\eta_\xi \in \Phi$, where each element η_ξ is negated iff $(\mathcal{K}, \sigma') \vDash \neg \eta_\xi$.

3. $[\alpha]\phi$: We can can re-write this as a formula

$$\kappa_\alpha \to \bigwedge_{\beta \in \Pi \setminus \{\alpha\}} \neg \kappa_\beta \wedge \bigwedge_{\sigma \in \Sigma} \psi_\sigma \to F_\sigma$$

where F_σ describes the possible states in the next step:

$$F_\sigma := \bigcirc \Big(\bigvee_{\sigma^i \in \Sigma . R_{\mathcal{A}}(\alpha)(\sigma, \sigma^i)} \psi_{\sigma^i} \wedge \bigwedge_{\sigma' \in \Sigma . R_{\mathcal{A}}(\alpha)(\sigma, \sigma')} F_{\sigma, \sigma'} \Big)$$

and $F_{\sigma,\sigma'}$ forces exactly the state σ' to exist and evaluates ϕ:

$$F_{\sigma,\sigma'} := \sigma' \wedge \bigwedge_{\sigma^k \in (\{\varsigma | R_{\mathcal{A}}(\alpha)(\sigma, \varsigma)\} \cup \{\sigma\}) \setminus \{\sigma'\}} \neg \sigma^k \wedge \varphi \wedge \phi$$

where φ is a conjunction of all elements $\eta_\xi \in \Phi$ so that η_ξ is negated iff $(\mathcal{K}, \sigma') \vDash \neg \eta_\xi$.

4. $[\alpha; \beta]\phi$: We re-write this as $[\alpha][\beta]\phi$ and recurse.

5. $[\alpha \cup \beta]\phi$: We re-write this as $[\alpha]\phi \vee [\beta]\phi$ and recurse.

6. $\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$, $\phi \to \psi, \bigcirc\phi, \ominus\phi, \phi\mathcal{U}\psi, \phi\mathcal{S}\psi$: We recurse on ϕ and ψ (if present).

It remains to be shown that these transformations result in an equivalent formula; however, this should be intuitively apparent. □
We will refer to this kind of restricted DDLTLB as *Sparse DDLTLB*.

4.3 Extensions to Sparse DDLTLB

We will now consider the aspects that differentiate the DDLTLB from our sparse DDLTLB. All of these extensions introduce an infinite number of states.

4.3.1 Quantifiers and Predicates

It is obvious that DDLTLB with quantifiers and predicates is not decideable, since it carries the full expressional power of first-order logic.

4.3.2 Arithmetical expressions and variables

Proposition 2: *Adding arithmetical variables and expressions to the sparse DDLTLB makes the resulting logic undecideable.*

Proof: We consider a non-deterministic Turing machine $M = (Q, A, \Delta, q_0, q_t)$, where Q is the set of states, $q_0 \in Q$ is the initial state, $q_t \in Q$ is the state of termination, $\Delta \in (Q \times A) \times (Q \times A \times \{-1, 0, 1\})$ is the transition relation, and $A = \{\emptyset, 1, \#, \$\}$

is the band alphabet. For a given input string $\mathcal{I} \in A^*$, M is said to terminate iff it assumes the state q_t after a finite number of non-deterministic operations on this string.
We now model M in a Kripke structure $\mathcal{K} = (\Pi, \Sigma, \pi, R_{\mathcal{A}}, \leq, R_{\mathcal{O}})$, where our set of propositions $\Phi = \{s, t, b_\emptyset, b_1, b_\#, b_\$\}$. We choose $\Sigma = Q \times \mathbb{Q} \times \{0,1,2,3\} \times \mathbb{Q} \times \mathbb{Z}$ and write $(\sigma, b_l, b, b_r, i) \in \Sigma$ for each element of Σ. Now we set our propositional valuation function as follows:

- $\pi((\sigma, b_l, b, b_r, i))(s) = tt \iff (i < 0 \vee \sigma = q_0 \wedge i = 0 \wedge b = b_0 \wedge b_r = b_{r,0})$ (where b_0 and $b_{r,0}$ will be described later)
- $\pi((\sigma, b_l, b, b_r, i))(t) = tt \iff \sigma = q_t$
- $\pi((\sigma, b_l, b, b_r, i))(b_\#) = tt \iff b = 0$
- $\pi((\sigma, b_l, b, b_r, i))(b_\emptyset) = tt \iff b = 1$
- $\pi((\sigma, b_l, b, b_r, i))(b_1) = tt \iff b = 2$
- $\pi((\sigma, b_l, b, b_r, i))(b_\$) = tt \iff b = 3$

We define exactly one action: $\Pi = \{a\}$. Now, we set $R_{\mathcal{A}}$ as follows:

$$R_{\mathcal{A}}(a)((\sigma, b_l, b, b_r, i), (\sigma', b_l', b', b_r', i')) \iff \Delta(\sigma, b, \sigma', b', c) \wedge i' = i + c$$

so that all of the following conditions apply:

- $c = -1 \to b' = \lfloor \frac{b_l}{4^{i'}} \rfloor \wedge b_l' = b_l - b' * 4^{i'} \wedge b_r' = b_r + b * 4^i$
- $c = 0 \to b_l' = b_l \wedge b_r' = b_r$
- $c = 1 \to b' = \frac{b_r}{4^i} \bmod 4 \wedge b_l' = b_l + b * 4^i \wedge b_r' = b_r - b' * 4^{i'}$

Finally, we set the variables b_0 and $b_{r,0}$ mentioned above. For this we define the translation function $f(x)\colon A \to \{0,1,2,3\}$ as follows:

$$f(x) = \begin{cases} 0 & \iff x = \# \\ 1 & \iff x = \emptyset \\ 2 & \iff x = 1 \\ 3 & \iff x = \$ \end{cases}$$

and set $b_0 = f(i_0)$, where $(i_0, i_1, ..., i_n) \equiv \mathcal{I}$. To set $b_{r,0}$, we also define the following recursive function:

$$g((i, j...)) = \begin{cases} 0 & \iff (i, j...) = () \\ f(i) + 4 * g(j...) & \text{otherwise} \end{cases}$$

and set $b_{r,0} := g(\mathcal{I}) - b_0$.
From this definition follows that the formula $ff\,\mathcal{U} s \wedge tt\mathcal{U} t$ (the first part of which can only be true if we start in a state where s holds, and the second part of which can only hold if we eventually reach any of the states where t holds) can be proven iff the Turing machine M terminates on an input vector of $\mathcal{I}$. Solving the decision problem for this kind of logic is therefore at least as hard as deciding the halting problem for Turing machines, which is undecideable. □

4.3.3 The Star operator

Proposition 3:*If it is known beforehand that the maximum size of all formulas we need to verify is below a certain bound l, we can emulate any expression $[\alpha^*]\phi$ with atomic*[6] *α for $|\alpha|$ by $[\alpha^{l+|\Sigma|^{l+1}}]\phi$, where $[\alpha^n]\phi$ means that ϕ holds after α is executed anywhere from 0 to n times.*

Proof: The only operators whose closure is questionable are the temporal past operators $\ominus$ and $\mathcal{S}$.
Closure under $\ominus$: For any syntactically correct formula with a bound of l, at most $l-1$ $\ominus$ operators may be present in the formula. In the worst case, they refer $l-1$ steps into the past. Considering all temporal traces

$$\rho = \langle ...(\sigma_{n-1}, \alpha), (\sigma_n, \alpha)\rangle$$

where σ_n is the final state described (non-deterministically) by our full sequence of emulated actions, we know that for any $i \in \{1...|\Sigma|\}$ there must exist a state σ_{n-i} where $\sigma_{n-i} = \sigma_n$, because of the finiteness of our set of states. Therefore, we already are in a non-deterministic loop spanning the full range of $l < (l + |\Sigma|^{l+1})$, and any formula (without the $\mathcal{S}$ operator) of the length l which does *not* hold after α has been executed several more times has already not held in the past.
Closure under $\mathcal{S}$: Almost identical to the proof for $\ominus$, except that, if $\phi\mathcal{S}\psi$ is chosen cleverly, it can refer back to the past for up to $|\Sigma|^{l-1}$ steps non-deterministically, so it certainly can not refer backwards more than $|\Sigma|^l$ steps for the full formula. $\square$

Obviously, this approach is inappropriate for unknown values of l.

5 Summary

We have shown that a subset of the DDLTLB, as proposed in [1], is decideable, while both of the implicit extensions made to it (predicates and arithmetic formulas) make it undecideable. For practical reasons, it would be desirable to include at least arithmetical formulas anyway; if the numbers allowed for those formulas are chosen from a finite set of numbers, the resulting logic will, of course, remain decideable.

Regarding Sparse DDLTLB, it appears to be likely that an optimized decision algorithm (e.g. based on the PTL decision algorithm from [4]) can be constructed to decide formulas presented in it. As basic idea for this tableaux-based algorithm, we might label satisfactory edges with the actions that satisfy them, and recurse on obligations. More efficient methods may be possible, though this is left as material for a future discussion.

References

[1] F. Dignum and R. Kuiper. Combining dynamic deontic logic and temporal logic for the specification of deadlines, 1997.

[2] F.Dignum, H. Weigand, and E. Verharen. Meeting the deadline: on the formal specification of temporal deontic constraints, 1996.

[3] David Harel. *First-order dynamic logic*, volume 68. Springer-Verlag Inc., New York, NY, USA, 1979.

[6]Note that it is possible to prove a similar limit for non-atomic actions.

[4] Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. A decision algorithm for full propositional temporal logic, 1993.

[5] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[6] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall International, New Jersey, USA, 1995.